GEOMETRIC ALGORITHMS AND SOFTWARE ARCHITECTURE FOR COMPUTATIONAL PROTOTYPING: APPLICATIONS IN VASCULAR SURGERY AND MEMS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Nathan Marshall Wilson

December 2002

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____

Charles A. Taylor (Principal Adviser)
Assistant Professor of Mechanical Engineering and Surgery

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____

Robert W. Dutton (Co-Adviser)
Professor of Electrical Engineering

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____

Frank R. Arko III
Assistant Professor of Surgery

Approved for the University Committee on Graduate Studies:

_____

ABSTRACT

Traditionally, engineers have constructed and tested physical prototypes to create new devices and improve on existing designs. Since the advent of digital computing over 50 years ago, significant efforts have been underway to supplement (and in some cases replace) the need for physical prototypes with computational prototypes created and simulated on a computer. Most of the commercially available design software systems existing today originated in support of the automotive, aerospace, defense, and semiconductor industries thus making it difficult to apply these existing systems in new application areas such as medicine.

This dissertation details a general, extensible, modular software framework developed for computational prototyping. The framework integrates the three major stages in computational prototyping: creating geometry, discretization, and numerical simulation. To highlight the versatility and extensibility of the framework, two specific applications were demonstrated.

The first area of application of the software framework was in the field of vascular surgery. When planning a surgical procedure to restore blood flow to the lower extremities for a given patient, vascular surgeons rely primarily on intuition and past experience to develop a surgical plan. In this work, a surgical planning system was developed enabling a vascular surgeon to create and test alternative operative plans prior to surgery for a given patient. Hemodynamic (i.e. blood flow) simulations were performed for the operative plans for two aorto-femoral bypass patients and compared with actual postoperative data. The information that can be obtained from hemodynamic simulation (e.g. wall shear stress) may be clinically relevant to future vascular surgeons planning surgical interventions.

The software framework was also extended for use in computational prototyping of micro-electro-mechanical systems (MEMS). While MEMS are often constructed utilizing integrated-circuit fabrication techniques, the size and aspect ratios of typical MEM structures differ significantly from those traditionally found in the VLSI community. In this work, geometric algorithms were developed to incorporate two- and three-dimensional process simulation from VLSI to construct three-dimensional geometric models for simulation-based design of MEM devices with particular emphasis given to geometric modeling of a radio-frequency micro-switch.

ACKNOWLEDGEMENTS

No PhD is done in a vacuum, and this was no exception. Having spent over 11 years in college, there are numerous people that influenced my professional development as well as enhanced my social life over these years. Here I'll try and summarize just the major characters in the "Nathan Wilson PhD Experience."

To begin with, the most influential person in the experience of a PhD student is their principal advisor. In my case, I had the good fortune to work closely over the years with two faculty members who served as my principal advisors. During the first years of my PhD, I worked closely with my co-advisor Professor Bob Dutton. Working for Bob has been a wonderful experience, and his dedication to his students is remarkable. It was thanks to Bob's support I made it through the early days of the PhD process, and I managed to travel to several international conferences as a representative of his Technology CAD research group.

I also want to thank my principal advisor, Professor Charley Taylor for his support over the years. It is easy to be supportive of people when things are going well, but the real test of a great advisor is when things look their worst. In my case, the biggest challenge I faced in graduate school was passing my qualifying exam. Charley really helped me get through the qualifying exam process, and I owe him a great deal for that.

In addition to my principal advisors, two additional faculty members have profoundly impacted my experience at Stanford. In particular, Professor Frank Arko III provided the clinical expertise critical for my work in simulation-based medical planning. I want to thank Frank for his patience, cooperation, and desire to help me understand the current treatment of vascular disease. Finally, I would like to thank Professor Tom Hughes. It was working with Tom that initially attracted me to Stanford, and his finite-element

classes and discussions about research were extremely important in shaping my PhD experience.

I want to thank the members of the Taylor and Dutton research groups. In addition to engaging in discussions relevant to my work, many of them have been friends over the years as well. In particular, I want to thank Ken Wang, Mary Draney, Joy Ku, Bev Tang, Chris Cheng, and David Parker and the rest of the Taylor group. I need to acknowledge that my work in simulation-based medical planning would have been impossible without the contributions Ken made during his PhD research. Mary's assistance with the acquisition and processing of MRA and PCMRI data was also critical to my work. In the Dutton group, I would like to especially thank Dan Yergeau, Edward Chan, and Michael Kwong. Dan has been a good friend over the years, and an amazing resource for my computational endeavors. Discussions with Edward about MEMS were invaluable to my research in that area.

In addition to work, the support of my close friends over the years has made my PhD experience a fulfilling one. I'd really like to thank Vineet Sarin, Chris Roat, Chris Hernandez, Laura Cook, Maya Beasley, Marc Glazer, Faye Steiner, Robert Schaffer, and Emily O'Connor. My friends were there for the good times and the bad, but I'm guessing in a few years we'll only remember the good times.

I also want to take this opportunity to thank the people from my undergraduate days at Virginia Tech. In particular, Professor Romesh Batra and Professor John Mahan were instrumental in motivating me to pursue an advanced degree. I also want to thank my friends from those days including Dave Young, Grant Corley, Steve and Jodie Mellinger, and Erica Lewkowicz.

I would like to thank my family. None of this would have been possible without the love and support of my Mom and Dad. They have been there for me from the time I started

kindergarten all the way though my PhD. I also want to thank my brother Seth for being there over the years.

I would like to thank Jill Higginson for diligently proofreading this thesis. Her help was greatly appreciated.

Finally, the people who paid some of the bills during my graduate career deserve some credit. I would like to thank DARPA (contract F30602-96-2-0308), the Intel Foundation, and NSF (contract ACI-0205741) for providing the funding for my PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PART I – INTRODUCTION TO COMPUTATIONAL PROTOTYPING

Part I of this thesis introduces the concepts and machinery of simulation-based design and computational prototyping. A general, extensible, modular framework is then described which is shown to be applicable to simulation-based medical planning (Part II) and MEMS design (Part III).

CHAPTER 1 INTRODUCTION

1.1 ENGINEERING DESIGN PROCESS

To design is to formulate a plan for the satisfaction of a human need [1]. The first step in design then is to clearly define a problem that needs to be solved, and identify which variables may be controlled to achieve the desired objective. Often a great deal of thought must be given to the specification of the design problem so that any solution can be obtained. For example, ending world hunger may be a worthy problem satisfying a basic human need, but is likely an ill-defined problem with too many possible variables to control.

Engineering design is the process in which scientific principles and the tools of engineering (e.g. mathematics, computers, graphics) are used to produce a plan which, when carried out, will satisfy a human need [1]. There is a wide range of tools available to the engineer to study a given problem. Past experience, federal guidelines, computer modeling, and production and testing of physical prototypes are all examples of the knowledge and tools an engineer may use to design a solution to a particular need. For the purposes of this dissertation and the problems discussed herein, it is useful to further subdivide engineering design into two sub-categories: component design and system design.

It is easiest to explain the difference between the categories by using the everyday example of automotive design. A major individual component, e.g. the powertrain, may be designed to meet a desired level of durability, performance, and cost. However, the optimal design of the powertrain assemblage may in fact deteriorate the overall system performance (e.g. fuel economy) if designed in isolation. Historically, it was commonplace for development teams to work in isolation due to limitations in communications and the high level of expertise needed to use certain design aids (e.g.

computer modeling). Increasingly, however, technological advances (particularly in the field of computation) have created a new paradigm of system-level design. As will be seen, system-level design (e.g. performance of a medical device after implantation) places additional requirements on computer-aided design tools.

The overall goal of the work presented in this dissertation was to develop and demonstrate a software architecture and framework applicable to multiple design problems. A common foundation was developed, with additional application-specific functionality incorporated as needed. To provide continuity and clarity for the rest of this thesis, definitions will be given in this chapter for traditional prototyping, computational prototyping, simulation-based design, and simulation-based medical planning.

1.2 TRADITIONAL AND COMPUTATIONAL PROTOTYPING

For the purposes of this work, an important distinction is drawn between traditional prototyping and computational prototyping. In this dissertation, traditional prototyping (or simply prototyping) will refer to the act of creating physical experimental models to test and iterate to a desired set of design objectives. For example, if one were interested in evaluating the fuel economy of five different proposed designs of a new type of automobile, it is possible that five prototype vehicles could be built and tested. This type of testing typically involves a large investment of time and can be costly. In addition, exploring the entire design space is often cost prohibitive, requiring the engineer to employ (sometimes educated) guesses in the design process. If unanticipated issues arise during testing (e.g. an unexpected failure mode), additional prototypes may be fabricated further increasing the cost and delaying the process of design.

Computational prototyping can be defined as the use of mathematical and physical models, implemented in computer software and solved on computer hardware, to systematically evaluate the efficacy of alternative conceptual designs. Computational prototyping has several significant advantages over traditional prototyping. First,

computational prototyping takes advantage of the amazing increase in computational power of the later part of the 20th century. For example, in 1964 the most powerful supercomputers in existence achieved 1 million floating point operations per second (1 Mfps). By 1999, one of NASA's supercomputers reached 100 billion floating point operations per second (100,000 Mfps). This is an almost unbelievable gain of five orders of magnitude in less than four decades. Second, computational prototyping is typically less expensive than equivalent physical prototyping. Third, the total time required for testing a computational prototype is usually less than its physical counterpart. In addition, computational prototyping provides the entire field of engineering quantities of interest (e.g. stress and strain). Finally, computational prototyping may provide alternatives to physical testing embroiled with ethical considerations (e.g. animal testing).

To summarize, there are tradeoffs between traditional and computational prototyping. The major strength of physical testing is that it may reduce the number of simplifying approximations that need to be made. The most significant drawback is that, for the foreseeable future, the major costs involved in physical testing are unlikely to decrease. The major advantage of computational prototyping is the ability to study more of the design space and to obtain detailed features of field variables of interest. The most significant drawback to computational prototyping is that it requires sufficient understanding of the physics involved in the utilization of a designed device. For at least the next decade it appears very likely that computational power will grow significantly while associated costs per Mfps will continue to decrease dramatically. This undoubtedly will continue to increase the importance of computational prototyping in engineering design in the future.

1.3 MEDICAL TREATMENT AND TRADITIONAL PROTOTYPING

The current paradigm in planning treatment for patients with vascular disease is similar to traditional prototyping found in engineering. Specifically, a surgeon will select a procedure for a particular patient based on past experience for patients with a similar state of disease. The experience gained from this patient will be selectively used when treating the next patient with similar symptoms. In this fashion, the surgeon is using a "build and test" approach where the design of a particular procedure is iteratively improved. The unfortunate downside to this approach from the viewpoint of an individual patient is that the surgeon cannot typically iterate in their individual treatment.

1.4 SIMULATION-BASED DESIGN SYSTEMS

Historically the tasks involved in iterative engineering design were divided between three groups of engineers: designers, analysts, and draftsmen. For example, a designer may make a clay model of a car. That model would be given to a draftsmen to create computer-aided design (CAD) drawings and a geometric solid model. This computational geometric model would be given to the analyst to perform numerical analysis. The results of the analysis would then be given back to the designer who would evaluate the results and the process would repeat until an acceptable design was achieved. This typical iterative design process is susceptible to human error and involves the coordination of multiple individuals to achieve success. This motivates the need for a system to enable a single user to carry out the steps of iterative design using computational prototyping.

Simulation-based design (SBD) systems are computer software systems that integrate model creation and modification, discretization, analysis, and visualization software programs within a common graphical user interface (GUI) to support the computational prototyping process for a specific application domain. The strength of a SBD system is the ability of a single or small number of users to perform the steps necessary for

computational prototyping tailored to their individual background while hiding intermediate steps (e.g. file conversions).

The desired objectives for a SBD system are to:

1. Provide a single consistent GUI customized for the needs and experience of the typical designer

2. Maintain and propagate relevant information through the process while minimizing possible inconsistencies and human error

3. Automate and streamline the individual tasks involved in the process

It is worth noting that general commercial integration environments do exist on top of which one can develop a SBD for a given application. An example of such an environment is the Adaptive Modeling Language (AML) [2]. The benefit of a commercial integration environment such as AML is that it greatly accelerates the process of creating an application-specific SBD system since much of the integration work has already been done. The drawback to an integration environment such as AML is that it can be rigid in the underlying functional components and this in turn can prevent the use of best-in-class component technology.

1.5 SIMULATION-BASED MEDICAL PLANNING

Simulation-based medical planning (SBMP), originally proposed by Taylor [3], refers to applying the methodology and machinery from the field of engineering for simulation-based design to assist in planning treatment for vascular disease in a patient-specific fashion. That is, SBMP enables a surgeon to evaluate different surgical options prior to treatment using patient-specific models of the vascular system. Once the geometric models representing the surgical options for a given patient have been created, the additional major steps in computational prototyping (i.e. discretization, simulation, and

visualization) can be performed enabling the surgeon to select the procedure most advantageous for the patient.

1.6 APPLICATION OF SBD AND SBMP SYSTEMS

SBD systems have only come into existence in the last decade of the 20th century. These systems typically consist of existing software packages coupled together in a proprietary framework developed internally at a large institution or corporation. These proprietary SBD systems are used in product design to reduce development costs and decrease the overall time-to-market for new designs. MEMCAD was the first system for MEMS design [4] and has since become the basis of two commercial products in use in the MEMS community. The "Stanford Virtual Vascular Laboratory (SVVL)" [5] was the pioneering system for vascular surgery applications, which enabled blood flow simulations in idealized vascular models.

1.7 SUMMARY OF PHD RESEARCH AND CONTRIBUTIONS

This thesis details the contributions made in the field of simulation-based medical planning and simulation-based design of micro-electro-mechanical-systems. Specifically a simulation-based design software system was developed with application to vascular surgery planning and MEMS design.

The first application of this system to be discussed will be for simulation-based medical planning. Prior to this work, it required months to go from medical imaging data to simulation results. More than fifteen separate computer programs were required to perform the steps of SBMP, and no single user could perform all of the steps in the process. In addition, no geometric models from actual surgical patients had been constructed and no system existed that could be used by a vascular surgeon to create a surgical plan prior to surgery. Finally, no method existed to prescribe experimentally determined velocity profiles as inflow boundary conditions.

Several unmet needs were identified for SBMP. First, it is critical to be able to build geometric models for real patients from medical imaging data in a clinically relevant time frame. Second, a system must exist that enables a vascular surgeon to create surgical plans prior to surgery for simulation. Finally, the system must enable the prescription of experimentally determined velocity profiles as boundary conditions for simulations.

A system was developed that addressed these needs and enables a vascular surgeon with the assistance of a technician to preoperatively evaluate different surgical procedures in a patient-specific fashion. Specific unique contributions of the system over previous work include:

1. Integrated and improved medial axis path planning algorithms
2. Improved 2-D image segmentation techniques including generalization with direct application to future work in 3-D image segmentation
3. Developed methods to propagate information through the SBMP system
4. Integrated novel segmentation and visualization capabilities to process flow data from Phase-Contrast Magnetic Resonance Imaging (PCMRI)
5. Improved robustness of alignment algorithms used in the construction of 3-D models from 2-D segmentations
6. Integrated SBMP-specific scientific visualization techniques
7. Created a robust, unified GUI for SBMP

The system detailed in Chapter 3 is the first system to enable the construction of geometric models for patients prior to surgery in a clinically relevant time frame. In addition, examples in Chapter 4 show surgical plans created by a vascular surgeon using the system that may enable a surgeon to preoperatively evaluate different surgical procedures in a patient-specific fashion in the future.

The second application to be discussed is a system for computational prototyping of micro-electro-mechanical systems. Geometric algorithms were developed enabling a MEMS designer to go from mask and process information directly to 3-D geometry for computational prototyping. The algorithms enabled the creation of geometry of realistic dimensions for MEMS simulation with different levels of physical accuracy. Specific contributions include:

1. An explicit implementation of a minimal idealized process specification
2. Integration of 2-D and 3-D physical process simulation relevant to MEMS
3. A "domain-decomposition" algorithm to reduce by orders of magnitude the computational expense in performing physical process simulation for certain classes of MEM devices
4. A prototype Internet-based MEMS geometry tool

Chapter 6 highlights the application of the methods developed including an example in which more realistic geometric models of a test RF switch structure were created. Demonstration of an Internet-based MEMS geometry tool is also shown.

1.8 ORGANIZATION OF PHD THESIS

This thesis is organized as follows: Part I, consisting of Chapters 1 and 2, introduces a general framework applicable for both simulation-based design and simulation-based medical planning. The major steps in the process and component technologies are introduced and discussed individually in Chapter 2. Part II (Chapters 3 and 4) introduces and demonstrates a system developed for simulation-based medical planning. Chapter 3 introduces the methods required specifically for vascular surgery planning, while Chapter 4 shows clinically relevant applications. Part III (Chapters 5 and 6) introduces and demonstrates a system for computational prototyping of micro-electro-mechanical

systems. Chapter 5 focuses on methods specific to MEMS prototyping. Chapter 6 shows MEMS-related examples and applications. Conclusions and future work for both vascular surgery planning and MEMS prototyping are discussed in Chapter 7. Finally, Appendix A provides additional implementation details and information regarding the graphical user interface for simulation-based medical planning.

.

CHAPTER 2 SOFTWARE ARCHITECTURE AND COMPONENT TECHNOLOGY

2.1 OVERVIEW

This chapter describes in detail the core functional areas in computational prototyping and outlines a general framework and software architecture that can be applied to a wide class of problems of engineering interest. First the core functional areas of surface evolution techniques, solid modeling, discretization (i.e. grid or mesh generation), numerical simulation, and scientific visualization are introduced. A cohesive framework is then described to join these individual components. Finally, to demonstrate application-specific customizations Chapter 3 and Chapter 5 show extensions of this framework for two particular applications.

2.2 COMPONENT TECHNOLOGIES

Historically the major components of computational prototyping were developed independently. This led to a vast body of literature in each major core functional area (usually with its own nomenclature, journals, and technical conferences). In each of the fields to be described, academic and commercial software was developed over at least the last three decades. Since few readers will be familiar with all of the areas relevant to computation prototyping, a brief introduction is presented for each major component with additional details available from the cited references.

2.2.1 Surface evolution

One may often think of computer-aided engineering as an engineer creating a proposed design (e.g. CAD drawings) and then wanting to predict its performance using computational prototyping. This is certainly the case in most instances and there are several large commercially available software packages intended for this purpose (e.g. [6], [7]). However, in many circumstances the actual fabrication process can deviate sufficiently from the specified design to significantly alter the performance (or failure) of a proposed design. It can be desirable in these cases to model the actual fabrication process used to construct the device (e.g. micro-fabrication techniques) which often means tracking an evolving surface as material is deposited or removed during manufacture. A general surface evolution technique is introduced in this section that is shown in Chapters 5 and 6 to have useful application in building geometric models for simulation of MEMS.

As will be discussed in detail in section 3.5.3, boundary evolution techniques can also be used for image segmentation and surface extraction. Although the volumetric image data to be introduced in the next chapter is not changing with respect to time, boundary evolution techniques provide a mechanism to enforce certain boundary surface characteristics and properties in the resulting segmentation. For example, the velocity functions to be discussed in section 3.5.3 evolve the boundary such that the resulting image segmentation satisfies certain smoothness criteria. While in some cases direct image segmentation techniques can be used in conjunction with post-processing algorithms to create segmentations with the desired properties, utilizing boundary evolution algorithms can lead to easier software implementations (particularly in 3-D) that can also handle more general cases.

There are numerous methods to track a boundary evolving with time. In the discussion that follows, n-space (i.e. $\Re^n$) will be divided into two distinct "material" regions. It is worth noting that the techniques discussed below can be generalized to more than two

material regions but this is non-trivial. The techniques will be separated into two broad categories by their method of surface representation: explicit and implicit. An explicit surface representation is one in which the boundary is specified directly (e.g. a set of line segments in 2-D or a surface triangulation in 3-D). An implicit representation is one in which the boundary is represented indirectly. An explicit boundary can be extracted from the implicit representation, but this requires an intermediate processing step (e.g. using a marching cubes algorithm for 3-D level set methods).

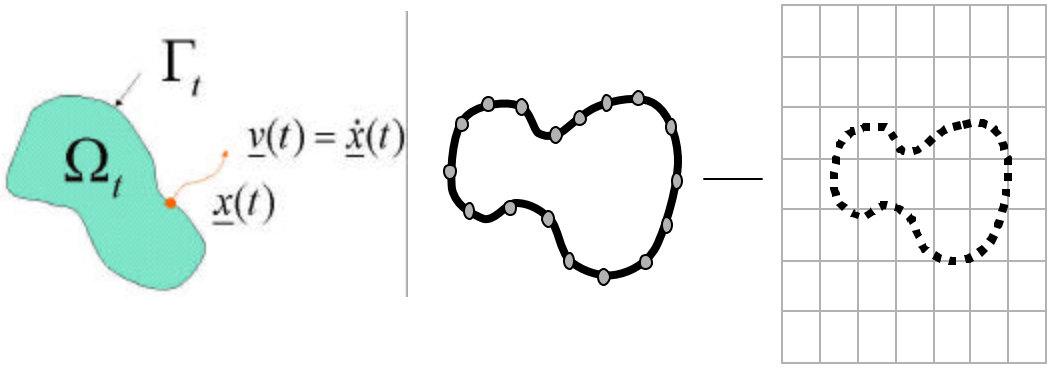
Explicit surface representations (also known as "marker" and "string" methods) have been widely used in many applications for 2-D surface evolution problems (see [8] and [9]). However, in 3-D the "bookkeeping" associated with these techniques becomes cumbersome particularly in the presence of topologic change (see section 2.2.2 for the definition of topology). This programmatic complexity motivated research into alternative methods such as implicit surface representations for boundary evolution problems.

A powerful surface evolution technique utilizing an implicit surface representation known as the "level set method" was introduced by Osher and Sethian [10]. Briefly, the level set method embeds a boundary of interest in a higher dimension space, and solves a partial differential equation governing the evolution of the interface (see Figure 2.1). That is, we define a scalar function ϕ such that:

$$f(\underline{x}(t),t) = 0 \qquad \underline{x} \in \Gamma_t \tag{2.1}$$

Using the chain rule and taking the time derivative of equation 2.1 (assuming sufficient differentiability) yields:

$$\begin{aligned} & f_t + \frac{\partial \underline{x}}{\partial t} \cdot \nabla f = 0 \Leftrightarrow f_t + \underline{v}(t) \cdot \nabla f = 0 \\ & \textit{where} \qquad \underline{v}(t) = \frac{\partial \underline{x}}{\partial t},\ \underline{x} \in \Re^n,\ t \in [0, T] \end{aligned} \tag{2.2}$$



(a) (b) (c)

Figure 2.1: The level set boundary representation. Figure (a) shows the abstract mathematical representation, while (b) shows 1-D boundary edges being embedded in a 2-D uniformly spaced grid (c).

Recalling the definition of the normal from differential geometry, and projecting the velocity function $\underline{v}(t)$ along the normal to the boundary, transforms the expression of (2.2) into the standard partial differential equation of the level set method:

$$\begin{aligned} & f_t + \underline{v}(t) \cdot \nabla f = 0 \quad \Rightarrow \quad f_t + F|\nabla f| = 0 \\ & \textit{where } \underline{n} = \frac{\nabla f}{|\nabla f|} \quad \textit{and} \quad \underline{v} = F\,\underline{n} \end{aligned} \tag{2.3}$$

All that remains is to pick a particular $f_0 = f(\underline{x}(0),0)$ and the normal velocity function F and we have an initial boundary value problem for the evolution in time of the hypersurface $f=(\underline{x}(t),t)$. A particularly useful choice for ϕ_0 is the signed-distance function:

$$|\nabla f_0| = 1 \quad t = 0,\ f \in \Re^n \tag{2.4}$$

Notice that thus far the development has been general without concern for the particular physics of the boundary evolution. Application-specific physics enters into the level set method only in two places: the speed function F and the projection of F from the

boundary (Γ_t) to the rest of the domain ($\Re^n$). Standard first-order accurate explicit finite-difference numerical methods for the solution of Hamilton-Jacobi equations are used to solve equation 2.3 as discussed in section 2.2.4. Additional details on the level set implementation used in the present work are described by Wang [11].

2.2.2 Solid modeling

Solid modeling refers to the development of three-dimensional models that are topologically complete (no missing faces or gaps) on a CAD system that uses one or more database techniques (i.e., CSG, B-Rep, or hybrid) specifically designed to modify, store, and display such models [12]. Solid modeling is essential for computational prototyping because it enables the manipulation of the geometry to investigate design alternatives.

There are two major solid modeling databases: Constructive Solid Geometry (CSG) and Boundary Representation (B-Rep). The CSG database structure defines and stores a solid as a series of unions, intersections, and differences of analytic or freeform shapes by boolean techniques. The topology of a CSG solid is implicitly defined (that is, the intersecting boundaries are mathematically implied). The B-Rep solid modeling database structure defines and stores a solid as a topological set of explicitly defined vertices, edges, and faces. The major commercially available solid modeling kernels ([13], [14], [15]) rely on the B-Rep database structure.

For the geometric algorithms discussed in the present work, it is important to understand the distinction between geometry and topology. Geometry is defined as the shape, size, and location of geometric elements such as points, lines, and planes. Topology refers to vertices, edges, and faces of a solid model that are either implicitly or explicitly defined. Figure 2.2 shows the relationship between topology and geometry for a unit cube. The B-Rep for the cube then consists of a database defining the use of faces, edges, and

vertices to define the region of the cube. Note that a topologically valid manifold solid can use a vertex an unlimited number of times, an edge exactly twice, and a face once.

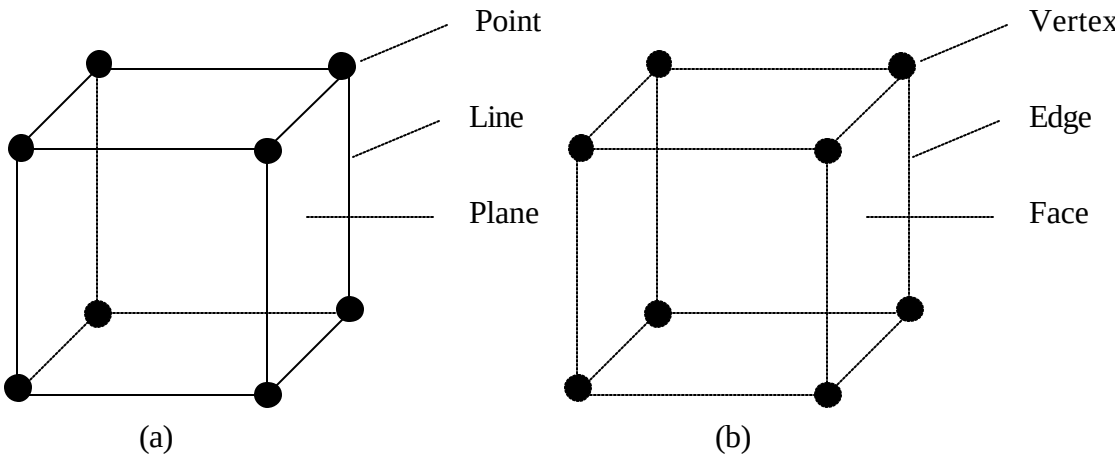


Figure 2.2: B-Rep of a unit cube. The B-Rep is a data structure that associates the geometric entities (a) with the topologic elements (b) to define a valid manifold solid object. In all, there are 33 topologic entities and 60 relations used to explicitly define a unit cube using a B-Rep.

In this work a "solid object" will refer to a valid topology with an associated valid geometry. All of the solid objects discussed throughout this dissertation should be assumed manifold unless explicitly specified otherwise. As seen in Figure 2.3, a solid model can be parametrically altered by changing the geometry attached to a specific topology. This will prove particularly useful in MEMS applications (see section 5.7.1.2).

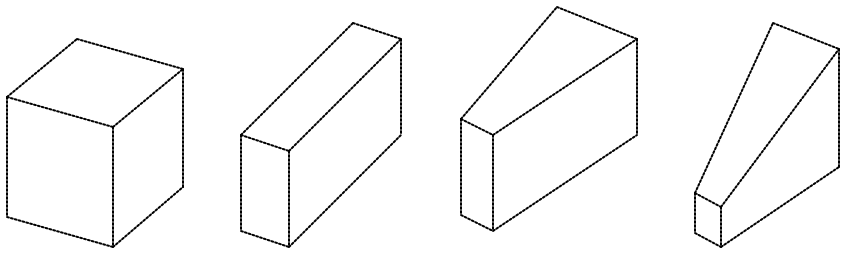


Figure 2.3: Parametric geometric changes to a topologic cube. All four solids shown in this figure have identical topology (given in Figure 2.2b) but different associated geometry.

2.2.3 Mesh generation

Discretization, also known as grid or mesh generation, is defined as the process of breaking up a physical domain into smaller sub-domains (usually called elements). Discretization is necessary in order to facilitate the numerical solution of partial differential equations. In other words, there are few analytic solutions for geometries of engineering interest thus the geometry must be divided into an aggregate of simpler pieces for analysis. Automatic mesh generation has been an area of intense research for decades, and a tremendous amount of literature and numerous algorithms have been developed. There are three fundamental challenges in the field: robustness, mesh quality, and computational efficiency in generating the mesh. A recent survey indicated there were over 80 commercial and academic meshing products available, of which 39 automatically generated tetrahedral ("tet") elements compared to 20 that performed unstructured hexahedral ("brick") mesh generation [16]. The current dominance of tetrahedral meshing can be attributed most notably to its ability to robustly mesh arbitrary, complex geometries. In addition, the use of tetrahedral elements often simplifies the process of adapting the mesh during simulation [17].

The following definitions will be useful in the discussion of mesh generation techniques applied in the present work [18]:

Definition: Given a set P of M unique points in n-dimensional space, an n-dimensional triangulation T^n is the set of N non-degenerate n-dimensional simplexes, s_i^n

$$T^n = \{s_1^n, s_2^n, s_3^n, \ldots, s_N^n\}$$

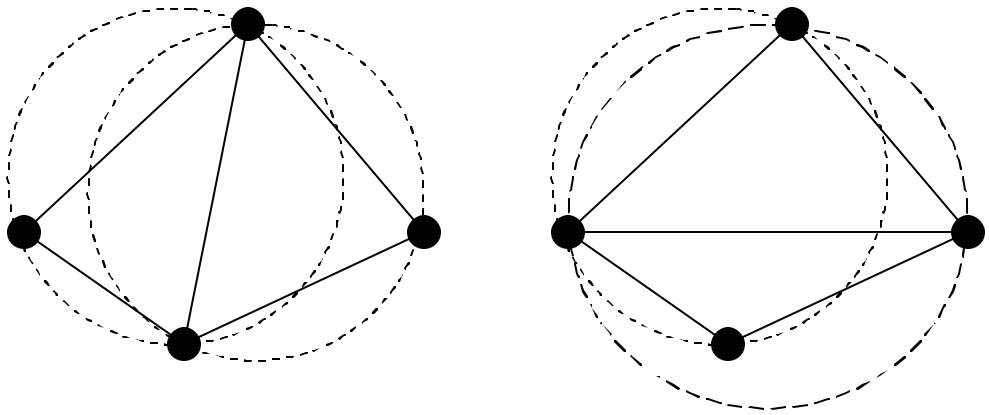
with the properties:

1. All vertices of each simplex $s_i^n \in P$
2. For each $i \neq j$, interior(s_i^n) $\cap$ interior(s_j^n) = $\varnothing$

3. The convex hull, C, of T^n is given by $C = \bigcup_{i=1}^{N} s_i^n$

4. The (n-1)-dimensional faces of s_i^n are either on the boundary of C and used by precisely one s_i^n, or in the interior of C and used by precisely two s_i^n.

Definition: If T^n of a point set P is such that the bounding hypersphere defined by the n+1 points of s_i^n contains no other points of P then that triangulation is a Delaunay triangulation T_D^n (see Figure 2.4).



(a) (b)

Figure 2.4: Delaunay criterion. The Delaunay criterion states that no other point in the triangulation can fall within the circumscribing sphere (circle in 2-D) of the points defining a simplex in the triangulation. Figure (a) shows a valid Delaunay triangulation of four points in $\Re^2$ while (b) shows a non-Delaunay triangulation of the same four points. In 2-D, the Delaunay criterion minimizes the maximum interior angle producing an optimal triangulation for a given set of points.

In the work presented here, two finite-octree based tetrahedral mesh generators derived from the same code base ([18], [19]) were used. The basic idea behind finite-octree methods is to decompose a complex geometry into simpler pieces and then mesh the individual pieces using a mesh generation technique (e.g. templates, Delaunay triangulation, etc.). An example quadtree decomposition (the 2-D analog of finite-octree) is shown in Figure 2.5. A structured (i.e. tensor-product) quadtree grid that completely fills the space occupied by the geometric model (i.e. bounding box) is initially created

based on a user-defined mesh density. Subdivision of the octants is then performed to reach a desired complexity of the geometry contained therein. Restricting the "level difference" or gradation between adjacent octants is essential to preserve mesh quality. After the geometry is decomposed, surface meshing is performed using projected 2-D Delaunay surface triangulation. Templates are used to create the interior volume mesh (i.e. octants contained completely inside of the geometric model), and 3-D Delaunay triangulation is used in the boundary octants (i.e. octants containing part of the geometric model boundary) to finish the meshing process.

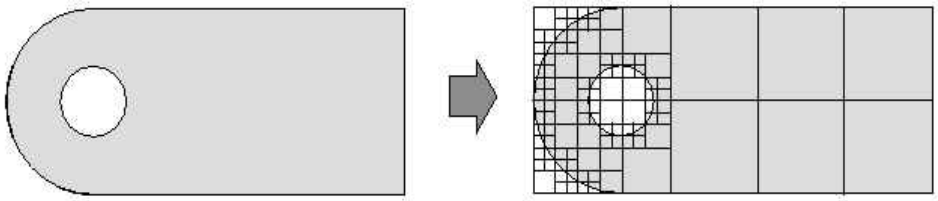


Figure 2.5: Quadtree decomposition. The figure (from [16]) shows an example quadtree decomposition (directly analogous to octree decomposition in 3-D) that is used to divide the geometry into less complex individual pieces for automatic mesh generation.

Finally, we note that several techniques to evaluate the quality of the discretization are used in this work. For the exterior surface mesh, a visual inspection may provide useful information. However, it is impractical to visualize the individual tetrahedral elements for large meshes, so geometric-based mesh quality indicators (see [20]) are used. Specifically, three mesh quality indicators will be discussed (see Figure 2.6):

1. Minimum solid angle
2. Radius ratio
3. Aspect ratio

In the present work, a combination of these quality indicators is used by an iterative mesh optimization algorithm to improve the overall quality of the mesh. It should be noted that there is current research interest in generating error estimators that include both solution

and geometric information (e.g. [21]) that may lead to more accurate solutions while requiring fewer elements.

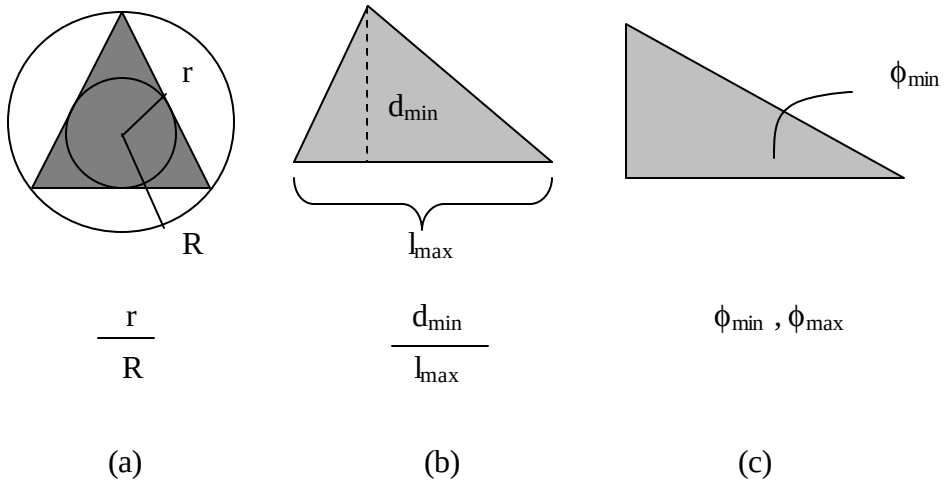


Figure 2.6: Geometric mesh quality measures. Shown are 2-D geometric mesh quality measures with direct analogies in 3-D. The radius ratio (a) is the ratio of the radius of the maximum inscribed circle (sphere in 3-D) over the radius of the circumscribed circle (sphere in 3-D). The aspect ratio (b) is a ratio of the minimum height to the maximum base length. The maximum/minimum interior (dihedral in 3-D) angle is shown in (c).

2.2.4 Numerical analysis

Computational prototyping requires that the performance of a device or design can be evaluated using a digital computer. Numerical methods used to solve the mathematical models (i.e. partial differential equations, PDE) play a pivotal role in simulation-based design systems. Common numerical methods include finite-difference schemes, finite-element methods, finite-volume methods, and boundary element methods. The choice of an appropriate numerical method depends on the PDE, the domain, the required boundary conditions, programming complexity, and personal preference. Two numerical methods are used extensively in this work. First, explicit finite-difference methods are used to solve the level set equations that are of the form of a Hamilton-Jacobi conservation law (see section 2.2.1). Second, finite-element methods are used to solve the equations governing blood flow (i.e. Navier-Stokes equations) and coupled elastodynamics and electrostatics for MEMS design.

Finite-difference techniques refer to a class of methods where the derivatives in a PDE are replaced with truncated Taylor series expansions. The method operates directly on the strong (i.e. classical) form of the problem. Most implementations of finite-difference techniques (including the one used in this work) utilize "stencils" to perform the necessary calculations. It is useful to recast the level set equation in a slightly different form from that of equation 2.3:

$$f_t + F_0\left|\nabla f\right| + \vec{U}(x,y,t)\cdot\nabla f = ek\left|\nabla f\right| \tag{2.5}$$

where:

F_0 = propagation expansion speed

$\vec{U}(x,y,t)\cdot\nabla f$ = passive advection term

$\vec{U}(x,y,t)$ = underlying velocity field

$ek\left|\nabla f\right|$ = speed based on curvature

The implementation of a first-order accurate, explicit forward Euler in-time finite-difference scheme with upwind differences for stabilizing the advective terms is given by:

$$f_{ij}^{n+1} = f_{ij}^{n} + \Delta t\left[\begin{array}{c} -[\max(F_{0ij},0)\nabla^{+} + \min(F_{0ij},0)\nabla^{-}] \\ -\left\{\begin{array}{l}[\max(u_{ij}^{n},0)D_{ij}^{-x} + \min(u_{ij}^{n},0)D_{ij}^{+x} \\ +\max(v_{ij}^{n},0)D_{ij}^{-y} + \min(v_{ij}^{n},0)D_{ij}^{+y}]\end{array}\right\} \\ +[eK^{n}{}_{i,j}((D_{ij}^{0x})^2 + (D_{ij}^{0y})^2)^{1/2}]\end{array}\right] \tag{2.6}$$

where $\vec{U} = (u,v)$, $K^{n}{}_{i,j}$ is the central difference operator approximation to the curvature expression in equation 2.5, and the D_{ij} represent one-sided difference operators (see [9] for additional details and higher-order accurate schemes). It should be noted that efficient methods are utilized in this work to solve the level set equations (see [11]).

The finite-element method will be introduced below to solve the equations of linear electrostatics (applicable to MEMS design). The finite-element method is used to solve the weak (or variational) form of the governing partial differential equation. The relationship between the strong form {S}, weak form {W}, Galerkin form {G}, and matrix form {M} are given in pictorial form [22]:

$$\{S\} \Leftrightarrow \{W\} \approx \{G\} \Leftrightarrow \{M\}$$

The strong form of the problem for electrostatics for linear-isotropic media (in the absence of free space charges) reduces to the elliptical partial differential equation known as Laplace's Equation:

$$\begin{aligned} &\nabla^2 f = 0 \quad in\ \Omega \\ &f = g \quad on\ \Gamma_g \\ &\frac{\partial f}{\partial n} = h \quad on\ \Gamma_h \end{aligned} \tag{2.7}$$

where:

ϕ = potential (i.e. voltage)

g = prescribed voltage

h = prescribed flux

The weak form corresponding to equation 2.7 is given by:

$$\begin{aligned} &Find\ u \in S, such\ that\ \forall\ w \in V \\ &a(w,u) = (w,h)_\Gamma \end{aligned} \tag{2.8}$$

where:

$$a(w,u) = e\int_{\Omega} w_{,i}\, u_{,j}\, d\Omega$$
$$(w,h)_{\Gamma} = \int_{\Gamma} wh\, d\Gamma$$
$$V = \{w \mid w \in H^1, w = 0 \; on \; \Gamma_g\}$$
$$S = \{u \mid u \in H^1, u = g \; on \Gamma_g\}$$

The Galerkin form is then given by:

$$\begin{aligned} &Find\ v^h \in S^h, \ such\ that\ \forall\ w^h \in V^h \\ &a(w^h, v^h) = (w^h, h)_{\Gamma} - a(w^h, g^h) \end{aligned} \tag{2.9}$$

where:

$$u^h = v^h + g^h$$
$$V^h = \{w^h \mid w^h \in H^1, w^h = 0 \; on \; \Gamma_g\}$$
$$S^h = \{u^h \mid u^h \in H^1, u^h = g \; on \Gamma_g\}$$

The matrix form is then found by selecting a particular set of finite-element basis functions (typically low-order polynomials) and solving the resulting system of linear equations for the nodal unknowns v^h (see [22] and [23] for additional details).

In summary, it is noted here that both of the above numerical methods converge in the limit as the element size and time step is reduced (for finite-differences the explicit forward Euler method requires that the Courant-Fredericks-Levy condition also be met [9]). In the case of the finite-element method, the Lax Equivalence Theorem guarantees convergence due to the stability and consistency of the Galerkin formulation. Unconditionally stable semi-discrete time integration methods were used when solving the Navier-Stokes equations (Chapter 4) and elastodynamics (Chapter 6). Finally, stabilized finite-element methods have been employed in the case of blood flow simulations (see section 3.9).

2.2.5 Scientific visualization

Scientific visualization is the formal name given to the field in computer science that encompasses user interface, data representation and processing algorithms, visual representations, and other sensory presentation such as sound or touch [24]. Simply stated, visualization is the process of exploring, transforming, and viewing data as images to gain understanding and insight into the data [25]. Given the vast amount of raw data in both imaging data and simulation results presented in this work, the keystone of a useful simulation-based design system is scientific visualization. Figure 2.7 shows a schematic of the visualization process.

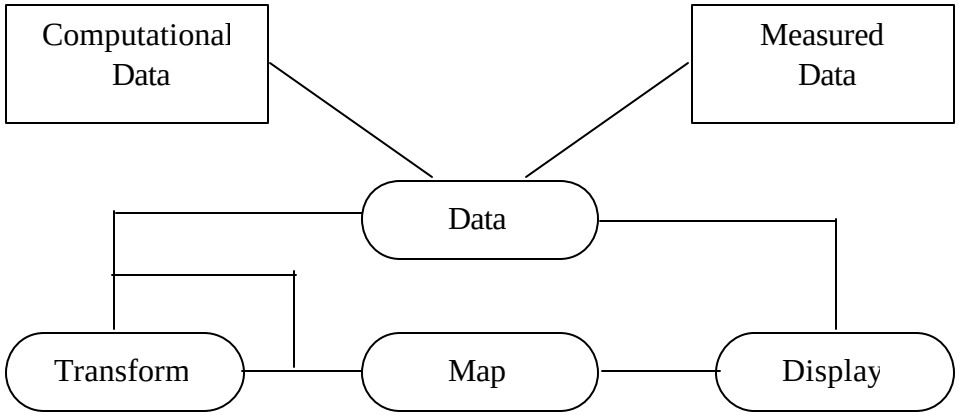


Figure 2.7: The visualization process. Data is transformed to extract and enhance information and the resulting data is mapped to the graphics system for display (adapted from [25]).

The fields of computer graphics and visualization are large, and the focus here will be on two particular methods of importance to the work contained herein. First, particularly useful in the field of surgical planning, is volume rendering. In this work, the broad definition of volume rendering as any method that operates on volumetric data to produce an image will be used. Volume rendering differs from surface rendering techniques (also used extensively in this work) in that surface rendering requires explicit triangulated representations of the geometry for visualization. The second common technique to be described below is referred to as surface decimation or surface smoothing. These play

particularly important roles when it is desired to use faceted geometry (e.g. from level set simulation) in subsequent solid modeling operations.

The utility of volume rendering is demonstrated in Figure 2.8. These volume rendered images show an abdominal aortic aneurysm in a patient prior to an endovascular aneurysm repair. The image is controlled by the user defining several "transfer" functions (for color, opacity, and gradient) that are then used to construct the images shown. Particularly if advanced techniques or specialized hardware is utilized, interactive frame rates can be achieved enabling the user to interact conveniently with the 3-D data [26]. While techniques such as isocontouring (e.g. using a marching cubes algorithm [25]) can be used to extract geometric primitives to render 3-D data, practical issues such as noise and limited image resolution can make this difficult. The power of volume visualization then is to utilize the human senses and facilities to "filter" the displayed information (such as ignoring the partial volume effects of a red tint on some of the bone surface). In this work, both hardware accelerated volume visualization [26] and software rendering using ray-casting were utilized. Both the software and hardware implementations allowed the mixing of volume visualization with geometric primitives (e.g. faceted solid models).

Another algorithm from the visualization community useful for the work contained in this thesis is known as surface decimation (also called mesh simplification). Decimation takes a given faceted surface representation and reduces the number of triangles or polygons used in the representation. Decimation accelerates graphical rendering and reduces the transmission costs of sending geometric models over the Internet. Numerous techniques exist for decimation (e.g. [27], [28], [29], [30]). Figure 2.9 shows an example of surface decimation using two different algorithms. While both are equally good for surface visualization, the surface in Figure 2.9e is more suited for the geometric algorithms discussed in Chapter 5 (due to the better quality of the triangles representing the surface). Unfortunately, the implementation of [30] is complex and is not widely available, which limits its use for geometric modeling in the present work.

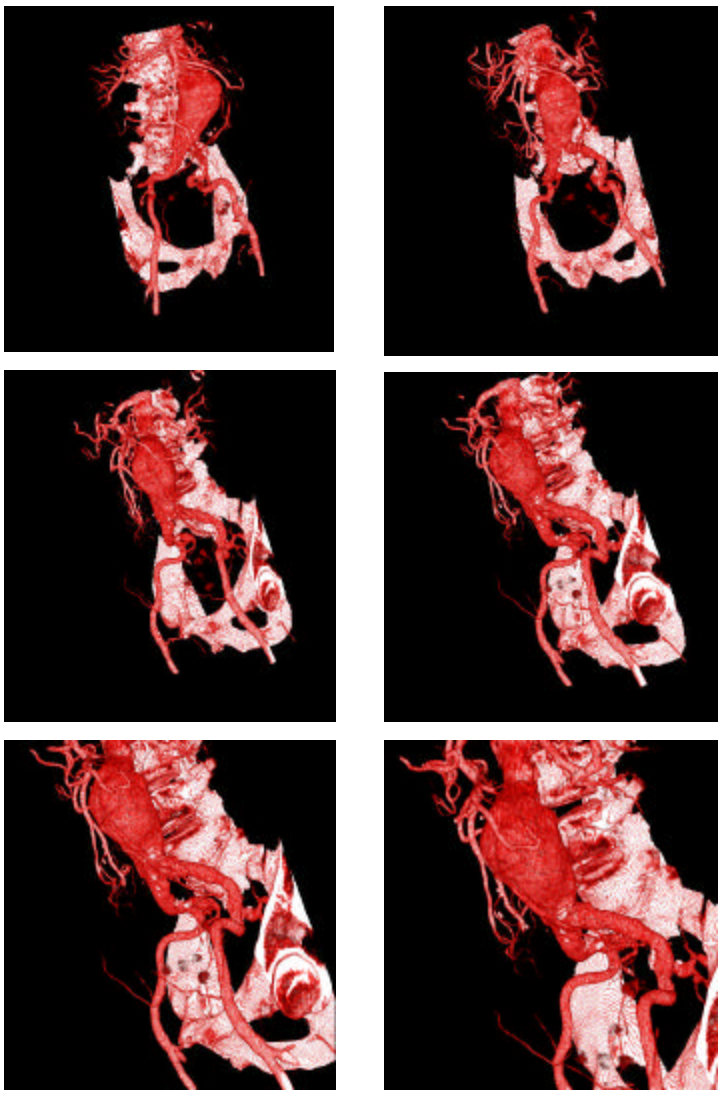


Figure 2.8: Images created using volume rendering. Six views of an abdominal aortic aneurysm (see section 3.2.2) created using software volume rendering found in [25].

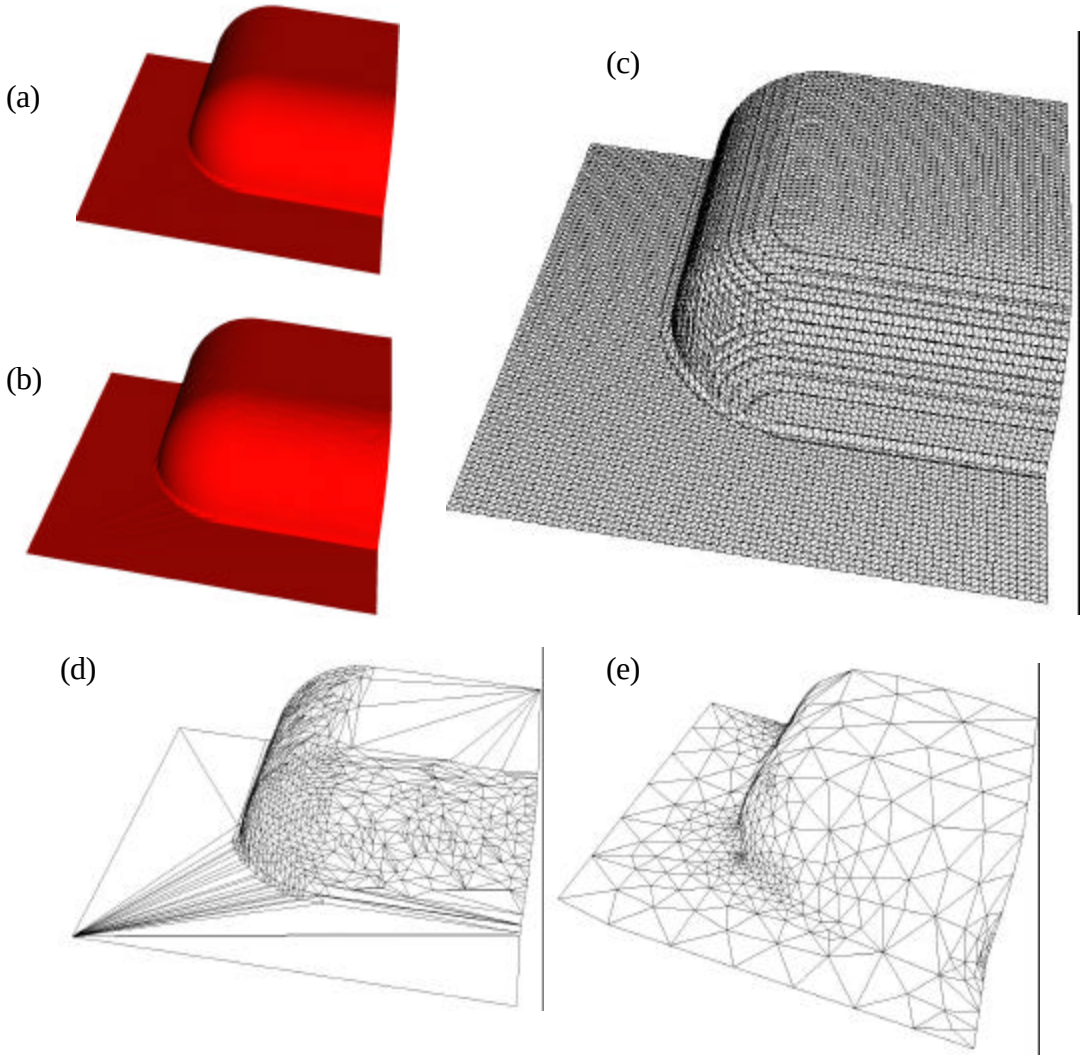


Figure 2.9: Example of surface decimation. Figure (a) shows the shaded surface mesh seen in (c) which consists of 4000 triangles. Figure (b) shows the shaded decimated surface given in (d). Visually, (a) and (b) appear to be the same surface. Figure (d) is a decimated version of (c) consisting of 1474 triangles created using a modified version of the algorithm in [29]. Figure (e) shows a decimated surface consisting of 658 triangles using the algorithm found in [30]. The surface shown in (e) is more appropriate for subsequent geometric processing or mesh generation because of the better quality of the surface triangulation.

2.3 SYSTEM ARCHITECTURE

The challenge of integrating diverse tools such as solid modelers and mesh generation software into a unified framework involves integrating tasks of varying computational expense and requirements. The framework needs to be general enough to handle different computational prototyping endeavors (e.g. surgical planning and MEMS design) with the ability to develop special application-specific algorithms. In addition, in a research setting it is desired to have a tool that enables rapid prototyping and quick testing of new algorithms. The overall requirements for the system can be summarized as:

1. Minimal overhead in the framework
2. Consistent application programmers interface (API) across modules to simplify development
3. Modular design to allow the inclusion / exclusion of functionality based on application needs
4. A high level interface useful for quick implementation of new functionality
5. A low level interface to enable the integration of efficient code for computationally intensive tasks
6. Enable the use of external kernels and libraries
7. A flexible data exchange mechanism to enable information transmission across modules
8. Enable the use of state-of-the-art computer hardware, software, development tools, and programming languages
9. Platform independence

A schematic of the framework (called **Geodesic**) developed to satisfy these requirements is shown in Figure 2.10. The figure shows three distinctive parts: a front-end integration environment, a data repository for an abstract data exchange between modules, and

modules roughly corresponding to the major tasks in simulation-based design. These parts are discussed in more detail below.

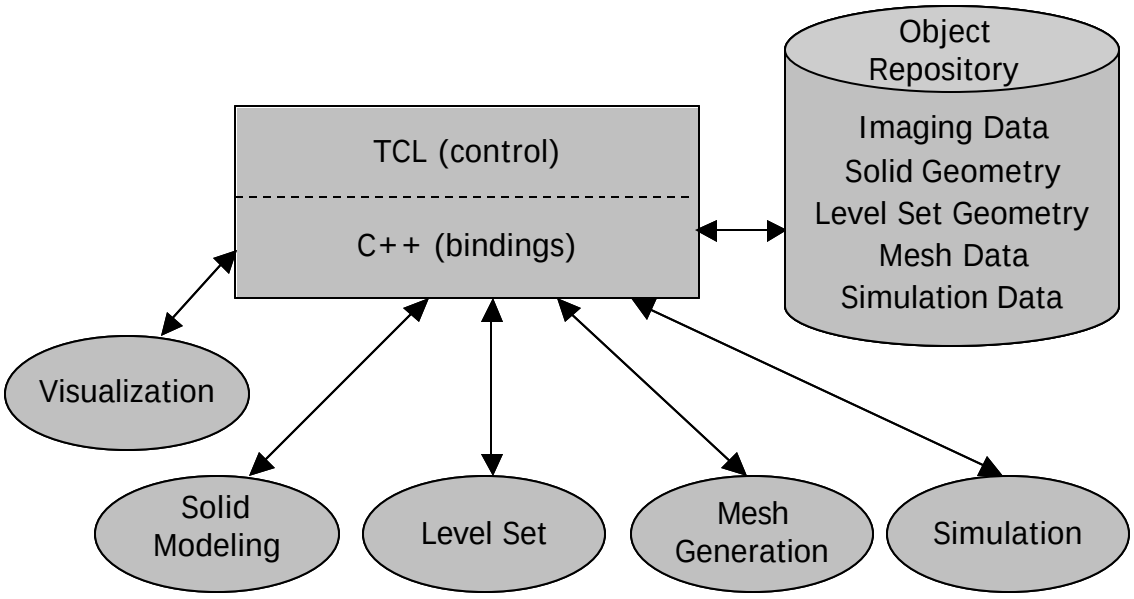


Figure 2.10: **Geodesic** modular software architecture. Three distinct parts exist in the framework: a front-end integration environment (shown as a rectangle), an object repository (shown as a cylinder), and five modules (shown as ellipses).

2.3.1 Front-end integration environment

To meet the desire for high level access and enable the inclusion of low-level code, a popular scripting language, the "Tool control language" (Tcl), was selected as a front-end integration environment [31]. In other words, Tcl combines the ease of a powerful scripting language with the ability to embed C/C++ code for computationally intensive operations. Scripting languages allow code to be written quickly and easily at the expense of computational efficiency. Many of the key algorithms in **Geodesic** were initially prototyped in Tcl and then rewritten in C++.

By creating custom high-level Tcl-bound commands to the underlying C++ code an intermediate level of access is provided for non-core developers. In addition, a companion library to Tcl named the "Toolkit" (Tk) was used to develop platform

independent graphical user interfaces providing user-friendly access to common tasks. The hierarchy of user and developer access is shown in Figure 2.11.

2.3.2 Abstract object repository

The object repository is a simple hash table of names (character strings) with corresponding object pointers. The repository allows for the basic operations of adding, deleting, listing, and querying of object type. This layer does not place any restrictions or requirements on the underlying structure of the objects contained in the repository, it only returns the pointers and calls instantiation and deletion methods as required. The details of the repository data object are shown in Figure 2.12. Minimizing the number of object types simplifies the communication between modules. While it may be possible to translate the fundamental objects in the repository to generic representations (e.g. solid models into STEP format [32]) to increase compatibility, this presents programmatic complexity and performance degradation and was not done in **Geodesic**. Also, because of the numerous useful algorithms and open-source code nature of the Visualization Toolkit (VTK) [25], several VTK classes were used to pass information between modules in **Geodesic** as shown in Figure 2.12.

2.3.3 Modules

There are five main modules in **Geodesic**: the solid modeling module, the level set kernel, the meshing interface, the visualization module, and the simulation module.

At the heart of geometric modeling is a solid modeler. By wrapping the solid modeling function calls used in a generic interface layer, **Geodesic** can be used with multiple solid modeling kernels. Because these packages inevitably exhibit relative strengths and weaknesses, it is useful to design for a plug-and-play modularity that enables the easy exchange of one implementation for another. This interoperability with multiple kernels is facilitated by designing the system to minimize the number of distinct function calls required to build a geometry. The current implementation can be used with two different

commercial solid model kernels ([13] and [14]). To date, only limited results have been achieved using a freely available solid modeler [33] due to limitations in its internal data representation and capabilities. The extension of **Geodesic** for use with other solid modelers (e.g. [15]) likely is straightforward.

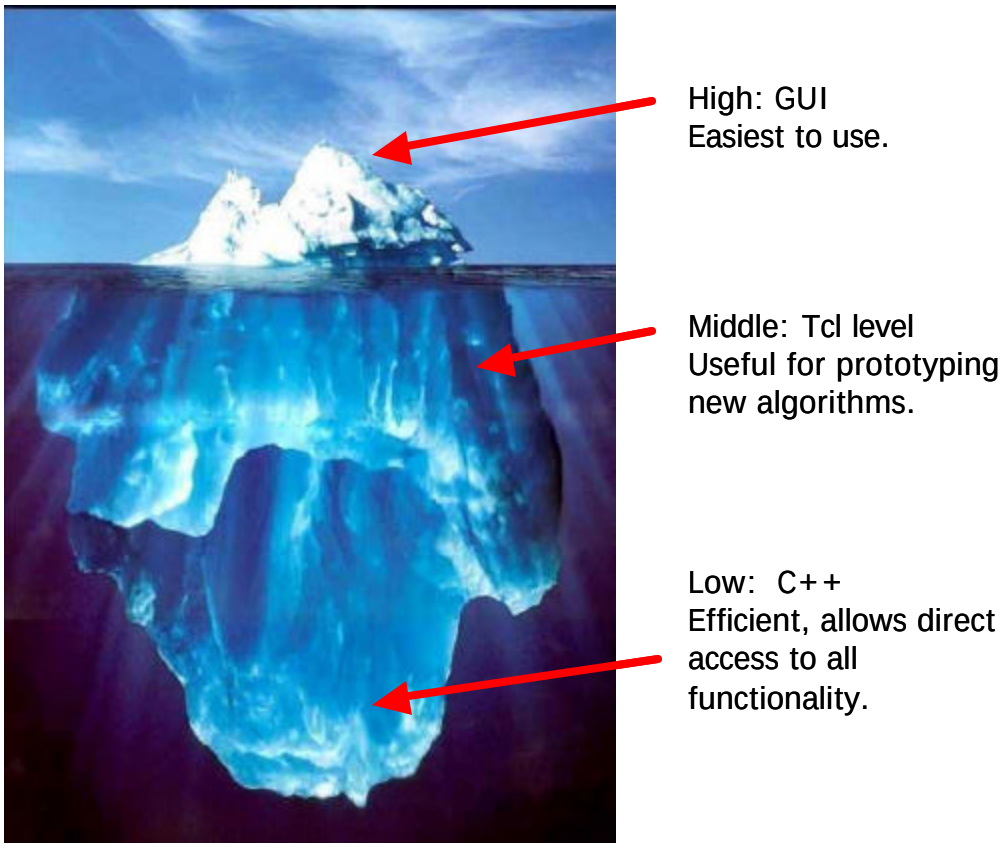


Figure 2.11: Levels of access in **Geodesic**. The graphical user interface (GUI) level hides most of the implementation details from the end-user (the tip of the iceberg), while advanced users and developers may also interact at the Tcl level (just below the surface of the water). Only core system developers interact at the lowest level where all of the functionality is completely exposed.

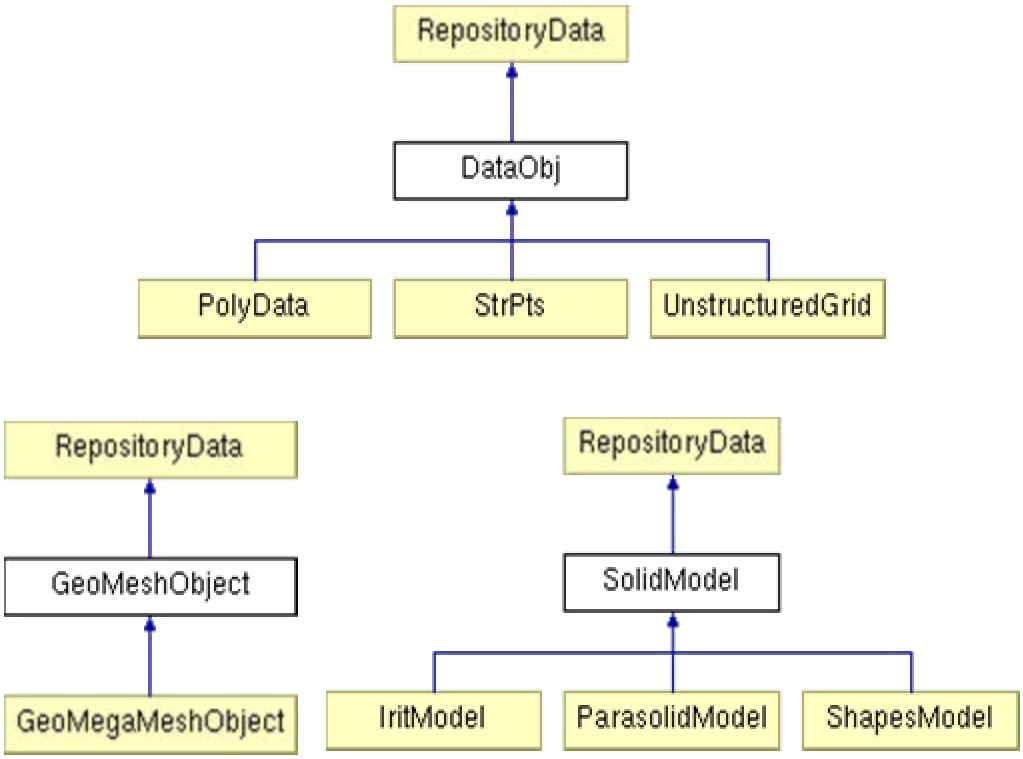


Figure 2.12: Repository hierarchy in **Geodesic**. Only three fundamental object types exist: visualization and simulation-related data (DataObj), mesh objects (GeoMeshObject), and solid model objects (SolidModel). The abstract DataObj class consists of three concrete dataset classes from VTK, the abstract GeoMeshObject class consists of a SCOREC/meshSim mesh database object, and the abstract SolidModel class consists of solid model objects represented in the internal format of Parasolid, Shapes, and Irit.

Geodesic contains a fully integrated general multi-dimensional level set kernel that can be used for image segmentation and physical process simulation (see [11]). Several performance-enhancing techniques are employed in the level set implementation found inside of **Geodesic**. These include using a compact storage scheme to reduce memory requirements and a technique known in the literature as narrow-banding that significantly reduces the computational requirements.

Geodesic also contains a generic meshing layer. In the current implementation, two automatic tetrahedral mesh generators ([18], [19]) are supported. Both of these mesh

generators provide boundary layer meshing. Boundary layer meshing has potential application in both simulation-based medical planning and MEMS simulation.

The visualization layer provides access to code to display graphical images on a computer screen. In the current implementation, this layer is a high-level interface to the Visualization Toolkit (VTK) that provides image processing, geometric, and volumetric visualization techniques [25]. VTK is an open-source cross-platform toolkit that is based on the OpenGL graphics API (see [34]). As previously mentioned, several VTK data objects are used extensively to communicate between modules in **Geodesic** because of their well-defined interface and powerful set of useful methods.

Finally, the simulation module can roughly be thought of as two distinct components. First, functionality to create required files for specifying boundary conditions is provided (part of this code resides in the meshing layer). Second, there is also embedded functionality to visualize the analysis results (known as “post processing”). The application chapters provide additional detail on the capabilities provided by the simulation layer.

PART II – SIMULATION-BASED MEDICAL PLANNING

Part I of this thesis introduced the general concepts and machinery of simulation-based design and computational prototyping. The next two chapters introduce and demonstrate additional functionality added to build an application (**ASPIRE**^{2}) on top of the **Geodesic** framework that enables vascular surgeons to evaluate different surgical procedures for a given patient prior to surgery. Using simulation to provide a vascular surgeon with additional information when planning a surgical procedure is known as simulation-based medical planning and was introduced by Taylor in 1999.

CHAPTER 3 SIMULATION-BASED MEDICAL PLANNING

3.1 OVERVIEW

This chapter introduces two particular types of clinically relevant vascular disease and describes the options currently available for the treatment of these conditions. The chapter also details functionality added into the **Geodesic** framework described previously to develop a software application (**ASPIRE**^{2}) that enables vascular surgeons to evaluate different surgical procedures for a given patient prior to surgery.

3.2 CARDIOVASCULAR DISEASE AND TREATMENT

Cardiovascular disease is one of the leading causes of death in the industrialized world [35]. As the median age continues to increase in the modern world, arterial disease will grow in importance due to its impact on quality of life and related medical expense. The work presented here focuses on two clinically relevant types of vascular disease: arterial occlusive disease of the lower extremities and abdominal aortic aneurysm. Before the idea of simulation-based medical planning is introduced formally in section 3.4, a brief overview of these two classes of disease along with current treatment options follows.

3.2.1 Aortoiliac occlusive disease

In symptomatic patients with occlusive disease of the lower extremities, the infra-renal abdominal aorta and the iliac arteries are the most common sites of obliterative atherosclerosis [36]. Atherosclerosis is a degenerative disease of complex origin which leads to a narrowing of the flow lumen preventing adequate flow (particularly during exercise) to the lower extremities. This inadequate flow, which results in claudication, is typically present in patients requiring direct anatomic surgical revascularization. Due to

the diffuse nature of arteriosclerosis disease, occlusive disease in the aortoiliac region frequently coexists with disease of the femoral arteries (see Figure 3.1).

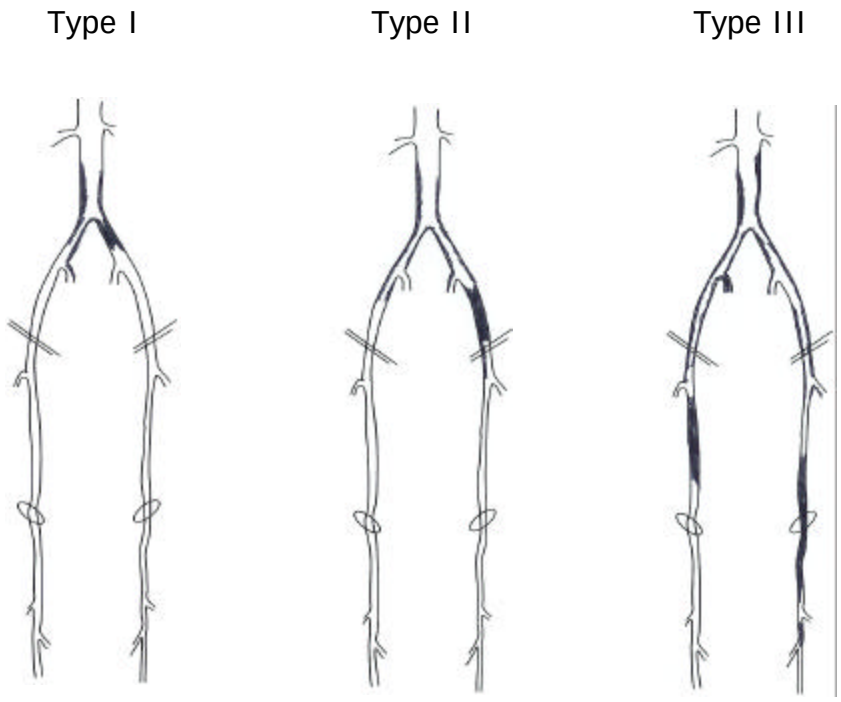


Figure 3.1: Patterns of aortoiliac occlusive disease. *Type I* consists of disease localized to the distal abdominal aorta and common iliac arteries. *Type II* consists of extensive intra-abdominal disease. *Type III* denotes multilevel disease (adapted from [36]).

Three major classes of treatment exist for aortoiliac occlusive disease. First, catheter-based endoluminal therapies such as angioplasty and stenting may be appropriate for localized occlusive disease. The use of angioplasty and stenting is receiving considerable attention today due to probable cost savings and decreased morbidity compared to traditional open surgical interventions. Second, extra-anatomic or indirect bypasses may be indicated for patients with serious heath conditions or who have experienced technical problems with past treatments (e.g. failure of a previous graft). Finally, the most commonly used procedure and the focus of the work presented in section 4.3 is direct anatomic surgical reconstruction.

Direct surgical reconstruction refers to the insertion of grafts replacing or providing alternative pathways to flow through the infra-renal aorta and iliac arteries. Figure 3.2 shows the two common types (classified by proximal anastomosis) of aorto-femoral bypass known as "end-to-end" and "end-to-side." With an end-to-end anastomosis, all of the flow in the infra-renal aorta enters the graft directly. With an end-to-side anastomosis, the body of the graft provides a second flow conduit causing the flow to be divided in some proportion between the distal native aorta and the bypass graft. While an end-to-end anastomosis is clearly indicated in patients with a coexisting aortic aneurysm, the decision in other cases is controversial [36].

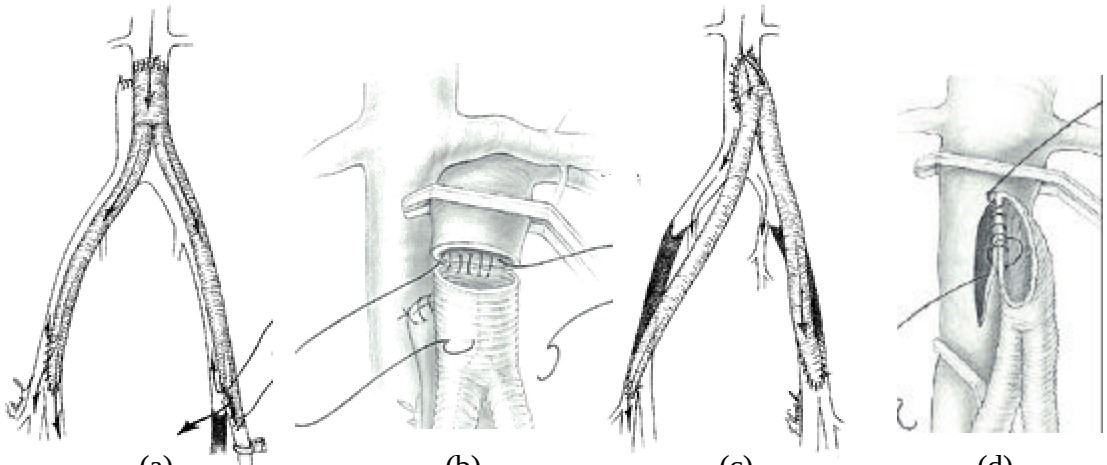


Figure 3.2: Two types of aorto-femoral bypass classified by proximal anastomosis (from [37]). A surgeon can elect to do an "end-to-end" bypass (a) or an "end-to-side" bypass (c). In an end-to-end bypass the graft body is attached directly to the infra-renal aorta (b), thus removing the distal native aorta from the flow path. In an end-to-side bypass, the graft is attached to the side of the aorta (d) providing an additional pathway for flow from the infra-renal aorta to the femoral arteries.

3.2.2 Abdominal aortic aneurysm

A second clinically relevant class of vascular disease to be studied here is abdominal aortic aneurysm (AAA) disease. Ruptured AAA's are the 15th leading cause of death overall and the 10th leading cause of death in men over the age of 55 years in the United States [38]. An aneurysm is defined as a focal dilation of at least 50% larger than

expected normal arterial diameter [39]. In practice, this typically translates to a cross-sectional diameter of about 3 cm for an AAA. The risk of rupture varies between 0.5% to 5% a year (for diameters of 4 to 5 cm) to 30% to 50% a year (for diameters greater than 8 cm). The paramount importance of diameter in determining AAA rupture risk is universally accepted [36]. Clinical opinion also holds that eccentric saccular aneurysms represent greater rupture risk than more diffuse, cylindrical aneurysms [36]. This clinical observation is in agreement with finite-element analysis of eccentric aneurysms [40].

For patients with a symptomatic AAA, surgical repair is usually appropriate due to low rates of operative morbidity. However, for patients with an asymptomatic AAA, the decision to operate is often difficult and depends on factors such as elective operative risk, AAA rupture risk, and life expectancy. The early (30-day) mortality rate after elective AAA repair in properly selected patients is 5% or less, whereas the early mortality rate after ruptured AAA repair averages 54% (not including deaths before repair) [36].

An emerging technique in treating AAA is known as endovascular treatment. Endovascular aneurysm repair involves the transluminal placement of a graft within the aneurysm that completely excludes the sac from the general circulation. The graft is held in place by a metal frame that supports all or part of the graft and provides a watertight proximal and distal seal. An important complication not found in open surgical repair of AAA's is endoleak. Endoleaks are defined by the persistence of blood flow outside the lumen of the endoluminal graft but within an aneurysm sac or adjacent vascular segment being treated by the graft. Endoleaks can be caused by an incomplete seal at the graft ends or by retrograde flow from patent lumbar arteries or other collateral vessels.

3.3 DIAGNOSTIC MEDICAL IMAGING

The two diagnostic imaging modalities used in the present work for geometric model construction are Computed Tomography Angiography (CTA) and Magnetic Resonance Angiography (MRA).

CT relies on using attenuation (reduction of energy) of x-rays as they pass through tissue to create an image. Modern CTA uses Spiral CT, single intravenous contrast bolus, small reconstruction intervals, and multi-planar reformatting during a single breath hold to construct detailed volumetric images. On the other hand, MRI is based on the reactions of various tissues to a magnetic field followed by a radio frequency pulse. MRA images are generated by taking advantage of the effects relating to the flow of blood relative to the stationary surrounding soft tissue.

While the exact resolution and dimensions vary from patient to patient, Table 3.1 lists representative parameters of the volumetric datasets used in the present work. In the end, MRA and CTA imaging produce a scalar set of data defined on an implicit three-dimensional (anisotropic) structured grid. The scalar value depends generally on the type of tissue, location in acquisition space, motion artifacts, and additional imaging modality-specific distortions.

3.4 SIMULATION-BASED MEDICAL PLANNING

The current paradigm for surgery planning for the treatment of vascular disease is shown in Figure 3.3. Briefly, diagnostic imaging data is acquired to assess the extent of aortoiliac disease in a given patient. The surgeon then relies on his/her past experience, personal bias, and previous surgical training to create a treatment strategy. The goal of this work was to create a simulation-based medical planning system for vascular disease that uses computational methods to evaluate alternative surgical options prior to treatment using patient-specific models of the vascular system (see Figure 3.4). Blood flow (hemodynamic) simulations enable a surgeon to see the flow features resulting from

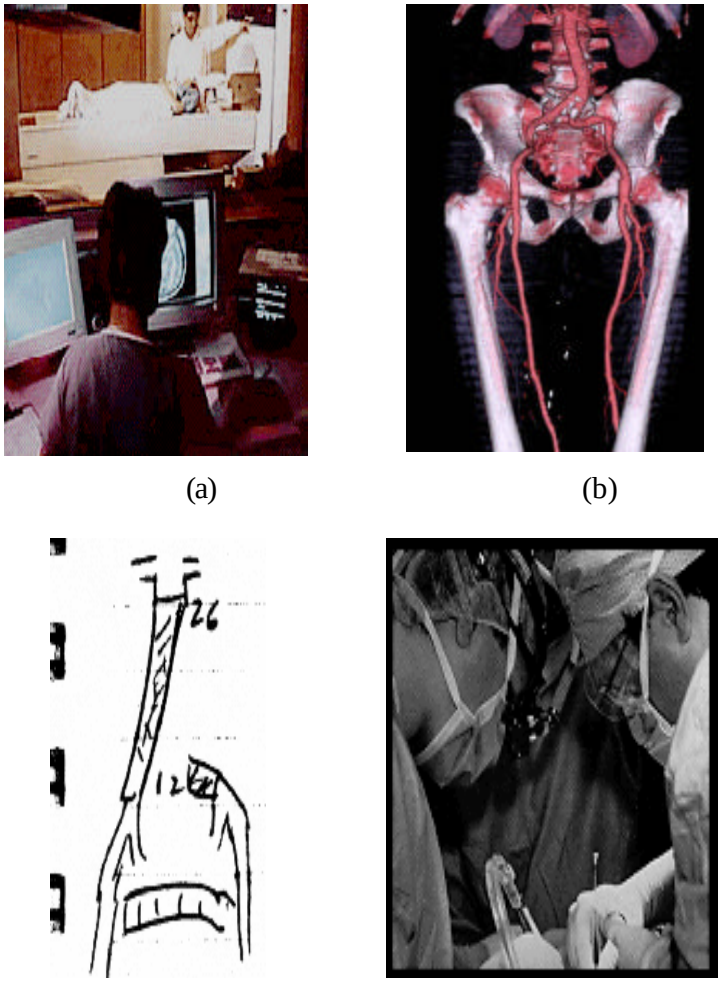
a proposed operation and to determine if they pose potential adverse effects such as increased risk of atherosclerosis and thrombosis formation. This type of information may in the future help reduce the controversy mentioned in the previous section over the proper care for certain patients.

Prior to this work, however, a significant bottleneck for simulation-based medical planning was the lengthy time required to build patient-specific geometric models and associated difficulties in discretization and numerical simulation. In this chapter, a software application, referred to as **ASPIRE^2** (the second generation Advanced Surgical Planning Interactive Research Environment), is described in detail. This software has been used to reduce the time required to build patient-specific geometric models from medical imaging data from several weeks to less than one day [41]. **ASPIRE^2** was designed for use by vascular surgeons (with the assistance of technicians) to preoperatively evaluate different surgical procedures for a given patient.

Table 3.1: Typical Volumetric Imaging Resolution of the Abdomen

Modality	**CTA**	**MRA**
Field of view (cm)	$34 \times 34 \times 44$	$44 \times 11 \times 44$
Resolution (mm)	$0.666 \times 0.666 \times 0.8^{*}$	$0.859^{+} \times 1.5^{++} \times 0.859^{+}$
Grid dimensions (R/L, A/P, S/I)	$512 \times 512 \times 551^{*}$	$512^{+} \times 76^{++} \times 512^{+}$
Total number of pixels	144,441,344	19,922,944
Dataset size (MB)	~ 300	~ 40

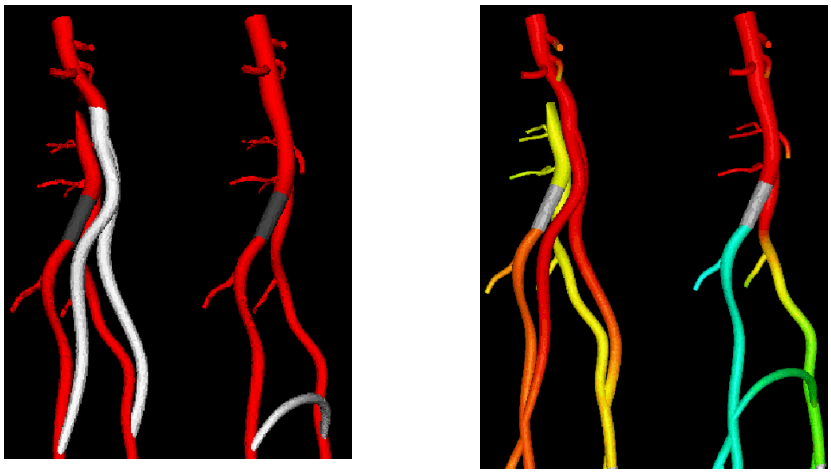
* slice thickness 1.25 mm, $^{+}$ 512 x 192 acquisition, $^{++}$ slice thickness 3.0 mm



(a) (b)

(c) (d)

Figure 3.3: Current paradigm in surgical planning. Diagnostic medical imaging is acquired (a) and visualized (b) from which a surgeon decides on an operative plan (c) based on past experience and personal bias. The surgeon then performs the operation (d) and the benefit of the procedure for the patient is assessed only after the operation.



(a) (b)

Figure 3.4: New paradigm of simulation-based medical planning [3]. Replacing pencil-and-paper plans with patient-specific surgical plans (a) created and simulated (b) on a computer enabling the evaluation of the efficacy of a procedure for a given patient prior to surgery.

3.4.1 Previous work

The first software system developed for studying hemodynamic factors for vascular adaptation and disease was the Stanford Virtual Vascular Laboratory (SVVL) [5]. This system predominantly relied on idealized models of the vascular system. Specifically, circular cross-sections were used to describe the surface of each artery branch to be included in the model and these individual artery models were joined (boolean addition) together to create a solid model representing the vascular system. This solid model could then be automatically meshed with unstructured tetrahedral elements using an automatic mesh generator. Finally, input files were automatically created for a finite-element fluid mechanics solver to perform hemodynamic simulation of the given vascular system. A major advantage of using idealized models was the ability to use parametric models, which enabled studies of a wide variety of vascular anatomy to be performed in a short period of time. The major shortcoming of the SVVL was its inability to build patient-specific preoperative geometric models from medical imaging data.

A conceptual prototype system for vascular surgery planning (named ASPIRE) has been described elsewhere [3]. This system had an Internet-based user interface specifically designed for a vascular surgeon. However, like the SVVL, this system also lacked the ability to build 3-D patient-specific geometric models from medical imaging data. In addition, the ASPIRE system did not permit the creation of actual bypass plans or the files necessary for simulation of three-dimensional blood flow.

Under continuous development since 1998, the **Geodesic** framework detailed in chapter 2 (see also [11]) serves as the foundation of the **ASPIRE^2** system. Originally intended for building geometric models from medical imaging data [11] and MEMS prototyping [42], this software framework was significantly extended in the present work to create the next-generation surgical planning system **ASPIRE^2**.

3.4.2 Detailed SBMP software system requirements

The major steps in the simulation-based medical planning process for vascular surgery are to:

1. Create a patient-specific preoperative geometric model from medical imaging data
2. Create geometric models of several surgical alternatives to be evaluated
3. Process additional data to obtain physiologic information to be used for boundary condition specification
4. Perform numerical simulation of the different surgical procedures
5. Visualize clinically relevant analysis results such as wall shear stress to evaluate the different surgical procedures

A software system for simulation-based medical planning must streamline the process of going from preoperative anatomic and physiologic data to analysis results in a robust and timely fashion. The overall effectiveness of the SBMP system depends equally on the performance of the individual component technologies and on the level of integration and

interface between the required components. That is, the system should utilize best-in-class commercial and academic software kernels and enable the use of custom code specifically written and optimized for vascular surgery applications. A practical consideration is that the system must minimize the time required by the vascular surgeon to create alternative surgical plans.

3.4.3 Software architecture for SBMP

Geodesic provided the software framework used in the present work to address the requirements for SBMP as outlined in section 3.4.2. Specifically, the **Geodesic** framework provided several key modules including: a level set module for segmentation of imaging data, a solid modeling module for geometric computation, and a meshing layer which queries meshes for information required to specify boundary conditions and creates input files for hemodynamic solvers. In addition, the integration of Tk into **Geodesic** provided the ability to create a GUI for easily creating alternative postoperative plans in **ASPIRE**^{2}.

The individual modules in **Geodesic** roughly correspond to the major tasks necessary to go from medical imaging data to an operative plan. Two of the modules, however, are used in multiple tasks. The solid modeling module is used in preoperative model construction, surgical planning, and also as a database to track important boundary condition information needed by the meshing layer. The visualization module plays a role in all of the tasks in the process (from viewing the image data to the analysis results) and as such plays a pivotal role in SBMP. Application-specific extensions to the **Geodesic** framework to create the **ASPIRE**^{2} system are detailed below.

3.5 PREOPERATIVE GEOMETRIC MODEL CONSTRUCTION

The first major step in the process of SBMP is to create a preoperative model from diagnostic medical imaging data. Figure 3.5 shows an overview of the major steps in the

solid model construction process in **ASPIRE^2**. Conceptually, a collection of cross-sectional segmentations is created for each vessel of interest in the image dataset. Then for each vessel a solid model is created representing the geometry of the given vessel. The last step involves creating a single solid model by combining the solid models of the individual vessels.

The following four subsections provide details regarding the process shown in Figure 3.5. Section 3.5.1 explains the methods implemented in **ASPIRE^2** for interacting directly with the volumetric image dataset. The intermediate step of creating medial axis paths (or vessel centerlines) required for the 2-D segmentation techniques used in the present work is detailed in section 3.5.2. Image intensity variations and modeling considerations often require that multiple methods of 2-D segmentation be used as described in section 3.5.3. Finally, section 3.5.4 explains how solid modeling operations are used to create geometric models of the flow domain given collections of 2-D cross-sectional segmentations.

3.5.1 Interacting with volume image data

As previously mentioned, imaging data can be represented by a set of scalar values defined on a structured 3-D grid. The most common use for medical imaging data is for diagnostic and visualization purposes. For example, a vascular surgeon may have image data acquired to confirm the diagnosis of a patient suspected of having vascular disease. Traditionally, most surgeons and radiologists look at sets of 2-D slices or static images of image data acquired in a 3-D volume. This typically requires the person to create a mental image of the patient's 3-D anatomy.

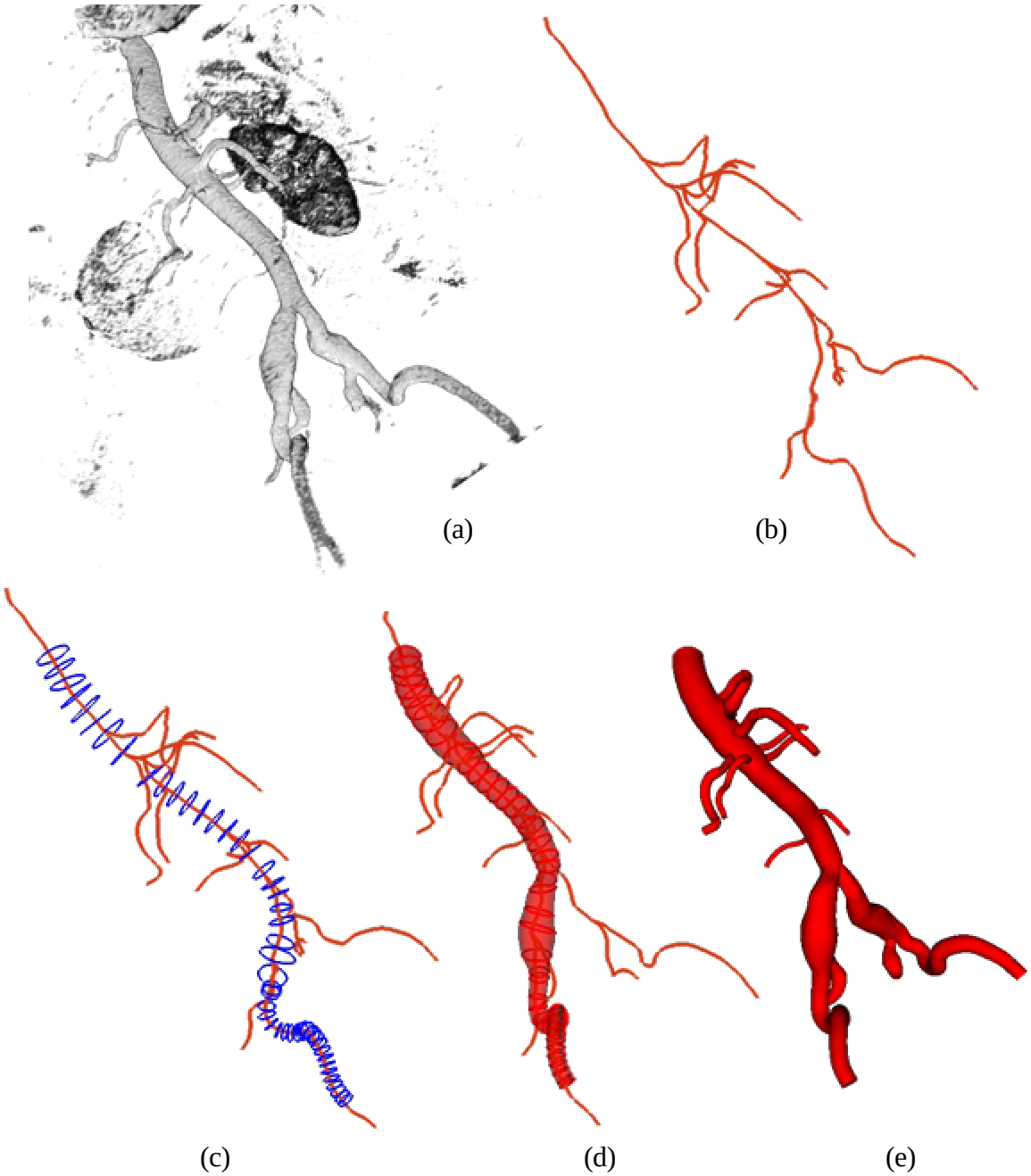


Figure 3.5: Overview of preoperative model construction. Initially, volumetric medical imaging data is acquired (a). Medial axis vessel paths are then extracted (b), followed by 2-D segmentation of the lumen boundary along each path (c). Solid models are then created for each vessel (d). Finally, the solid models representing the vessels are joined (boolean addition) to create a solid model representing the flow domain (e).

Numerous techniques exist to process and visualize medical imaging data. These vary in computational complexity and convey different amounts of information. There is no single correct method to visualize the data, and often several different techniques may be used during geometric model construction for a given clinical case. Figure 3.6 shows the image visualization methods available in **ASPIRE^2**. It should be noted here that while only one type of acquired data exists (i.e. the scalar image intensity), a second derived dataset is created from the image intensity and is referred to in this work as "potential" data (see section 3.5.3 for an explanation of this terminology). The potential is by definition the magnitude of the gradient of the image data at a given grid location (determined numerically by standard central-difference finite-difference stencils). The magnitude of the gradient is typically highest at the transitions between different tissues and the lumen boundary.

The available visualization methods can be classified into two broad categories: continuous and discrete. Continuous methods allow a grayscale or variation of color depending on the scalar quantity visualized. In **ASPIRE^2**, two continuous methods exist to visualize the volumetric image data. First, the user can visualize slices of image data parallel to the major coordinate axes (see Figure 3.7). Note that this method is the closest visualization technique in the system to the way radiologists typically view diagnostic imaging data. Changing the color value displayed for a given scalar intensity is usually referred to as "window leveling." Window leveling creates a non-linear color mapping function to help distinguish features in the image data. Figure 3.7 highlights how the visual content of the image slice changes based on the window level selected by the user. The second continuous visualization technique, volume rendering, was introduced in section 2.2.5. Figure 2.8 shows an example of volume rendering of a CTA dataset while Figure 3.8 shows a volume rendered MRA dataset.

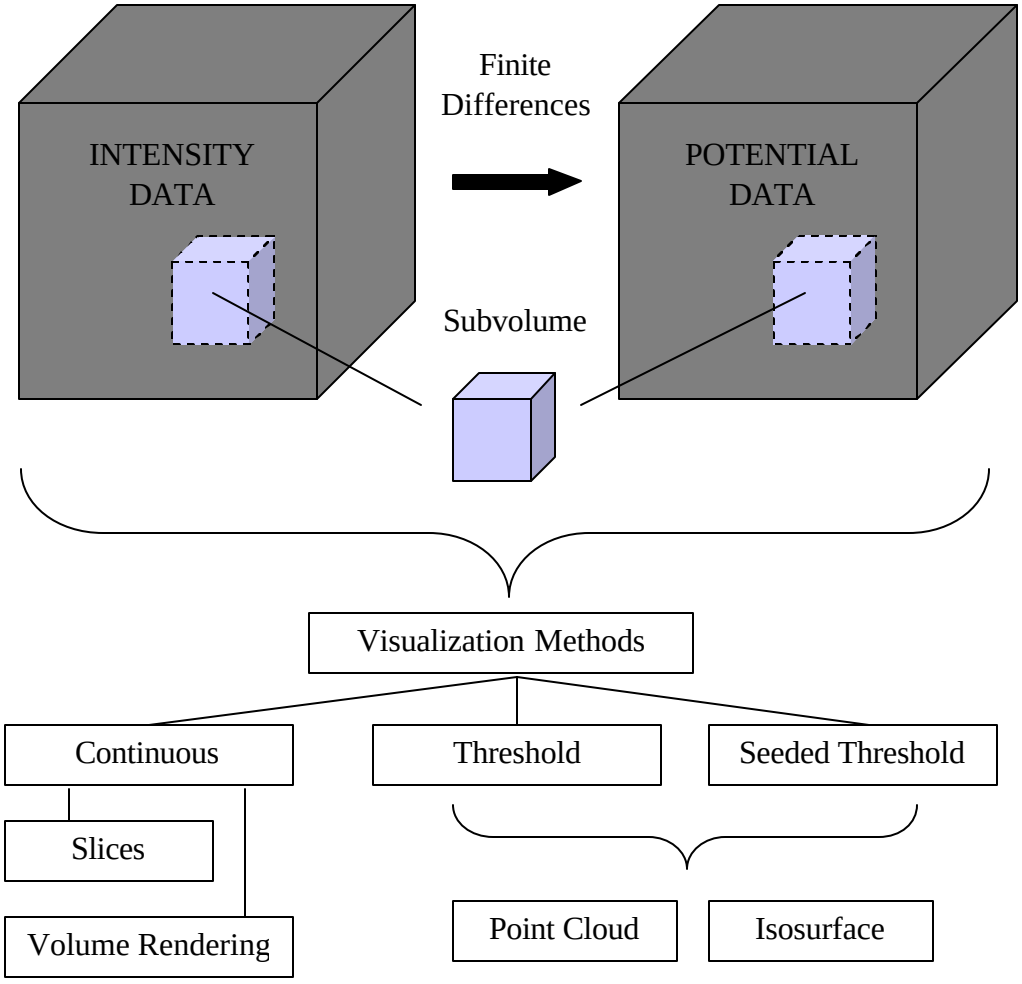
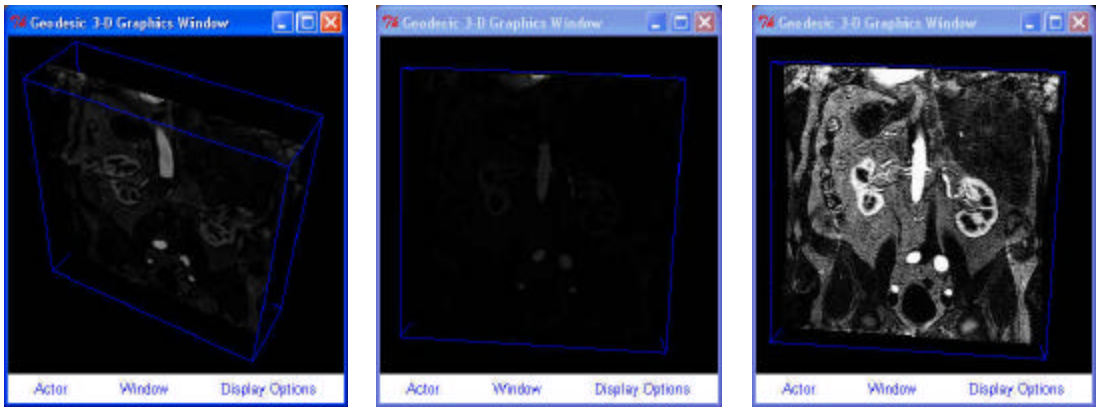
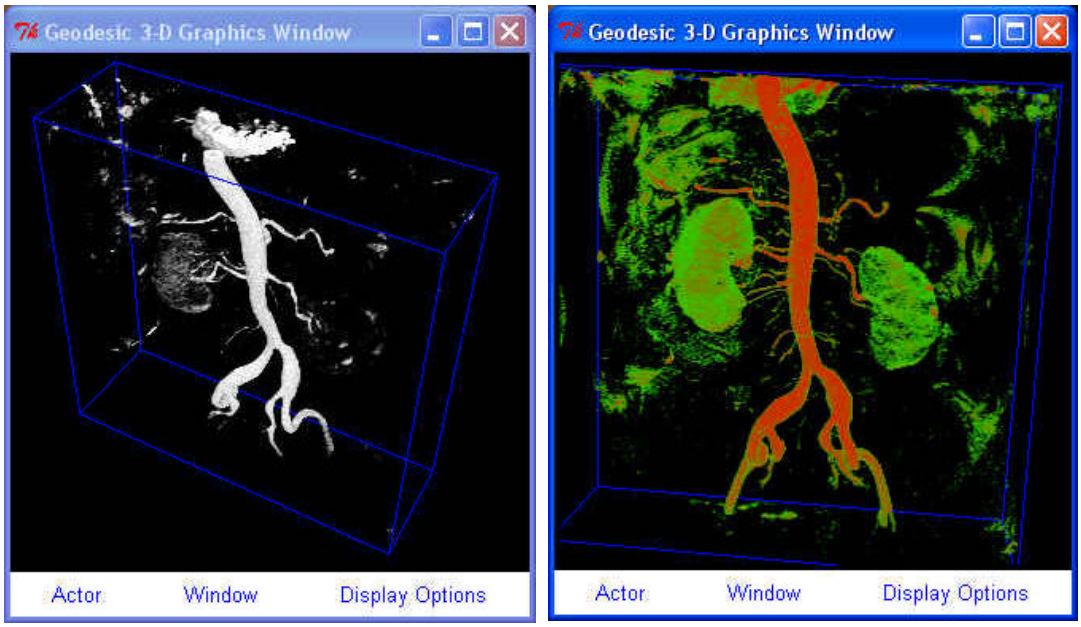


Figure 3.6: Methods for visualizing volumetric image data in **ASPIRE**[2]. The image intensity data is acquired with MRA or CTA (see section 3.3), and the magnitude of the intensity gradient (referred to in this work as potential) is calculated using central-difference finite-difference stencils. The visualization techniques divide into two broad categories: continuous and discrete. The continuous methods provide a grayscale or variation of the color based on the image data, while the discrete methods utilize image segmentation (i.e. thresholding) to create an image. Figure 3.7 through Figure 3.13 show examples of the methods listed in this figure.



(a) (b) (c)

Figure 3.7: Visualizing primary image slice planes. Image slices parallel to the coordinate axes can be displayed as shown in (a). The mapping between scalar intensity and color can be altered to remove or highlight detail from the image data (referred to as "window leveling") as seen in (b) and (c) respectively.



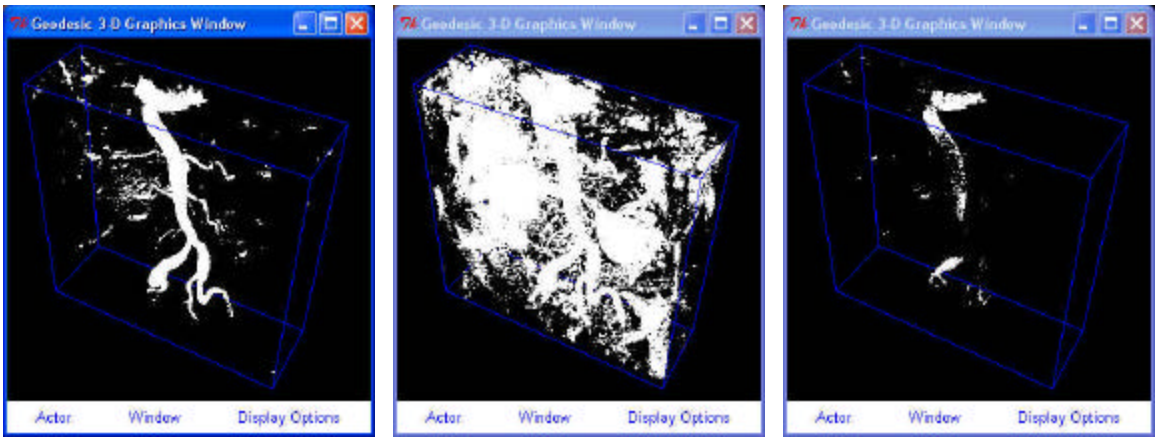
(a) (b)

Figure 3.8: Volume rendering of a MRA dataset. The displayed image is controlled by the color, opacity, and gradient transfer function specified. Figure (a) shows a grayscale color map, while (b) uses an appropriate color and opacity transfer function to have the kidneys highlighted in a different color than the aorta.

The other volumetric visualization techniques, under the discrete category, rely on some form of image segmentation. By definition, image segmentation refers to the act of classifying a region as "in" or "out" of an object of interest based on its scalar value. In this work thresholding will be used to refer to the act of image segmentation based solely on image intensity value. Normally the user defines a scalar value (i.e. threshold) above which the pixel is considered inside the object (e.g. vessel) of interest. Figure 3.9 shows the visualization technique referred to as "point cloud." A point cloud is created by displaying a dot in the center of each voxel that has a value in a user-selected range (above a minimum scalar value and below a maximum scalar value). A second technique is to create an isosurface (surface of constant scalar value) in the image dataset, as shown in Figure 3.10.

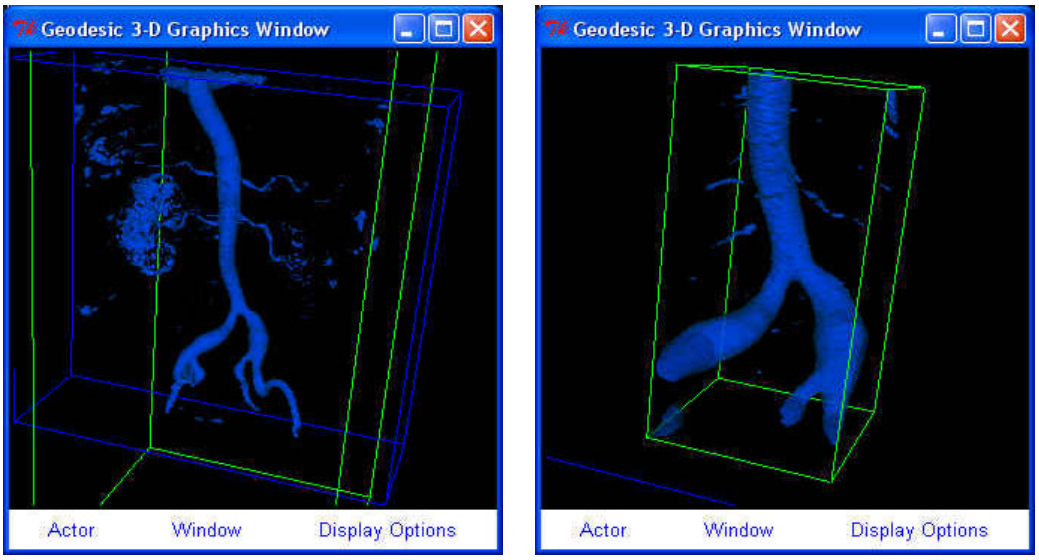
Dealing with the entire volume dataset can be a computational challenge (particularly for CTA). For this reason, methods were implemented to visualize a subvolume as seen in Figure 3.11. An additional visualization method, known as a "seeded threshold" is shown in Figure 3.12. For a seeded threshold the user must specify a voxel contained in the object of interest as well as the threshold values. The connectivity of the dataset is used so that any voxel satisfying the threshold criteria but not simply-connected to the seed voxel is discarded.

It is worth mentioning that the implementation of **ASPIRE^2** enables the combination of most of the visualization techniques described above. Figure 3.13 shows an example of using more than one visualization method. In addition, as described below, geometric models, meshes, and analysis results can be visualized in the same window as the image data. This flexibility to mix and match various visualization techniques and data makes **ASPIRE^2** a powerful system for simulation-based medical planning.



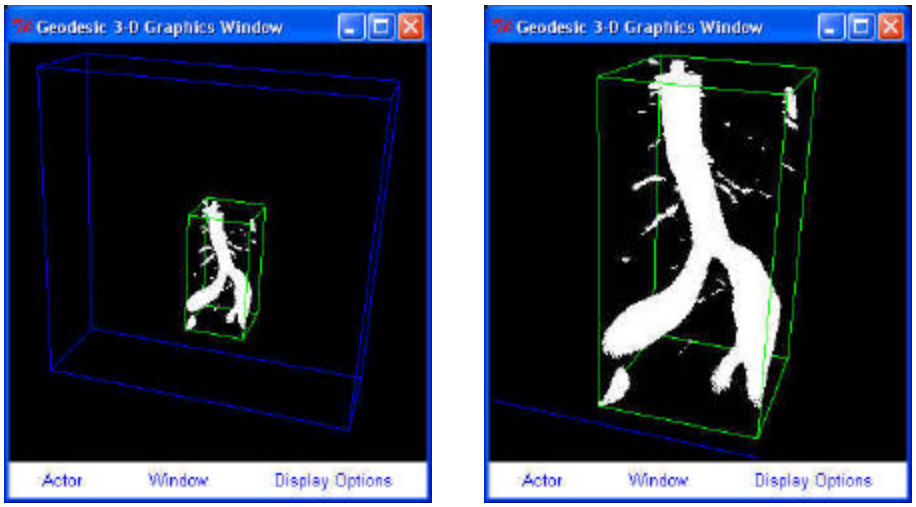
(a) (b) (c)

Figure 3.9: Using point clouds to visualize a MRA dataset. A point cloud is created by displaying a point at the center of all image pixels within a user specified scalar range. Figures (a-c) show different visualizations of the same dataset created by selecting a different minimum threshold value.



(a) (b)

Figure 3.10: Visualizing an isosurface of a MRA dataset. An isosurface is created at the user specified intensity value. Figure (a) shows an example of visualizing a volumetric dataset while (b) shows an isosurface created for a subvolume of the same dataset.



(a) (b)

Figure 3.11: Visualizing a subvolume of a MRA dataset with a point cloud. Figures (a) and (b) show the visualization of a subvolume of a dataset using a point cloud.

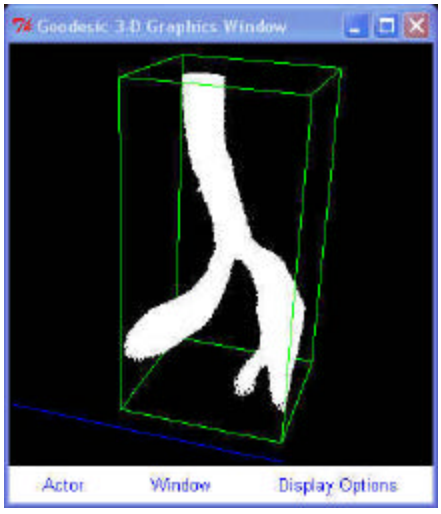


Figure 3.12: Visualizing a seeded threshold with a point cloud. To create a seeded threshold the user must specify an image pixel and only pixels within the desired threshold range that are simply-connected to the specified pixel are displayed. This technique is useful in removing noise and vessels not of primary interest (compare to Figure 3.11b).

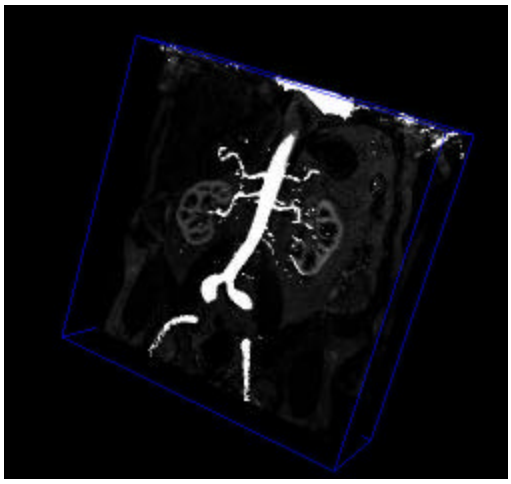


Figure 3.13: Combining image visualization methods in **ASPIRE^2**. The system developed in this work enables utilization of multiple image visualization techniques simultaneously. The figure shows a point cloud displayed with an image slice plane. Using multiple image visualization methods at the same time can greatly enhance the information conveyed in the displayed image.

3.5.2 Creating vessel paths

An axial path along each vessel of interest is required to be able to segment perpendicular cross sections as described in the next section. The **ASPIRE^2** system incorporates two useful methods (which can be combined) to create vessel paths. First, the user can select a set of points in space and an analytic curve (i.e. cubic spline) will be fit to these points to create a path. Allowing the user to manually select path points is particularly well suited for the operative planning to be discussed in section 3.6. Whenever possible, however, automatic or semi-automatic vessel path extraction is desirable. For example, having the user select the points along the path of a tortuous vessel such as the aorta may be cumbersome. The second method integrated into **ASPIRE^2** for determining vessel paths was an automated algorithm developed by Paik [43].

The Paik algorithm numerically approximates the medial axis for a given vessel. By definition, a point inside of an object is defined to have a maximal disk associated with it if the largest circle centered at that point touches the shape boundary at two or more

points. The medial axis is defined to be the set of all points with maximal disks. Having a path along the medial axis will prove advantageous for segmentation as described in the next section.

Figure 3.14 shows the process of creating an approximate medial axis path as implemented in **ASPIRE**^{2}. The user provides three inputs to the automatic path planning algorithm: a starting point, an ending point, and threshold values to create a binary segmentation of the image data. In this work, the simplest possible binary segmentation is created such that an image voxel is classified as inside or outside based on a minimum and maximum threshold value. The algorithm starts by creating an initial arbitrary path between the starting and ending point. This is followed by an iterative procedure to recover the approximate medial axis path. The central operation in each step is a parallel "thinning" where pixels are removed from the exterior surface of the segmentation analogous to peeling layers of an onion. After each layer is removed, the path points are reevaluated to minimize the Euclidean distance between the starting and ending point. The thinning is repeated until the entire binary segmentation has been removed (i.e. all the layers have been peeled away from the onion). It is worth noting that the parallel thinning operation can split the original segmentation into any number of subvolumes as the thinning is performed. When topologic change occurs (i.e. the binary segmentation splits into subvolumes), points from the path of the previous iteration are used as necessary to ensure that a path connecting the starting and ending point is always created during the current iteration. The pseudo code (adapted from [43]) for this iterative process is shown in Figure 3.15.

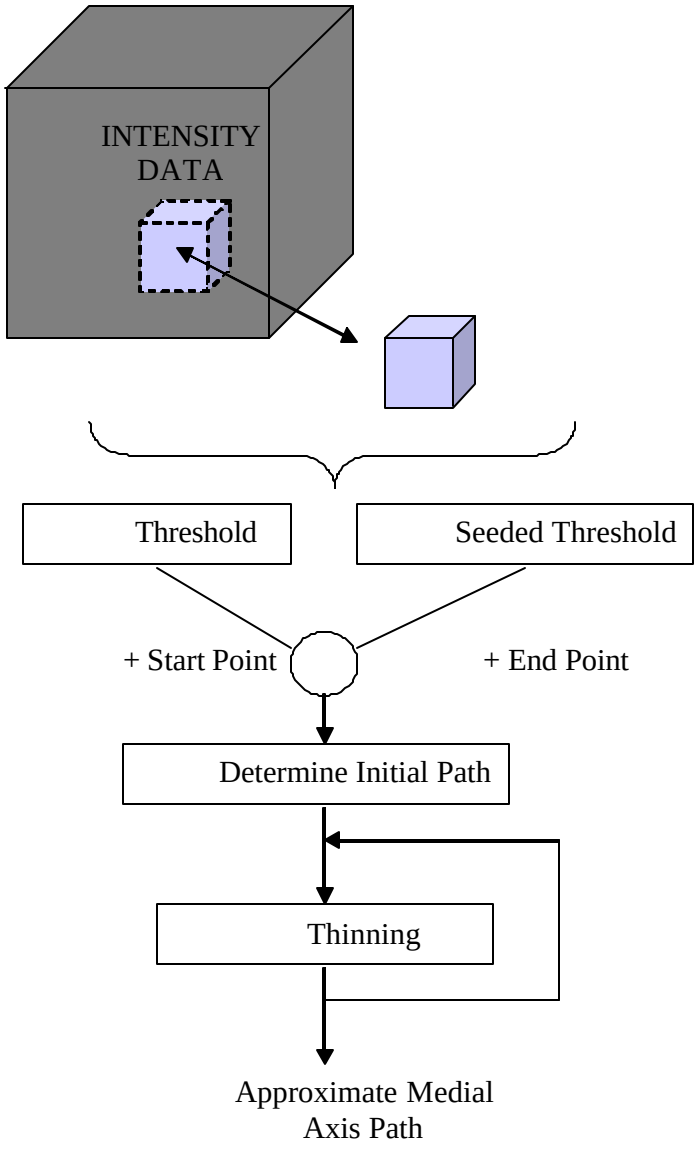


Figure 3.14: Creating a medial axis path for a given vessel. The user provides a starting point, ending point, and binary segmentation created from the image data. An arbitrary initial path is created connecting the specified starting and ending point and then is iteratively refined (through a process referred to as thinning) to recover the medial axis path of the given vessel.

```
Path := Path_initial
Surface := identify_surface
While exists(Surface) do
   Delete(Surface)
   Surface := identify_surface
   Make_Euclidean_distance_map(Surface, V_end)
   Path_next := empty_path
   V := V_start
   While V ≠ V_end do
      Add_to_path(Path_next, V)
      If exists(shallowest_next_neighbor(Surface,V))
         V_next := shallowest_next_neighbor(Surface,V)
      Else
          V_next := shallowest_next_neighbor(Path,V)
      End if
      V := V_next
   End while
   Add_to_path(Path_next, V_end)
   Path := Path_next
End while
```

Figure 3.15: Pseudo code of the iterative thinning process to recover a medial axis path (adapted from [43]). Starting with a binary segmentation and an arbitrary initial path between V_{start} and V_{end}, the pixels on the exterior surface of the segmentation are removed in parallel (analogous to peeling the layers of an onion) and the approximate medial axis path is recovered.

Several sources of error exist for the medial axis algorithm. First, "bubbles" may arise in the binary segmentation from inhomogeneities in the contrast agent and/or noise in the signal. Second, the thinning process occurs nonisotropically, moving faster in the direction of the greatest voxel dimension. The thinning process inherently moves faster in diagonal directions than in the directions of the acquisition axes. Finally, significant vascular disease can make it difficult to extract certain vessel paths. These technical challenges tend to make the path planning process semi-automatic for surgical patients. That is, in general the path will consist of a combination of user-selected and automatically calculated points.

Once a preliminary path has been created, it is usually smoothed as shown in Figure 3.16. This is required to minimize the kinks and abrupt changes of orientation of a perpendicular slice plane moving along the vessel path. Four methods of smoothing are provided in the current implementation and can be used in any sequence and repeated multiple times as required. First, the path can be sampled at some user-selected interval which typically smoothes the resulting path. Second, the user can interactively remove points to reduce undesirable changes in the path. A third path smoothing technique takes advantage of the fact that several of the sources of error contribute to "high frequency noise" in the preliminary path. This motivates a spatial smoothing technique based on Fourier transforms. The preliminary path is parameterized as a function of path length (i.e. X(d), Y(d), Z(d) where $0 \leq d \leq$ path length). A fast Fourier transform (FFT) is then performed on each coordinate independently (i.e. X(ω), Y(ω), Z(ω)) followed by an inverse FFT keeping only a finite number of modes (i.e. $p_i = \langle X(d_i), Y(d_i), Z(d_i) \rangle$ where $0 \leq d_i \leq$ path length).

Spatial smoothing does not take into account the underlying image data used to create the preliminary path. To address this limitation, a fourth smoothing technique referred to as "threshold realignment" can also be applied. This technique takes perpendicular cross-sections along the path at a user-defined interval and performs a 2-D image segmentation

given a threshold value (Figure 3.17). If a closed contour exists, the existing path point for this slice is discarded and the center of the contour is used in its place. With the proper selection of spacing and threshold parameters this technique can significantly improve the path for some datasets.

Figure 3.18 shows an example of automatically creating a path for the abdominal aorta. The steps outlined in this section are repeated for every major vessel of interest. While factors such as anatomic variability and modeling objectives determine the exact number of paths created for a given image dataset, typically on the order of ten paths are created.

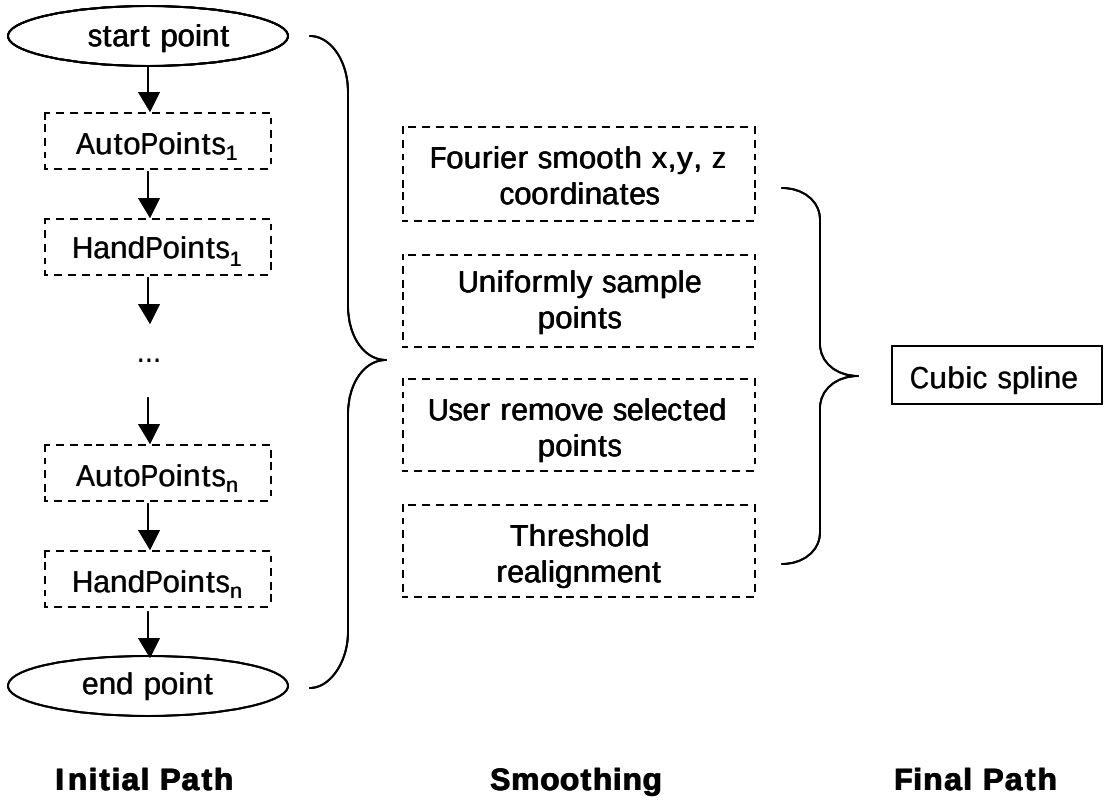


Figure 3.16: Creating a path in **ASPIRE**[2]. An initial path consists of some combination of points selected by hand and calculated automatically by the algorithm detailed in Figure 3.14. Four different methods can be applied in any combination to smooth the initial path points. The final path consists of a cubic spline through the smoothed path points.

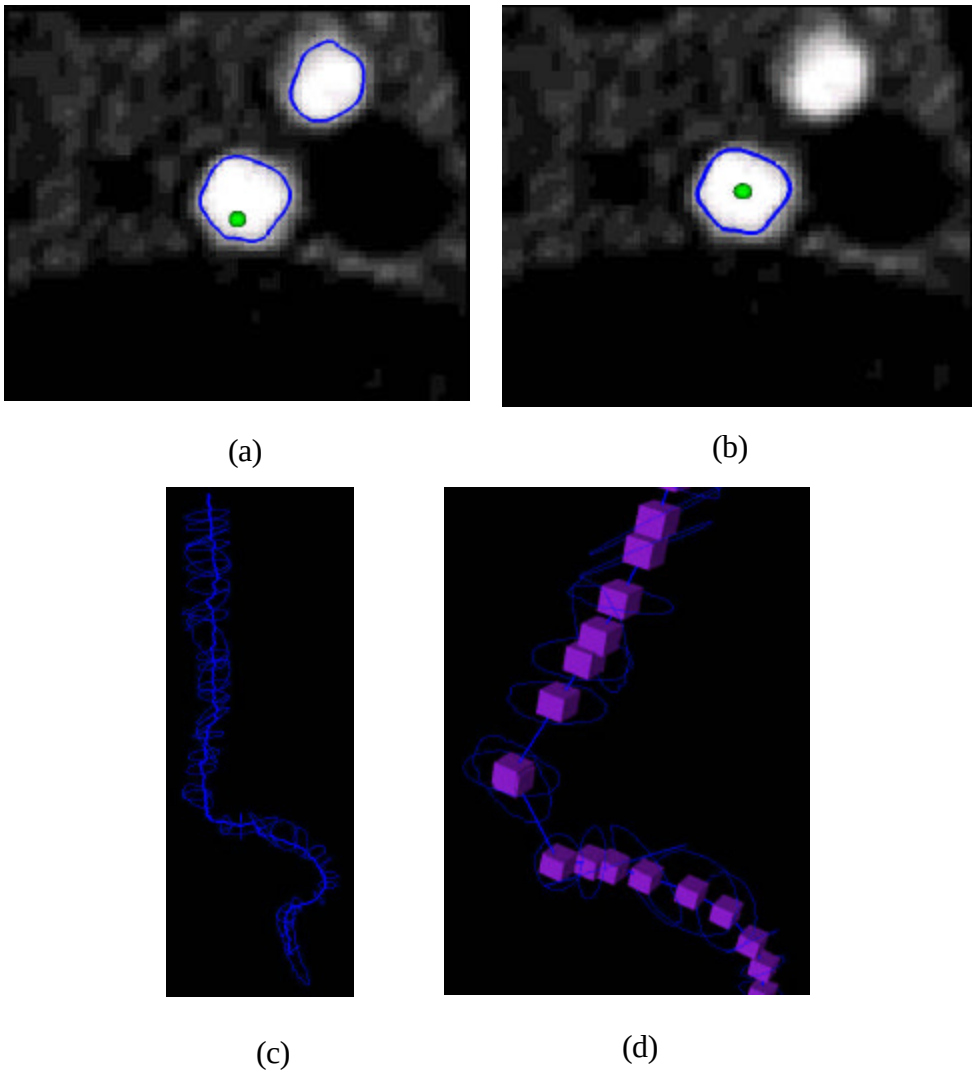


Figure 3.17: Smoothing a path using threshold realignment. This method of smoothing utilizes some of the code described in the next section for 2-D segmentation to smooth and center the path. Starting from the first point in the initial path, 2-D threshold segmentations of the image data are calculated at a user-specified interval along the initial path. If a closed contour exists containing the i^{th} path point shown as the green dot in figure (a), the initial i^{th} path point is replaced with the center of the current segmentation (b). Figure (c) shows all of the 2-D contours used to smooth the initial path, and (d) shows the resulting smoothed path. For some datasets with appropriately selected threshold and sampling parameters this technique significantly improves and smoothes the path by centering it in the desired vessel.

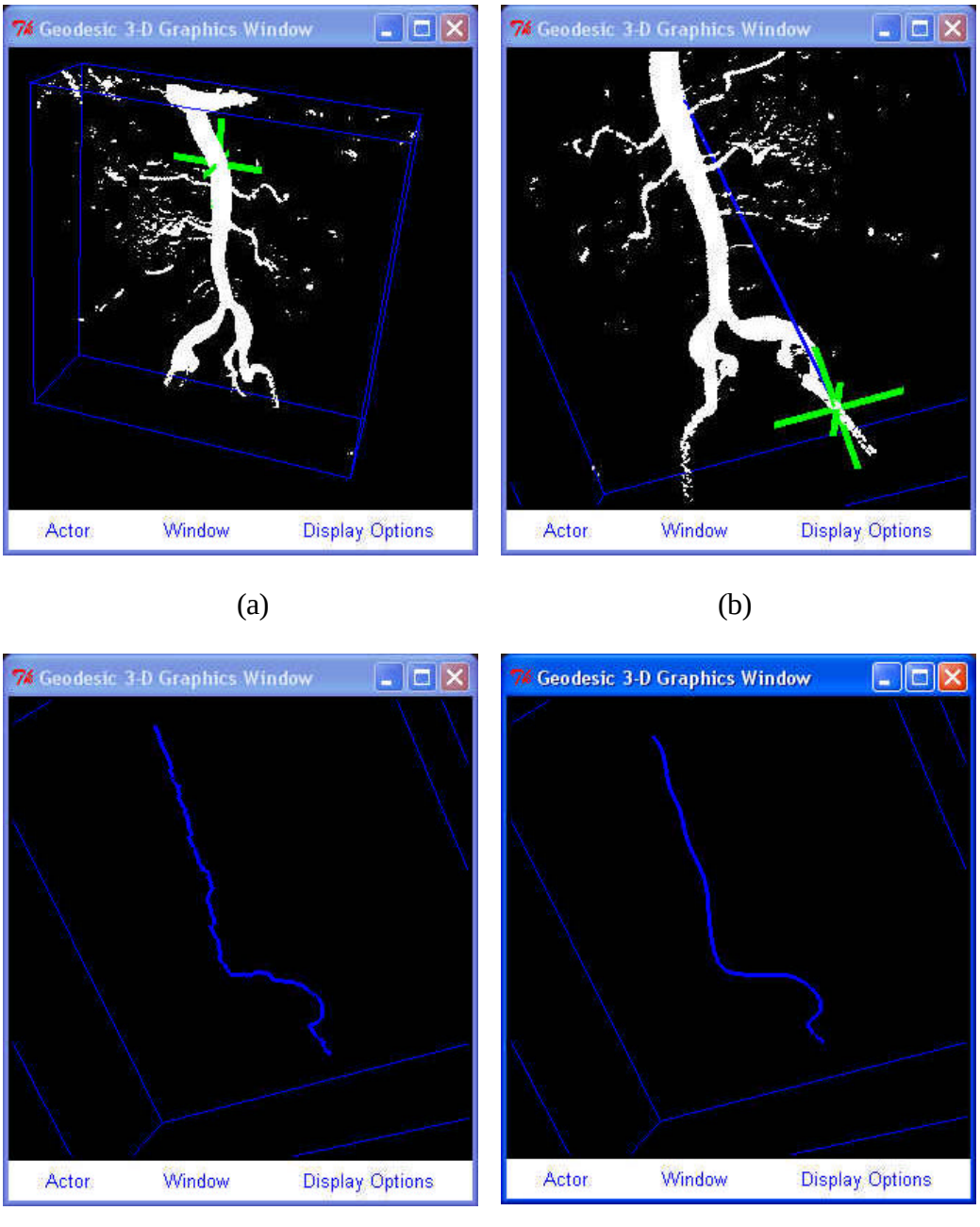
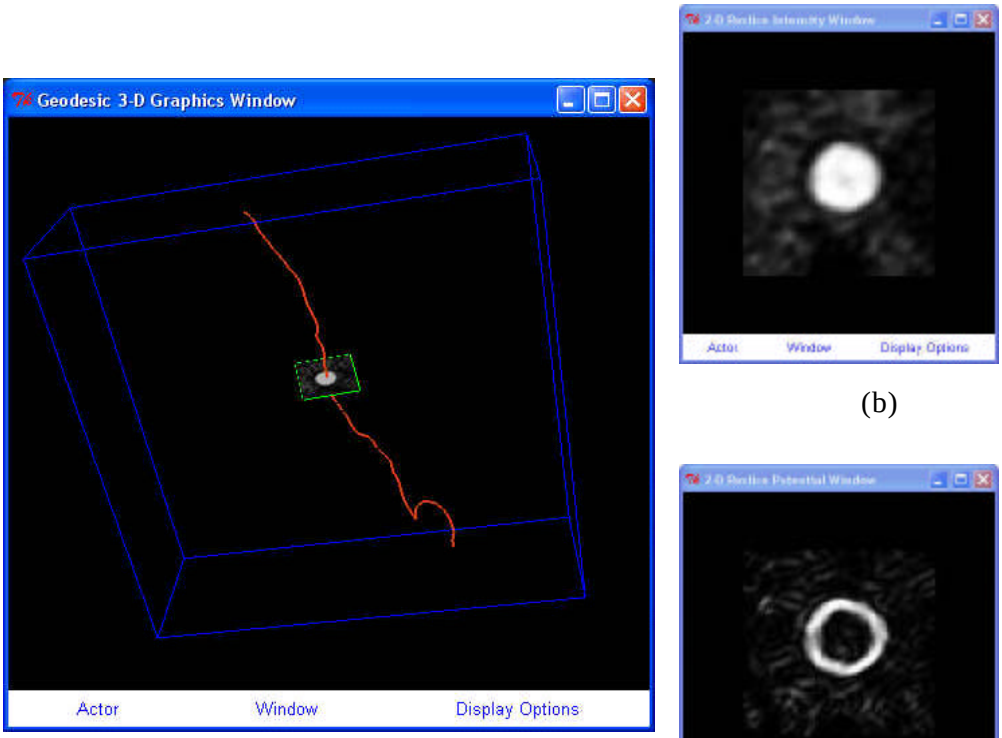


Figure 3.18: Example of creating a vessel path for the abdominal aorta. A starting point (a) and ending point (b) of the path was specified and a path was automatically extracted using the Paik algorithm. Fourier smoothing was applied to the approximate medial axis path (c) to achieve the desired final path shown in (d).

3.5.3 Two-dimensional image segmentation

Given a vessel path, the next step in the process is to perform 2-D image segmentation in planes perpendicular to the path. Figure 3.19 shows a 2-D reslice of the image data in a trimmed plane perpendicular to the path. The plane can be interactively moved along the path, and the user can select various locations to perform 2-D segmentation. Figure 3.20 shows the four different methods available for segmentation. Two of these techniques are straightforward. First, a GUI allows a user to hand draw a desired contour, typically done by tracing along a lumen boundary. A second technique is to create analytic curves (currently circles or ellipses) of desired dimensions. This is often done when the data is too noisy or lacks sufficient resolution to properly segment a cross-section.

The two techniques most commonly used for segmenting larger vessels are thresholding and the level set method. In this case, thresholding refers to creating an isocontour at a desired scalar intensity value and selecting the closed contour (if it exists) containing the path point. This is shown in Figure 3.21 and Figure 3.22. Notice that the selection of two different threshold values leads to visibly different segmentations in Figure 3.21. In the present work, it is hypothesized that the best approximation of the true lumen boundary occurs at the maximum value of the gradient (or roughly in the center of the annulus shown in Figure 3.21b). Thus the value and the resulting segmentation shown in Figure 3.21c are considered the proper choice for the image shown in the figure. Due to variations in contrast agents (and magnetic field in the case of MRA), a single vessel may require the use of different threshold values along its path. This usually increases the user interaction required. In addition, Figure 3.23 shows the case of a slice located near a vessel bifurcation in which it is impossible to get a closed boundary with any selected threshold value.



(a)

(b)

(c)

Figure 3.19: Slicing volumetric image data along a path. The user can slice the data in a trimmed plane perpendicular to a vessel path for 2-D segmentation. Figure (a) shows a path along the aorta with a perpendicular slice (i.e. trimmed plane) of image data created by tri-linear interpolation from the volumetric image data. Figure (b) shows the 2-D view of the image intensity slice, while (c) shows the corresponding potential data.

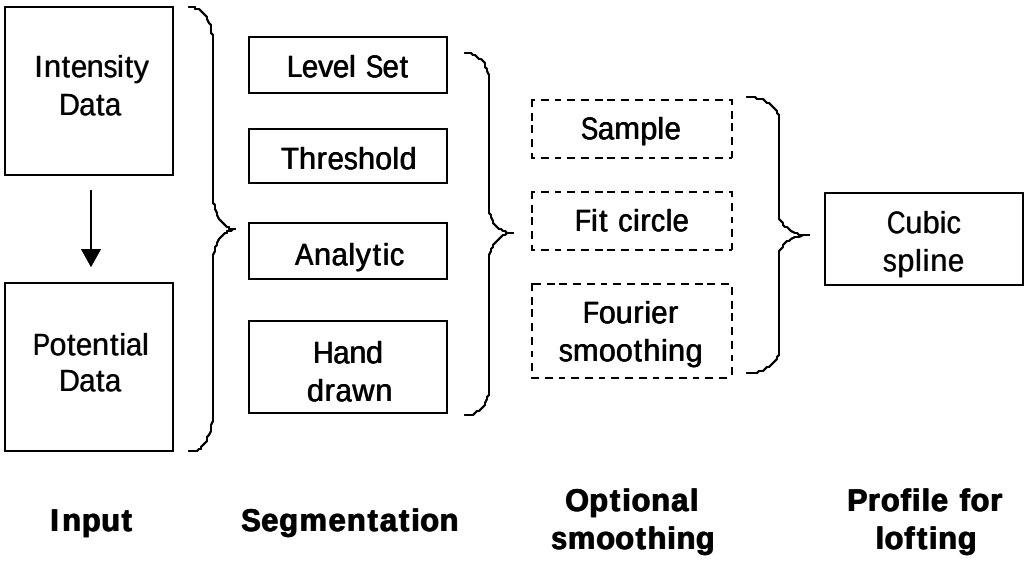


Figure 3.20: Four two-dimensional segmentation techniques in **ASPIRE**^{2}. Depending on the segmentation method, the input consists of image intensity data, potential data, or a combination of the two. It is not uncommon to use multiple segmentation techniques for extended vessel paths. Three optional smoothing techniques can be applied to the segmented lumen boundary, and the final profile is always fit with a cubic spline for lofting purposes.

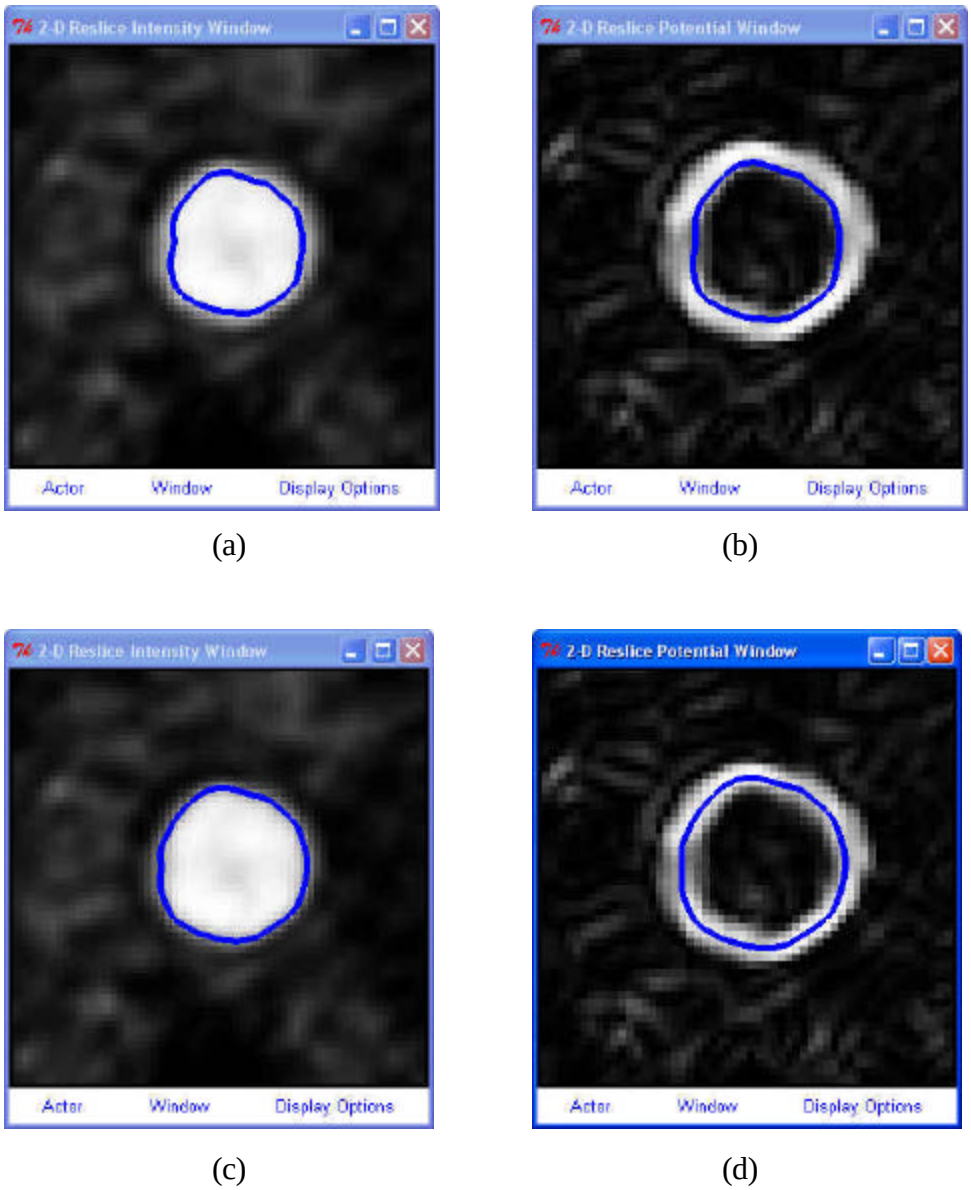


Figure 3.21: Two-dimensional threshold image segmentation. Figure (a) shows a two-dimensional segmentation (shown in blue) along with the image intensity data for a particular threshold value. The same segmentation is shown in (b) along with the potential data. Figure (c-d) shows a two-dimensional segmentation created with a different threshold value for the same image data. Notice that the segmentations are visibly different depending on the threshold value selected. It is hypothesized that actual lumen boundary is at the center of the potential annulus, so the threshold value for (c) is the desired choice.

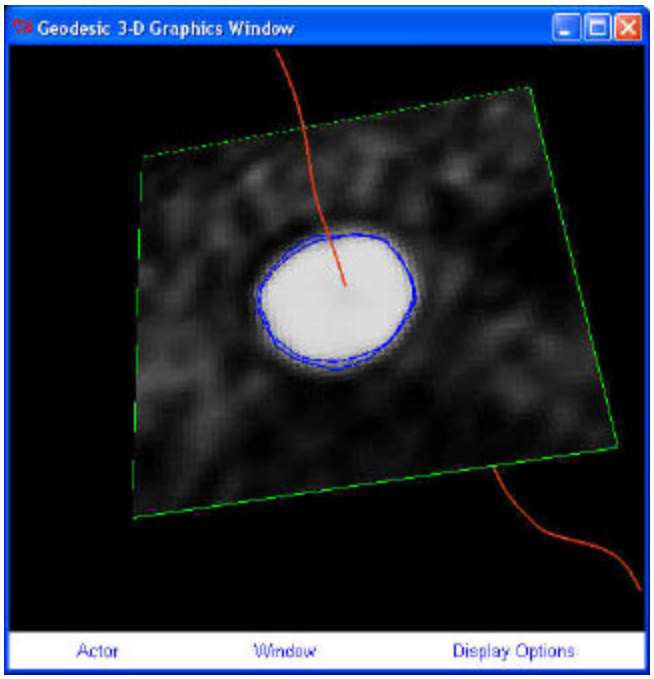


Figure 3.22: Two-dimensional image segmentation oriented in 3-space. After a 2-D segmentation is created as seen in Figure 3.21, an affine transformation is applied to properly orient the segmentation in 3-D.

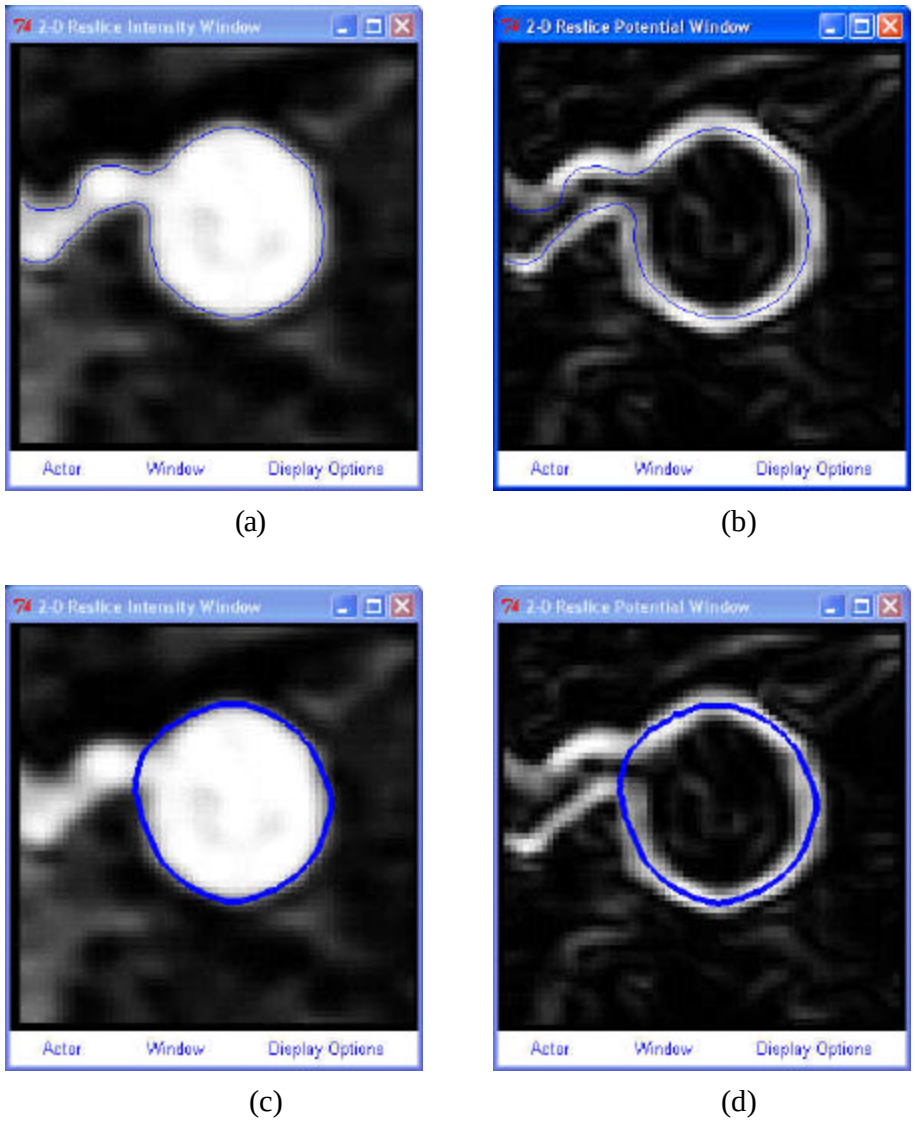


Figure 3.23: Two-dimensional image segmentation at a vessel branch. Figures (a) and (b) shows an attempt to create a threshold segmentation for the given image slice at a vessel branch point. No threshold value can be selected that will create a closed profile. Figures (c) and (d) show a level set segmentation can be created for the same image data. That leads to a closed profile necessary for subsequent lofting operations.

The second technique for lumen boundary extraction is based on the level set method introduced in section 2.2.1. Using the level set method for image segmentation was originally proposed by Malladi [44]. The major practical challenge using the level set method for different boundary evolution problems is finding appropriate velocity functions. In the work presented here the following two velocity functions proposed by Wang [11] were used:

$$v_i = + (k_{threshold} - k_i)\ e^{-e\,P} \tag{3.1}$$

$$v_i = \begin{cases} -e_k (k_i - k_{upper}) & k_i > k_{upper} \\ e_k (k_{lower} - k_i) & k_i < k_{lower} \\ e_p (\nabla P \cdot \underline{n}_i) & k_{lower} \le k_i \le k_{upper} \end{cases} \tag{3.2}$$

where:

v_i = normal velocity at node i

$\varepsilon, \varepsilon_k, \varepsilon_p$ = user defined constants

κ_i = surface curvature at i

κ_{upper}, κ_{lower}, $\kappa_{threshold}$ = user-defined curvature constraints

$P = |\nabla I|$ = magnitude of the intensity gradient (i.e. potential)

$\underline{n}_i$ = outward surface normal at node i

The details of the implementation and selection of appropriate parameters for the level set method using equation 3.1 and equation 3.2 are described elsewhere [11]. Here, the major concepts are briefly reviewed. The segmentation process consists of two stages. The first stage uses equation 3.1 to grow an initial circle placed inside of the lumen boundary until it is "near" the interior vessel wall. The parameter ε is selected so that the velocity is very small ($v \rightarrow 0$) when the change in gradient associated with the lumen boundary is encountered ($P >> 0$). This first stage produces segmentations very similar

to those found by thresholding a particular value of P for image slices such as Figure 3.21b. However, the curvature feedback (i.e. $k_{threshold} - k_i$) in equation 3.1 provides both smoothness and prevents the segmentation from "bleeding" out of openings in the potential band as seen in Figure 3.23d.

Once the segmentation has satisfied the stopping criteria for the first stage, it is used as the seed (i.e. the initialization) for a second level set evolution. This second stage uses the velocity equation 3.2 and the parameters are selected such that points on the segmentation boundary are attracted to the highest values of the gradient in their local neighborhood. The parameter ε_p effectively defines a width around the maximum values of the magnitude of the gradient that the final segmentation ideally will occupy. This band around the maximum potential values is referred to as the "potential well" and the segmentation can be thought of as being attracted to and falling into the well. A representative segmentation using the two-stage process is shown in Figure 3.23c. With an appropriate choice of the level set parameters (in particular the allowable curvature) the segmentation doesn't "bleed" into the smaller vessel seen in Figure 3.23.

Two significant enhancements over the previous implementation for 2-D image segmentation [11] were used in work presented here. First, the use of solid modeling operations in evaluating a geometric-based segmentation stopping criterion was eliminated. As discussed below, this has important implications for the generalization of the velocity functions used here for 3-D image segmentation. Second, a segmentation server was implemented to utilize distributed computing when performing multiple segmentations. Utilizing distributed computing can significantly reduce the wall-clock time needed to construct a patient model.

The selection of the constants in equation 3.1 and 3.2 is done such that $v \rightarrow 0$ as the desired lumen boundary is recovered. An additional parameter defined by the user is a cutoff velocity, V_{stop}. When every nodal velocity is less than V_{stop}, the boundary is only changing slightly and convergence on the lumen boundary is assumed. However, due to

several technical considerations, it cannot be guaranteed that this stopping criterion will be met. To address this, Wang proposed the use of the following geometric quality metric to describe the similarity between two shapes A and B:

$$f_x = \frac{A(X \cap Y)}{A(X)} \qquad f_y = \frac{A(X \cap Y)}{A(Y)} \tag{3.3}$$

$$Q(X,Y) = \sqrt{f_x \cdot f_y} = \sqrt{\frac{A(X \cap Y)^2}{A(X) \cdot A(Y)}} \tag{3.4}$$

where:

f_x = indication of spatial matching with respect to X

f_y = indication of spatial matching with respect to Y

A(.) = spatial extent operator (area in 2-D, volume in 3-D)

Q = linearized region matching indicator

Note that Q ranges from [0,1] with Q=0 indicating completely disjoint regions and Q=1 corresponding to entirely coincident, or exactly matching, regions. Wang used solid modeling operations to calculate the intersection A(X∩Y). The commercial solid modeling kernels [13,14] used in this work are relatively robust at calculating the 2-D intersections required to evaluate A(.). However, the kernels lack sufficient robustness to calculate the necessary intersections required when performing 3-D level set segmentations. Here a new mechanism is proposed to eliminate the need for solid modeling operations by performing the intersection directly on the level set representation. As discussed elsewhere [9], the following set operations can be performed directly on the level set representation:

$$A(X \cup Y) = \min(\phi_X, \phi_Y) \qquad \text{(union)} \tag{3.5}$$

$$A(X \cap Y) = \max(\phi_X, \phi_Y) \qquad \text{(intersection)} \tag{3.6}$$

$$A(X - Y) = \max(\phi_X, -\phi_Y) \qquad \text{(subtraction)} \tag{3.7}$$

Substituting equation 3.6 into equation 3.4 eliminates the use of a solid modeling kernel to evaluate the geometric quality metric. It should be noted that the method here is slightly more memory intensive because a duplicate copy of the previous level set solution must be maintained for the intersection calculation. However, the increase in robustness and overall reduction in computational cost make this an attractive method and it is used exclusively in the work presented here.

A performance enhancement used in this work was distributed computing. When performing 2-D segmentation, segmentation at one location along a path is completely independent of a segmentation being performed elsewhere along the same path. For this reason, performing the segmentations can be thought of as "embarrassingly parallel" since no communication between the evolutions at various path positions is required. Figure 3.24 details a prototype distributed computing facility inside of **ASPIRE^2**. Briefly, the main **ASPIRE^2** application creates a set of files consisting of 2-D reslices of the volumetric image data (and potential data) at the desired segmentation location. In addition, a parameters file to control the segmentation is also created. The location of these filenames and a request for handling is set to a separate process (called the "batch server") running on the local host. This "batch server" then distributes the requests to remote hosts (or other local processors) using a simple load-balancing scheme. The prototype implementation relies on secure sockets (via secure shell) and RSA authentication to transparently move files between different machines and execute processes remotely. The remote machines can be SGI, Sun, and HP workstations as well as personal computers running Microsoft Windows 2000/XP (running under Windows requires Cygwin [45] to provide the ssh server). The major drawback to the current server implementation is the unnecessary overhead associated with repeatedly opening and closing secure sockets on the same machines. One way to overcome this limitation is

to create a special server application that runs on the remote machines and listens on a dedicated port waiting for segmentation requests.

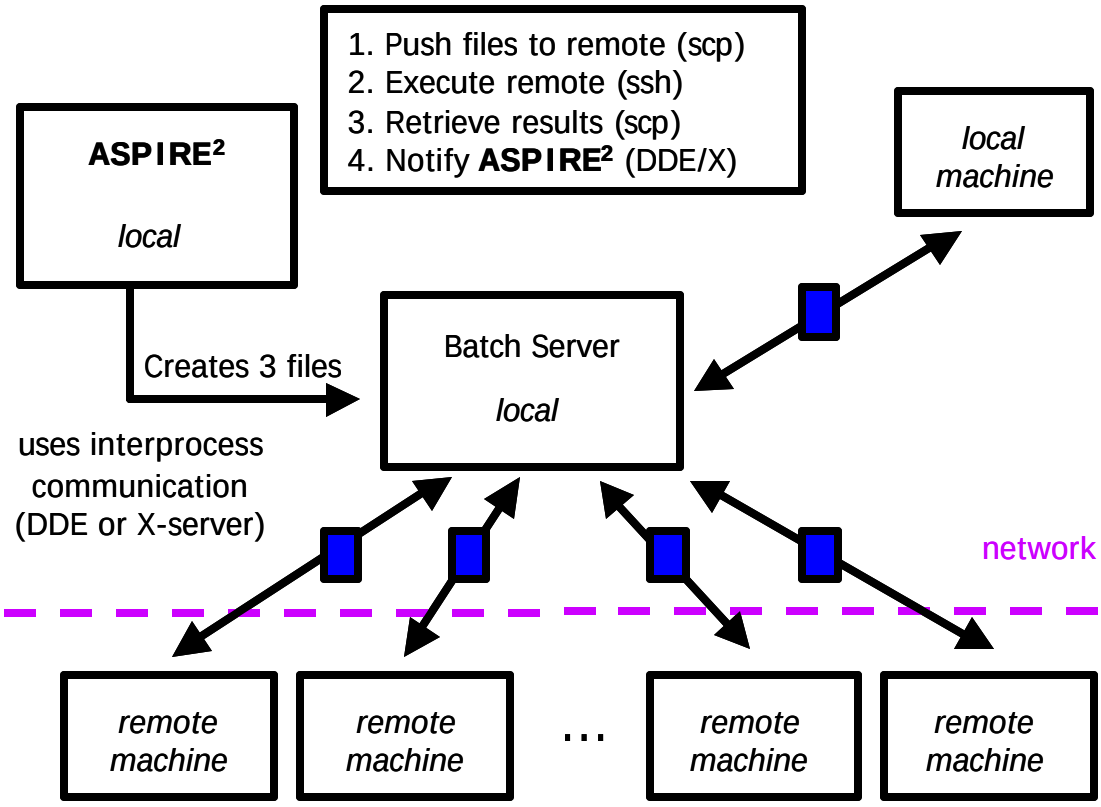


Figure 3.24: Segmentation batch server. A distributed batch server enables the use of additional local processors or remote machines over the network. The **ASPIRE²** application generates three files (i.e. image slice, potential slice, parameters file) on the local machine and sends a request to the batch server (also running locally) to handle the segmentation for the given slice. The batch server uses a simplistic load balancing scheme to distribute the segmentations over the available machines. When the segmentation process is complete on the remote machine, the batch server is notified and the batch server in turn passes the segmentation back to the **ASPIRE²** system. The batch server utilizes dynamic-data-exchange (DDE) on Windows platforms and the X-server on UNIX for interprocess communication. The data is passed in the form of files between the local and remote machines using secure connections (via ssh/scp). The batch server explicitly runs **Geodesic** in command mode on the remote machine to perform the desired segmentations thus not requiring servers to be running on the remote segmentation machines.

Table 3.2 shows the wall-clock time running a set of 73 segmentations on an assorted number of remote machines. Ideal scaling is seldom achieved due to network overhead, loads on the remote machines, and so forth. In addition, as the number of time steps increases per segmentation, the scaling improves because the computational time in performing the segmentation tends to surpass the time required to transfer files and create secure connections. Finally it is worth noting that excellent scaling is achieved on a dual processor PC since the network related delays are eliminated.

Table 3.2: Wall-clock Time to Perform 73 Segmentations in Parallel

Number of Processors	Wall-clock Time (seconds)	Time vs. Single Processor Time
5 remote machines, 2 local processors	180	40%
2 local processors	262	57%
1 local processor	453	100%

3.5.4 Creating solid models from 2-D segmentations

Once a set of cross-sectional segmentations have been created for a vessel of interest, the next step in the process is to create a solid model for the given vessel. It should be noted that the spacing and locations of the cross-sections could have a significant impact on the solid created for a given vessel (see [11]).

For each vessel path there exists an ordered set of curves defining the geometry for the given vessel. In **ASPIRE**^{2}, these ordered sets are referred to as "groups." All of the 2-D segmentation curves are associated with one and only one vessel. A two-stage process is used to create a preoperative solid model. First, a "lofted" solid is created for each branch (see Figure 3.25). This results in a collection of solid branches. Second, a boolean addition (union) is performed to join the individual branches. The result is then a single bounded solid region representing the blood flow domain as seen in Figure 3.26.

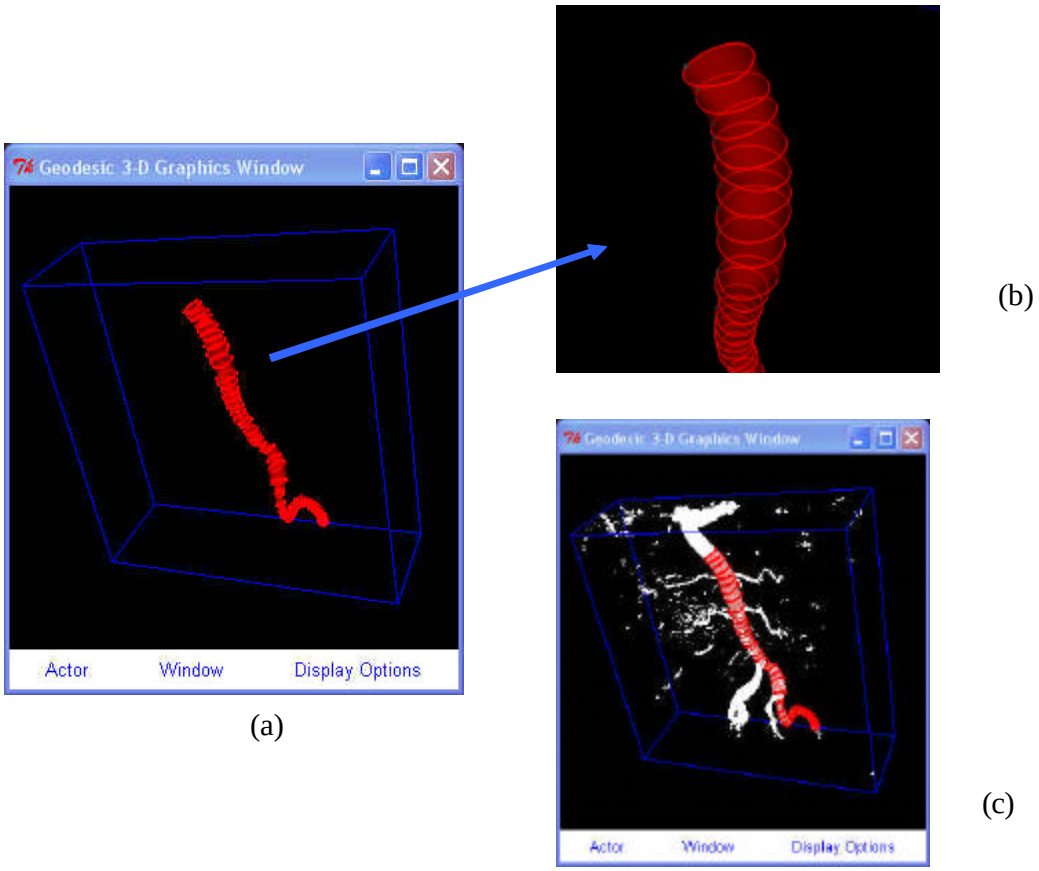
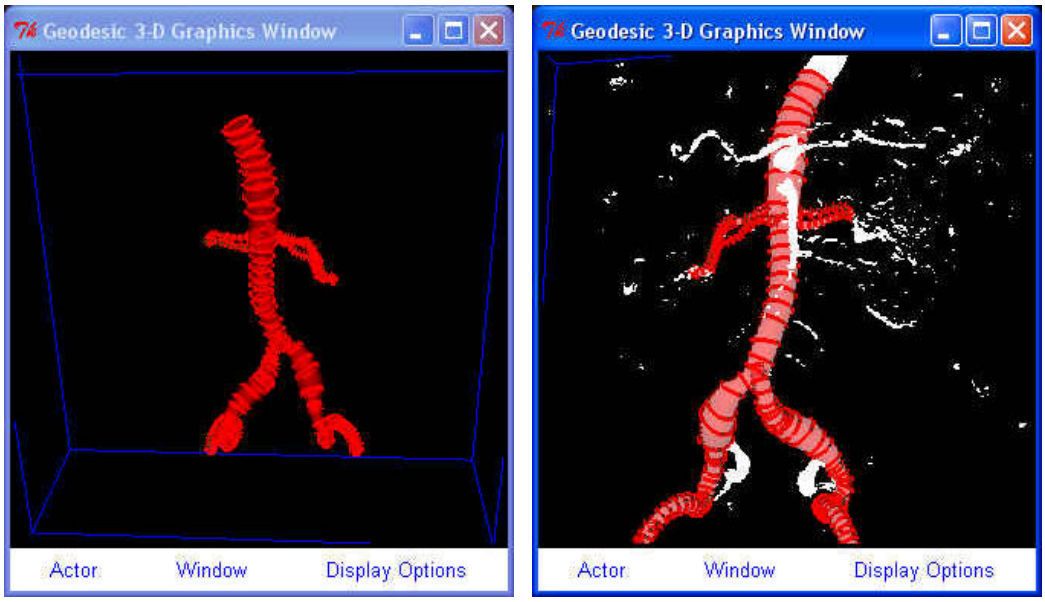


Figure 3.25: Lofting a solid for a single path. The figure shows an example of lofting a solid consisting of segmentations of the aorta, left common iliac, and left external iliac arteries. Figure (a) shows the lofted solid (semi-transparent) along with the segmentations (curves) used to create the solid. Figure (b) shows a close-up of the inlet of the lofted vessel, while (c) displays the solid branch along with a point cloud visualization of the image data.

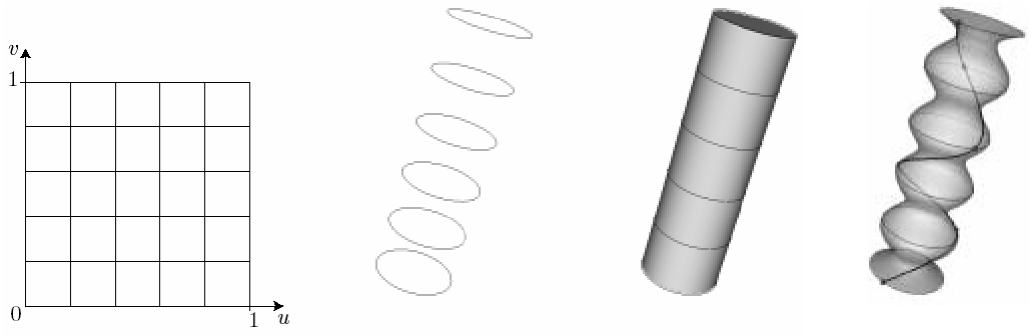


(a) (b)

Figure 3.26: Solid model created from six vessel branches. The process to create a solid model of the hemodynamic flow domain involves creating a lofted solid model for each individual branch and then performing a boolean addition (i.e. union) of the individual solid vessel models. Figure (a) shows an example of a solid created from six vessel branches. Figure (b) shows the solid displayed with a point cloud visualization of the volumetric image data.

Figure 3.27 gives more insight into the lofting process of a single vessel. Each 2-D segmentation becomes an isocurve (i.e. v = constant) in the parametric direction u. Since a path position is associated with each cross-section created, the curves are arranged logically in sequence from the beginning to the end of the path (i.e. v increases along the path). The challenge lies in aligning the profiles to avoid artificial twist. This twist can be eliminated by requiring that points that lie close to each other on the lumen boundary be close to each other in parametric space (u,v). Figure 3.27d shows an example of a surface constructed with profiles whose circumferential parameterizations are rotated with respect to one another. Several computationally efficient methods for profile alignment have been proposed (see [11]) that work in most cases. To address the robustness requirements for surgical planning, however, the alignment algorithm shown in Figure 3.28 was implemented. The basic idea behind this algorithm is to align the

closest two points on each profile and rearrange the segmentation points as required. This leads to an algorithmic complexity of $O((k\text{-}1)n^2)$ where k is the number of profiles being aligned and n is the number of points in a single profile. While the algorithmic complexity may initially be of some concern, it should be noted that typically $k < 100$ and $n < 1000$ so the wall-clock time for calculating this alignment for a single vessel tends to be less than 30 seconds on a modern PC (e.g. 1.7 GHz Pentium IV processor).



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

Figure 3.27: Creating a solid by lofting a set of cross-sectional segmentations (from [11]). A unit size parametric patch (u,v) is created as shown in (a) and the parametric curves in (b) become lines of constant v. The lofting operation then creates a NURBS surface from the profiles representing the side of the topologic cylinder with the caps of the cylinder being planar surfaces bounded by the first and last parametric curves (c). The alignment of the parametric curves in (b) should be such that points close to each other in physical space should be near each other when the curves are embedded in the parametric space (u,v). Figure (d) shows an example in which the misalignment of the parametric curves in (b) can lead to undesired results.

It is worth noting that face entity tags are attached to the solid model faces during the lofting process shown in Figure 3.27. That is, the boundary of each solid vessel consists of three faces. Two planar surfaces correspond to the "caps" (or ends) of the vessel branches while a Non-Uniform-Rational-B-Splines (NURBS) surface represents the side. The planar faces are tagged at the time of creation with the name of the group used to create the vessel. With the exception of the aorta, the boolean operations used to create the preoperative solid model always eliminate one of the two original caps of a given vessel incorporated in the final solid (by design one end of each branch vessel is

completely inside of a parent vessel and eliminated by the union operation). In addition, as described in section 3.8.1, the aorta is often trimmed by the location of a PCMRI image slice to prescribe inflow boundary conditions. This intersection of the PCMRI plane with the aorta is tagged with a convenient name (i.e. "inflow"). The end result of this tagging process is that the all of the faces requiring boundary conditions for analysis (see section 3.9) are tagged during the solid model creation process. As described in section 3.7, this propagation of information through the modeling process reduces the possibilities for user error and streamlines the process.

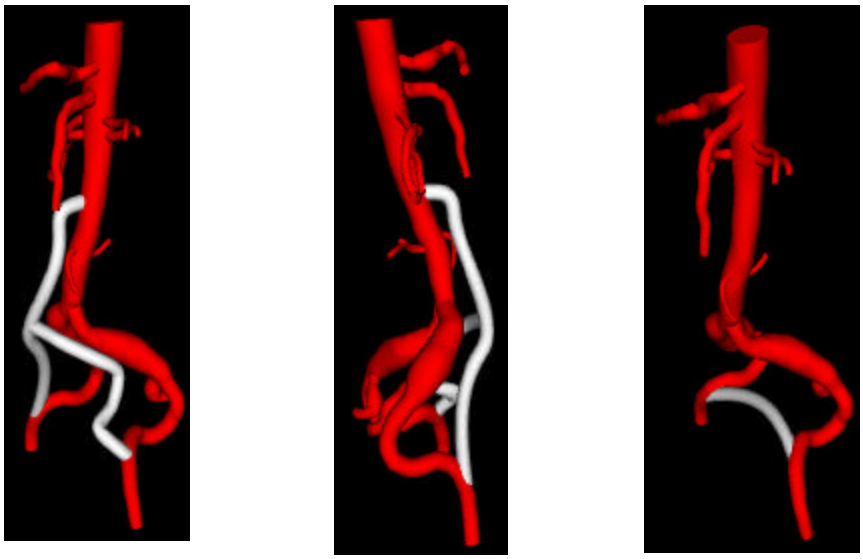
3.6 POSTOPERATIVE GEOMETRIC MODEL CONSTRUCTION

Once a preoperative geometric model has been created, the next step is to have a surgeon plan several surgical alternatives. For example, Figure 3.29 shows two different potential bypass procedures for a given preoperative model. The surgical planning examples in Chapter 4 will focus primarily on aorto-femoral bypass (AFB) procedures. In general, the following are requirements for a system to be able to perform the postoperative model construction necessary for simulation-based medical planning:

1. Provide appropriate anatomic information to select the location and path of the bypass graft
2. Provide the ability to create realistic proximal and distal anastomoses
3. Reasonably approximate the geometry of the bypass graft (including bifurcating grafts)

```
INPUT:  refPts ==  An ordered set of points representing the i^th
                   profile
        srcPts ==  An ordered set of points representing the i^th+1
                   profile

OUTPUT: dstPts ==  An ordered set of points for the i^th+1 profile
                   such that the closest point defining the
                   destination curve is aligned with the closest
                   point in the reference curve

d^2_min = VERY_LARGE_FLOAT

for ( i = 0; i < numPts; i++) {
   for ( j = 0; j < numPts; j++) {
      ix = i ; iy = j
      for ( k =0; k < numPts; k++) {
         d^2 = |refPts_ix - srcPts_iy|^2
         if (d^2 < d^2_min) {
            closestPt_ref = ix ; closestPt_dst = iy
            d^2_min = d^2
         }
         ix++ ; if (ix == numPts) ix = 0
         iy++ ; if (iy == numPts) iy = 0
      }
   }
 }

 if (closestPt_ref == closestPt_src) {
  // No re-alignment necessary
  dstPts = srcPts
  return dstPts
 }
 if ( closetPt_src > closestPt_ref ) {
    dx = closestPt_src - closestPt_ref
 } else {
    dx = closestPt_src + (numPts -closestPt_ref)
 }

 dstPts = reorder_pts_starting_with_dx(srcPts, dx)
 return dstPts
```

Figure 3.28: Pseudo code for profile alignment. The idea behind this algorithm is that the closest point on the i^{th} segmentation profile should be aligned with the closest point on the $i^{th}+1$ segmentation profile being used to loft the vessel solid model.



(a) (b) (c)

Figure 3.29: Example of two surgical plans for a human patient. Figures (a) and (b) show two different views of an aorto-femoral bypass, while (c) shows a femoral-femoral bypass. These surgical plans were created in the same rendering window as the volumetric image data to facilitate the creation of anatomically correct (i.e. physically possible) plans.

The **ASPIRE**^{2} system addresses these needs. Specifically, a GUI enables a surgeon to create the path or paths the bypass will follow (two or more paths are required in the case of a bifurcating graft). These paths are created within the same visualization window as the image data enabling the use of the visualization methods described in section 3.5.1 to provide anatomic context for the bypass procedure. In addition, the selection of the path points at the proximal and distal anastomoses allows limitless flexibility in the take-off angle of the anastomosis. Typically analytic cross-sections (circles and ellipses) are then fit along the path to approximate the distended geometry of the graft. The rule-of-thumb for the Dacron implanted bypass grafts used today in many AFB procedures is an enlargement of ~30% of the graft body and ~15-20% enlargement of the graft limbs over the device dimensions prior to surgery. Finally, it should be noted that the most

significant geometric approximation to bypass planning in the **ASPIRE**^{2} system likely occurs at the bifurcation of the graft. That is, the same difficulties in approximating bifurcations in the preoperative models using the methods described in section 3.5.4 also exist when creating the postoperative models.

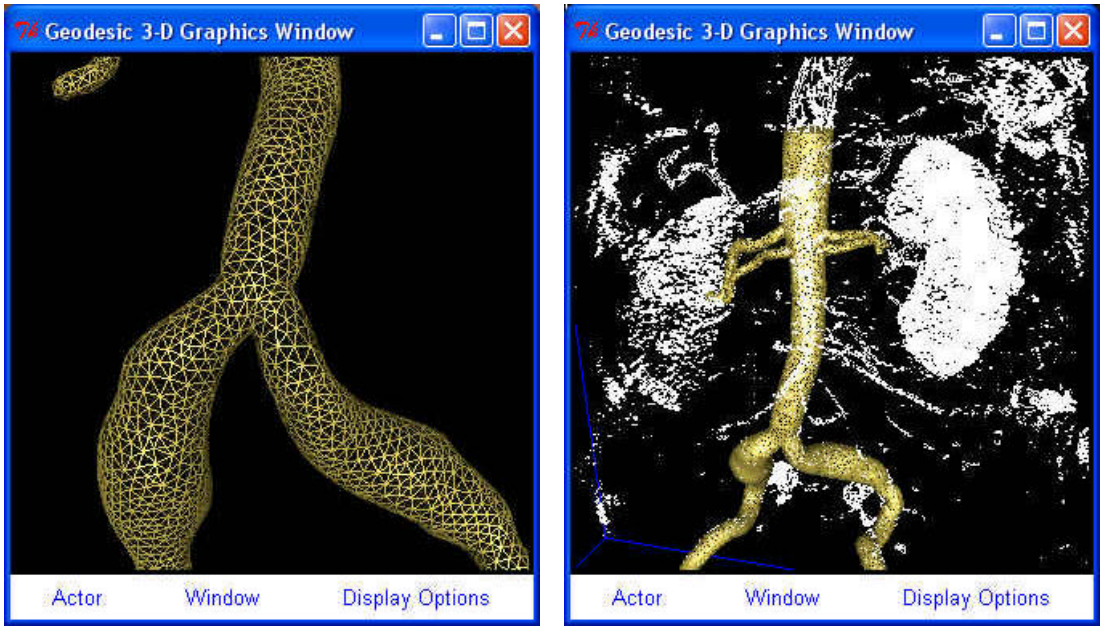
3.7 MESH GENERATION

Once a geometric model of interest exists, the next step in the SBMP process is usually to generate a finite-element mesh for simulation. Figure 3.30a shows an example of a mesh for a preoperative model. Note that even though a volumetric mesh is generated, only the exterior surface mesh is visualized in Figure 3.30. A strength of the **ASPIRE**^{2} system is the ability to visualize the finite-element mesh within the volumetric image data (see Figure 3.30b). As discussed in section 2.2.3, an automatic isotropic tetrahedral mesh generator is used to create meshes directly from the solid models created (see sections 3.5.4 and 3.6). It should be noted that improvements such as selective mesh refinement (particularly adaptive mesh refinement) are being investigated that may greatly reduce the computational cost required to achieve a given level of solution accuracy in the future. In addition, since the **ASPIRE**^{2} system automatically tracks solid model faces requiring boundary condition specification as described in section 3.5.4, the process of writing out nodes and element faces needed for boundary condition specification is trivial after a mesh is generated.

3.8 BOUNDARY CONDITION SPECIFICATION FROM PCMRI DATA

When MRA imaging data is acquired for the purpose of simulation-based medical planning, typically planar slices of experimental data are also acquired providing temporally and spatially resolved velocity fields. The technique used in the present work is known as cine-Phase-Contrast-Magnetic-Resonance-Imaging (PCMRI). PCMRI refers to a family of MR imaging methods that exploit the fact that nuclear spins that move

through magnetic field gradients obtain a different phase than static spins, enabling the production of images with controlled sensitivity to flow [46]. With appropriate selection of imaging parameters, the technique can be used to quantify volumetric flow and provide insight into the spatial velocity fields in a given planar slice location [47].

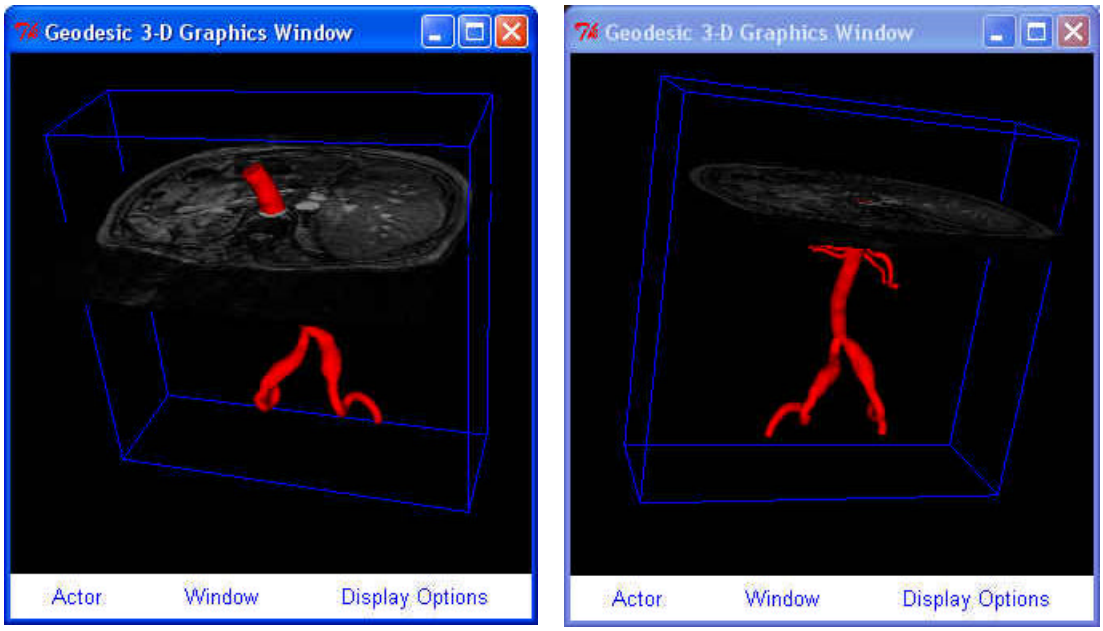


(a) (b)

Figure 3.30: Visualizing the exterior surface of a volumetric mesh. Figure (a) shows a close-up of the mesh near the aortic bifurcation. Figure (b) shows the finite-element mesh displayed with a point cloud visualization of the volumetric image data.

Using PCMRI data to prescribe boundary conditions presents several challenges. First, the actual vessel deforms over the cardiac cycle. Since the simulations used in this work rely on rigid-wall approximations (see section 3.9), the temporally varying lumen must be mapped onto a rigid inlet surface as described in section 3.8.3. Second, care must be used when the volumetric flow from multiple PCMRI slices is used to prescribe various inlet and outlet flows. In reality, the net inflow of blood during a single cardiac cycle into the modeled vascular domain is equal to the net outflow during a single cardiac cycle. Due to uncertainty and experimental error in the PCMRI data, however, it is often the case that the mean volumetric flow rates calculated from different slice locations will

not satisfy conversation of mass. Also, *in vivo* the instantaneous volumetric inflow is not required to equal the volumetric outflow since the vessel walls deform providing a form of capacitance in the system. However, the incompressible rigid-wall approximation does require instantaneous conservation of mass. This leads to modifications of the volumetric flow waveforms as discussed in section 4.3.1.4.



(a) (b)

Figure 3.31: Trimming a solid model with the location of a PCMRI slice. When prescribing boundary conditions directly at the inlet of the flow domain from PCMRI data, it is necessary to trim the solid model using the location of the PCMRI slice in 3-space. Figure (a) shows the solid model and the PCMRI data before the trimming operation, and (b) shows the solid after the trim operation.

3.8.1 Trimming geometric models

If the experimentally determined velocity field is to be mapped directly onto the inlet of the model, it is typically required that the model be "trimmed" at the location of the PCMRI plane. Figure 3.31 shows an example of this process for a model constructed from MRA data. Initially the geometric model is created to include the aorta proximal to supra-celiac PCMRI slice plane location. This plane is then used to trim the model as

shown in Figure 3.31b. Theoretically, the inlet surface to the model created by this trimming operation should correspond to the time-averaged cross-section contained in the PCMRI data. This is typically not the case, though, due to imaging limitations (spatial resolution, noise, partial volume effects, etc.).

3.8.2 Creating continuous velocity maps from PCMRI data

Most of the applications of PCMRI to date have been for volumetric flow rate or volume fraction calculations (see [47]). The phase data experimentally acquired corresponds to temporally discretized and spatially averaged velocity values in a given image voxel. For volumetric flow calculations, the volumetric flow is usually calculated as shown in Figure 3.32, where the through-plane velocity is assumed to be a piecewise constant function. While this is useful for volumetric flow calculations, the discontinuous nature of this representation is undesirable when creating velocity maps to be specified as the inlet boundary condition for a 3-D hemodynamic analysis.

In this work, a novel technique was used to create a C_0 continuous representation for each velocity vector component from the raw PCMRI data. First, a lumen boundary was segmented for each time frame from the intensity image data using one of the techniques shown in Figure 3.20. Next, the signed distance of each pixel centroid from the lumen boundary was calculated. The sign enables the classification of each pixel centroid as inside or outside of the lumen (Figure 3.33a). In addition, the distance optionally enables the exclusion of pixel centroids within a certain tolerance, ε, of the segmented boundary. Given the lumen boundary segmentation and the set of interior points, a 2-D constrained Delaunay triangulation is created (Figure 3.33b). This triangulation provides a topologic relationship between the interior pixel centroids and the points on the lumen boundary.

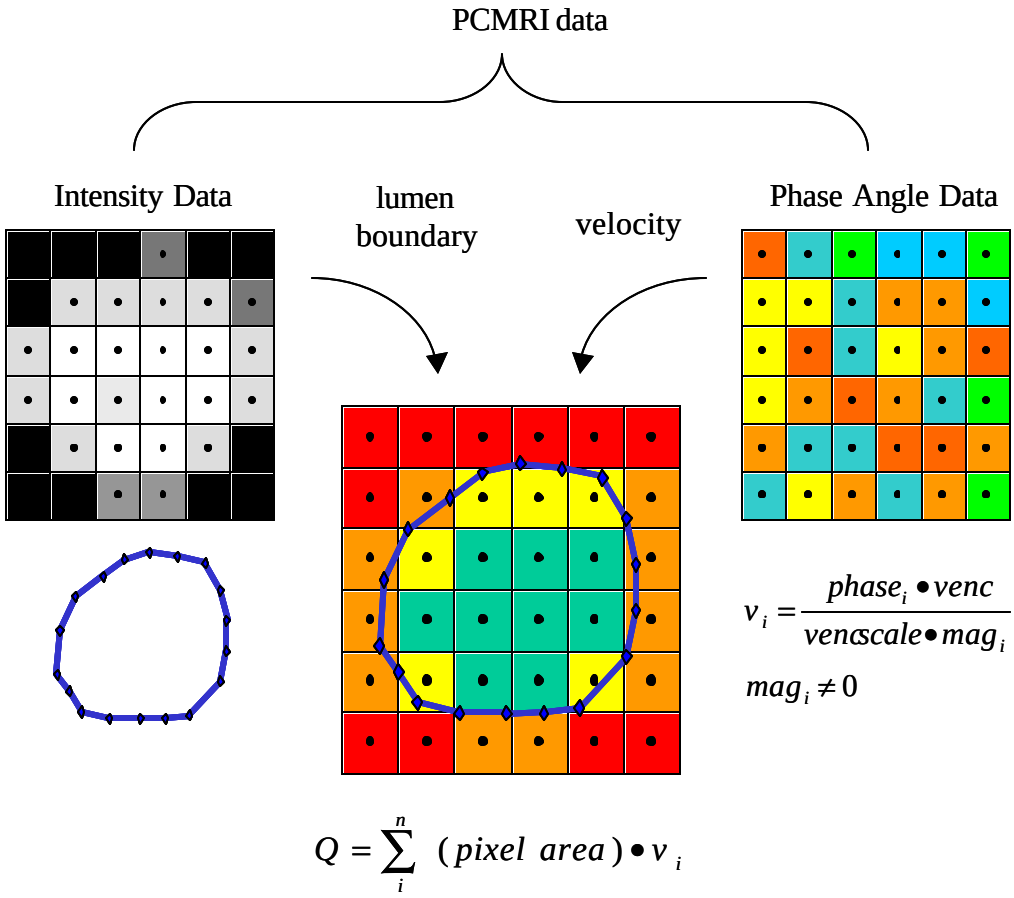
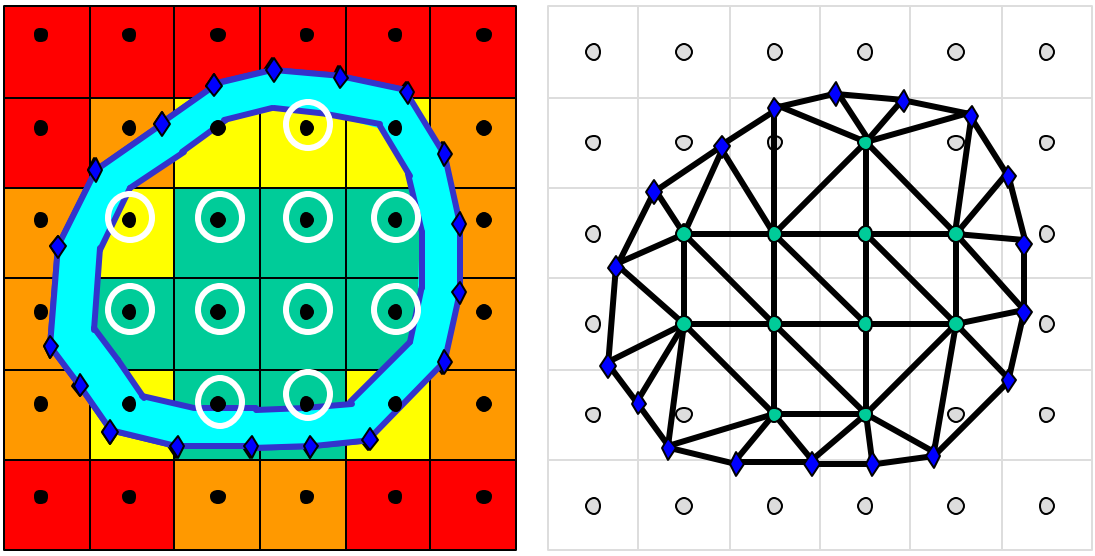


Figure 3.32: Calculating volumetric flow from PCMRI data. In general, MRI produces a signal intensity and phase angle. The phase angle is usually ignored, except in the case of PCMRI where information about the blood velocity field is encoded in the phase angle during acquisition and then can be extracted to calculate volumetric flow rate. Note: *n* is the set of all voxels whose centroid falls inside of the segmentation. This corresponds to green and yellow squares. Orange and red squares are ignored and do not contribute to the volumetric flow rate.



(a) (b)

Figure 3.33: Creating a C^0 velocity map from PCMRI data. For a given time frame, a segmentation (shown in blue) based on the magnitude data is created. The signed distance from the lumen boundary is calculated for all of the pixel centroids in the image. If the signed-distance is negative, the pixel centroid is added to the set of points in the velocity map (white circles in (a)). The user can optionally ignore centroids inside of the lumen boundary within an ε of the segmentation boundary (shown in light blue in (a)). A constrained-Delaunay triangulation is performed to generate a topologic relationship between the interior lumen points and the points comprising the piecewise-linear segmentation boundary (b). The velocity values for the corresponding image pixels are assigned to the interior lumen points, and the boundary points get assigned either the value from the nearest pixel centroid or zero depending on the user's preference.

For each interior point of the triangulation, the velocity vector of the corresponding image pixel is assigned to the point. For the boundary points, the user can decide to have the through-plane component either assigned a value of zero (assuming no-slip at the lumen boundary) or the value from the closest pixel centroid. Once each node is assigned a velocity vector, standard bilinear finite-element basis functions are used to interpolate vector component values at the interiors of the triangles of the triangulation. The volumetric flow rate is then calculated by integrating the bilinear interpolation functions over all the triangles in the mesh shown in Figure 3.33b and summing the results:

$$Q = \sum_{i=1}^{N} q_i \tag{3.8}$$

$$q_i = \int_{\Omega} v(x, y; t)\, d\Omega \tag{3.9}$$

For implementation purposes and to maximize the potential for code reuse elsewhere in the **ASPIRE**^{2} system, equation 3.9 was evaluated numerically using 2-point Gaussian numerical integration with standard bi-linear interpolation functions (with $v_3 \equiv v_4$ for triangles):

$$q_i \approx \sum_{k=1}^{2} \sum_{j=1}^{2} \sum_{i=1}^{4} w_k \; w_j \; J \; (N_i \, v_i) \tag{3.10}$$

where:

w_k = Gaussian integration weight k

J = Jacobian determinant

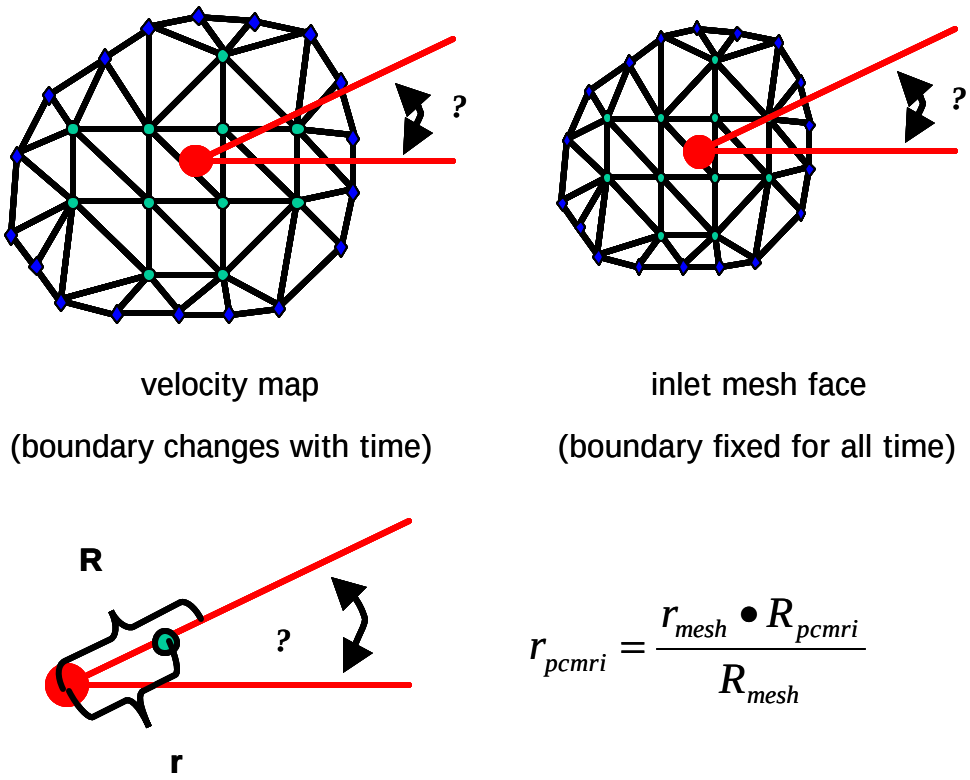
N_i = finite-element basis function for node i

v_i = nodal velocity

3.8.3 Mapping from PCMRI data onto a stationary mesh

As mentioned previously, in general the lumen boundary in the PCMRI data is changing with respect to time. However, since rigid-wall approximations are being used, the temporally varying spatial velocity map from PCMRI needs to be mapped to the stationary inlet of the finite-element mesh. A simple mapping is employed as shown in Figure 3.34. This mapping assumes convexity of the lumen boundary and the inflow boundary of the finite-element mesh. The mapping shown in Figure 3.34 accounts for rigid translation of the lumen in the PCMRI data with respect to the model constructed from the MRA data. However, any rotation between the two will contribute to error in

the inflow boundary condition. An isotropic scaling is performed on the projected velocities to preserve volumetric flow rate. More sophisticated mappings may address some of the limitations of the current mapping strategy.



$$r_{pcmri} = \frac{r_{mesh} \bullet R_{pcmri}}{R_{mesh}}$$

$$v_{node} = v_{pcmri}(x_{ctr} + r_{pcmri}\cos(q), y_{ctr} + r_{pcmri}\sin(q))$$

Figure 3.34: Mapping from a temporally varying spatial velocity map to a stationary inlet of a finite-element mesh. The mapping assumes convexity of the lumen boundary and the inflow boundary of the finite-element mesh. The centroids of the velocity map and the inlet mesh are aligned before the mapping, eliminating the effect of rigid body translation. However, any rotation between the velocity map and the inlet mesh will contribute directly to error in the mapping.

3.8.4 Prescribing velocity boundary conditions based on volumetric flow rate

It is often the case that one has only a volumetric flow rate and not a spatial velocity map such as the one provided by PCMRI data. This is typically the case for "small" vessels, where insufficient spatial resolution of the PCMRI data does not yield a usable velocity

field but does provide an approximation of the mean volumetric flow rate. In addition, mean volumetric flow through a vessel (e.g. the renal arteries) is often determined by taking slices above and below the branch location of the given vessel and using conservation of mass to calculate the unknown flow rate(s). In these cases, a useful approximation of the velocity field is given by assuming an analytic velocity profile derived for fully developed pulsatile flow for a Newtonian fluid in a rigid cylinder. By expressing the volumetric flow using N terms of a Fourier series,

$$Q(t) \approx \sum_{n=0}^{N} B_n e^{i n w t} \tag{3.8}$$

where:

B_n = Fourier coefficient

ω = angular frequency = period/(2π)

The Womersley axisymmetric velocity profile is given by (see [5]):

$$w(r,t) = \frac{2B_0}{p\,R^2}\left[1-\left(\frac{r}{R}\right)^2\right] + \sum_{n=1}^{N}\left\{\frac{B_n}{p\,R^2}\left[\frac{1-\dfrac{J_0\left(a_n \dfrac{r}{R} i^{3/2}\right)}{J_0\left(a_n i^{3/2}\right)}}{1-\dfrac{2J_1\left(a_n i^{3/2}\right)}{a_n i^{3/2} J_0\left(a_n i^{3/2}\right)}}\right]\right\} e^{i n w t} \tag{3.9}$$

where:

w = axial component of velocity

R = vessel radius

J_0, J_1 = Bessel functions of the first and second kind, respectively

$a = R\sqrt{w/n}$ = Womersley number

υ = kinematic viscosity

In the **ASPIRE**^{2} implementation the resulting profile is mapped onto the mesh using the same techniques as described in section 3.8.3. Note that for even slight deviations from a circular cross-section or in the presence of insufficient length to generate fully developed flow this may be a poor approximation to the actual velocity profile.

3.9 ANALYSIS

In reality, the vessel wall deforms during the cardiac cycle, blood behaves like a non-Newtonian fluid, and turbulent and non-periodic flow may be present particularly in the case of vascular disease. In the work presented here, however, as a first-order approximation the vessel walls are assumed to be rigid, the fluid Newtonian, and the boundary conditions periodic. Hemodynamic simulation with these simplifications consists of solving the incompressible Navier-Stokes equations:

Given $\underline{f} : \Omega \times (0,\mathrm{T}) \rightarrow \Re^3$, $\underline{g} : \Gamma_\mathrm{g} \times (0,\mathrm{T}) \rightarrow \Re^3$, $\underline{h}: \Gamma_\mathrm{h} \times (0,\mathrm{T}) \rightarrow \Re^3$, and $\underline{u_0}: \Omega \rightarrow \Re^3$. Find $\underline{u}(\underline{x},\mathrm{t})$, and $p(\underline{x},\mathrm{t})$ $\forall \underline{x} \in \Omega$, $\forall \mathrm{t} \in [0,\mathrm{T}]$ such that

$$
\begin{aligned}
& r\left(\underline{u}_t + (\underline{u}\cdot\nabla)\underline{u}\right) = \nabla\cdot\underline{\underline{s}} + \underline{f} && (\underline{x},t) \in \Omega\times(0,T) \\
& \nabla\cdot\underline{u} = 0 && (\underline{x},t) \in \Omega\times(0,T) \\
& \underline{u}(\underline{x},0) = \underline{u}_0(\underline{x}) && \underline{x} \in \Omega \\
& \underline{u} = \underline{g} && (\underline{x},t) \in \Gamma_g\times(0,T) \\
& \underline{t} = \underline{h} && (\underline{x},t) \in \Gamma_h\times(0,T)
\end{aligned}
\qquad (3.10)
$$

where:

$\underline{u}$ = velocity vector

p = pressure

Ω = flow domain

Γ = boundary of flow domain ($\Gamma_\mathrm{g} + \Gamma_\mathrm{h} = \Gamma$, $\Gamma_\mathrm{g} \cap \Gamma_\mathrm{h} = \varnothing$)

ρ = density

$\underline{f}$ = volumetric forcing function

$\underline{g}$ = prescribed velocity boundary conditions

$\underline{h}$ = prescribed traction boundary conditions

$\underline{u}_0$ = initial velocity field (assumed divergence free)

$\underline{\underline{s}} = -p\underline{\underline{I}} + \frac{m}{2}(\nabla\underline{u} + \nabla\underline{u}^T)$ (i.e. Newtonian fluid)

$\underline{t} = \underline{\underline{s}}\,\underline{n}$ is the traction vector

m = kinematic viscosity

In the present work, either $\underline{u}$ or p (not both) is specified at each boundary leading to a well-posed problem.

In addition to velocity and pressure, a passive scalar transport problem is solved during the simulation to study scalar clearance time. The strong form of the transport problem is:

Given: $f_0: \Omega \to \Re^3$, $\underline{u}(\underline{x},t)$ and $p(\underline{x},t)$ from (3.10), find f $\forall \underline{x} \in \Omega$, $\forall t \in [0,T]$ such that

$$\begin{aligned} & r f_t + r(\underline{u}\cdot\nabla f) = \nabla\cdot\left(\underline{\underline{k}}\,\underline{f}\right) + f && (\underline{x},t) \in \Omega\times(0,T) \\ & f(\underline{x},0) = f_0(\underline{x}) && \underline{x} \in \Omega \\ & f = g && (\underline{x},t) \in \Gamma_g \times (0,T) \\ & \nabla f \cdot \underline{n} = h && (\underline{x},t) \in \Gamma_h \times (0,T) \end{aligned} \qquad (3.11)$$

where:

ρ = density

f_0 = initial scalar field

$\underline{\underline{k}} \equiv k\underline{\underline{I}}$ = isotropic diffusitivity matrix

f = volumetric source term

g = prescribed scalar

h = prescribed flux

A commercial finite-element code, Spectrum™ [48], was used to perform the calculations reported here. Briefly, the code uses the stabilized finite-element method introduced by Hughes et. al. [49]. A two-step stagger solution strategy was employed where the following three staggers were solved in succession for each step:

1. A matrix-free iterative GMRES solver [50] was used to solve the coupled problem for $\underline{u}$ and p with ϕ held fixed.
2. A conjugate-gradient solver was used to update the pressure solution p while $\underline{u}$ and ϕ were held fixed.
3. A matrix-free iterative GMRES solver was used to solve for ϕ while $\underline{u}$ and p were held fixed.

The meshes were decomposed using a domain decomposition algorithm [51] enabling calculations to be performed in parallel. The computations were done on a 128 processor (400 MHz IP35 MIPS), 32-node SGI Origin 3800 supercomputer with 64 GB of shared memory. While the wall-clock computation time for the calculations in this thesis varied based on the system load, number of processors used, time step size, and number of cardiac cycles solved, most of the simulations (400 time steps per cardiac cycle, 6 cardiac cycles, approximately 2 million elements running on 32 processors) required roughly 48 hours of wall-clock time per simulation. This implies that the computational cost in performing the analysis needs to be considered when evaluating the clinical application of SBMP tools using current computer technology.

3.10 HEMODYNAMIC VISUALIZATION

The results of the hemodynamic simulations described in the previous section are hundreds of megabytes (often gigabytes) of analysis results. As discussed in section

2.2.5, scientific visualization transforms the millions of scalars and vectors produced during the analysis into visual images that can be used to understand the flow solutions. Figure 3.35 shows an overview of the analysis visualization (i.e. post-processing) capabilities integrated into the **ASPIRE^2** system.

Three specific examples highlight the benefit of the integrated visualization environment provided by **ASPIRE^2**. First, the analysis results can be viewed directly inside the same window with the volumetric image data to provide additional anatomic context to the solutions. In addition, the user can specify the location of a cut plane based on a PCMRI slice enabling the direct comparison of finite-element results to experimentally acquired velocity results. Second, information used in the construction of the solid model can be used directly to query the analysis results. For example, the user can plot velocity and pressure along a medial axis path. Finally, additional functionality is integrated in **ASPIRE^2** to calculate post-processing quantities of interest such as wall shear stress.

A surgeon may want to review velocity profiles, pressure distributions, and scalar clearance time for a proposed surgical procedure. In addition, several post-processed quantities are also of interest because of their theorized role in atherosclerosis. Specifically, instantaneous wall shear stress, mean wall shear stress, shear stress pulse, and oscillatory shear index (OSI) are all derived quantities of the solution of interest to a vascular surgeon. By definition, the instantaneous wall surface traction vector (i.e. shear stress) is given by:

$$\underline{t}_s = (\underline{\underline{s}} - s_{nn}\underline{\underline{I}})\,\underline{n} \qquad (\underline{x},t) \in \Gamma \times (0,T) \tag{3.12}$$

where $s_{nn} = \underline{n} \cdot \underline{\underline{s}}\,\underline{n}$, $\underline{n}$ is the unit outward surface normal, and $\underline{\underline{s}}$ is defined as in equation 3.10.

The following useful quantities representing different temporal averages of equation 3.12:

$$t_{mean} = \left| \frac{1}{T} \int_0^T \underline{t}_s \, dt \right| \tag{3.13}$$

$$t_{pulse} = \frac{1}{T} \int_0^T \left| \underline{t}_s \right| dt \tag{3.14}$$

$$OSI = \frac{1}{2} \left(1 - \frac{t_{mean}}{t_{pulse}} \right) \tag{3.15}$$

Chapter 4 shows examples of using the visualization methods described in this section to clinically relevant cases of aortoiliac occlusive disease and abdominal aortic aneurysms. In addition, Chapter 4 briefly discusses the role of these visualization methods in related validation work of the methods described in this chapter.

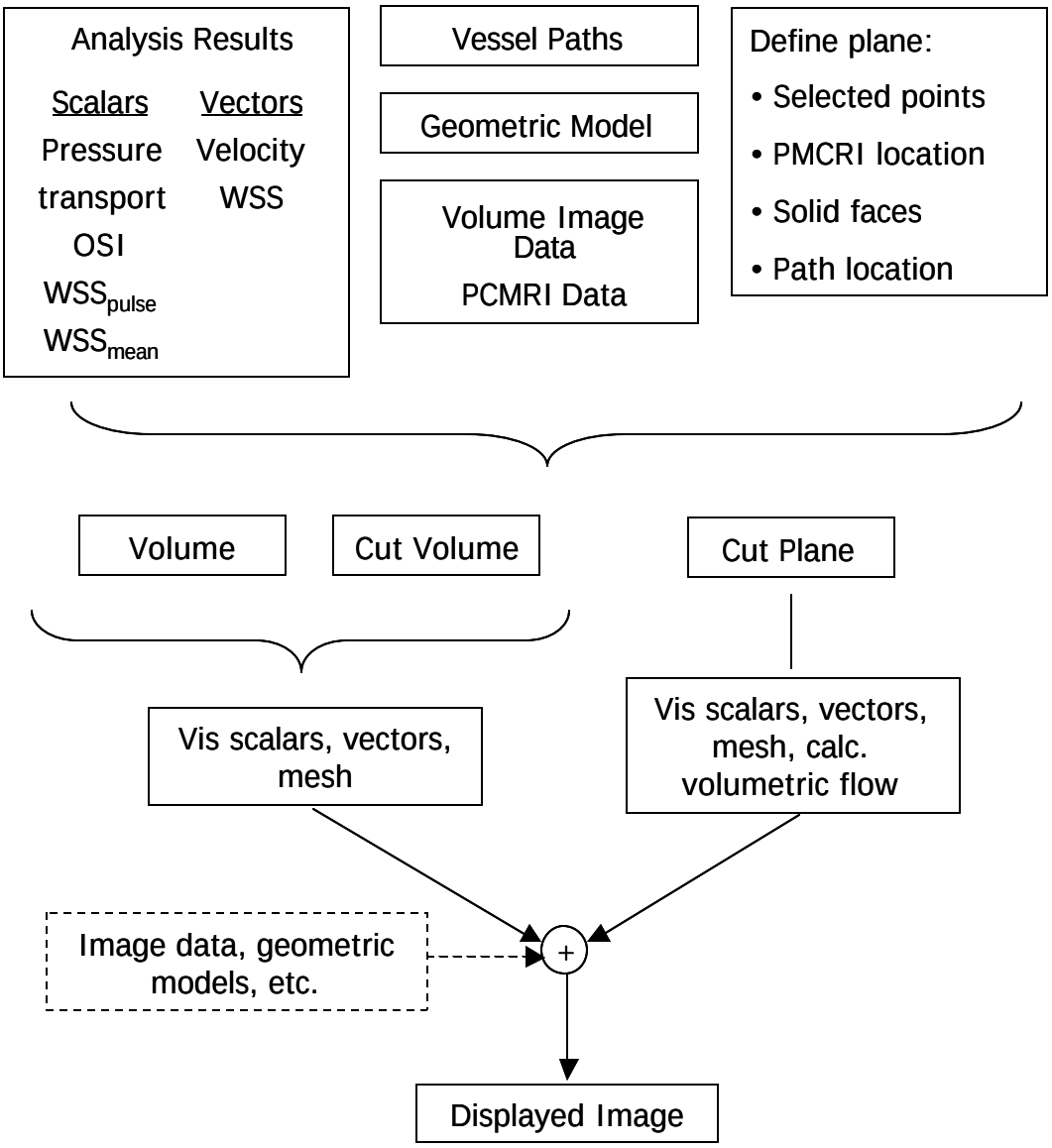


Figure 3.35: Post processing visualization options in **ASPIRE**^{2}. The idea behind the visualization GUI was to combine standard visualization techniques (e.g. displaying the velocity vectors) with more application-specific visualizations (e.g. slicing the analysis results using the location of a PCMRI plane). In addition, the high level of integration in **ASPIRE**^{2} enables the user to visualize the analysis results along with the imaging data and geometric models.

CHAPTER 4 VASCULAR APPLICATIONS

4.1 OVERVIEW

The focus of this chapter is to demonstrate the application of the **ASPIRE^2** system detailed in the previous chapter to clinically relevant cases of vascular disease. For surgical planning, two patient case studies are presented where medical imaging data was acquired prior to and following aorto-femoral reconstruction. In addition, an example of flow through an abdominal aortic aneurysm is presented to demonstrate future applications of this system to study flow phenomenon. As a prelude to the clinical applications, simpler examples are studied to help provide validation and insight into the limitations of the methods being employed.

4.2 VALIDATION

The major sources of error in the SBMP process can be roughly divided into categories closely linked to the steps of the process given in section 3.4.2:

1. Geometric inaccuracies in the preoperative solid model
2. Geometric inaccuracies introduced in the operative planning
3. Numerical inaccuracies in the computation
4. Inaccuracies in physiologic data provided by PCMRI
5. Modeling assumptions (e.g. rigid walls, boundary conditions, etc)

The following four sections provide additional insight and information into the sources of error. The discussion of inaccuracies in physiologic data and inaccuracies introduced in the operative planning process will be deferred until section 4.3.

4.2.1 Geometric inaccuracies in the preoperative solid model

There are two major sources for geometric inaccuracy in the preoperative solid model. First, the imaging data can contain errors introduced by limitations of the acquisition method. This is particularly true in MRA, where complex flow phenomena can degrade the signal intensity of blood during a contrast enhanced scan. In addition, nonlinearities in the magnetic field can seriously degrade the quality of MRA data (see below). A second source of error arises when using the methods described in section 3.5.4 for constructing models. Selecting inappropriate segmentation parameters, for instance, can degrade the quality of the models constructed even in the presence of image data with adequate resolution (see [11]). The rest of this section details the correction for an error specific to Magnetic Resonance Imaging (MRI).

It is useful for this discussion to define the terms *in vivo* data and *in vitro* data. When imaging data is acquired of a living organism, it is referred to as *in vivo.* *In vivo* data has the benefit of representing the entity of interest in its native physiologic environment, subject to the forces and conditions actually exerted on it. However, determining the validity of *in vivo* data is difficult due to these physiologic parameters that exist inside of a living organism. Thus, *in vitro* data can be used for validation of several of the methods being applied in this work. *In vitro* data is acquired in an artificial environment designed to eliminate sources of error, simplify data acquisition, and provide additional means of determining quantities of interest.

MRI is a particularly useful technique for simulation-based medical planning because it can provide both physiologic and volumetric geometric information. The MRI scanner used in this work was developed and manufactured by General Electric [52]. Ideally, the scanner creates linearly varying gradients of the magnetic field across the image volume. However, in practice, non-linearity of the magnetic field gradients exists that must be accounted for or significant geometric errors occur in the volumetric image data. Image data post-processing techniques were originally proposed in 1983 by Glover and Pelc

[53] to correct for these so-called “gradient warping” (“gradwarp” for short) effects. In the commercially available clinical volumetric scans used in the present work, gradwarp effects are corrected independently in two of the primary coordinate directions of each image slice but not in the direction between slices. An *in vitro* test phantom shown in Figure 4.1 was used to quantify the distortion in the “slice plane” direction due to this lack of gradwarp correction [54]. In general, data can be acquired in three orthogonal directions: coronal, sagittal, and axial. The test phantom was placed in the scanner and volumetric image data was acquired in the three primary directions without moving the phantom. As the three views in Figure 4.2 indicate, highly visible error occurs in the volume data using the standard volumetric image scans. Ideally the extracted geometry for each tube from each acquisition direction should occupy the same location in physical space, and the tubes should remain straight.

Figure 4.3 shows a reformatted maximum intensity projection (MIP) that gives a more quantitative description of the volumetric error. This view helps clarify that the distortion is minimal at the center and increases significantly away from the center of the image volume (referred to as isocenter). Figure 4.4 and Figure 4.5 shows a quantitative study indicating the promise of an experimental slice direction gradwarp correction algorithm provided by General Electric [55,52]. Finally, Figure 4.6 shows the clinical significance of the correction in a supra-celiac aortic reslice from a volumetric image dataset. As these results indicate, the gradwarp correction is clearly necessary for SBMP when using MRA data and all of the clinical results presented in this work utilize gradwarp correction in the slice plane direction.

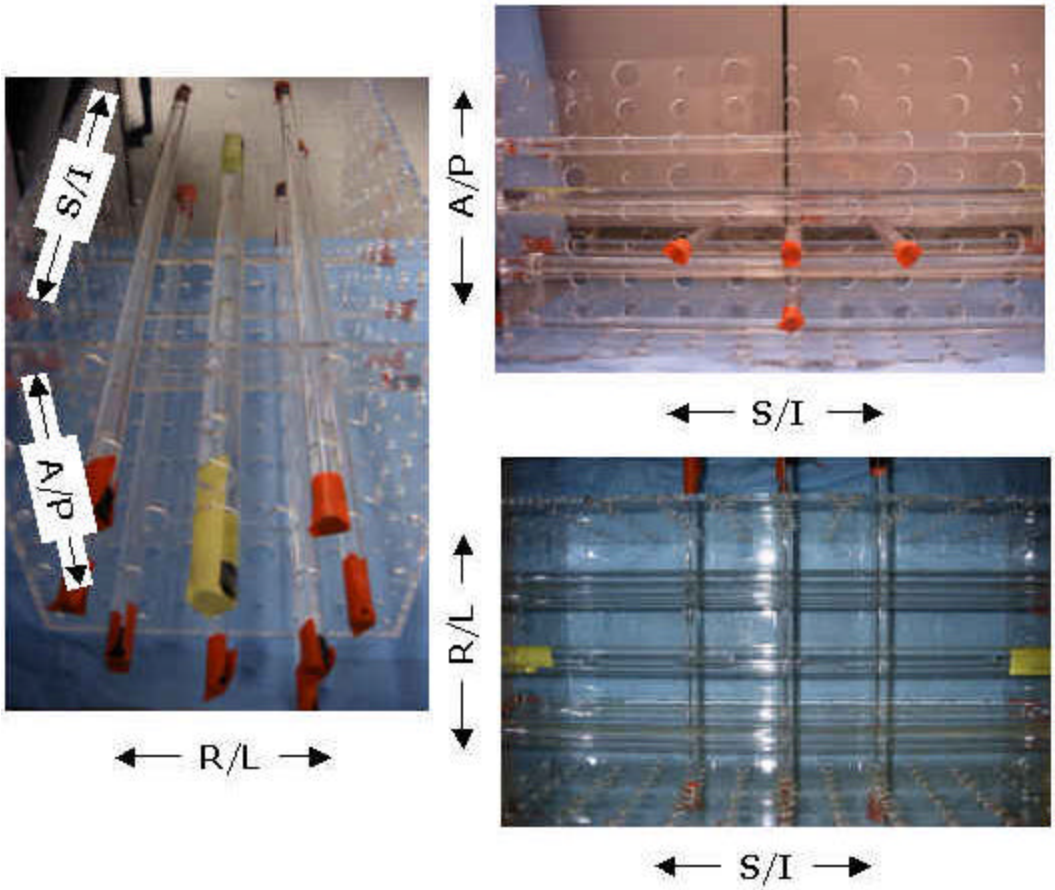


Figure 4.1: Pictures of the physical test phantom. The test phantom consists of a rectangular solid-shaped box made from Plexiglas with an array of evenly spaced holes allowing rigid tubes of various known diameters to be inserted. The rigid tubes are filled with a contrast agent (Gadolinium) and sealed with plastic corks covered with tape. The configuration shown above corresponds to the configuration imaged using MRA shown in Figure 4.2 and Figure 4.3.

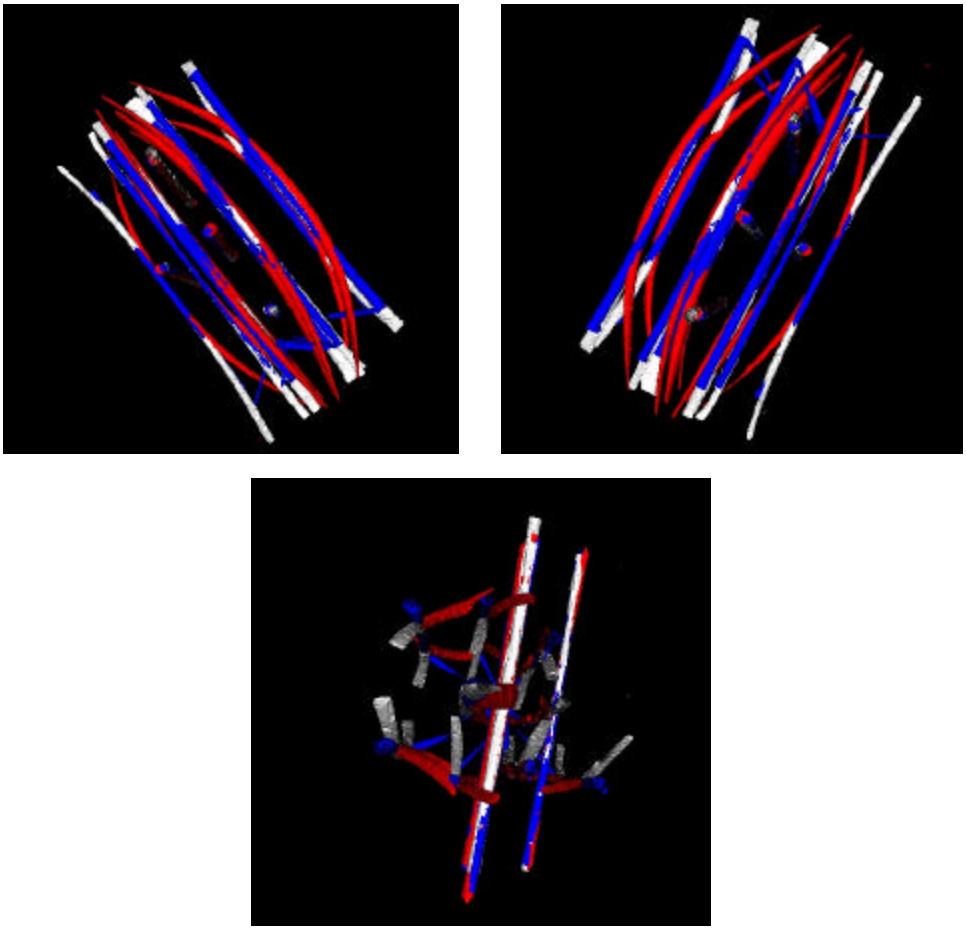


Figure 4.2: Views of the reconstructed phantom from three MRA datasets. The blue corresponds to the axial acquisition, the red the coronal acquisition, and the white the sagittal acquisition. Since straight rigid tubes were used and the phantom was not moved between acquisitions, the highly visible differences in the views shown above correspond to error in the acquisition predominantly due to gradwarp effects.

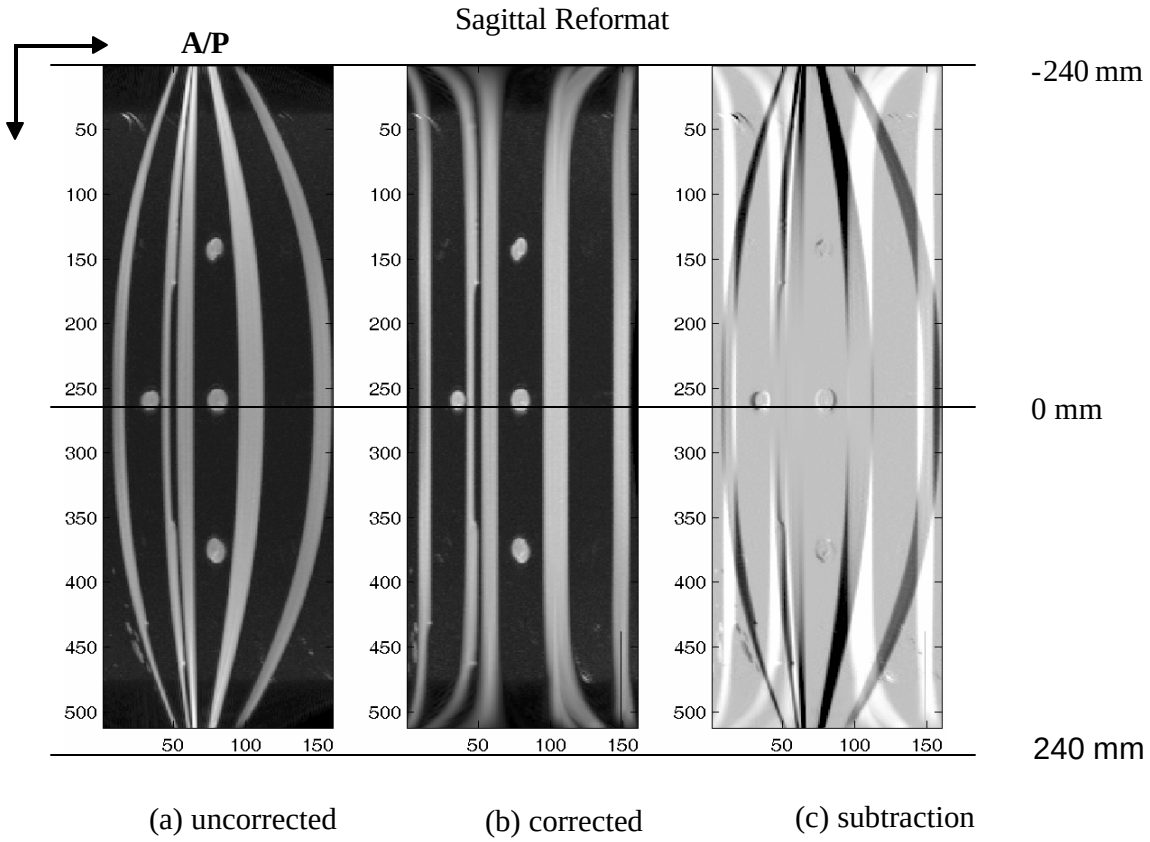


Figure 4.3: Reconstructed sagittal MIP showing effect of gradwarp correction. The most commonly used clinical scan direction is coronal, and the best view of the slice plane direction gradwarp effects is the sagittal reformat shown in this figure. The rigid straight tubes appear curved in the uncorrected image (a) and straighter in the corrected image (b). Figure (c) shows the subtraction of (b) – (a), where white and black correspond to large differences in the pixel value and gray indicates nearly identical values. Figure (c) illustrates the gradwarp effects are least significant at the center of the image volume, and increase significantly away from the center.

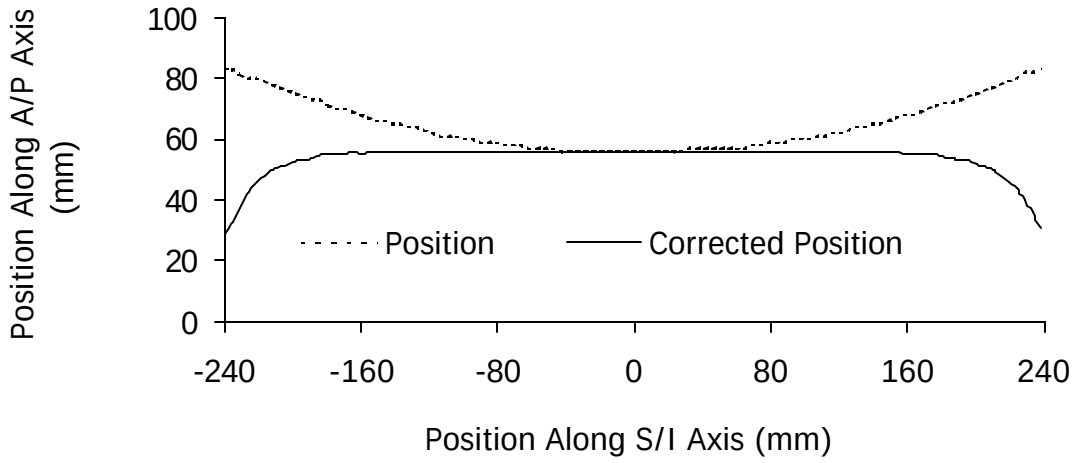


Figure 4.4: Position in A/P direction along one tube in phantom. A single tube aligned with the S/I coordinate axis near the center of the test phantom was selected and the centerline of the tube was plotted as a function of S/I coordinate. The gradwarp corrected data (solid line) indicates an improvement in the location of the tube in the image volume.

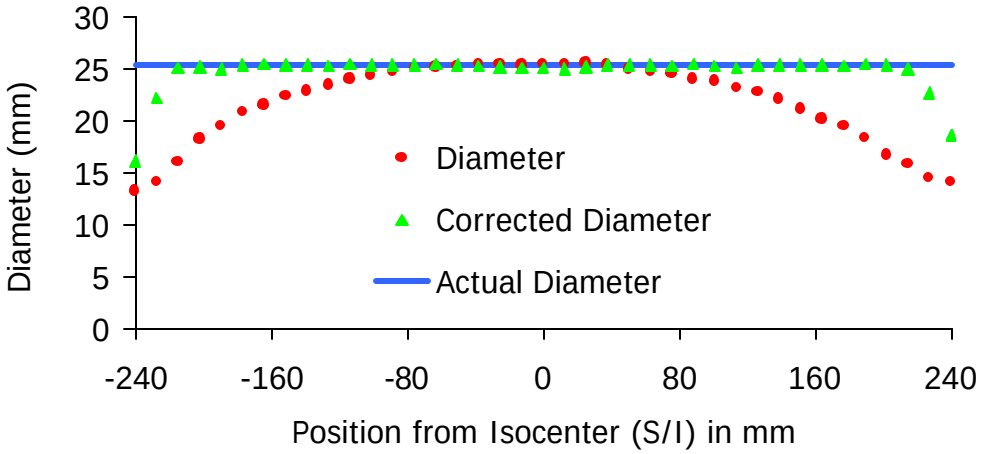
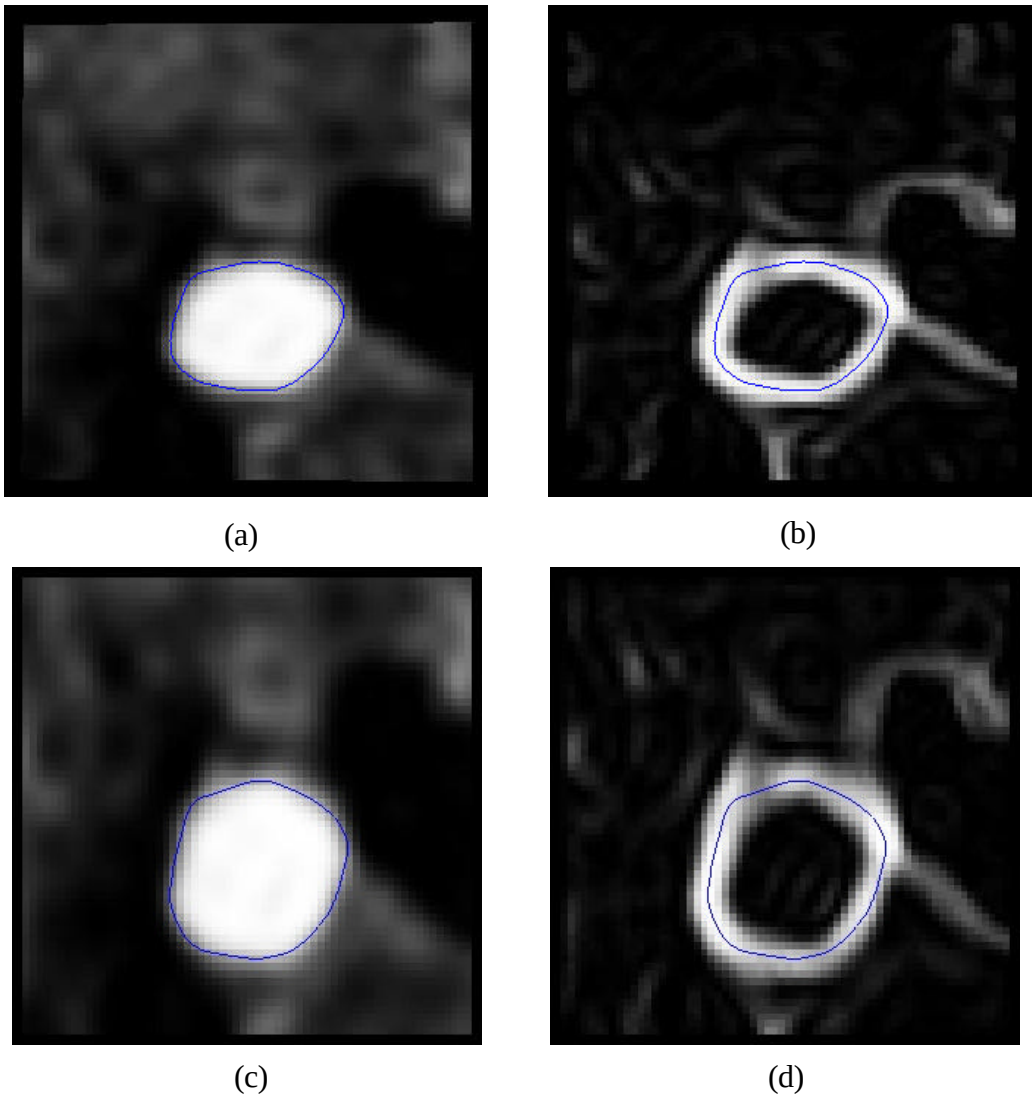


Figure 4.5: Diameter along length of one tube in phantom. From the coronally acquired MRA dataset, a single tube aligned with the S/I coordinate axis was selected and the cross-sectional area along the length of the tube was calculated as a function of S/I coordinate. The cross-sectional area for the tube from the gradwarp corrected data is in better agreement with the actual diameter of the rigid tube.



(a) (b)

(c) (d)

Figure 4.6: Gradwarp corrected supra-celiac aorta reslice. A planar reslice of the volumetric MRA data for patient #1 is shown where (a) corresponds to the uncorrected data and (c) corresponds to the gradwarp corrected data. A level set segmentation (shown in blue) was created for each dataset from the potential data shown in (b) and (d). The cross-sectional area from the uncorrected data was 258.56 mm^2 while the cross-sectional area for the corrected data was 364.48 mm^2. The cross-sectional area for the corrected data is in closer agreement with the average cross-sectional area from PCMRI data acquired at the same location for this patient (429.74 mm^2).

4.2.2 Validation of the numerical methods

In general, the finite-element method employed is a convergent numerical method. Thus, as the mesh is refined and the time step is reduced the results will continue to improve and approach the exact solution. Few known analytic solutions to the Navier-Stokes equations exist, but as discussed in section 3.8.4 an analytic solution does exist for fully developed flow in a rigid straight axisymmetric tube. This motivates performing a convergence test of applying a pulsatile plug flow at the inlet of a sufficiently long straight rigid tube and studying the outlet velocity pattern and pressure distribution as the mesh is continually refined.

A cylinder with a length of 6 cm and diameter of 0.4 cm was subjected to a periodic plug inflow velocity boundary condition given by equation 4.1:

$$V(t) = \overline{V}\left(1 + \sin(2p\,\frac{t}{T})\right) \tag{4.1}$$

Where the period, T, was 0.2 seconds, the mean velocity ($\overline{V}$) was 13.5 cm/sec, and the viscosity of the fluid was 0.04 poise. A constant zero pressure boundary condition was weakly enforced on the cylinder outlet. The cylinder was discretized and the solution strategy given in section 3.9 was used to solve for the fluid flow through the cylinder.

Three meshes of varying spatial resolution were created (a coarse mesh with 22,253 elements, a refined mesh with 130,217 elements, and a more refined mesh with 828,961 elements). Selected simulation results are shown in Figure 4.7 through Figure 4.11. The volumetric inflow and outflow was calculated for each mesh and a representative waveform is shown in Figure 4.7. Figure 4.7 indicates that mass is conserved by the flow solver. The accuracy of the velocity solution improves as the mesh is refined as seen in Figure 4.8. Figure 4.9 shows that the outlet velocity profile has converged throughout the period for the refined mesh. Figure 4.10 shows that the centerline velocity reaches a constant value approximately 3 cm from the inlet of this cylinder indicating fully

developed flow. Finally, Figure 4.11 demonstrates linear variations in pressure as the flow becomes fully developed – an important assumption in Womersley theory.

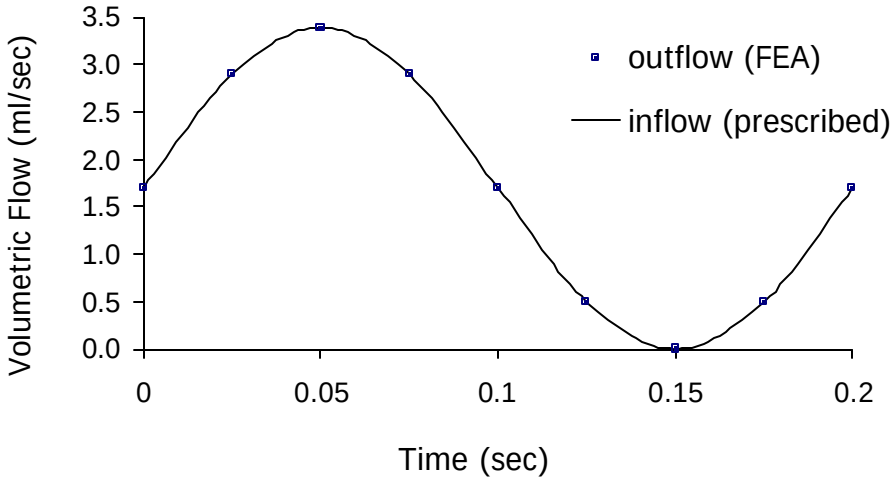


Figure 4.7: Conservation of mass in the flow solver. Since the model is rigid and the fluid is modeled as incompressible, the volumetric outflow at the outlet of the cylinder should equal the volumetric flow rate at the inlet of the cylinder.

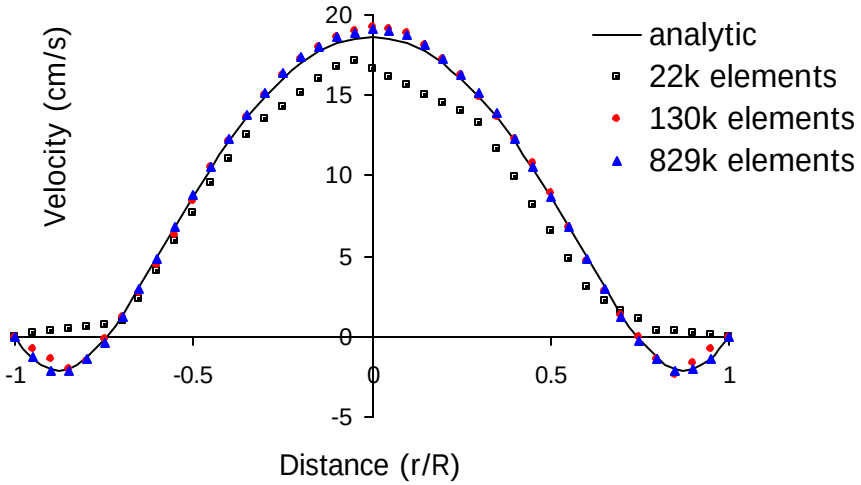


Figure 4.8: Convergence of velocity profile at outlet of cylinder (for t/T=0.625). As the mesh density is increased, the numerical solution approaches the Womersley analytic solution.

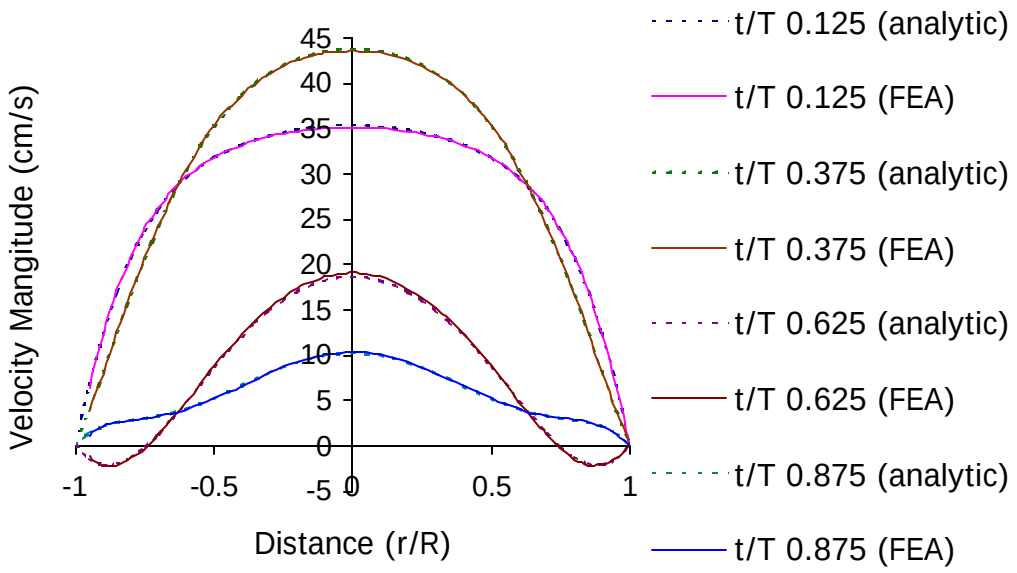


Figure 4.9: Velocity profile at outlet for four time points. The chart shows the numerical solution agrees very well with the analytic solution throughout the periodic cycle (829k element mesh).

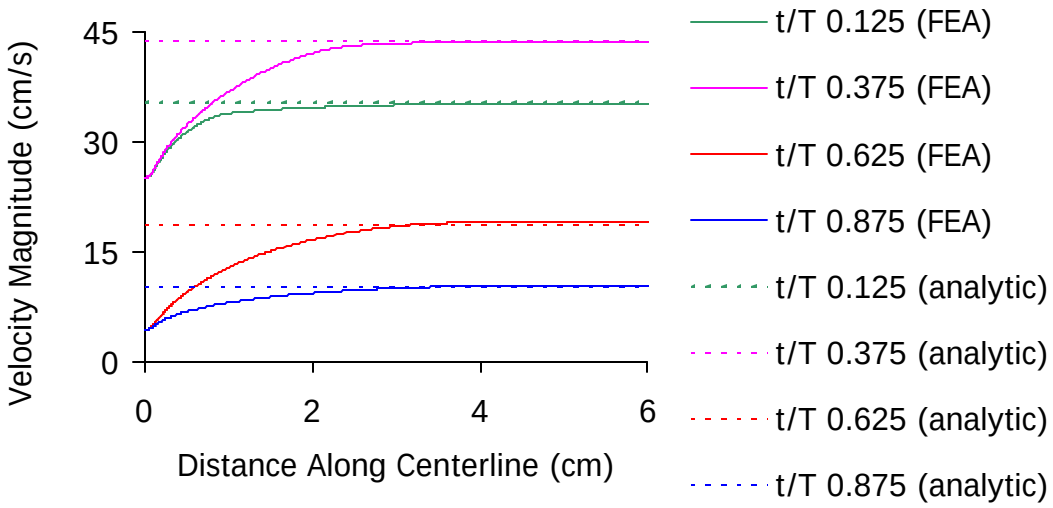


Figure 4.10: Velocity magnitude along cylinder axis. The figure shows the initial pulsatile plug flow develops into fully developed flow approximately 3 cm from the cylinder inlet (829k element mesh). Note that the diameter of the cylinder was 0.4 cm.

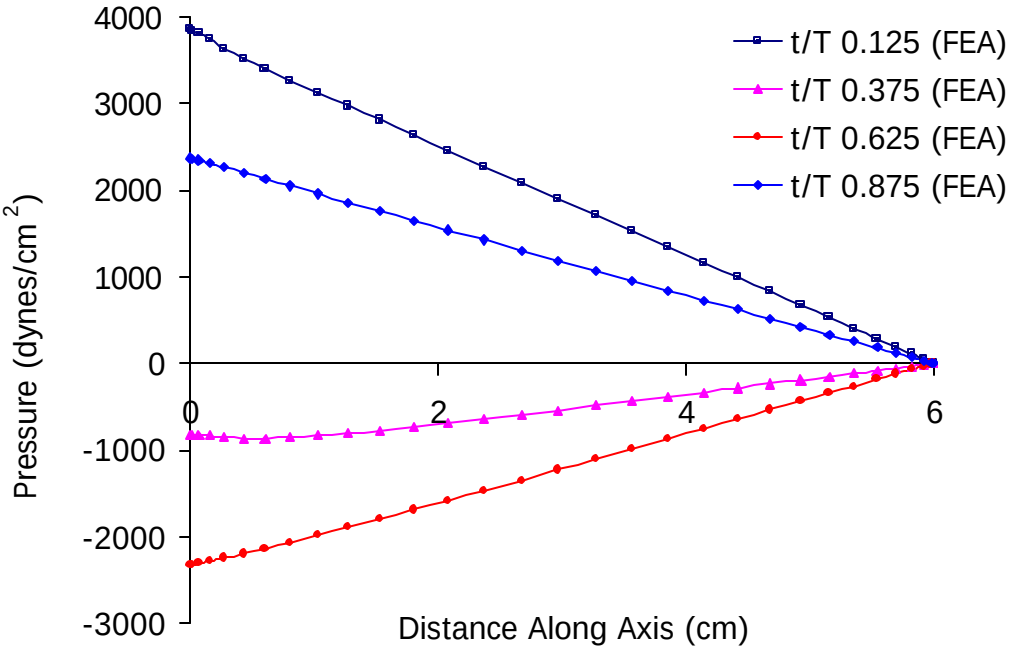


Figure 4.11: Pressure along cylinder axis (829k element mesh). A key assumption in deriving the Womersley analytic theory for pulsatile flow in a rigid axisymmetric vessel is that the pressure varies linearly along the length of the vessel.

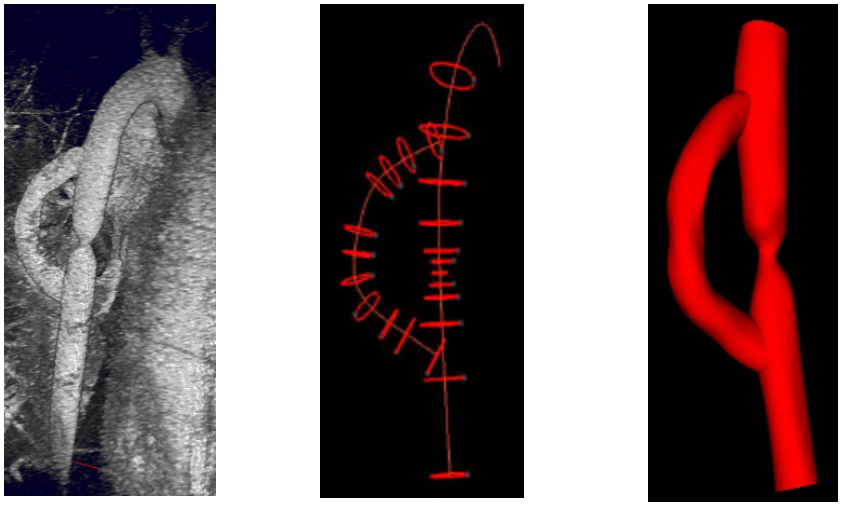
4.2.3 *In vivo* validation of volumetric flows in arterial bypass grafts

While *in vitro* experiments are useful, the medical community also utilizes animal studies to achieve more realistic *in vivo* conditions without putting human subjects at risk. The real benefit of the animal studies in the present context is the ability to design experiments that facilitate validation of the methods used in simulation-based medical planning. The **ASPIRE**[2] system was utilized by Ku [56] to perform preliminary *in vivo* validation studies of blood flow in arterial bypass grafts. Thoraco-thoraco aortic bypass procedures were performed on eight pigs each with an artificially created aortic stenosis. The results of numerical simulation were then compared with the experimentally determined volumetric flow distributions between the native aorta and bypass graft. Figure 4.12 shows MRA data and a solid model constructed (using the methods from

section 3.5) from the imaging data collected for one of the pigs. Figure 4.13 shows an example of a mesh and simulation results for one of the pigs. Mesh convergence studies and sensitivity studies on image segmentation results are also detailed in [56].

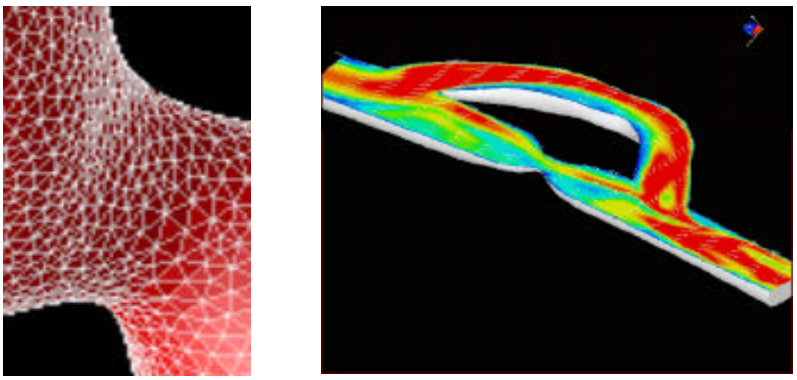
4.2.4 Importance of inlet boundary conditions

In addition to studying *in vivo* volumetric flow distributions, the animal study described in the previous section provided useful data to investigate the impact of inlet boundary conditions on the velocity patterns obtained from simulation. Much of the work to date has focused on using idealized velocity patterns (plug-like or Womersley profiles) to prescribe volumetric flow rates at the inlet of simulation domains (e.g. [5], [56]). As described in section 3.8, PCMRI data provides temporally and spatially resolved velocity patterns. A numerical study was performed to assess the impact on the resulting flow distributions and velocity profiles when experimentally determined velocity patterns were prescribed at the inlet of the pig model compared to idealized velocity patterns with an identical volumetric flow rate [57]. The study concluded that the volumetric flow



(a) (b) (c)

Figure 4.12: Solid model construction of a bypass model for a pig with an artificially created stenosis. Figure (a) shows the image data, (b) shows the paths along which selected level set curves have been extracted, and (c) shows the resulting solid model after the joining of the bypass and aorta branches.



(a) (b)

Figure 4.13: Mesh and simulation results of a pig thoraco-thoraco aortic bypass. Figure (a) shows a close up of the surface mesh near the stenosis, while (b) shows the magnitude of the flow velocity (blue low, red high) in the aorta and bypass.

distribution between the aorta and the bypass was independent of the exact velocity pattern at the inlet, but that the resulting velocity patterns varied significantly (particularly in the flow domain closer to the inlet boundary conditions) depending on the inflow boundary conditions used. Figure 4.14 shows representative results from this study indicating the observed differences in the flow patterns. Since velocity patterns and quantities derived from velocity (i.e. wall shear stress) impact the long-term successfulness of a given surgical intervention, this work motivates utilizing experimentally determined velocity patterns for boundary conditions when possible.

4.3 AORTO-FEMORAL BYPASS PLANNING

Once a surgeon has decided that a patient requires direct surgical revascularization and elects to perform an Aorto-Femoral Bypass (AFB), three major design decisions are made to plan the procedure. First, the surgeon needs to select the type (e.g. material, method of fabrication, etc.) of graft and the appropriate size (e.g. standard sizes of bifurcated Dacron grafts include 14 mm $\times$ 7 mm, 16 mm $\times$ 8 mm, 18 mm $\times$ 9 mm, and 20 mm $\times$ 10 mm). Inappropriate sizing can lead to inadequate restoration of flow or graft

limb occlusion. A second decision is whether to use an end-to-end or an end-to-side proximal anastomosis as discussed in section 3.2.1. Finally, the location, angle, and length of each distal anastomosis must be determined.

The following two case studies demonstrate the application and limitations of the **ASPIRE**[2] system for planning an AFB procedure. The first case involves a 67 year-old female patient while the second case involves a 55 year-old male patient.

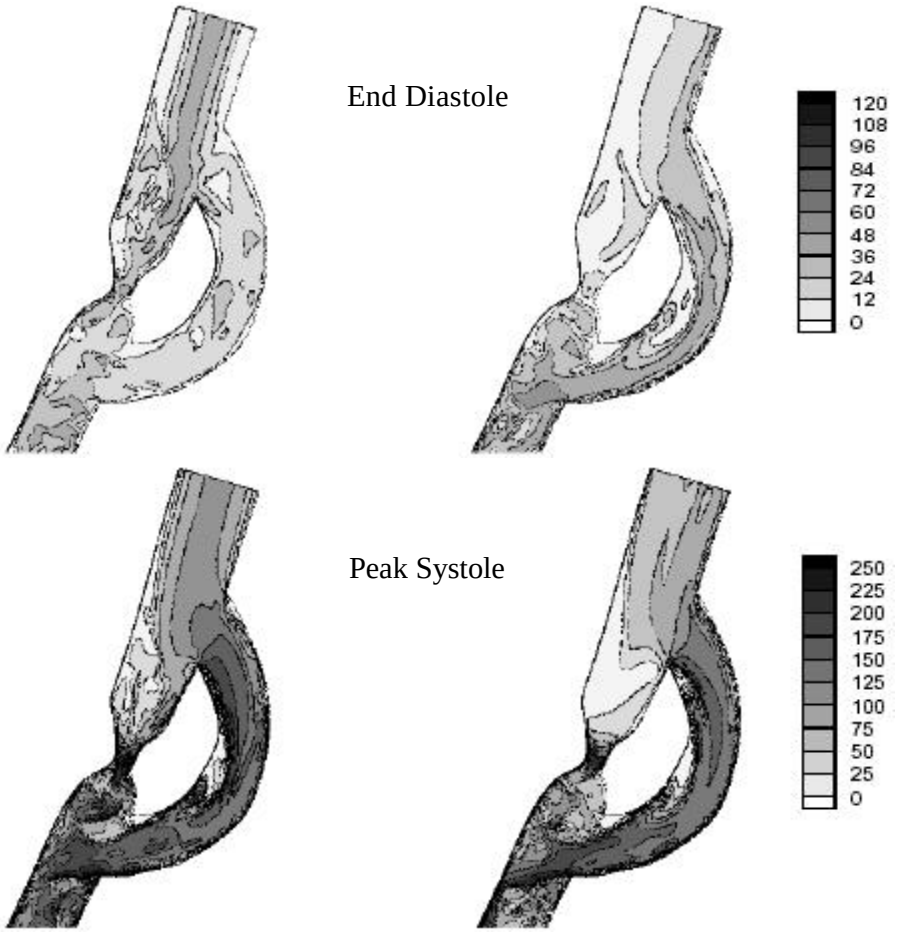


Figure 4.14: Velocity magnitude in pig bypass transverse cross-section (from [57]). First, contours of velocity magnitude (in mm/s) along a transverse cross-section at end diastole, for the Womersley profile (left) and the three-component profile (right) are shown. Second, contours of velocity magnitude along a transverse cross-section, at peak systole, for the Womersley profile (left) and the three-component profile (right) are shown.

4.3.1 Patient #1

4.3.1.1 Case history

A 67 year-old female with rest pain and calf claudication at 20 feet with a past medical history significant for peripheral vascular disease (ankle brachial index 0.50 bilaterally), chronic obstructive pulmonary disease, nicotine abuse, and cerebral vascular occlusive disease (60% carotid stenosis) was studied. Noninvasive imaging was consistent with atherosclerotic occlusive disease of the iliac and femoral arteries bilaterally.

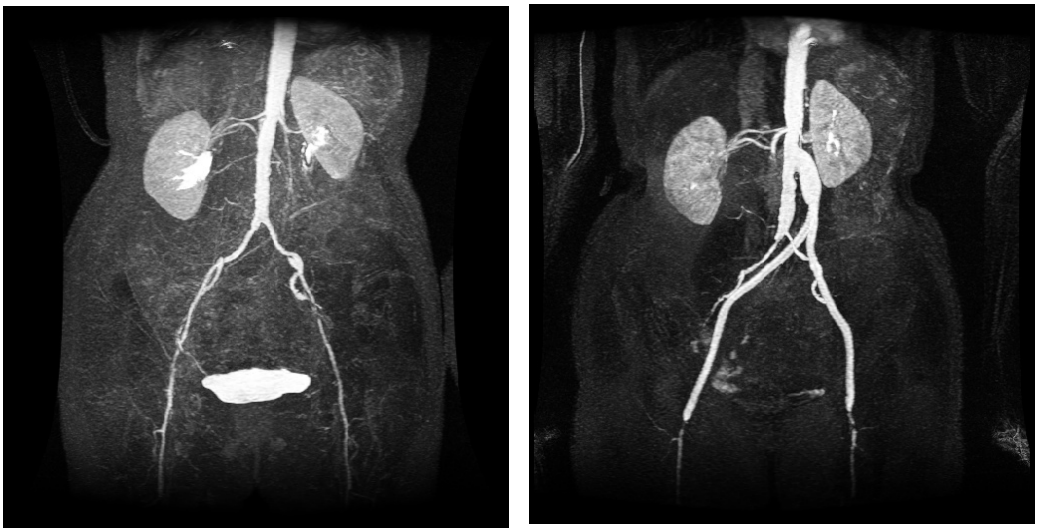
Imaging data was acquired preoperatively (Figure 4.15a) to obtain geometric information (using MRA) and quantify volumetric flow (using PCMRI) in the aorto-iliac-femoral system. Approximately two weeks after the acquisition of the preoperative data the patient underwent an end-to-side aorto-femoral reconstruction. A Dacron bypass graft with nominal dimensions of 14 mm x 7 mm was used.

Postoperative image data was acquired approximately 6 months after the operation (Figure 4.15b). The patient at that time was symptom free (i.e. felt better than before the operation), although diagnostic imaging and testing indicated a progression of the disease. There was mild narrowing of the profunda femoris arteries distally and the external iliac arteries that were bypassed had occluded completely.

4.3.1.2 Preoperative model construction

Prior to the patient's surgery, a preoperative geometric model shown in Figure 4.16a was constructed from the patient's volumetric MRA dataset. Partially due to inconsistencies between this patient's MRA and PCMRI data, the problems with gradwarp detailed in section 4.2.1 were discovered. Once General Electric provided software to correct gradwarp effects in the slice plane direction, a new preoperative model was constructed (several months after the patient's operation) and that model is shown in Figure 4.17a.

Constructing geometric models for surgical patients presents significant challenges not found in volunteer subjects [11] and artificially created disease in otherwise healthy pigs



(a) (b)

Figure 4.15: MIP of preoperative and postoperative MRA for patient #1. Figure (a) shows the extent of the occlusive disease preoperatively (notice the tight stenosis in the patient's left external iliac), while figure (b) shows the bypass graft and the occlusion of the external iliac arteries postoperatively.

[56]. In this particular case, the combination of a female patient with significant occlusive disease caused many of the vessels of interest to be small in relation to the resolution of the volumetric image data (in certain cases only a few image pixels were seen across the diameter of the vessel). The image resolution and diffuse vascular disease led to three specific geometric modeling challenges discussed below.

The first geometrically challenging region, shown in Figure 4.18a, was proximal to the renal vessels in the abdominal aorta. It was initially unclear if the celiac trunk, superior mesenteric artery, or both were patent. While the image data showed what appears to be only a single vessel branching off of the aorta, it seemed plausible that two vessels could branch off very close to each other and blur together within the limitations of the image resolution. After a consultation with a radiologist, however, it was determined that only the superior mesenteric artery was patent and it branched from the aorta in the region shown in Figure 4.18a.

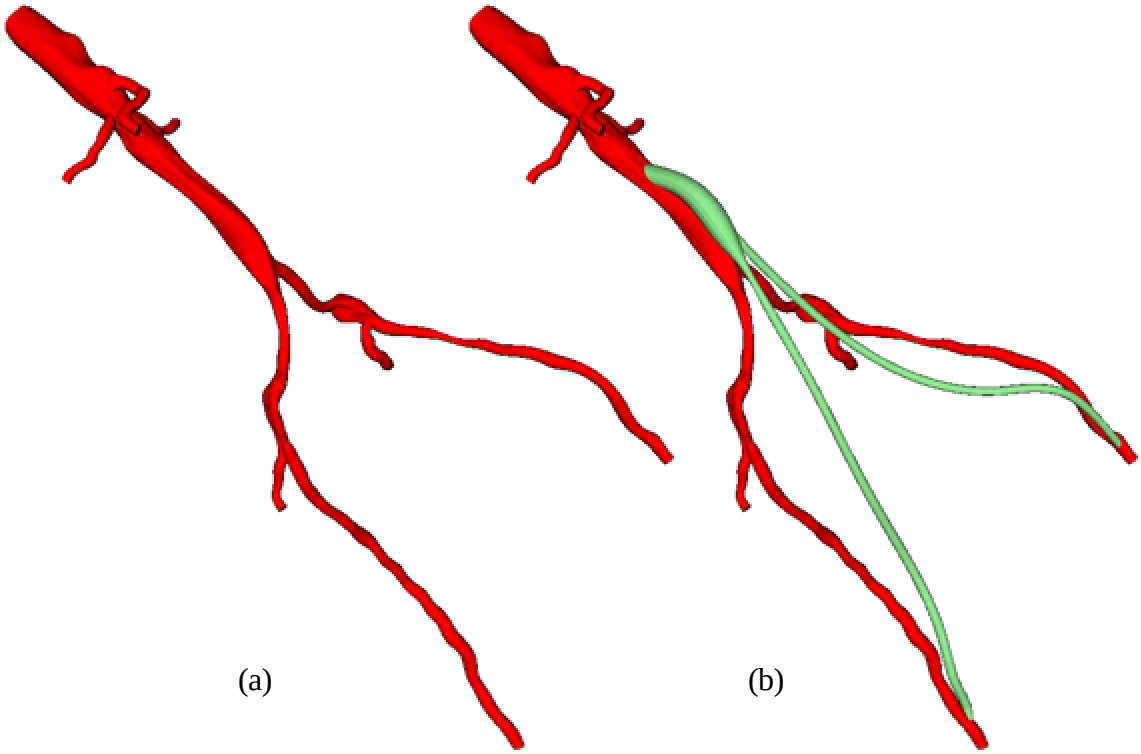


Figure 4.16: Original operative plan for patient #1. The preoperative model (a) and operative plan (b) was constructed prior to the patient's surgery. However, due to the gradwarp-related issues discussed in section 4.2.1, this model was not used for simulation.

A second challenging area occurred near the branch point of the left internal iliac artery off of the left common iliac artery (see Figure 4.18b). Several different anatomical possibilities seem to exist to explain the imaging data shown. For example, this area could be an aneurysm (with flow recirculation causing a drop in signal intensity). It also could be a stenosis of the external iliac artery, with the left internal and external iliac arteries being "twisted" around each making them appear as seen in the figure. Upon consultation with a radiologist and the attending vascular surgeon, it is believed that this region represents an "intimal flap" causing limited flow in the "hole" of the "donut." Given the difficulty of modeling this region using the lofting techniques described in section 3.5.4 and the fact that this region is removed from the areas of principal interest, it was approximated as a mild aneurysm in the model shown in Figure 4.17a.

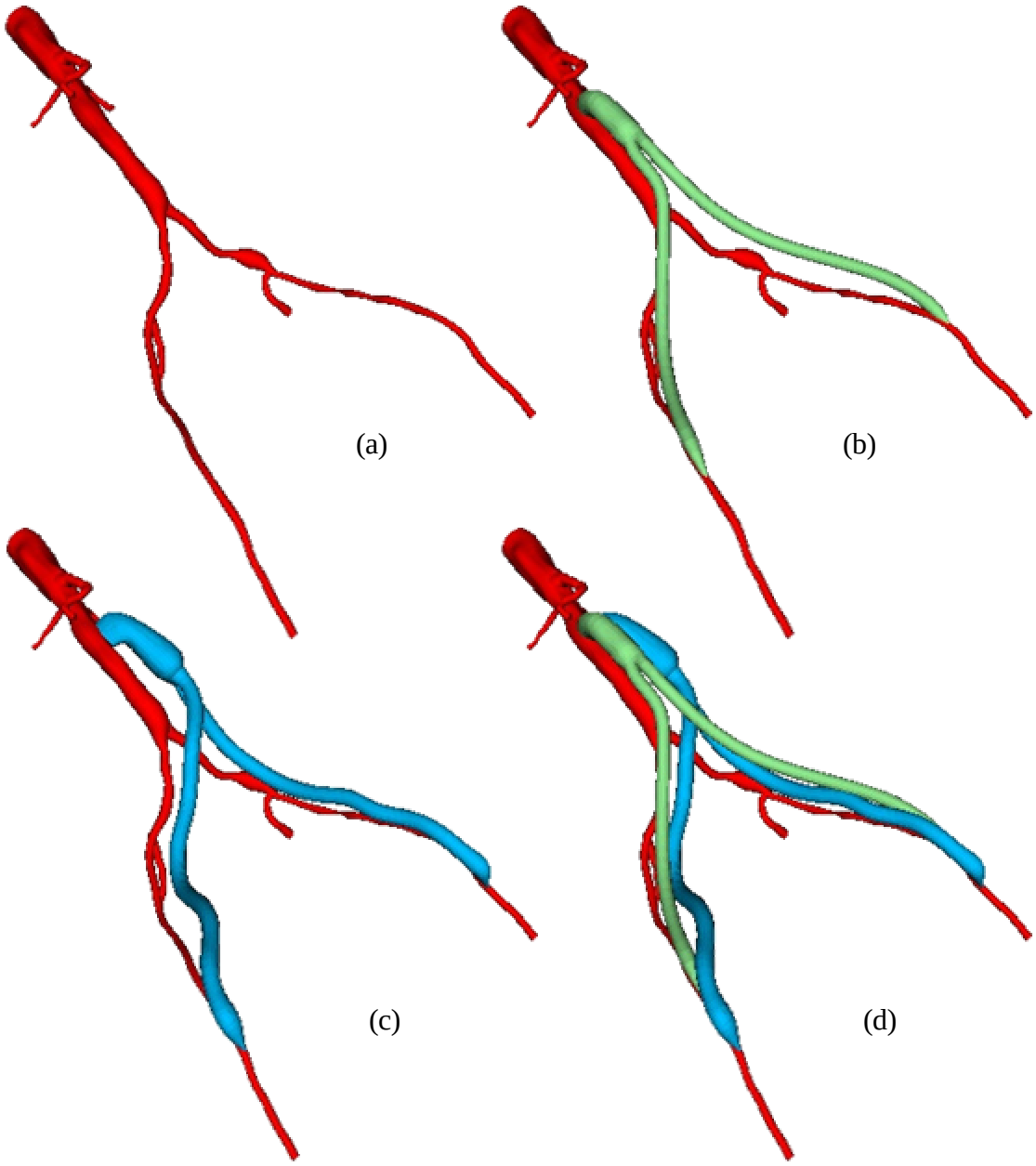


Figure 4.17: Two bypass plans for patient #1. Figure (a) shows the preoperative model constructed prior to surgery. Figure (b) shows the plan created by the vascular surgeon, and (c) shows the plan created using the postoperative MRA dataset as a guide. Figure (d) shows the two bypass plans displayed together to highlight the geometric differences in the two plans.

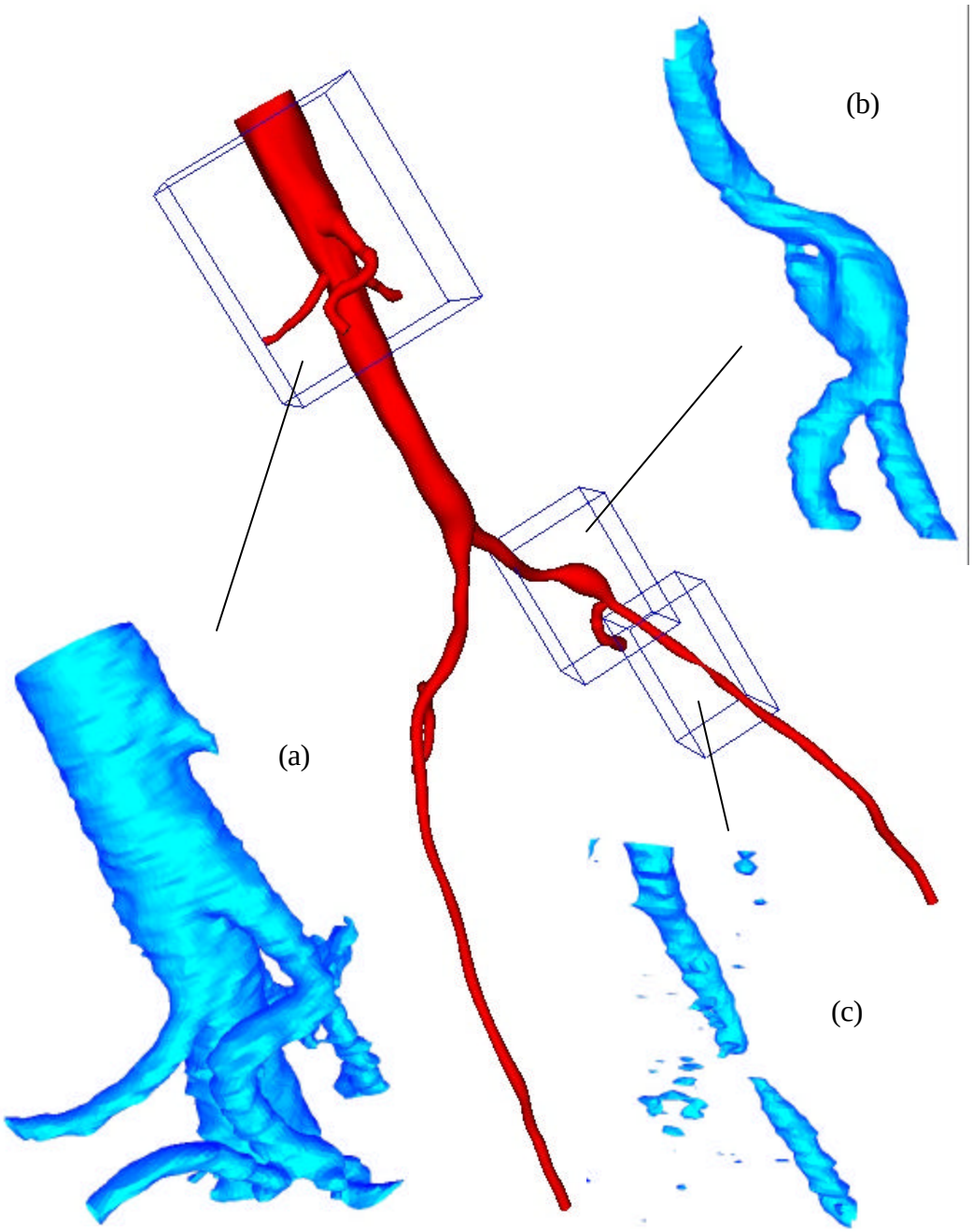


Figure 4.18: Anatomic variability caused by vascular disease. The preoperative solid model is shown (in red) with three views created from the MRA data using thresholding. Figure (a) shows the celiac trunk is not patent due to advanced vascular disease. A dissection near the branch of the left internal iliac artery is shown in (b). Figure (c) shows a tight stenosis that causes the vessel to appear disjoint due to limitations in the image resolution and the threshold value selected.

Finally, the imaging data made it difficult to ascertain the level of stenosis and patency of the left external iliac artery. In the opinion of the surgeon, the external left iliac artery was severely occluded but patent. He estimated a 90% stenosis based on other diagnostic imaging data, but due to meshing considerations this was modeled as a 70% stenosis. The enlargement of the stenosis stems from the use of isotropic mesh generation and the need to guarantee that at least several finite-elements existed across the diameter of vessel for blood flow simulations. The use of local mesh refinement in the neighborhood of the stenosis will in future studies enable modeling of a higher grade stenosis without significantly increasing the overall mesh density.

Table 4.1 shows the number and type of segmentations used to create the preoperative

Table 4.1: Segmentations Used to Create Preoperative Model for Patient #1

Vessel	**Aorta***	**Left Iliac+**	**SMA**	**Renal**		**Internal Iliac**		**TOTALS**
				Right	**Left**	**Right**	**Left**	
Total profiles	40	32	16	10	9	15	10	132
Level set profiles	40	31	16	10	9	15	10	131
Analytic: circle	0	1	0	0	0	0	0	1
Analytic: ellipse	0	0	0	0	0	0	0	0
Hand drawn profiles	0	0	0	0	0	0	0	0
Smoothed profiles	0	0	0	0	0	0	0	0
Replaced with circles	29	31	16	10	9	15	10	120
Pasted profiles	0	0	0	1	1	4	1	7
Scaled profiles	0	2	2	1	0	1	1	7

* the aorta group contains segmentations for the aorta, right common iliac, right external iliac, and right femoral arteries, + the left iliac group contains segmentations for the left common iliac, left external iliac, and left femoral arteries

model shown in Figure 4.17a. As indicated in the table, a mixture of level set and analytic profiles were used to create the preoperative model. In addition, due to the limits of the resolution of the MRA and the state of disease, many of the level set segmentations were fit with circles of equivalent area as shown in the table. It should be noted that no guidelines have been established regarding which segmentation method is appropriate under which circumstances, so this may lead to significant inter-operator variability in the resulting model constructed.

4.3.1.3 Operative planning

As mentioned in the previous subsection, there were problems with the MRA data that could not be corrected until several months after the patient's operation. The surgeon did, however, create an operative plan using **ASPIRE**^{2} based on the original MRA data prior to surgery and that plan is shown in Figure 4.16b. Once Beta gradwarp correction software was available from collaborators at General Electric [55], a second preoperative model was constructed as shown in Figure 4.17a. To prevent bias in his surgical plan created for this study, the surgeon created a new operative plan (Figure 4.17b) prior to reviewing the postoperative MRA data in detail. The surgical plan was constructed using three vessel paths (graft body, graft left limb, and graft right limb) and consisted of a combination of circles and ellipses lofted to create solids representing the flow domain of the bypass graft. The anastomoses were constructed using ellipses, transitioning to circular cross-sections of 14 mm diameter in the bypass body. The bypass limbs consisted of circular cross-sections of 7 mm diameter, with ellipses used to create the distal anastomoses.

In addition, for validation purposes a second plan was created to be similar to the actual operation. The objective in creating this plan was to use the knowledge and data gained from the postoperative MRA to create a plan as close as possible to the actual operation on the preoperative model. There is an important distinction between the procedure to be described below and simply creating a model from the postoperative data. In particular,

the external iliac arteries have completely occluded in the postoperative scan, making it impossible to build a model directly from the postoperative scan to study the flow in the external iliac arteries immediately following the surgical procedure. However, using the postoperative data the dilation of the graft and the proximal takeoff angle as a result of the actual surgery can be accurately modeled. To achieve an operative plan as similar as possible to the real surgery, the following steps were performed:

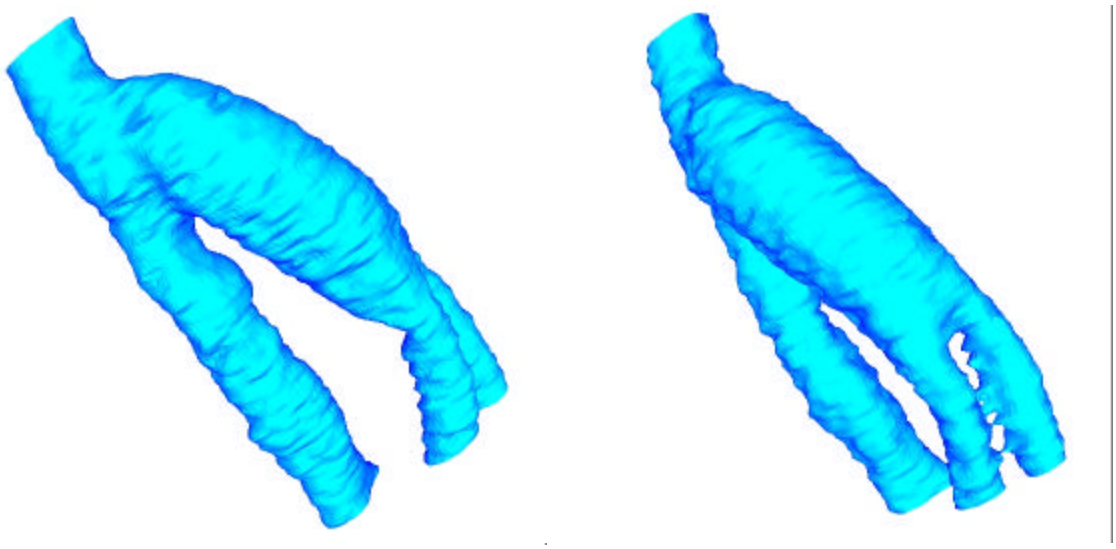
1. A preoperative model was constructed from the preoperative MRA data.

2. Three points were selected on the model (the location of aortic bifurcation, the left internal iliac bifurcation, and the right iliac bifurcation) and the preoperative model was rigidly rotated into the same space as the postoperative MRA data.

3. Paths for the left and right limbs of the bypass were automatically extracted from the postoperative MRA data.

4. A surgical plan was constructed using the paths from step 3 as a guide. Specifically, the paths from step 3 were used explicitly until approximately the S/I coordinate of the internal iliac bifurcations. The paths then maintain the approximate shape of the real bypass but were extended and stretched so that the bypass would intersect the preoperative model at the approximate S/I location of the distal anastomosis as seen in the postoperative MRA.

The result of this procedure is shown in Figure 4.17c. The two different surgical plans are displayed together in Figure 4.17d. Notice the difference between the two plans. In particular, the proximal anastomosis in the surgeon's plan takes off predominantly in the anterior direction while the actual anastomosis takes off between the anterior and patient-left directions. The geometry of the actual anastomosis from the postoperative MRA data is shown in Figure 4.19. The dimensions of the bypass graft in the postoperative MRA data were approximately 18 mm $\times$ 8.5 mm, reflecting an effective dilation of 29% and 21% of the graft over the nominal dimensions of 14 mm $\times$ 7 mm.



(a) (b)

Figure 4.19: Views of the proximal anastomosis from the postoperative MRA dataset. An isosurface of the postoperative MRA dataset in the neighborhood of the proximal anastomosis is shown from two different angles.

4.3.1.4 Processing PCMRI data

PCMRI data was acquired in this case study to quantify the volumetric flow through several arteries of interest. Many previous studies have focused mainly on quantifying volumetric flow in the supra-celiac and infra-renal aorta in healthy subjects (e.g. [58,59]), partially due to the large size of the aorta in relation to the in-plane image resolution. Simulation-based medical planning, however, also requires the quantification of volumetric flow in small (relative to pixel dimension) and diseased vessels such as the common iliac arteries. In this and the following case study, the vessels were roughly divided into three categories: large vessels, medium-size vessels, and small vessels.

For large vessels (typically over 10 image pixels in diameter), the level set segmentation technique described in section 3.5.3 was used to determine the lumen boundary from the potential data. Figure 4.20 shows two representative frames of data from the supra-celiac aorta of patient #1. The in plane image resolution in this acquisition was 1.25 mm × 1.25 mm. This resulted in approximately 20 pixels across the lumen during peak systole in the

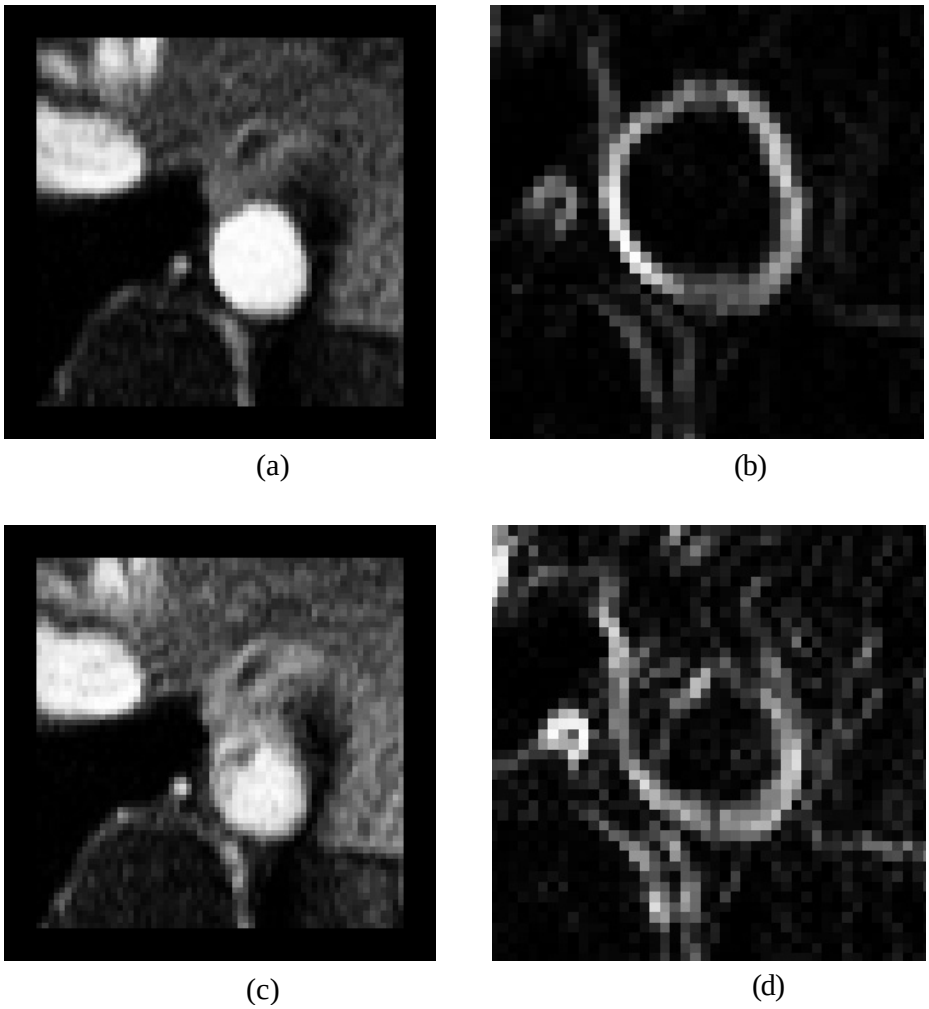
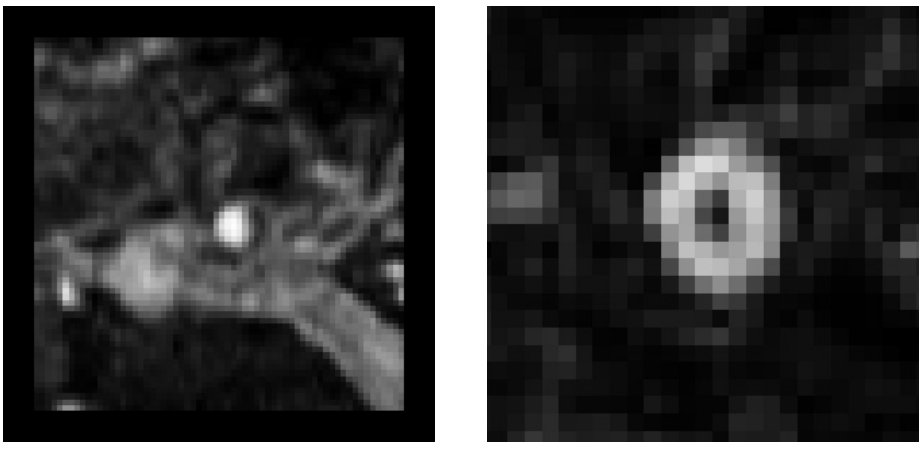


Figure 4.20: Two different time points of the supra-celiac aorta acquired using PCMRI. During peak systole (a), the supra-celiac aorta is approximately 20 pixels across and the lumen boundary is clearly visible in the zoomed view of the potential data shown in (b). However, during parts of diastole, the lumen boundary is not as clearly defined as seen in (c) and the corresponding zoomed potential data (d).

supra-celiac aorta (Figure 4.20a). For medium size vessels (roughly between 5 and 10 pixels across the vessel diameter) as seen in Figure 4.21, a threshold of the image intensity data was used to segment the lumen boundary. Finally, for small vessels with five or fewer pixels across the diameter (e.g. Figure 4.22) the individual pixels were selected and used in calculating the volumetric flow rate through the artery. This is done interactively, where the user can select an individual pixel in the image data and a volumetric flow waveform for that pixel will be displayed. The user then decides if the waveform corresponds to what would be expected through the vessel of interest, and includes the pixel if desired. This technique introduces significant user-dependence into the volumetric flow calculation since the inclusion / exclusion of a single image pixel can significantly alter the mean volumetric flow through a small artery of interest.

It should be noted that even for major arteries with 20 image pixels across the diameter, the lumen boundary is not always well defined. For example, a selected image frame from diastole in the supra-celiac aorta (Figure 4.20c) shows the ambiguity that can occur. The potential data at this time point does not clearly define a lumen boundary. While

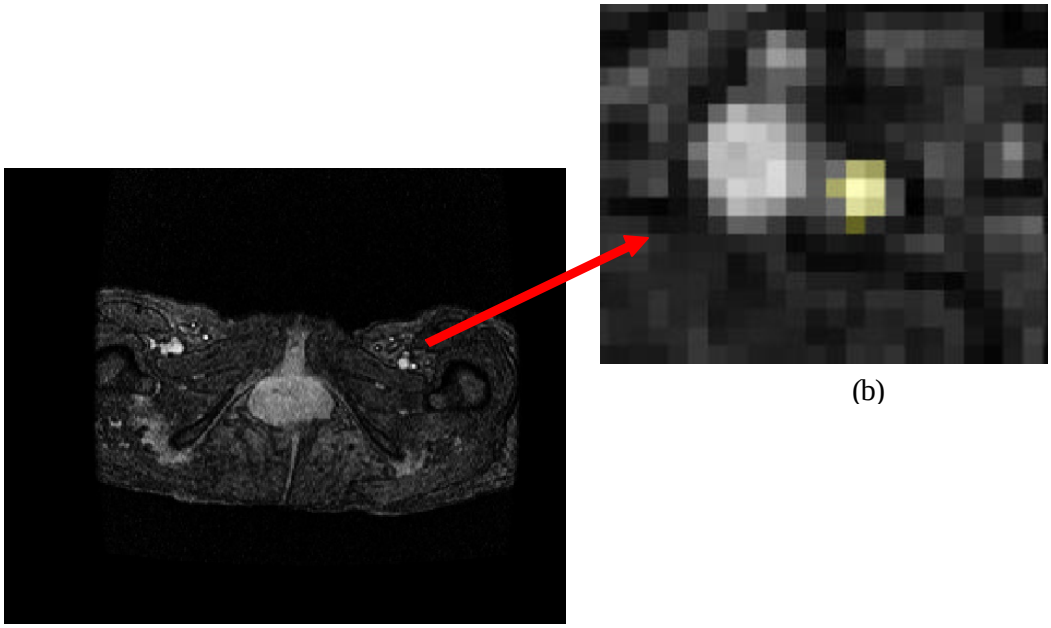


(a) (b)

Figure 4.21: Single time point for the right common iliac acquired using PCMRI. The vessel shown in (a) is approximately 5 pixels across, and the corresponding (zoomed) potential data is shown in (b). There is insufficient resolution to use the level set segmentation for vessels of this size, so a threshold segmentation of the intensity data is used instead.

ASPIRE[2] provides several options to create lumen boundaries (see Figure 3.20) that may be applicable to this type of situation, the predominant technique used here was to copy segmentations from previous frames in the cardiac cycle where the boundary was well defined. This assumes the vessel does not change significantly between time points in the PCMRI acquisition, which may be reasonable given the increased rigidity of diseased vessels such as the ones studied here.

For this case study, seven slices of PCMRI data were acquired preoperatively as shown in Figure 4.23. During acquisition the plane locations are selected to be perpendicular to a given vessel of interest. Only the through-plane component of velocity was acquired to



(a) (b)

Figure 4.22: Single time point showing a femoral artery acquired using PCMRI. For small vessels such as the femoral arteries (or heavily diseased vessels) less than 5 pixels in diameter, the user interactively selects the pixels defining the flow lumen using a combination of the intensity and velocity information. Figure (a) shows the entire slice of PCMRI data, while (b) shows a zoomed section around the femoral artery. The user clicks on each pixel and views the velocity information encoded during the acquisition and decides if the profile matches an expected pattern. If it does, the pixel is selected by the user for the volumetric flow rate calculation and is highlighted in yellow as seen in figure (b).

improve the temporal resolution of the data. Figure 4.24 through Figure 4.26 show the volumetric flow waveforms calculated from the preoperative PCMRI data using **ASPIRE**^{2}. As discussed in section 3.8, flow waveforms are often combined to calculate flow waveforms for smaller vessels of interest (e.g. internal iliac arteries). Figure 4.27 shows an example of the difficulties in combining flow waveforms. Assuming physiologic conditions stay constant between scans and neglecting any patent lumbar arteries, the flow in the aorta just above the aortic bifurcation should be equal to the sum of the flow in the right and left common iliac arteries. However, for the data acquired experimentally for this patient the flow waveforms are visibly different depending on how flow is calculated in the right common iliac artery as seen in Figure 4.27. Interestingly, in this particular example the mean flow for the two curves only differs by 1%, but this is not generally the case as both the waveforms and mean volumetric flow rates can vary.

As discussed in section 3.8.2, there are multiple methods to calculate volumetric flow rates from PCMRI data. Table 4.2 summarizes a comparison between a custom software program using the methods described in [47] and the implementation in **ASPIRE**^{2} discussed in section 3.8.2. As discussed elsewhere [47], a common post-processing technique used to eliminate the effects of eddy currents leading to erroneous velocity in otherwise stationary tissue is known as "baseline correction." As seen in Table 4.2, differences in lumen segmentation between the two software programs for this case study can have a larger effect on mean volumetric flow rate than baseline correction. For the purposes of this work, baseline correction was not used. Finally, Table 4.3 shows the mean cross-sectional lumen area for each vessel of interest in the PCMRI data.

Postoperative PCMRI data was acquired six months after the operation at the same time as the postoperative MRA data shown in Figure 4.15b. The segmentation and processing issues are similar to those described above. Table 4.4 summarizes the mean volumetric flow rates calculated using **ASPIRE**^{2} from the postoperative PCMRI data.

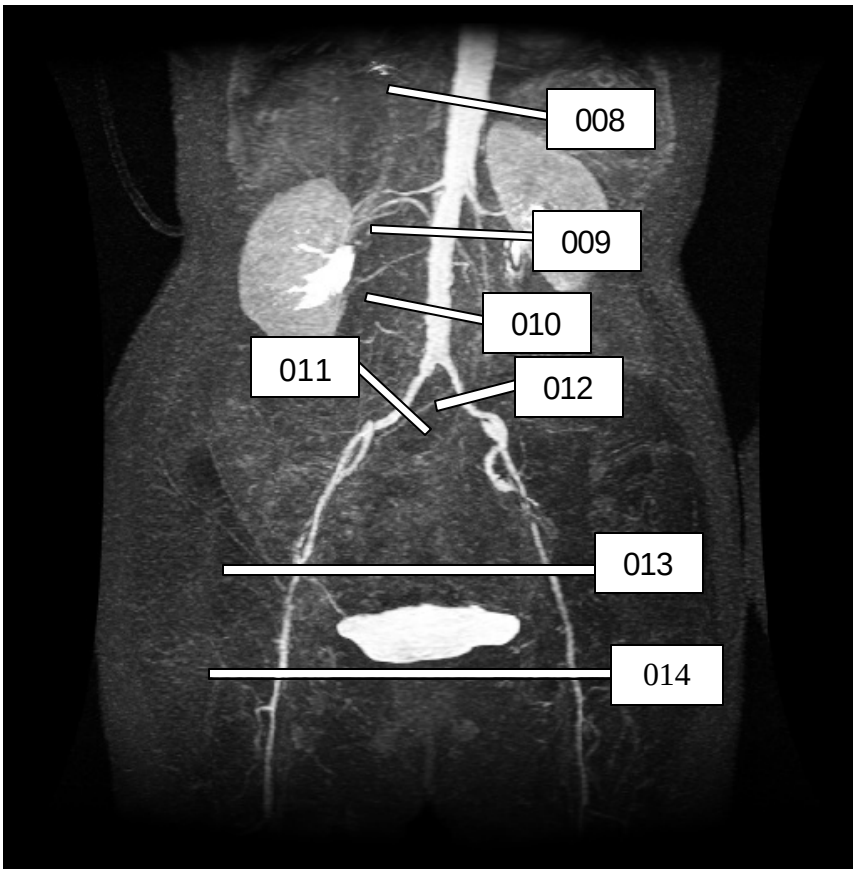


Figure 4.23: PCMRI slice plane acquisition locations for patient #1. Seven planes of through-plane velocity information were acquired at the locations shown in the figure.

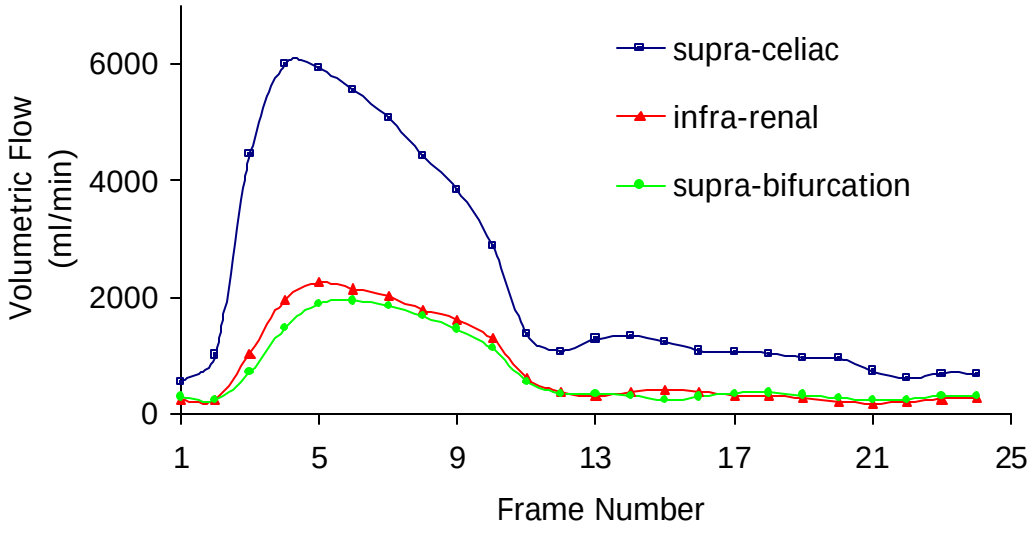


Figure 4.24: Volumetric flow waveforms in the abdominal aorta. The supra-celiac aorta, infra-renal aorta, and aorta just before the aortic bifurcation are shown.

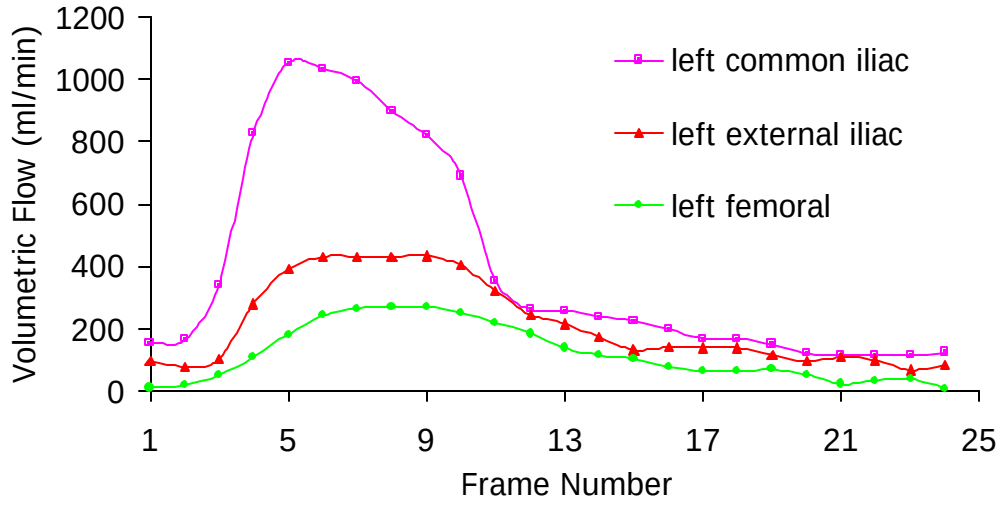


Figure 4.25: Volumetric flow waveforms for the patient's left side. The left common iliac artery, left external iliac artery, and left femoral artery are shown.

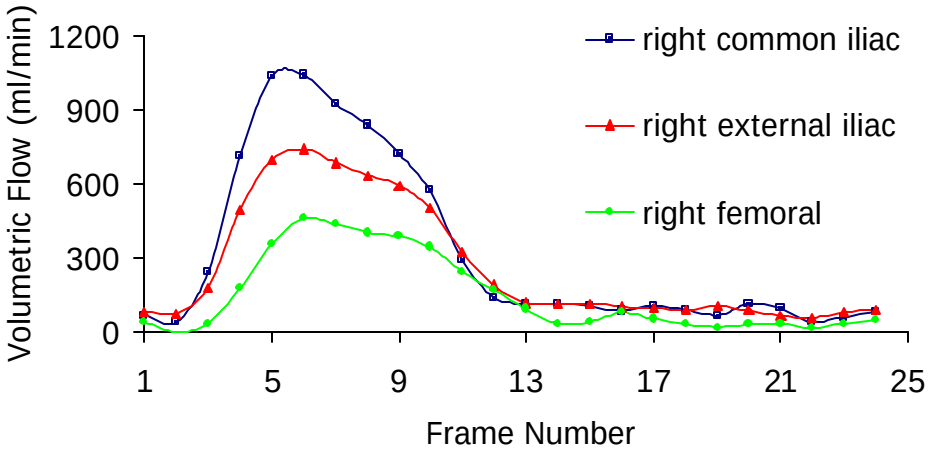


Figure 4.26: Volumetric flow waveforms for the patient's right side. The right common iliac artery, right external iliac artery, and right femoral artery are shown.

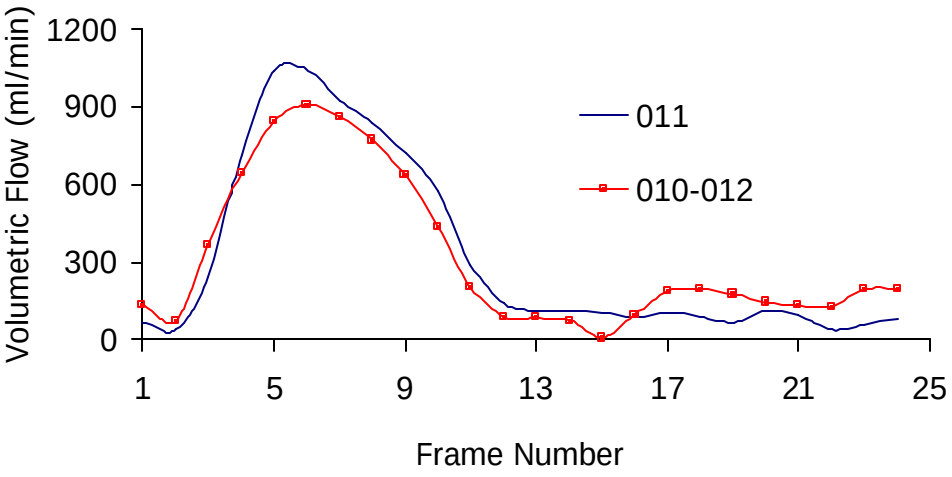


Figure 4.27: Volumetric flow waveform in right common iliac artery. The chart shows the waveform calculated two different ways. First, the chart shows series 011 (see Figure 4.23) that was acquired in the right common iliac artery. Assuming no physiologic changes between acquisitions and neglecting any patent lumbar arteries, conservation of mass implies that subtracting the left common iliac artery (series 012) flow waveform from the supra-aortic bifurcation flow waveform (series 010) should yield the right common iliac artery flow waveform. While the mean flow rate for the two curves shown in the figure differ by only 1%, the flow waveforms are noticeably different.

Table 4.2: Comparison of Methods to Calculate Volumetric Flow

Series*	Location	Mean Volumetric Flow (ml/min)				Difference++ Between ASPIRE[2] and vcalc+ (%)
		vcalc+			ASPIRE[2]	
		Baseline Correction			No Baseline Correction	
		None	Linear	Change (%)		
8	Supra-celiac aorta	2339.60	2270.82	-2.94	2246.64	-1.06
9	Infra-renal aorta	882.30	966.86	9.58	795.77	-17.70
10	Supra-bifurcation	858.36	866.34	0.93	715.81	-17.38
11	Right common iliac	327.56	411.39	25.59	319.94	-22.23
12	Left common iliac	359.47	394.00	9.61	399.83	1.48
13	Right infra internal iliac	290.37	265.02	-8.73	262.97	-0.78
13	Left infra internal iliac	208.44	199.28	-4.39	213.89	7.33
14	Right femoral	124.88	136.16	9.03	148.42	9.00
14	Left femoral	95.17	116.47	22.38	118.80	2.00

* refers to Figure 4.23, + custom software from [47], ++ difference between **ASPIRE[2]** mean flow rates with no baseline correction and those calculated by vcalc using linear baseline correction

Table 4.3: Vessel Cross-sectional Area and Diameter from PCMRI Data

Series*	Location	Mean Area (mm^2)	Effective Diameter (mm)
8	Supra-celiac aorta	417.97	23.07
9	Infra-renal aorta	110.30	11.85
10	Supra-bifurcation	109.00	11.78
11	Right common iliac	39.30	7.07
12	Left common iliac	45.10	7.58
13	Right infra internal iliac	29.70	6.15
13	Left infra internal iliac	20.30	5.08
14	Right femoral	20.30	5.08
14	Left femoral	12.50	3.99

* refers to Figure 4.23

Table 4.4: Mean Volumetric Flow Rates from Postoperative PCMRI Data

Location	Mean Volumetric Flow Rate (ml/min)
Supra SMA	2171
Infra-renal supra bypass	492
Right common iliac	90
Left common iliac	113
Bypass body	230

4.3.1.5 Preparing for analysis

Based on previous experience with the software components used in the present work, it was known that meshes on the order of two million elements could be simulated and processed in a reasonable amount of time and effort with available computational resources. In addition, due to current memory limitations imposed by the flow solver the mesh could not exceed 4.5 million elements (the other components in the **ASPIRE**[2] system currently support meshes up to 6 million elements). With these limitations in mind, the target for the current case study was to create most of the meshes with approximately 2 million elements and a refined mesh consisting of 4 million elements for testing convergence issues. Four tetrahedral meshes were generated from the models shown in Figure 4.17 as summarized in Table 4.5. The table also summarizes the relevant geometric mesh quality statistics for the meshes indicating that the elements were of high geometric quality. A representative exterior surface mesh is shown in Figure 4.28. All of the meshes were generated on a standard PC (Pentium IV 1.7 GHz CPU with 3GB RDRAM) in less than 30 minutes of wall-clock time per mesh.

Table 4.5: Mesh Statistics for Patient #1

Model	Mesh Statistics		Mesh Quality Statistics*				
	Elements	Nodes	Dihedral Angles (%)		Aspect Ratios (%)		
			< 145°	<160°	< 6	< 10	< 20
Preoperative	2,651,625	510,416	99.98	100	96.95	99.81	100
Plan 1	1,474,844	294,573	99.97	100	96.16	99.75	100
Plan 2	1,827,475	361,378	99.97	100	96.33	99.77	100
Plan 2	4,099,441	791,577	99.98	100	96.96	99.82	100

* see section 2.2.3 for definitions of geometric mesh quality indicators

As discussed in section 3.9, rigid-wall and incompressible fluid assumptions were used in this work. The result of these assumptions is that instantaneous conservation of mass must be enforced at all inlet and outlets of the flow domain. As shown in Figure 4.29, however, the actual compliance of the vessel walls creates a phase lag in the volumetric flow waveform from the supra-celiac aorta to the femoral arteries. Aligning the normalized flow waveforms as seen in Figure 4.30 still does not satisfy the instantaneous conservation of mass requirement. Since the waveforms in Figure 4.30 have a similar characteristic shape, an average of the normalized profile was created as shown in Figure 4.31. This characteristic shape was then scaled by the desired inlet / outlet volumetric flow rates for the prescribed velocity boundary conditions. That is, an analytic Womersley profile for fully developed pulsatile flow in a rigid tube of effective diameter with a specified mean flow was generated for each boundary face given the flow waveform in Figure 4.31. Thus, as long as the mean volumetric flow rates prescribed at the various inlets / outlets satisfy conservation of mass, instantaneous conservation of mass will be achieved since the flow waveforms will differ only by the maximum amplitudes.

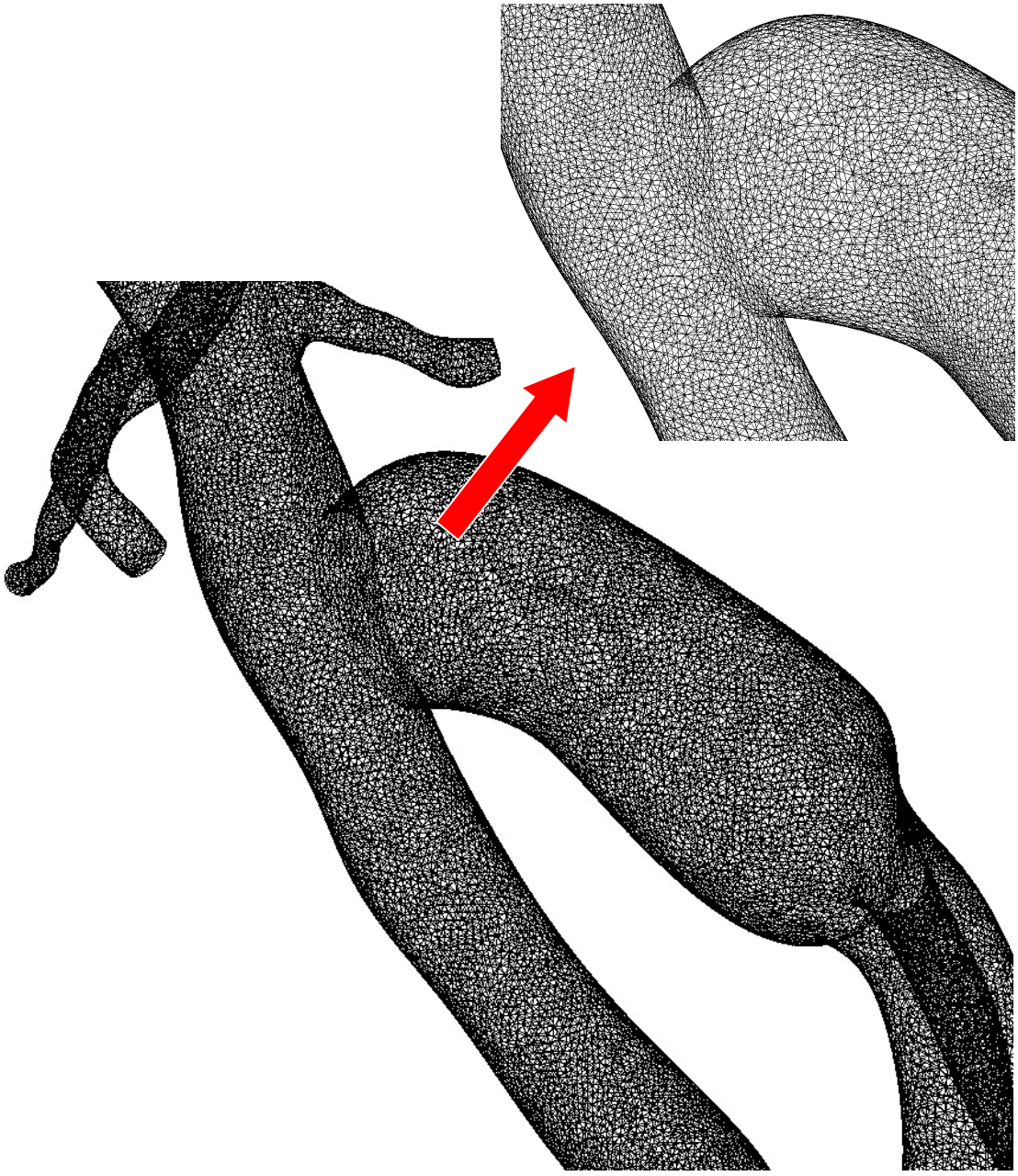


Figure 4.28: Exterior surface mesh for patient #1 plan #2. The mesh consists of 1.8 million tetrahedral elements.

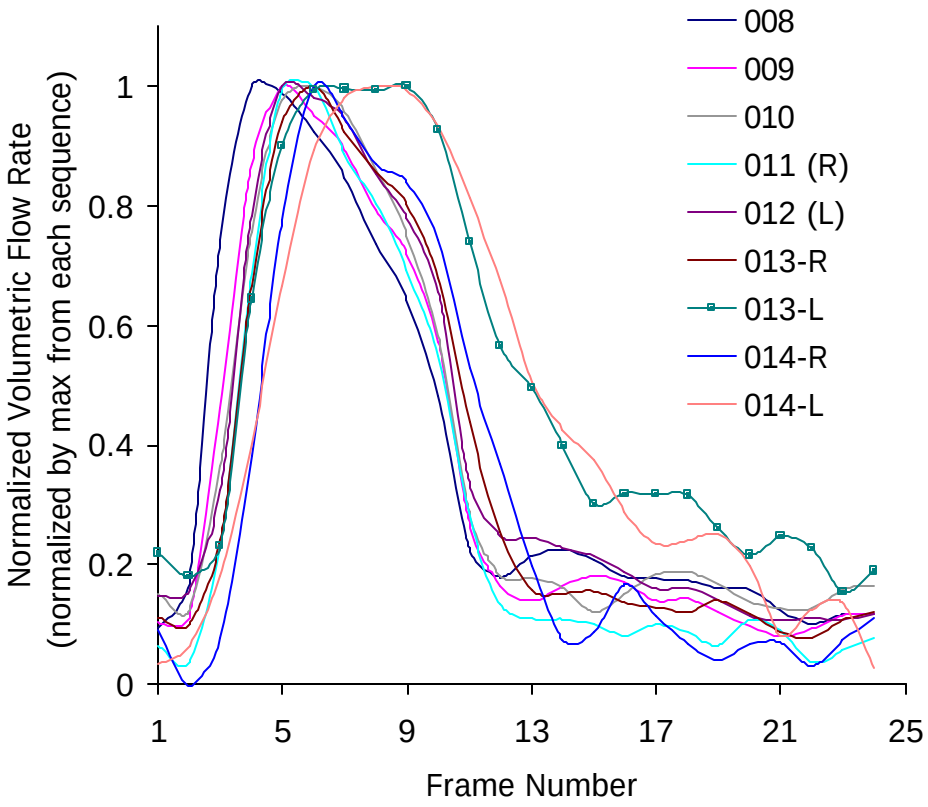


Figure 4.29: Normalized volumetric flow waveforms. The series numbers (i.e. 0xx) correspond to those shown in Figure 4.23. Each flow waveform was normalized by its maximum instantaneous volumetric flow rate.

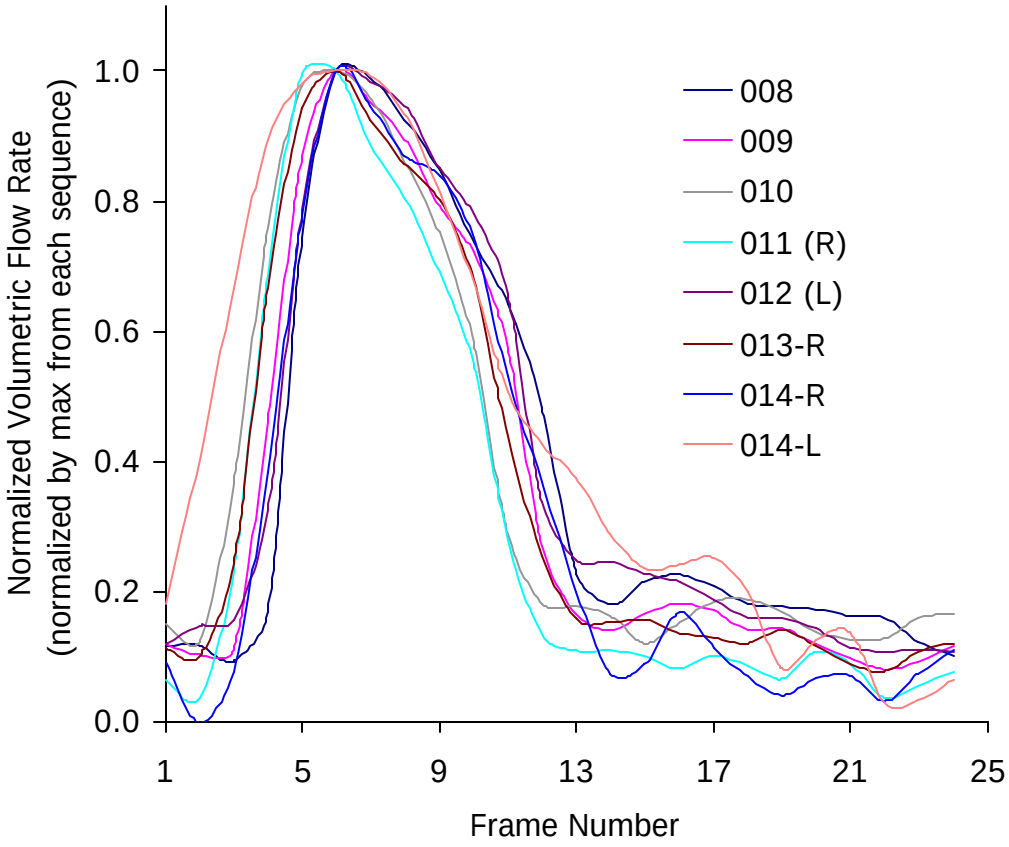


Figure 4.30: Aligned peaks for normalized flow waveforms. The curves from Figure 4.29 were shifted so that the maximum instantaneous volumetric flow rate occurred at the same time in each artery (curve 013-L was dropped due to differences in its shape). The motivation for this shift is the modeling assumptions of incompressibility and rigid walls which imply that the system has no capacitance.

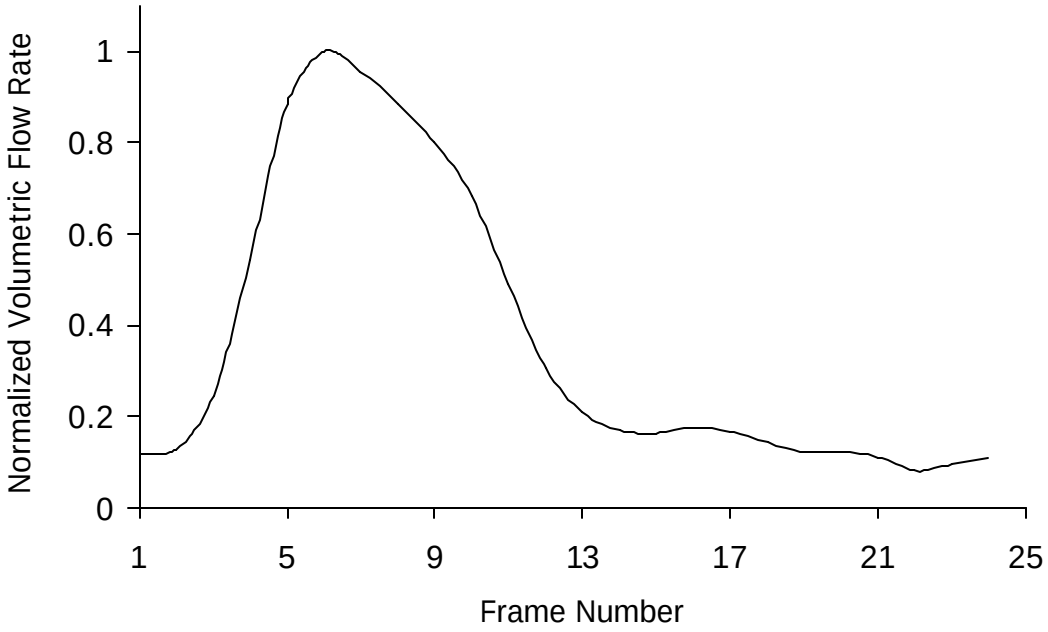


Figure 4.31: Average normalized aligned volumetric flow waveform. This curve represents the point-wise arithmetic average of the flow waveforms in Figure 4.30.

Figure 4.32 indicates the two sets of boundary conditions used in this case study. First, Figure 4.32a shows that prescribed velocities were used at the aorta inlet, superior mesenteric artery outlet, and renal artery outlets. Since during the imaging study only supra-celiac and infra-renal slices were acquired, a simple flow distribution was assumed between the superior mesenteric artery and renal arteries as shown in the figure. Zero pressure boundary conditions were applied at the outlets corresponding to the femoral and internal iliac arteries. This zero pressure boundary condition effectively sets the resistance of the downstream vascular bed of the internal iliac arteries to be the same as the femoral arteries (both are zero), which may not be an accurate assumption. For this reason, a second set of boundary conditions was also tested (shown in Figure 4.32b) where prescribed velocity boundary conditions were applied for the internal iliac arteries with flow determined from the postoperative PCMRI data. Table 4.6 summarizes the volumetric flow rates used for both sets of boundary conditions. In the case of the non-

circular inflow, the analytic solution was mapped onto the inlet using the same mapping technique described in section 3.8.3.

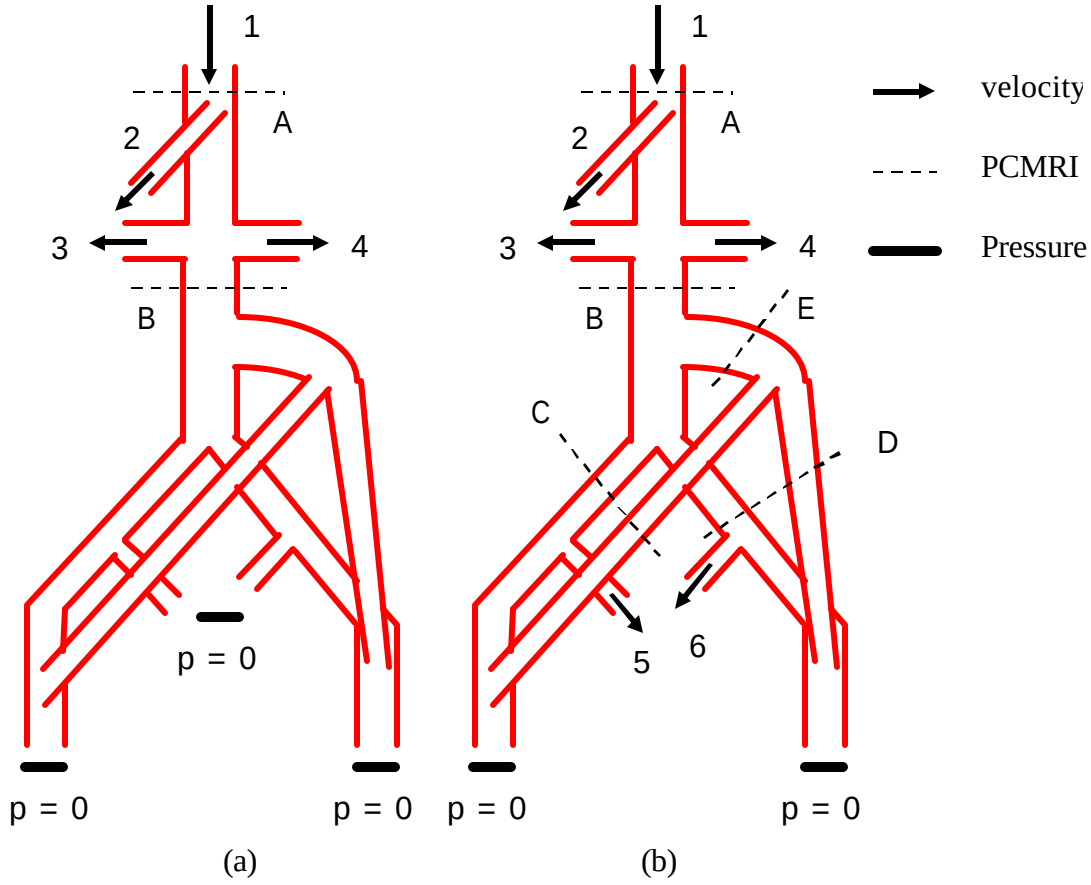


Figure 4.32: Boundary conditions used for patient #1 simulations. Two different sets of boundary conditions were examined. Figure (a) shows the boundary conditions where all of the outlets below the aortic bifurcation were set to zero pressure, while (b) has prescribed velocity boundary conditions at the outlets corresponding to the internal iliac arteries. Womersley analytic profiles were applied at all prescribed velocity boundaries (see section 3.8.4). The PCMRI slice plane locations shown correspond to the postoperatively acquired data, and it was this data that was used to prescribe volumetric flow rates. The assumed mean volumetric flow rate for the superior mesenteric artery was 0.5*(A-B) while the mean volumetric flow rate for each renal artery was assumed to be 0.25*(A-B).

Table 4.6: Prescribed Volumetric Flow Rates

Location*	Calculation	Mean Volumetric Flow Rate (ml/min)
1: Inflow	1.0 * A	2171
2: SMA	0.5 * (B - A)	839
3,4: Renals	0.25 * (B - A)	420
5: Right Iliac^{+}	(C/(C + D)) * (B - E)	116
6: Left Iliac^{+}	(D/(C + D)) * (B - E)	146

* see Figure 4.32, $^{+}$only for Figure 4.32b

4.3.1.6 Simulation results

The primary objective of a surgical intervention for aortoiliac occlusive disease is to restore adequate flow to the lower extremities. As can be observed from the data given in section 4.3.1.4, however, infra-renal flow postoperatively can at times be lower than the preoperative infra-renal flow at a resting state. This is due to normal physiologic variation (e.g. flow diversion due to digestion) and the fact that inadequate flow typically happens during an exercise-induced physiologic state (not at rest). Since many surgical patients spend a majority of their time in a sedentary state, simulations at rest may provide insight into disease progression. Mean volumetric flow through the different arteries of interest at rest is still a quantity of interest since it may provide insight into locations of low flow susceptible to thrombus formation. Table 4.7 summarizes the mean volumetric flow rates for the 1.8 million element mesh for the two different boundary conditions shown in Figure 4.32.

Both boundary conditions indicate extremely low (retrograde) flow in the external iliac arteries during resting conditions. Recall (see Figure 4.15b) that postoperatively the external iliac arteries occluded. The simulation results showing low flow in the external

iliac arteries (an order of magnitude lower than the preoperative flow rates) could explain the clinically observed occlusion.

Table 4.7: Mean Volumetric Flow Rate for Two Surgical Plans*

Location	BC: Figure 4.32a		BC: Figure 4.32b	
	Plan 1 (ml/min)	**Plan 2 (ml/min)**	**Plan 1 (ml/min)**	**Plan 2 (ml/min)**
Infra-Renal Aorta	492+	492+	492+	492+
Bypass Body	241	242	259	279
Bypass Left Limb	125	127	135	148
Bypass Right Limb	116	115	123	130
Distal Native Aorta	251	250	233	213
Right External Iliac	-5	-7	-2	-1
Left External Iliac	-6	-5	-5	-4
Right Femoral	111	108	121	129
Left Femoral	119	122	130	145
Right Internal Iliac	116+	116+	107	97
Left Internal Iliac	146+	146+	134	121

* 1.8 million element mesh, +explicitly prescribed by boundary conditions

It should be noted that the mean volumetric flow in the distal native aorta and the bypass body varied by less than 0.6% between the 1.8 million element and 4.1 million element mesh. While the refined mesh is not of sufficient resolution to explicitly declare a converged solution (with respect to volumetric flow distribution), the results are consistent with the 1.8 million element mesh being sufficiently refined.

Several additional simulations were performed for plan #2 to test the effects of key simulation parameters on the solution. Velocity patterns in the bypass body were studied

to examine the effect of time step size. Figure 4.33 shows an example of the through-plane velocity component (plotted as an oriented elevation plot) for a cut plane through the bypass body. Three different time step sizes were tested: 200, 400, and 800 steps per cycle. The results were then visually compared over one cardiac cycle. All three time step sizes led to nearly identical volumetric flow rates, but visibly different velocity patterns as shown in Figure 4.34. The solution with 200 steps per cycle was smoother than the solutions with a larger number of steps per cycle. While 800 steps per cycle appears to better capture the velocity field extrema during the cardiac cycle, the 400 steps per cycle appears to capture the main features of the velocity patterns at half the computational cost.

Periodicity of the solution was also examined in the bypass body as shown in Figure 4.35. The simulations were run for a total of six cardiac cycles. With periodic boundary conditions and unknown initial conditions as is the case in these simulations, it is assumed that the initial conditions will "pollute" the solution for several cardiac cycles. This is observed in Figure 4.35 as the solution reaches a periodic solution after the 4th period.

On the basis of these results, it was assumed that using 400 steps per cycle and running the simulations for at least four cardiac cycles is an appropriate balance of solution accuracy with computational cost. It is noted that periodicity and convergence of the solution in one region of the model (such as the bypass body) does not guarantee periodicity and convergence elsewhere. Derived quantities (e.g. wall shear stress), regions with complex flow, and areas of geometric complexity (e.g. a stenosis) may require higher mesh densities, more steps per cardiac cycle, and more periods to reduce effects of initial conditions. In addition, it is possible for the flow to be non-periodic in certain regions even with periodic boundary conditions.

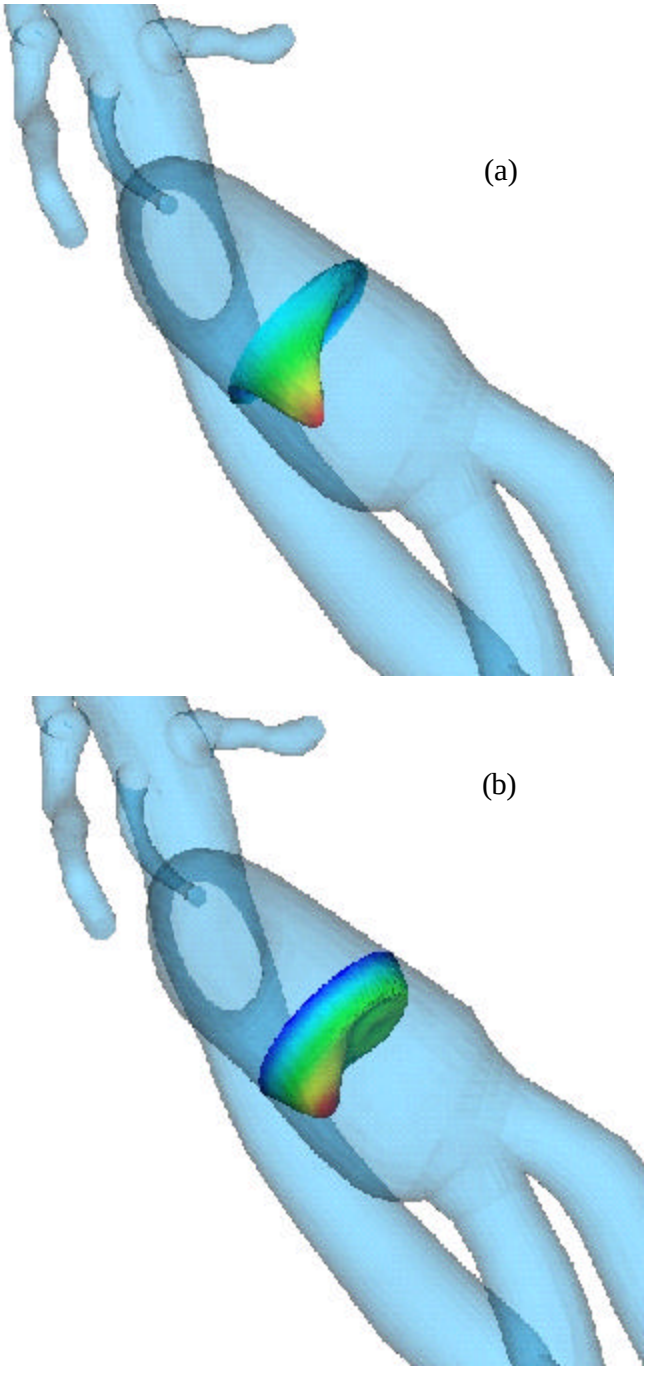


Figure 4.33: Through-plane velocity in bypass body from simulation for plan #2. Figure (a) shows a slice of the simulation results plotted as an oriented elevation plot during diastole while (b) is during systole.

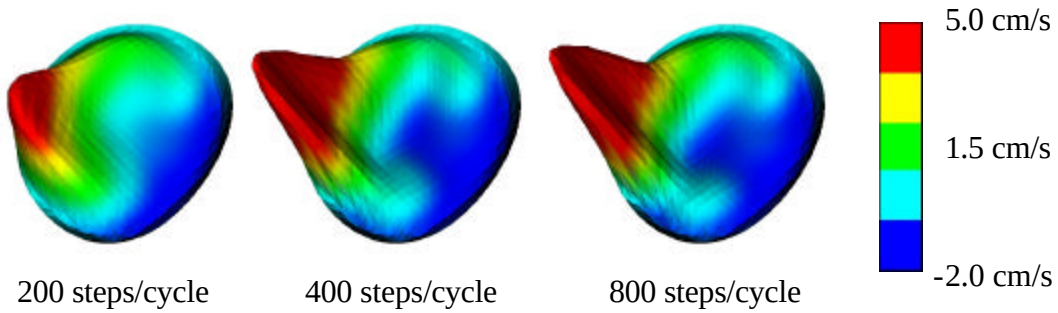


Figure 4.34: Effect of time step size on velocity profiles in the bypass body of plan #2. The figure shows the effect of the time step size on through-plane velocity during early diastole. During diastole the volumetric flow is lower, but more time steps per cardiac cycle are required to resolve the peak through-plane component of velocity as shown in the figure.

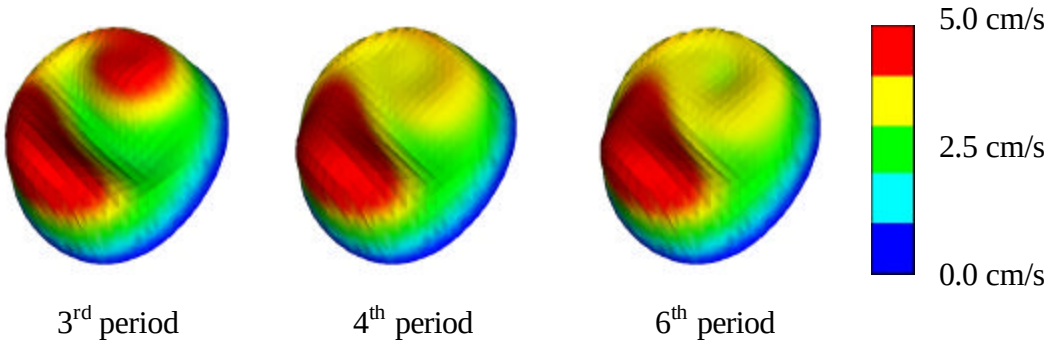


Figure 4.35: Checking for periodicity in velocity profiles in the bypass body of plan #2. The figure shows the through-plane component of velocity just after peak systole. It required approximately four cardiac cycles for the solution to become periodic in the bypass body for plan #2. If the flow solution is periodic in a given location, it typically takes several simulated cardiac cycles for the solution to become periodic due to inaccuracies in the initial conditions.

While volumetric flow distribution is of paramount concern initially following a surgical intervention, other quantities such as high particle residence time and low mean wall shear stress are theorized to be factors in disease progression and long term graft patency. Figure 4.36 shows a scalar clearance time simulation run over 16 cardiac cycles. The scalar is initialized to a value of 1.0 in the entire domain (shown in red) with a Dirichlet

boundary condition at the inlet of the scalar equal to zero. Zero flux is assumed through the vessel walls, and the outlet boundaries allow scalar advection and diffusion from the flow domain. The scalar is passively advected and diffused by the blood flow. The figure indicates that the external iliac arteries have a very high scalar clearance time that may be related to a high particle residence time. This is also consistent with the occlusion of these vessels seen postoperatively. Another region of interest is near the bypass bifurcation. This area may be a region that promotes thrombus formation due to stagnant flow.

Figure 4.37 and Figure 4.38 show mean wall shear stress results. Mean wall shear stress is a clinically important quantity since a surgeon wants to avoid creating regions of low mean shear stress as these locations may be involved in disease progression leading to atherosclerosis and intimal hyperplasia and possible occlusion of the bypass graft. Figure 4.38 shows results from a simulation run on the preoperative model. An interesting area of low mean wall shear stress appears on the posterior aortic wall just above the aortic bifurcation. The aortic bifurcation is the most common place for abdominal aortic aneurysms to form.

4.3.1.7 Case summary

This case study highlights the functionality of **ASPIRE^2** to build geometric models from MRA data, process PCMRI data for flow information, and perform and visualize the results of hemodynamic simulation. This is the first study to combine imaging data and numerical simulation in a patient-specific fashion to study the effects of surgical design choices on the resulting hemodynamic characteristics for an aorto-femoral bypass patient. The study also highlights the practical challenges in processing MRA and PCMRI data, and the restrictions of certain modeling assumptions such as rigid-wall approximations.

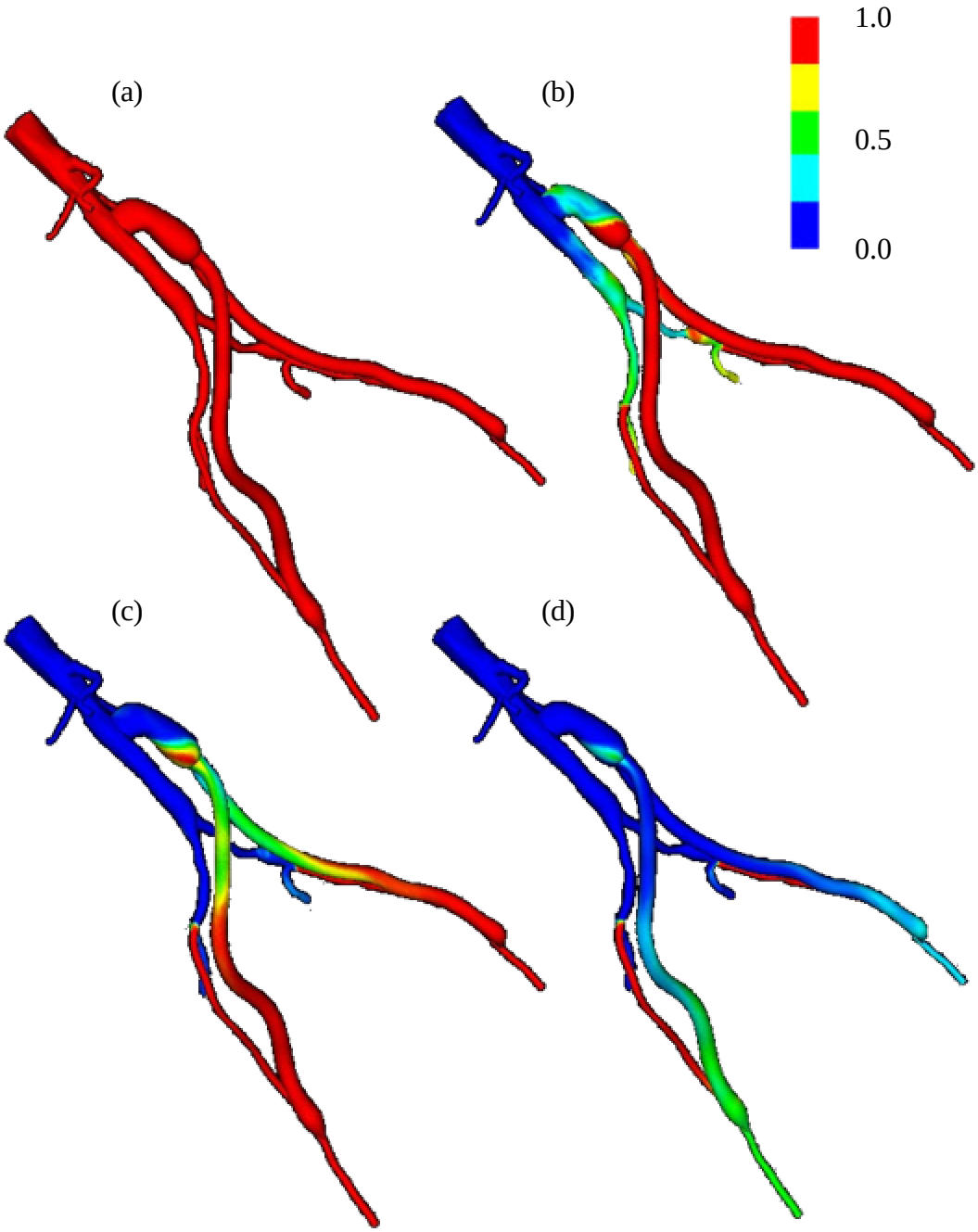


Figure 4.36: Scalar clearance time for plan #2. Initially, the scalar field is set to 1.0 (shown in red) with a zero scalar Dirichlet boundary condition at the inlet. The scalar is then advected and diffused out of the flow domain (blue corresponds to a scalar value of zero). The vessel walls have a scalar flux of zero.

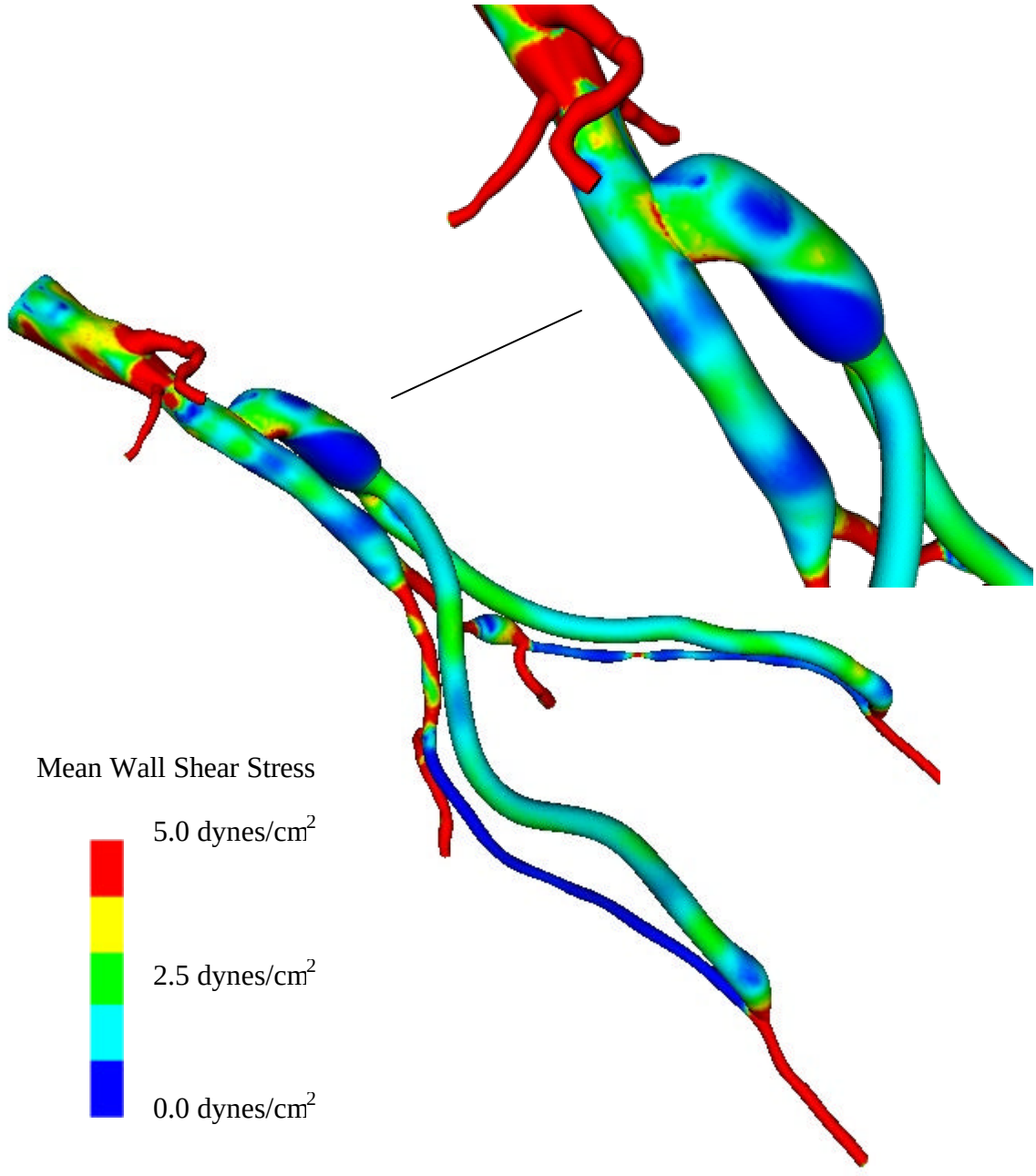


Figure 4.37: Mean wall shear stress for plan #2. Regions of low mean wall shear stress (shown in blue) are theorized to be sites of high risk for the development of atherosclerosis.

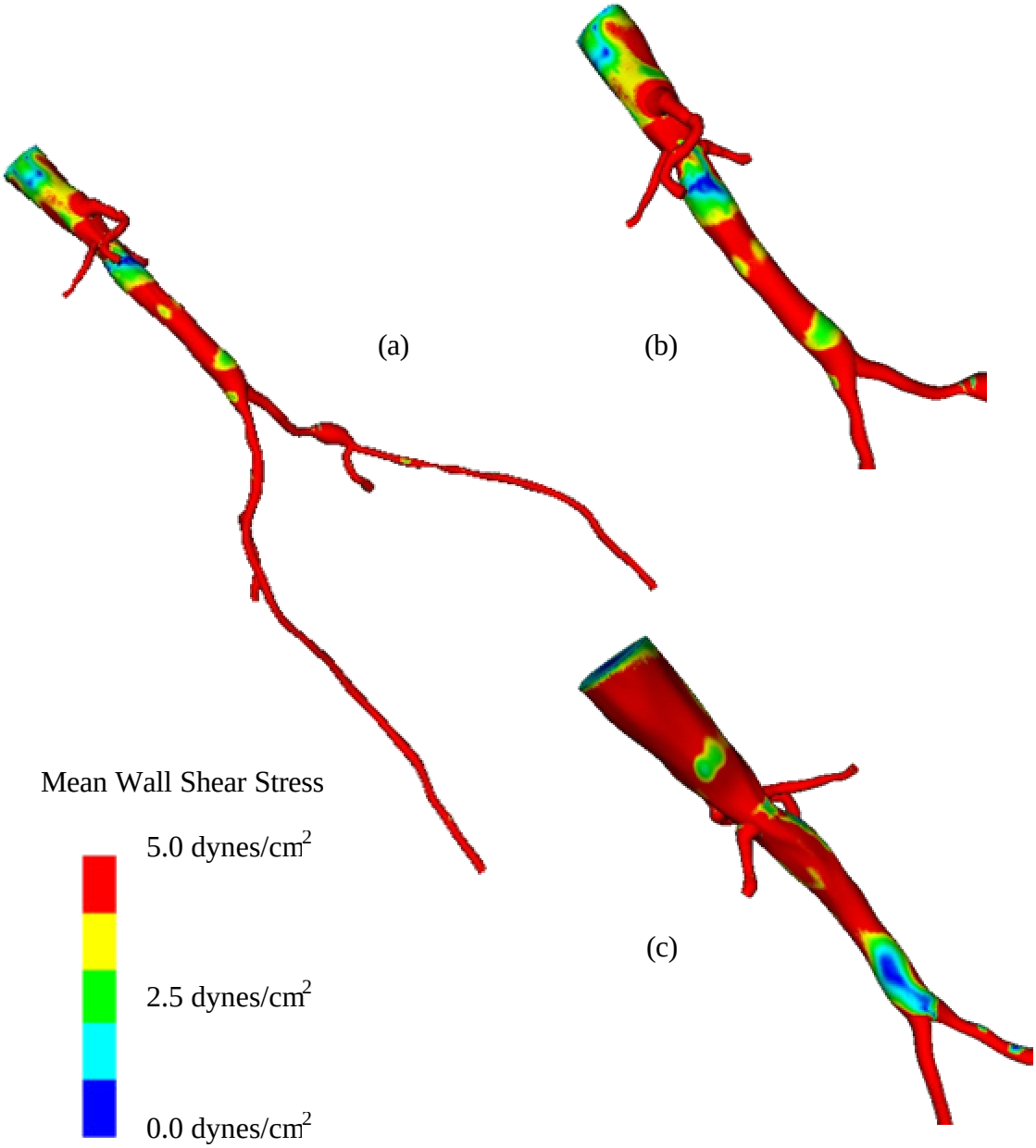
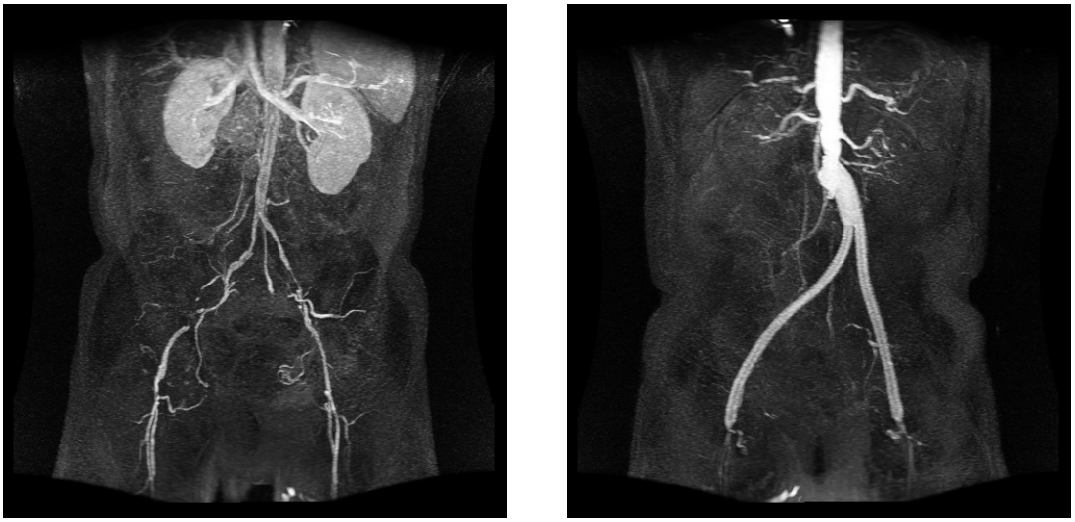


Figure 4.38: Mean wall shear stress for preoperative model. Figure (a) shows the mean wall shear stress for the entire preoperative model. Figure (b) shows a zoomed anterior view of the infra-renal aorta, while (c) shows a posterior view of the infra-renal aorta. Notice the region of low mean wall shear stress located just proximal to the aortic bifurcation, an area predisposed to atherosclerosis.

4.3.2 Patient #2

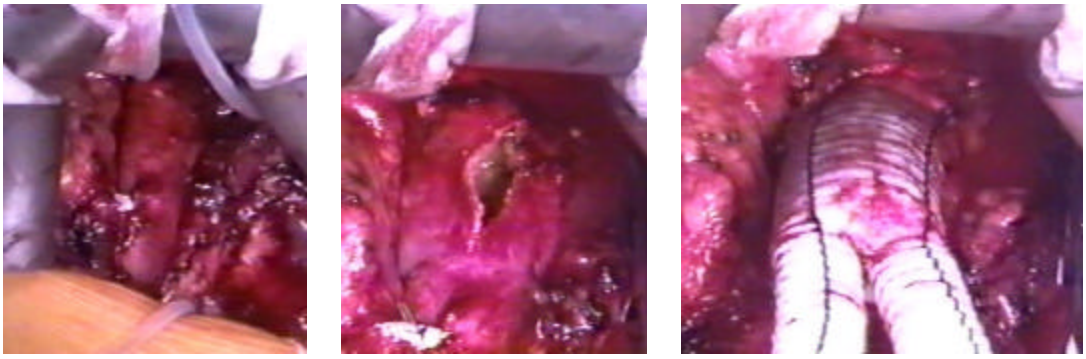
4.3.2.1 Case history

A 55-year old male was experiencing severe pain in his legs during mild exercise. After examination, it was determined he had the following relevant medical history: peripheral vascular disease (ankle brachial index 0.4 bilaterally), chronic obstructive pulmonary disease, nicotine abuse, and coronary artery disease. Noninvasive imaging was consistent to severe aorto-iliac-femoral occlusive disease. Angiography confirmed the diagnosis and an aorto-femoral reconstruction was planned. Preoperative medical imaging data was acquired two days prior to his AFB procedure (see Figure 4.39a). The preoperative data indicated a patent right internal iliac artery, a long segment occlusion of the right external iliac artery, an occluded left common iliac artery with diminished filling of the left internal iliac artery, and a long segment stenosis of the left external iliac. A highly visible collateral vessel (possibly branching from the superior mesenteric artery) was feeding the vascular bed supplied by the left internal iliac artery. An end-to-side aorto-femoral bypass procedure was performed. The surgery for this case study was video taped, and Figure 4.40 through Figure 4.42 show the proximal and distal anastomoses created during the operation for this patient. Postoperative MRA data was acquired nine days after the operation (see Figure 4.39b). The postoperative MRA clearly indicated a complete occlusion of the distal native aorta. Figure 4.43 shows an isosurface of the proximal anastomosis from the postoperative MRA. The attending radiologist noted that the right internal iliac artery was still patent, although no longer receiving flow from the right common iliac artery. Eleven days after the operation PCMRI data was acquired to quantify the volumetric flow in the native aorta and bypass. The operation was successful and the patient was symptom free.



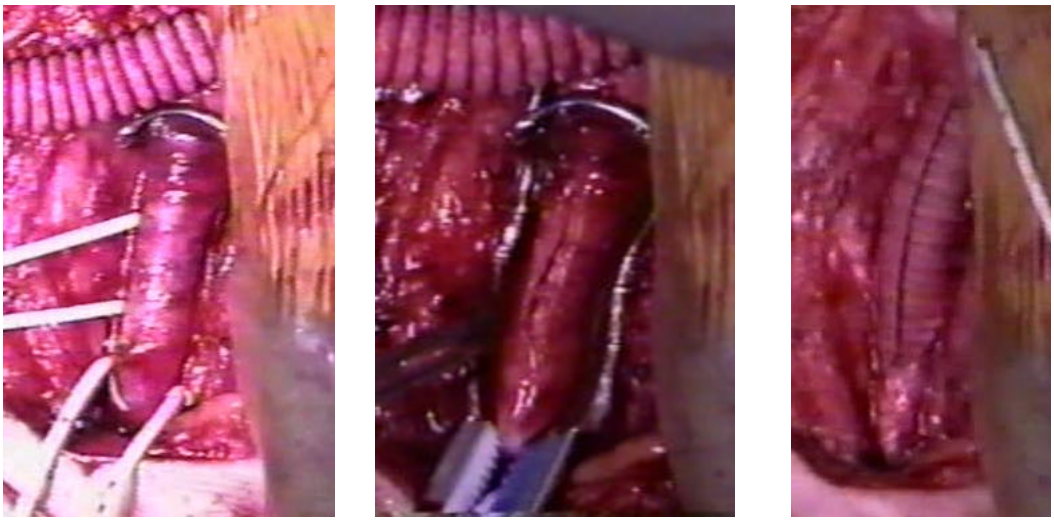
(a) (b)

Figure 4.39: MIP of preoperative and postoperative MRA for patient #2. Figure (a) shows the extent of the occlusive disease preoperatively (notice the tight stenosis in the left and right external iliac arteries), while figure (b) shows the bypass graft. Notice that postoperatively the distal native aorta and the external iliac arteries are occluded. The postoperative data was acquired nine days after the surgery.



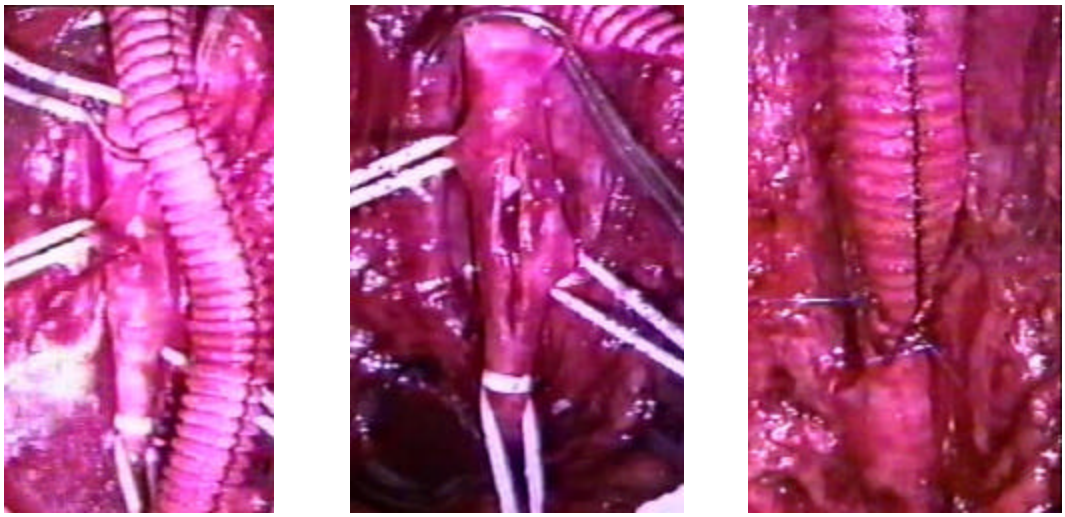
(a) (b) (c)

Figure 4.40: Proximal anastomosis for patient #2. Figure (a) shows the infra-renal aorta prior to the creation of the anastomosis, (b) shows the incision and (c) shows the anastomosis after the bypass body has been attached to the aorta.



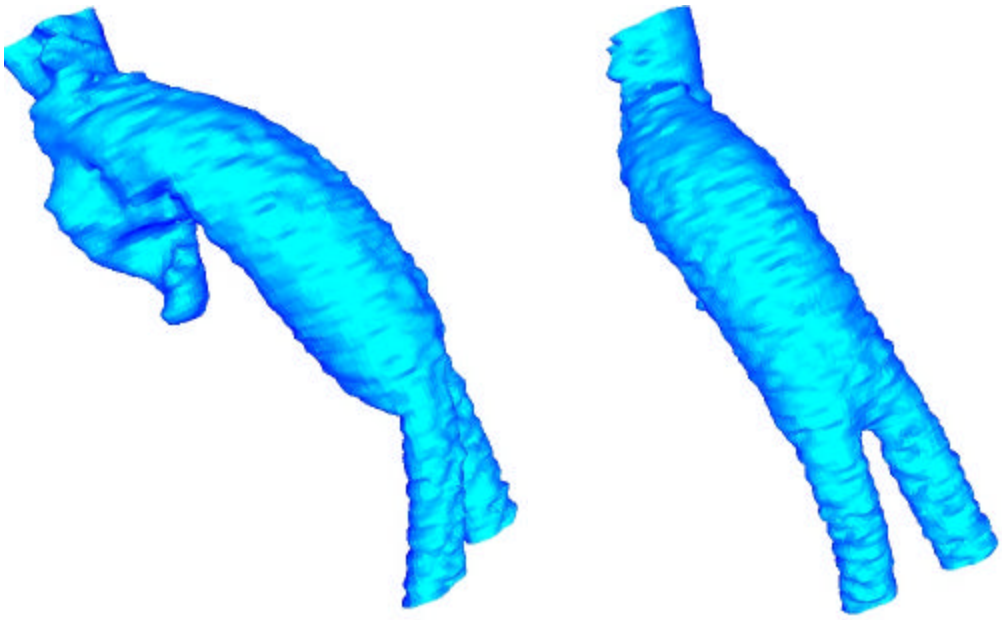
(a) (b) (c)

Figure 4.41: Right distal anastomosis for patient #2. Figure (a) shows the common femoral artery prior to the creation of the anastomosis, (b) shows the incision and (c) shows the anastomosis after the bypass has been attached to the common femoral artery.



(a) (b) (c)

Figure 4.42: Left distal anastomosis for patient #2. Figure (a) shows the common femoral artery prior to the creation of the anastomosis, (b) shows the incision and (c) shows the anastomosis after the bypass has been attached to the common femoral artery.



(a) (b)

Figure 4.43: Views of the proximal anastomosis from the postoperative MRA dataset for patient #2. An isosurface of the postoperative MRA dataset in the neighborhood of the proximal anastomosis is shown from two different angles. Notice that the distal native aorta occluded after surgery.

4.3.2.2 Preoperative model construction

Prior to the patient's surgery, a preoperative model shown in Figure 4.44a was constructed. Table 4.8 gives a summary of the 2-D segmentations used to create the model. Due to the significant vascular disease present, the arteries distal to the aortic bifurcation presented a geometric modeling challenge. On the patient's right side, for example, the right external iliac artery was occluded and a collateral vessel branching off of the right common iliac artery had enlarged to feed the right femoral artery. The vessel was tortuous and small compared to the image resolution. Unlike patient #1, the profunda femoris arteries were highly visible in the scan (with several generations of branching) so they were included in the preoperative model.

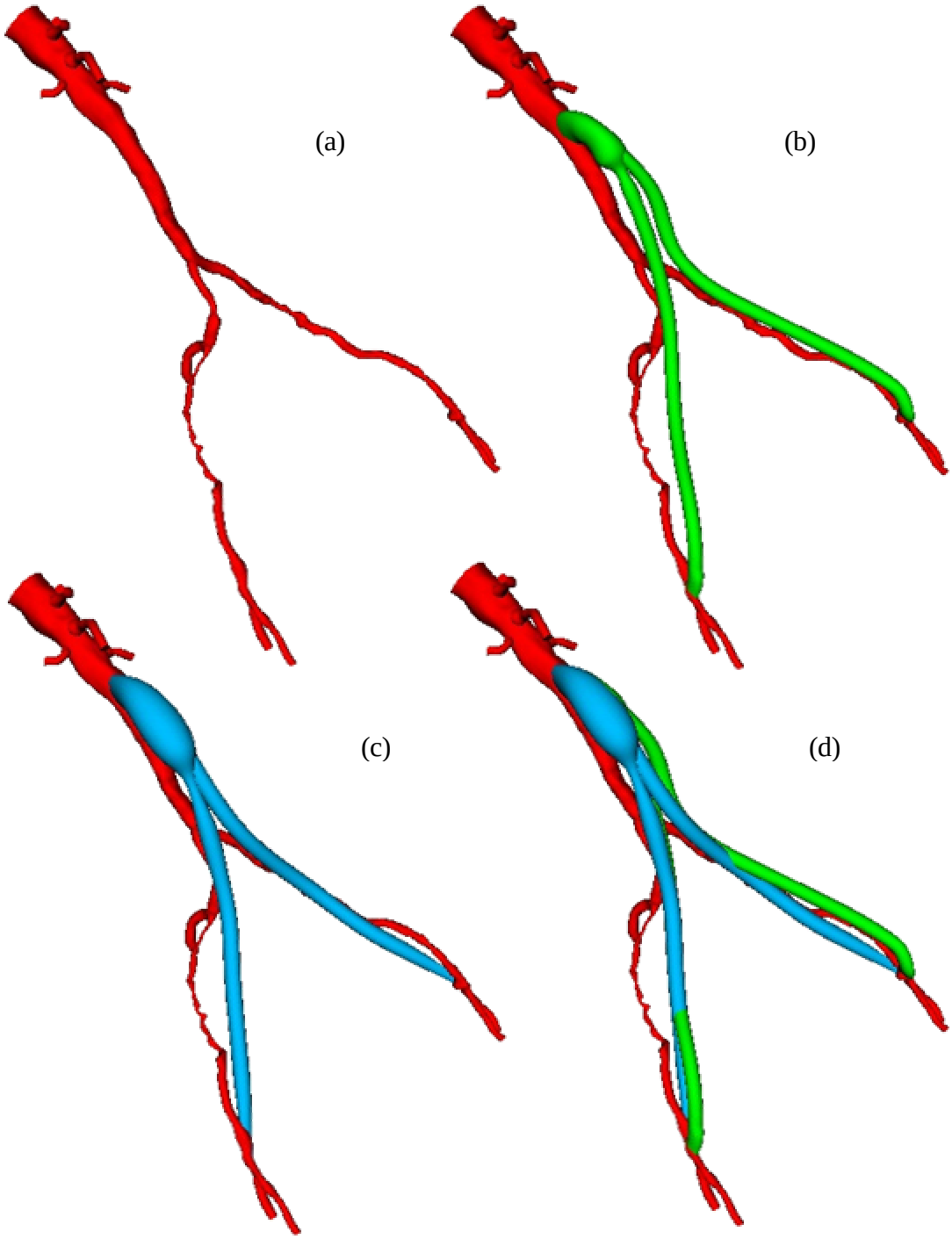


Figure 4.44: Two bypass plans for patient #2. Figure (a) shows the preoperative model, (b) shows plan #1 and (c) shows plan #2. Figure (d) shows both plans together to highlight differences between the two plans.

Table 4.8: Segmentations Used to Create Preoperative Model for Patient #2

Vessel	Aorta*	Right Iliac+	Celiac Trunk	SMA	Renal		Right internal Iliac	Profunda Femoris		Totals
					Right	Left		Right	Left	
Total profiles	88	53	7	10	7	6	8	8	6	193
Level set profiles	32	11	0	0	0	0	0	0	0	43
Threshold profiles	56	42	7	0	0	0	8	8	6	127
Analytic: circles	0	0	0	10	7	6	0	0	0	23
Analytic: ellipse	0	0	0	0	0	0	0	0	0	0
Hand drawn profiles	0	0	0	0	0	0	0	0	0	0
Smoothed profiles	88	53	4	0	0	0	7	7	6	165
Replaced with circles	0	9	4	0	0	0	1	2	0	16
Pasted profiles	1	1	1	0	0	0	2	1	1	7

* aorta group includes aorta, left common iliac, left external iliac, and left femoral artery, + right iliac group includes right common iliac, right external iliac, and right femoral artery

4.3.2.3 Operative planning

Prior to the operation, the surgeon created two operative plans as shown in Figure 4.44. Plan #1 considered the option of using a 14 mm × 7 mm Dacron graft with an assumed uniform dilation of 14% (Figure 4.44b). Plan #2 considered the option of using a 16 mm

× 8 mm Dacron graft with an assumed dilation of 22.5% for the bypass body and a 12.5% dilation of the bypass limbs. In addition, the location and path of the bypass is different in both surgical plans as highlighted in Figure 4.44d. The variation in the location and path of the bypass is motivated both by the different size grafts and the desire of the surgeon to test how differences in the procedure can impact the resulting flow. However, the location of the graft is limited by the anatomical structures of the body during actual placement.

A 16 mm × 8 mm Dacron graft was used in the actual surgery. The grafts are cut and trimmed specifically for each patient. When designing an anastomosis, the surgeon cuts the circular graft cross-section at an angle to create an elliptical cross-section to attach to the native vessel. This is done because it is believed the resulting anastomosis leads to better hemodynamic conditions. The proximal anastomosis of the graft measured 27 mm x 16 mm prior to attachment to the native aorta. Based on the postoperative MRA data, the bypass body had a nominal diameter of 22 mm (37.5% dilation) and the bypass limbs were nominally 10 mm in diameter (25% dilation). For this case study, both surgical plans underestimated the dilation of the graft *in vivo* under physiologic pressures.

4.3.2.4 Processing PCMRI data

In theory, processing the PCMRI data should resemble that for patient #1. In practice, however, there were acquisition problems that made processing the PCMRI data for this case study particularly difficult. Properly selecting encoding parameters can be a challenge, and for several of the slices of preoperatively acquired PCMRI data aliasing occurred (i.e. velocity wraparound since the velocity value is encoded in the phase angle). When acquiring PCMRI data, the operator sets a parameter (VENC) which is related to the maximum expected magnitude of velocity in a given slice location. Keeping the VENC low increases the accuracy of the velocity values obtained, but risks aliasing. In this case study, however, higher than expected velocities were observed in the slices through the common iliac arteries preoperatively and aliasing occurred. Technical

difficulties caused the postoperative MRA data and the PCMRI data to be acquired at different times. Since the data was acquired on different days, the patient's physiology and position in the magnet could have changed between acquisitions adding an additional source of error. Due to the patient anatomy and boundary conditions used in the present work, only supra-celiac and infra-renal aorta volumetric flow rates were needed. From the preoperative scan, these were determined to be 1.8 L/min in the supra-celiac aorta, and 0.7 L/min in the infra-renal aorta. Based on an estimate from a preoperative PCMRI slice plane at the aortic bifurcation, it appeared that 67.5% of the distal aortic flow was entering the right common iliac artery (~ 470 ml/min) and 32.5% of the flow was entering the left common iliac artery (~ 230 ml/min). Note that the right common iliac artery was providing flow to both the internal iliac artery and the femoral artery on the patient's right side.

4.3.2.5 Preparing for analysis

Tetrahedral meshes were generated for each plan shown in Figure 4.44. The mesh for plan #1 consisted of 1,781,661 elements (355,413 nodes) and the mesh for plan #2 consisted of 1,937,880 elements (382,393 nodes). The boundary conditions used for this case study are shown in Figure 4.45. The three-component velocity data obtained experimentally from the preoperative PCMRI data was prescribed at the mesh inlet. Approximately 1.1 L/min of combined outflow was prescribed in the superior mesenteric, celiac trunk, and renal arteries. This effectively prescribed an infra-renal flow of 0.7 L/min. A simple distribution was assumed (22.7% to each renal artery, 27.3% to the celiac trunk and superior mesenteric artery) for the branch vessels between the supra-celiac and infra-renal image slices. The flow waveforms of the prescribed analytic boundary conditions were scaled versions of the inlet waveform.

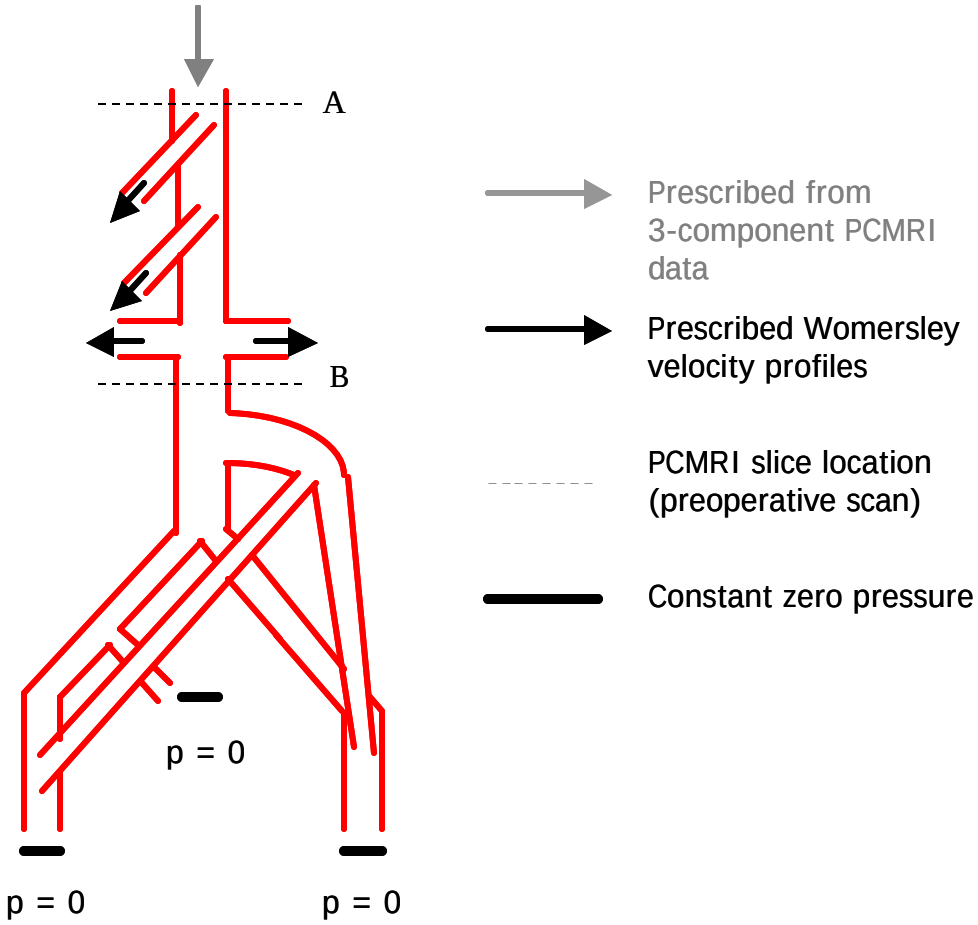


Figure 4.45: Boundary conditions used for patient #2 simulations. Three-component nodal velocities were imposed from PCMRI data at the inlet, with Womersley analytic profiles being imposed at the outlet boundaries of the celiac trunk, superior mesenteric, and renal arteries. All of the outlets below the aortic bifurcation were set to zero pressure. The PCMRI slice plane locations shown correspond to the preoperatively acquired data, and it was this data that was used to prescribe volumetric flow rates. The assumed mean volumetric flow rate for the superior mesenteric artery and celiac trunk was 0.273*(A-B), while the mean volumetric flow rate for each renal artery was assumed to be 0.227*(A-B).

4.3.2.6 Simulation results

Blood flow simulations were performed assuming a blood viscosity of 0.04 poise and density of 1.06 g/cm^3. A total of six cardiac cycles were simulated with 400 time steps per cardiac cycle. The mean volumetric flow rate results from the 6th period of the simulations are summarized in Table 4.9. The simulations show flow in the distal native

aorta between 170-200 ml/min. Recall that the postoperative MRA data indicated that the distal native aorta occluded within 9 days of the surgery (Figure 4.39b). The results from the simulations presented here are not necessarily consistent with the clinically observed occlusion of the distal native aorta. In addition, as was the case for patient #1, clinically relevant quantities such as mean wall shear stress (Figure 4.46) and oscillatory shear index (Figure 4.47) were calculated.

Table 4.9: Mean Volumetric Flow Rates for Two Surgical Plans

Location	**Plan #1 (ml/min)**	**Plan #2 (ml/min)**
Infra-renal aorta	693*	693*
Bypass body	521	490
Bypass right limb	264	276
Bypass left limb	255	213
Aorta distal bypass	172	202
Left external iliac	42	56
Right common iliac	135	147
Right external iliac	-2	-3
Right internal iliac	138	149
Left femoral	114	133
Right femoral	110	134
Left profunda	178	136
Right profunda	153	139

* explicitly prescribed by boundary conditions

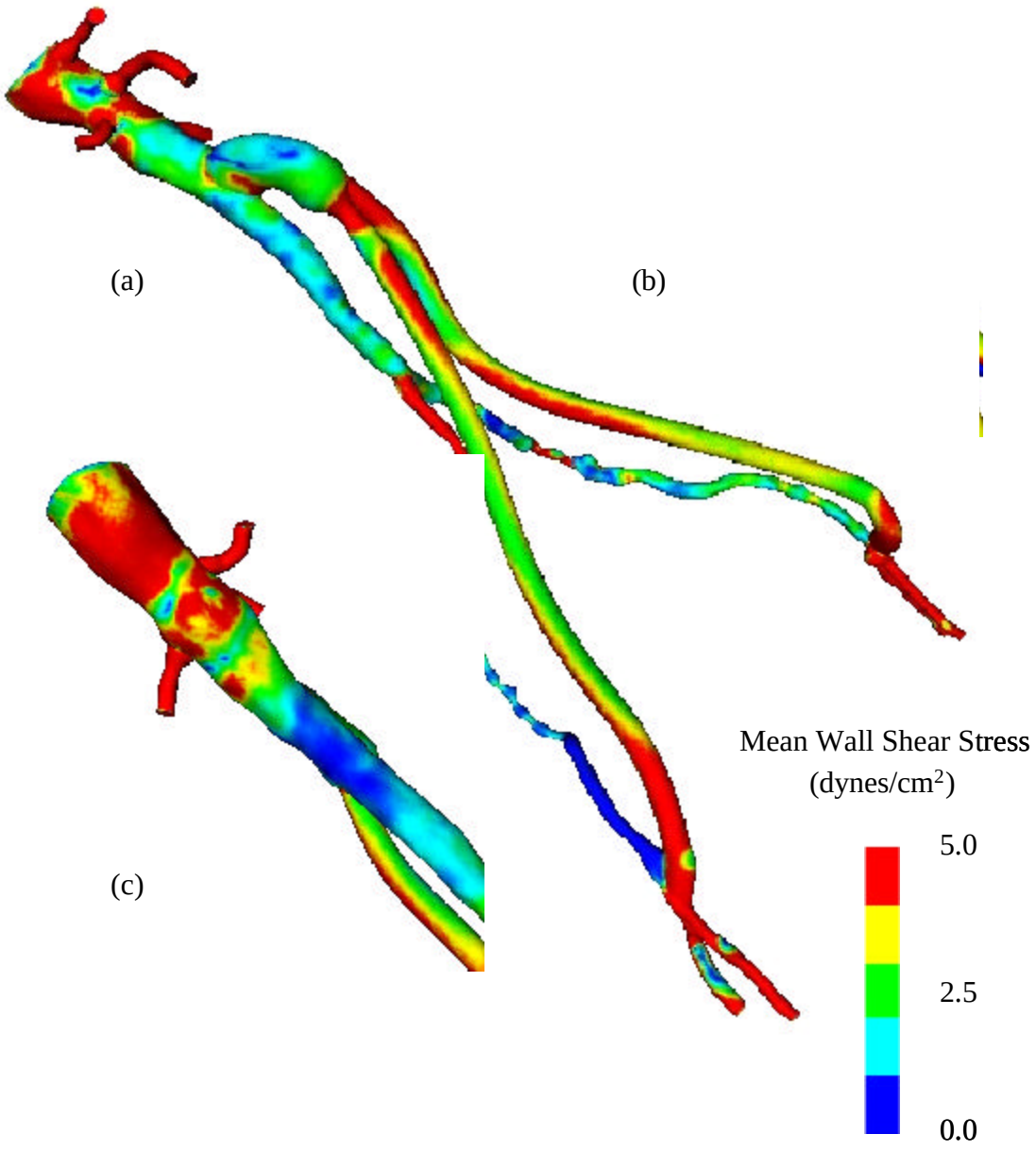


Figure 4.46: Mean wall shear stress for plan #1. Figure (a) shows the entire model, while (b) shows an anterior view and (c) shows a posterior view of the proximal anastomosis.

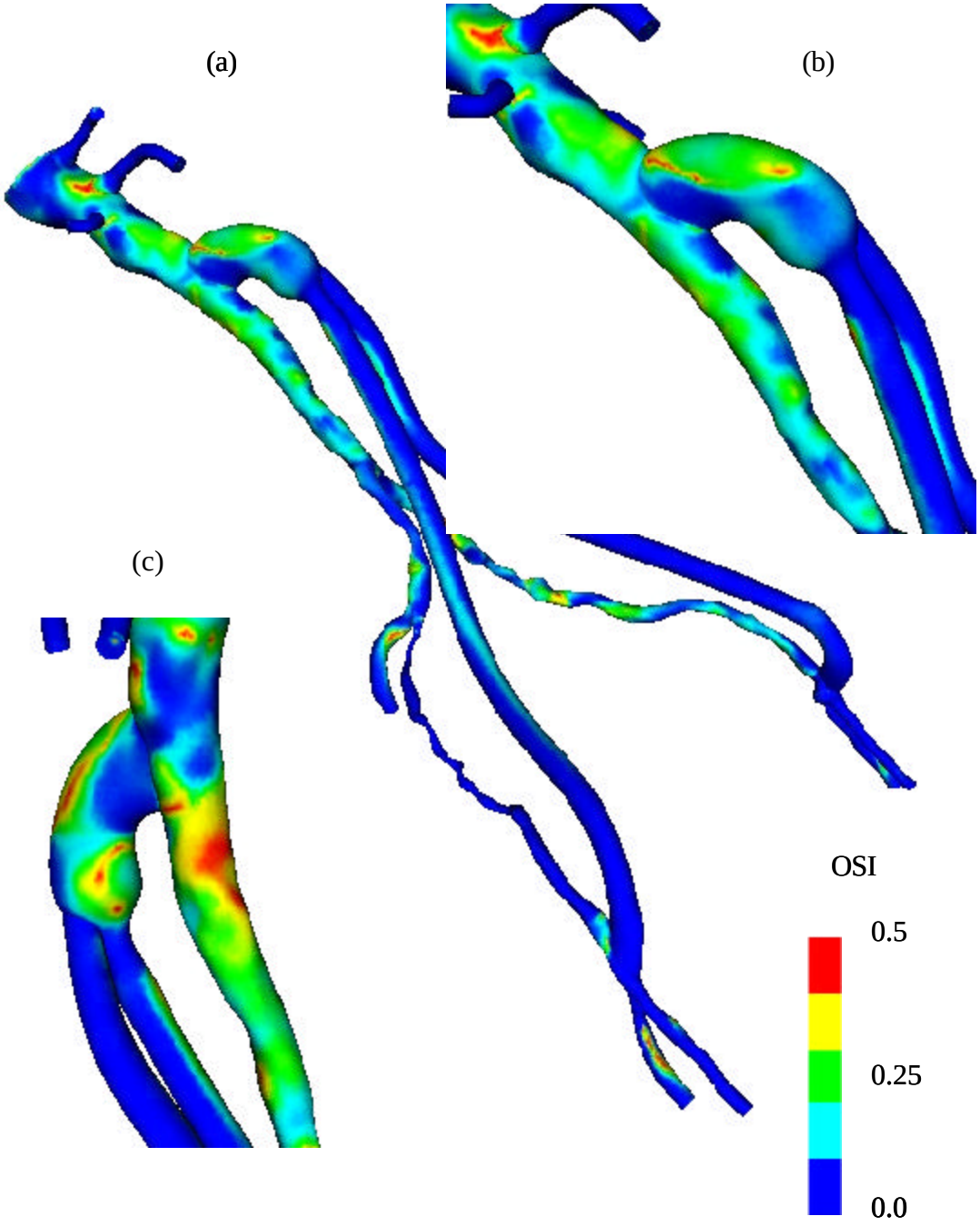


Figure 4.47: Oscillatory shear index (OSI) for plan #1. Figure (a) shows the entire model, while (b) shows an anterior view and (c) shows a posterior view of the proximal anastomosis.

4.3.2.7 Case summary

The significance of this case study was the demonstration of model construction and operative planning in a clinically relevant time frame. The preoperative model and PCMRI data was processed in less than 24 hours after the data acquisition, and the surgeon was able to create two surgical plans prior to performing the actual surgery. Each plan required about one hour of the surgeon's time. The data for this case study presented additional processing challenges compared to patient #1 due to difficulties with the PCMRI acquisition. In addition, the closure of the native distal aorta was not expected. There are two possibilities to explain the apparent inconsistency of the simulation results with the observed clinical outcome. First, the simple boundary conditions used may not have adequately captured the postoperative flow distribution. Current research into improved boundary conditions may remove this limitation in the near future [60]. Second, it is possible that when the surgeon clamped the native aorta (see Figure 3.2) during the operation, thrombus or plaque was dislodged and traveled downstream impeding flow in the distal native aorta. It is currently unknown how frequently debris becomes dislodged due to surgical interventions.

4.4 ABDOMINAL AORTIC ANEURYSM

The previous two examples focused on aorto-iliac occlusive disease. This case study demonstrates application of **ASPIRE**^{2} to other clinically relevant vascular disease research such as flow in abdominal aortic aneurysms.

4.4.1 Case history

A 76 year-old male patient entered the hospital with symptoms of back and abdominal pain. The relevant medical history included: chronic renal insufficiency, severe hypertension, and renal artery stenosis. The diagnosis was a symptomatic AAA with impending rupture. An endovascular aneurysm repair was performed with an AneuRx (Medtronic Inc., Santa Rosa, CA) stent graft.

4.4.2 Preoperative model construction

While the previous examples of occlusive disease focused on using MRA volumetric image data, this case study highlights the use of CTA data. Figure 4.48 shows a volume rendered image of the patient preoperatively. As clearly visible in the figure, CTA contains bone which produces a signal of comparable intensity (albeit higher) to that of the contrast enhanced blood. While clearly visible bone is useful in providing anatomic context, it is a disadvantage when the bone is in close proximity of a vessel to be segmented. Since the aorta follows and is in close proximity to the spinal column, the selection of appropriate level set segmentation parameters can be challenging for cross-sections such as shown in Figure 4.49a. In addition, calcifications such as the one shown in Figure 4.49b present challenges is selecting level set segmentation parameters and preventing the lumen boundary from erroneously falling inside of the calcification. Finally, since modern CTA provides significantly higher geometric resolution than MRA, detailed features of the lumen boundary can be resolved (e.g. Figure 4.49a,c) increasing the complexity of the geometric models generated which impacts downstream processing such as mesh generation and simulation.

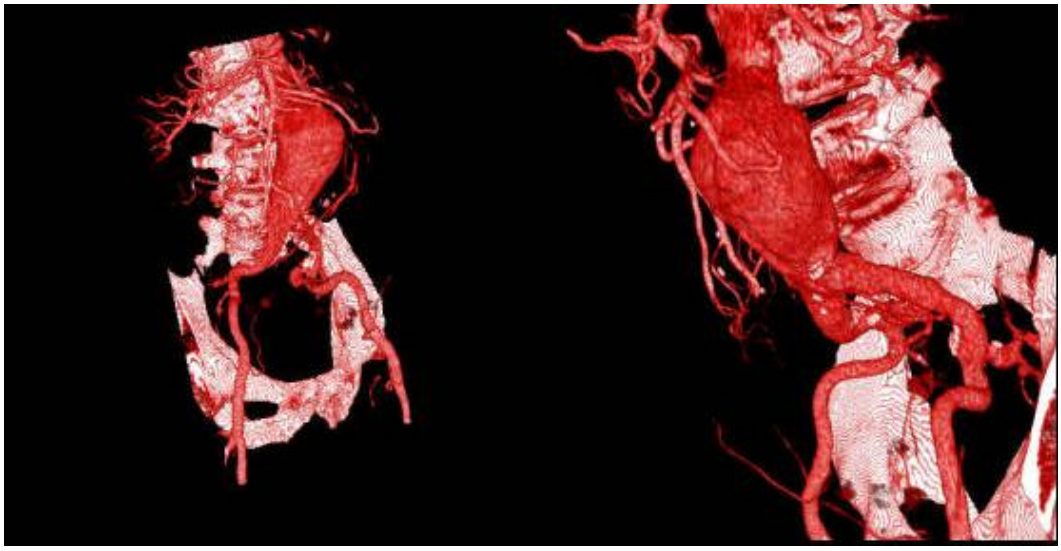


Figure 4.48: Two views of AAA for 76 year-old male patient. The figure shows two views of the preoperatively acquired volume rendered CTA dataset.

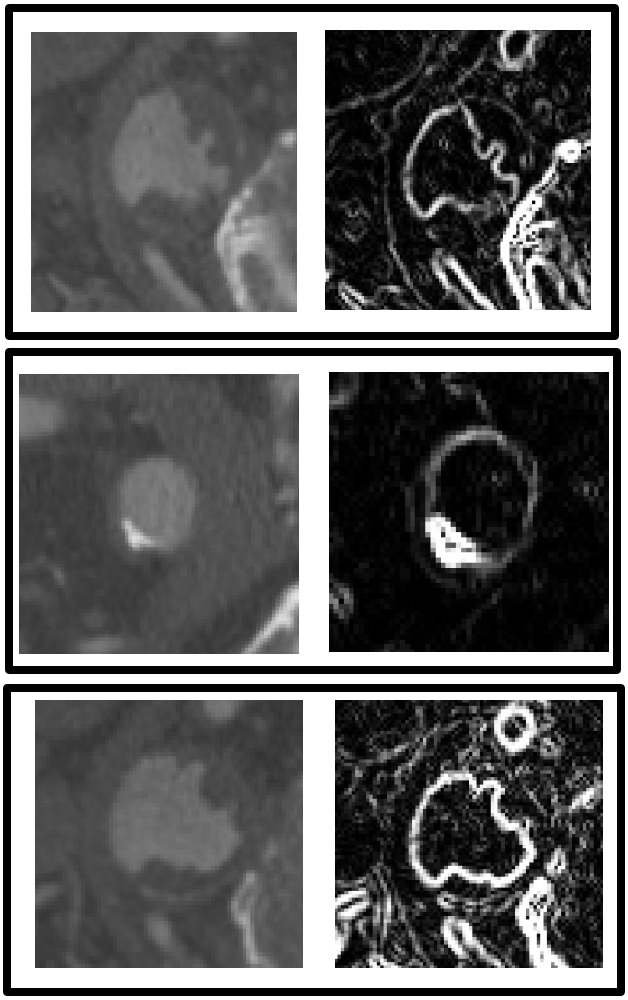


Figure 4.49: Issues inherent to segmenting CTA data. The two major challenges in processing CTA data compared to MRA data derive from the high signal of bone (and calcifications) and the increase in image resolution in CTA. Figure (a) shows a slice of the infra-renal aorta where the proximity of the spine can cause segmentations problems. Figure (b) shows the high signal from a calcification in a diseased vessel that adds difficulty in selecting appropriate values for level set segmentation parameters. Figures (a) and (c) show the resolution possible in a modern CTA scan defining a complex flow lumen.

The preoperative model was constructed in similar fashion to the previous examples with one exception. First, in addition to the approximate medial axis paths extracted as described in section 3.5.2, a path parallel to the S/I coordinate axis through the aneurysm was created. Since the aneurysm is approximately aligned with the S/I coordinate

direction, this straight path enables the use of closely spaced 2-D segmentations of the lumen boundary that would not be possible if the curved medial axis path was used. Figure 4.50 shows the AAA created from 2-D cross sections. It is worth noting that the complex geometric nature of the aneurysm and its proximity to the aortic bifurcation provide motivation for future work in 3-D image segmentation techniques.

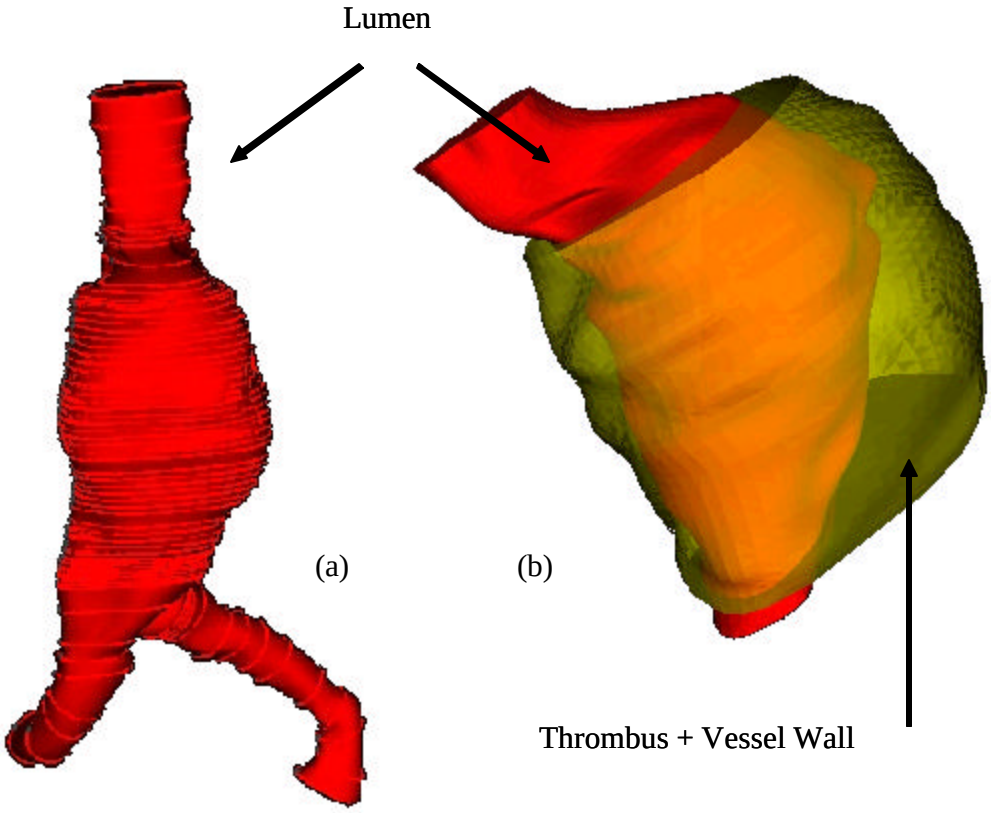


Figure 4.50: Preoperative solid model of AAA. Figure (a) shows the model created from a set of 2-D level set segmentations. Figure (b) shows a close up of the model with a translucent solid representing the thrombus and vessel wall. The solid representing the thrombus and vessel wall was segmented by hand from the imaging data and steps similar to those for creating the preoperative model were performed. The only additional processing step is to subtract the solid representing the lumen from the solid representing the vessel wall, thrombus, and lumen.

Another noteworthy difference between this and the previous examples was the ability to create a solid representing the thrombus and vessel wall around the aneurysm. This is due to a combination of the increased geometric resolution of CTA over MRA and the

significant thickness of the thrombus and vessel wall (with respect to in-plane pixel dimension). With proper initialization, it may be possible to extract the boundary of the vessel wall using the level set method. However, in this particular example hand-drawn segmentations were created. The segmentations were then smoothed and a solid representing the lumen, thrombus, and vessel wall was created. To recover a solid representing just the thrombus and the vessel wall, the solid model representing the flow lumen was then subtracted. While no simulations were performed on the resulting solid representing the wall of the aneurysm, the ability to create geometric models of the wall will be required in future applications studying AAA rupture.

For simulation purposes, a second solid model with a smoother surface was also created. The segmentations used to create the model shown in Figure 4.50 were smoothed with only the three lowest order Fourier modes of each contour being maintained. The inlet of the geometric model for the smoothed model was extended in the superior direction to remove the proximity of the prescribed velocity boundary conditions from the aneurysm.

4.4.3 Preparing for analysis

Two meshes were generated for this case study. The first consisted of 1,078,006 elements (201, 530 nodes) corresponding the geometry show in Figure 4.50. The second mesh was for the smoothed aneurysm model and consisted of 250,305 elements (48,982 nodes).

Since CTA only provides geometric information, an infra-renal mean volumetric rate of 0.7 L/min was assumed. In addition, the flow waveform was assumed to be the same as Figure 4.31. Boundary conditions were applied as shown in Figure 4.51.

4.4.4 Simulation results

Blood flow simulations were performed assuming a viscosity of blood of 0.04 poise and density of 1.06 g/cm^3. The simulations were run for six cardiac cycles with 400 time

steps per cardiac cycle. The simulations required a special residual control term be added to the formulation to prevent the numerical solutions from becoming unbounded. This may be indicative of transitional or turbulent flow inside of the aneurysm. In addition to the flow study, a scalar clearance time simulation was performed on the 250,305 element mesh. These simulations were performed for 200 time steps per cardiac cycle for 34 cardiac cycles.

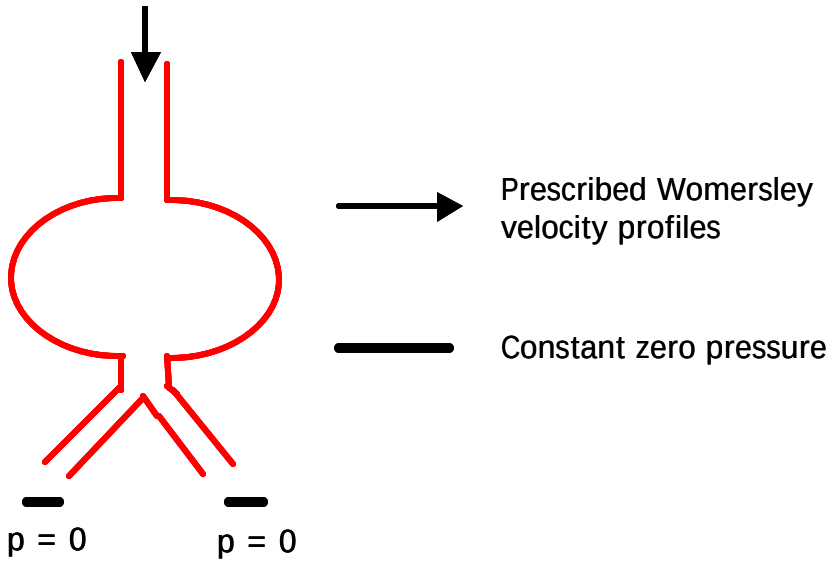


Figure 4.51: Boundary conditions used for AAA simulation. A Womersley analytic profile was prescribed at the inlet and zero pressure boundary conditions were applied at the outlets. The mean volumetric flow rate was assumed to be 0.7 L/min in the infra-renal aorta.

To highlight visualization functionality available in the **ASPIRE**^{2} system, Figure 4.52 shows the combination of simulation results from the one million element mesh with imaging data in the same visualization window. In addition to velocity information, Figure 4.53 shows the results of a scalar clearance time simulation. The figure shows the posterior wall and the proximal anterior wall of the aneurysm has a higher scalar clearance time than the rest of the aneurysm.

4.4.5 Case summary

The importance of this case study is that it shows construction of a preoperative model from CTA data. The geometric resolution of CTA provides both processing challenges and unique opportunities to increase the geometric resolution of models used for SBMP. Future studies of flow in the abdominal aorta may provide insight into hemodynamic conditions that promote aneurysmal formation and disease progression. Combining the geometric accuracy of CTA with flow information obtained using MRA may be possible in the future.

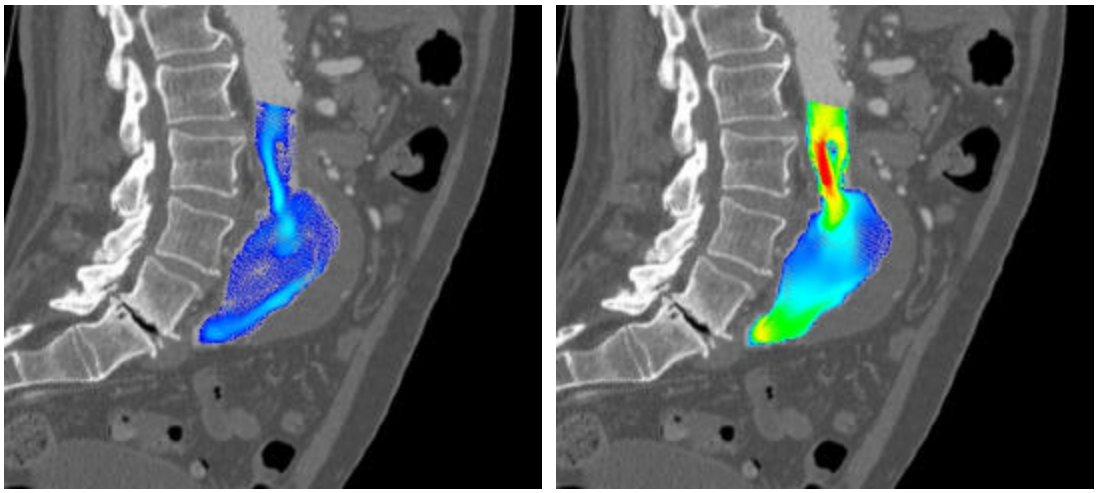


Figure 4.52: Visualization of simulation results with imaging data. A grayscale planar slice of the CTA data is shown with velocity vectors from a corresponding planar slice through the simulation results. The left figure corresponds to a time point from diastole and the right figure to a time point from systole.

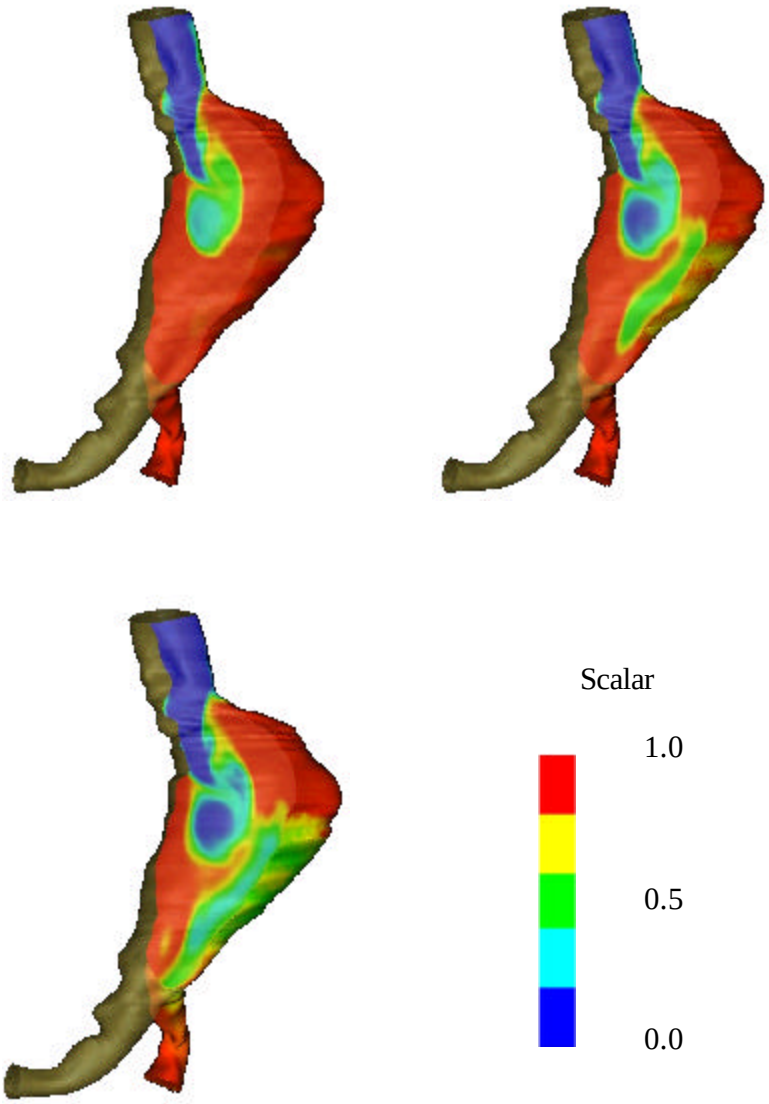


Figure 4.53: Scalar clearance time. The simulation shows the posterior wall of the AAA and part of the proximal anterior wall have a significantly higher scalar clearance time than the rest of the aneurysm.

PART III – MEMS PROTOTYPING

The next two chapters demonstrate a system (named **^{s}vfab**) built on top of the **Geodesic** framework outlined previously. The goal of this system is to use the simulation-based design methodology discussed in part I for micro-electro-mechanical-system design.

CHAPTER 5 COMPUTATIONAL MEMS PROTOTYPING

5.1 OVERVIEW

This chapter introduces the field of micro-electro-mechanical-systems (MEMS) and motivates the use of computational prototyping for device design. A client-server architecture is then described that enables Internet-based computational prototyping of MEMS. The remaining part of the chapter then details contributions to the field of geometric modeling for microsystems implemented as extensions to the **Geodesic** framework described in Chapter 2.

5.2 MEMS AND THE NEED FOR COMPUTATIONAL PROTOTYPING

Transducers, by definition, convert one form of energy into another. Two important classes of transducers are sensors and actuators. In this thesis, the focus will be on sensors. A sensor will be defined as a device that transforms some form of energy into an electrical signal. For example, a room thermostat is a sensor that transforms thermal energy into an electrical signal that may be used to display the current temperature of the room on a digital display. If a sensor is fabricated using micromachining technology developed for the integrated-circuit (IC) industry, it is referred to as a microsystem or micro-electro-mechanical-system (in this work, all microsystems are referred to as MEMS whether or not they involve mechanical forces).

Several leading authorities in the field believe that design tools are critical to the future and the growth of the MEMS market. For example, Karen W. Markus, director of the MEMS Technology Applications Center at MCNC has stated [61]:

> *MEMS foundries cannot effectively serve their customers without a set of engineering design tools that enable remote, non-expert, users to interface*

> *with those foundries. There are no such packages that free the designer from needing a comprehensive understanding of the fabrication process or the nuances of polygon-level layout design.*

In addition, Eric Peeters from Xerox writes [62]:

> *Several microelectromechanical systems have achieved commercial success. The barriers can still be formidable, though, and the path to success is often much different for MEMS than it was for mainstream semiconductors. Maturing software for comprehensive modeling and design will help in the future.*

Stephen Senturia, a leading expert in the field of MEMS simulation, commented on the state-of-the-art in [63]:

> *Technologies for fabricating a variety of MEMS devices have developed rapidly, but computational tools that allow engineers to quickly design and optimize these micromachines have not kept pace. Inadequate simulation tools force MEMS designers to resort to physical prototyping.*

Analysis of MEMS, however, is a challenge; the structures are geometrically complicated, electromechanically coupled, and inherently three-dimensional. The multi-domain physics for MEMS include: electrical, mechanical, and fluidic interactions. Effects of each domain need to be accurately simulated to correctly predict device performance. In addition, IC-based fabrication techniques require the modeling of complex material behavior and processing such as etching, deposition, and thermal processes such as diffusion (and oxidation). The size and aspect ratios of typical MEMS structures differ significantly from those traditionally found in the very-large-scale integration (VLSI) community. These differences make it difficult to construct geometric models using standard Technology CAD (TCAD) process simulation tools.

5.3 KEY ISSUES IN COMPUTATIONAL PROTOTYPING FOR MEMS

In the MEMS field, it is becoming widely accepted that representing the 3-D geometry of devices is critical for simulation-based design. Many commercial and academic systems

for MEMS analysis rely on purely geometric operations to create the structures. Notable examples include MEMCAD, Intellicad, and Solidis [4, 64, 65, 66, 67, 68]. While tools exist for creating geometry for the VLSI community, there are unique challenges posed by MEMS. Modeling challenges include the complexity and aspect ratios of typical structures and residual stresses that can create initial curvature in ideally flat devices. The typical dimensions of MEM switches (such as fabricated using the MUMPS process) are on the order of several hundred microns in length, a width on the order of 1/10 the length, and thickness of only several microns. These dimensions cause significant problems for deposition and etching process simulators such as SPEEDIE [69]. If full 3-D physical process simulations were performed, it would take excessive amounts of computing resources including time and memory that exceed current limitations of software and hardware. In addition, commonly occurring features such as large flat regions do not gain geometric accuracy from 3-D deposition and etching simulation compared to using 1-D or 2-D simulation.

A second issue concerning microsystem simulation is that available commercial tools are designed for a "traditional" computing environment. A traditional computing environment consists of a single user working at a fixed workstation running software locally with licenses tied directly to the given machine (or local cluster of machines). Commercial microsystem design tools were not designed to minimize platform dependence or take advantage of the latest computing technology (e.g. the Internet). The field of MEMS provides an ideal target for Internet-based simulation software for the following four important reasons:

1. Emerging market. Unlike the automotive and aerospace industries which have been around and utilizing simulation tools for decades, the MEMS market is relatively new and use of simulation in the design process has not yet become deeply entrenched using existing simulation paradigms.

2. Cost. A few large companies that can devote significant manpower and financial resources to computational prototyping drive the automotive and aerospace industries. In sharp contrast, MEMS is a field driven by smaller specialized firms and universities with tighter financial constraints.

3. Size of models. A typical model for an aircraft involves millions of unknowns (i.e. degrees of freedom). However, many micromechanical devices of interest can be reasonably simulated with tens-of-thousands of degrees of freedom.

4. Standardized processes. Due to economies of scale in IC fabrication, several popular commercial processes (e.g. MUMPS) currently exist. Since fabrication runs take several months, significant time is lost between runs if one is iteratively designing a prototype. This provides opportunities for "virtual fabrication runs" which can be done using Internet-based simulation tools between fabrication cycles.

This chapter details a MEMS computational prototyping system that was developed from the ground up to take full advantage of state-of-the-art technology in computer hardware, software, and the Internet. Key design decisions in the software architecture and engineering tradeoffs of computational efficiency for automation were made to enable Internet-based prototyping. After discussing the needs of the typical microsystem developer, an overview of the system is given.

5.4 SIMULATION ENVIRONMENT SYSTEM REQUIREMENTS

The architecture of a software system for microsystem design must take into account the needs and skills of a typical designer. Market research indicates typical MEMS design groups are small, without individuals who are solely dedicated to computer simulation [70].

Here several basic assumptions can be made of the typical designer:

1. Mechanical/Electrical engineer with a masters degree or equivalent work experience
2. Minimal background in computer simulation
3. Desire to utilize new tools which supply useful and timely feedback on new designs

The potential users are then designers who want to use computer simulation to help guide their development efforts. Under these assumptions, Figure 5.1 shows the desired iterative design cycle using the example of a RF switch.

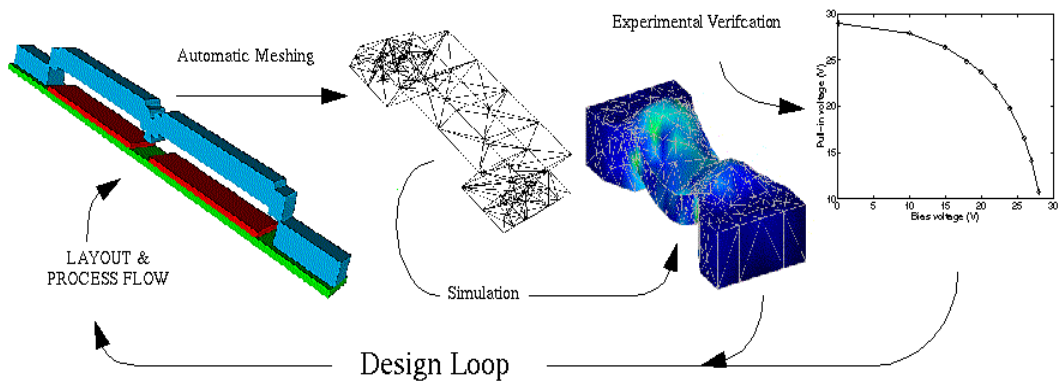


Figure 5.1: Desired simulation-based design loop for a micro-electro-mechanical switch.

In general, a simulation-based design system for MEMS devices consists of four major component technologies (see Chapter 2):

1. Geometric modeling (e.g. physical process emulation or simulation)
2. Discretization (i.e. mesh generation)
3. Simulation (e.g. coupled electro-mechanical analysis)
4. Visualization (e.g. displacements, charges, stress)

These components need to be sufficiently integrated so that information can travel from one module to another without significant user intervention. Two quite different

modeling methodologies are being pursued among the MEMS CAD community. The "homogeneous" approach is to develop tightly integrated component simulation tools and design methodology. Alternatively, a "heterogeneous" approach is one that wraps distinctly different end-user software packages to create a "meta" tool. The MEMCAD system is one well-known example that uses commercial and other academic tools (ABAQUS, IDEAS, and Fastcap) [68]. The system detailed below relies on using a heterogeneous collection of appropriately chosen external software kernels to create a versatile system that provides maximum automation of the model creation and simulation process.

A client-server architecture is now described that enables Internet-based computational prototyping of MEMS. This is followed with a detailed explanation of the server application that includes geometric modeling capabilities for microsystems implemented as extensions to the **Geodesic** framework.

5.5 CLIENT-SERVER ARCHITECTURE FOR INTERNET-BASED PROTOTYPING

As part of this work, a prototype of an Internet-based system for MEMS simulation was developed [71]. It focused on using mainstream, current web-based technology while highlighting possible future directions in relevant Internet technology. It proposed a paradigm of automation that limits the information passed back and forth between a browser-based client and the back-end server. In the present implementation, geometric models are generated from layouts and a process specification, then meshed automatically, and simulated using *hp*-adaptive finite-element techniques. The web- and browser-based environment provides file transfer, simulation control, and visualization of simulated results.

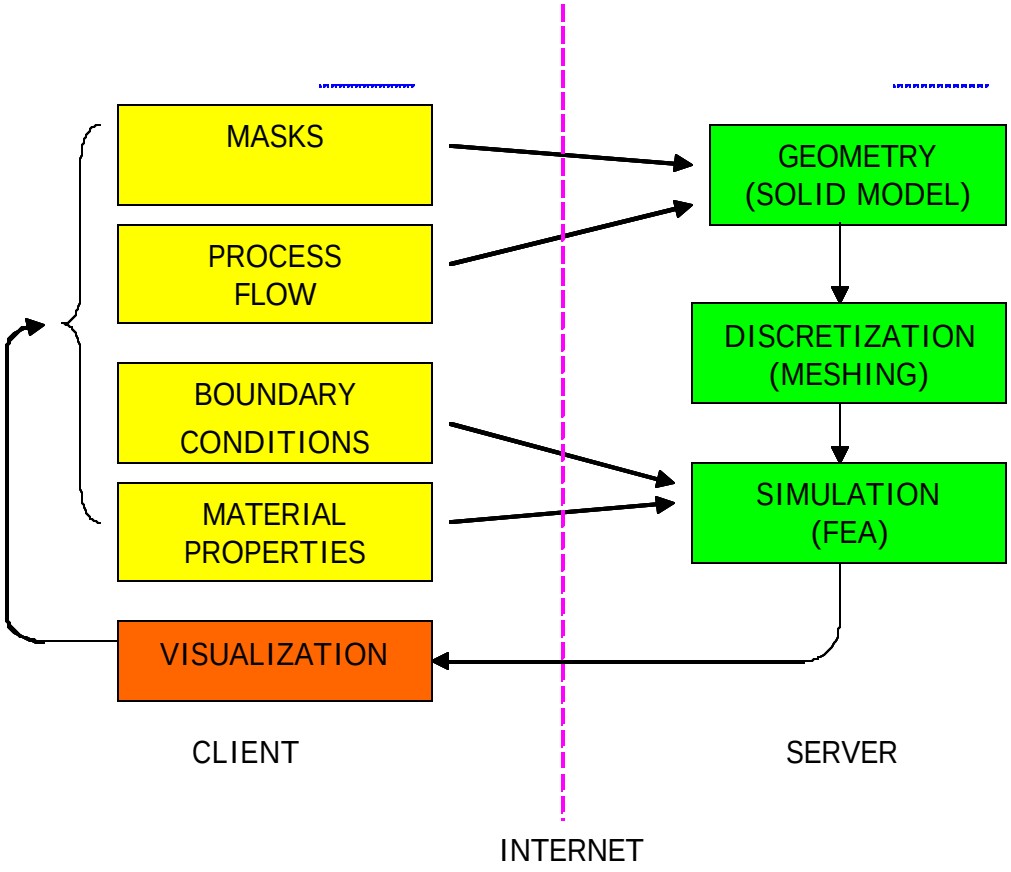


Figure 5.2: Schematic of client-server architecture for Internet-based MEMS prototyping.

The system takes advantage of a client-server paradigm as shown schematically in Figure 5.2. The yellow boxes (masks/layout, processing flow, boundary conditions, and material properties) indicate the user-provided information. The left side of the diagram also indicates the steps that are carried out on the designer's machine (referred to as the "client"). The right side of the figure corresponds to the processes run on the "server." The server can be the same machine as the client, a different machine on the same local network, or a remote machine accessed via the Internet, even through a proxy.

There are three main steps carried out on the server:

1. *Geometric Modeling*. This step takes the user-defined process flow and masks for the layout and creates a solid model representing the three-dimensional device.

2. *Discretization*. This step automatically breaks the solid model into smaller non-overlapping discrete pieces (called a mesh) needed for simulation.

3. *Simulation*. This step simulates the device using multi-physics finite-element analysis techniques to obtain electrical and mechanical characteristics of the structure.

In the overall system, the boxes on the left correspond to computationally inexpensive tasks, while the boxes on the right can require significant computational resources. The communication between the left and the right side is transaction oriented (not interactive) and thus can be done efficiently by transmitting small- to medium-sized blocks of data. The latency issues inhibiting interactivity include both network bandwidth issues and the computational time required for geometric modeling and simulation.

5.5.1 Client application

The client side application is presented through a frames-capable web browser as seen in Figure 5.3. The toolbar on the left allows selection of several submenus, which in turn update the toolbar and/or the main viewing window to guide the user interactively through the process of going from a design to device simulation. Most of the submenus consist of dialogs for information entered directly using standard HTML form objects. The information is transferred using standard HTTP POST/GET methods and server-side CGI scripts. The use of these standard transaction mechanisms enables access to a server across a firewall, provided that an HTTP proxy is available.

As an example transaction, consider the required step of passing the layouts created on the user's local machine to the server for geometric modeling. Typically the masks will

be created in a standard layout editor (e.g. L-Edit, Magic, etc.) and exist on the client machine in the form of a text file. The user will then specify the desired filename in a standard HTML form object that uses the HTTP POST method to pass the information to the server. The server receives this information via the HTTP port and runs an appropriate CGI script to create a copy of the file on the server. An identical mechanism can be used to transfer a file containing the fabrication process flow (using the Composite CAD Process Definition Specification [72]).

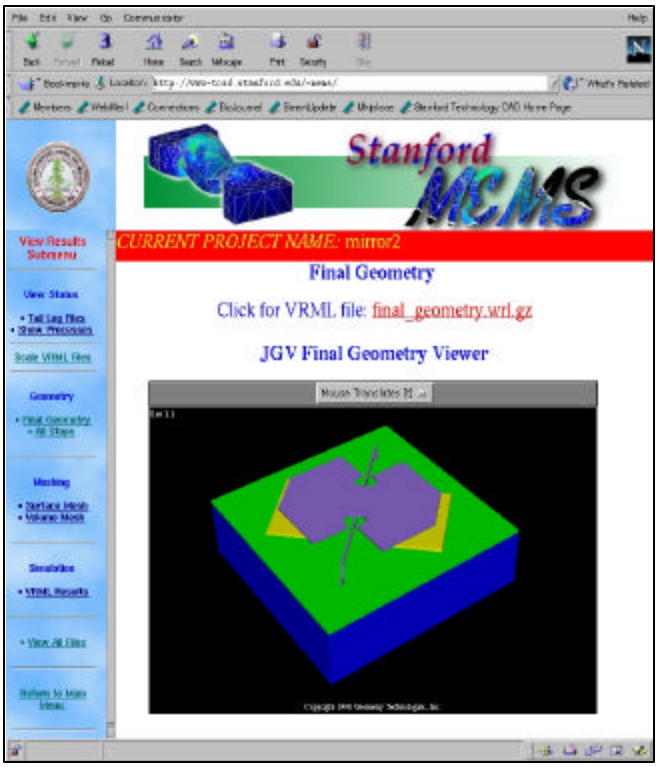


Figure 5.3: User interface (i.e. a web-browser) to the web-based MEMS prototyping environment. The toolbar on the left enables selection of several submenus, which in turn update the toolbar and/or the main viewing window to guide the user interactively through the process of going from a design to device simulation.

The user interface was implemented primarily with standard HTML, instead of Java, for greater portability among browsers and to avoid some of the security restrictions of Java (i.e. Java applets cannot access the local file system which would have prohibited

uploading of local files containing mask and process flow information). Unfortunately, the transaction-oriented nature of HTTP introduces latencies that are too large for highly interactive use. Tasks such as visualization that require a high level of interactivity or do not map well into a forms-based interface are implemented using Java applets that are downloaded to the browser. Examples of applets utilized in this implementation include an interactive geometry visualization applet (JGV) discussed below and a user-friendly applet to create a process flow. The process flow applet uses an HTTP POST method to communicate with the CGI scripts on the server for reasons stated above.

Visualization of geometry and simulation results on the local machine is an important step in the design process. Several visualization techniques were tested. First, visualization of the geometry and the surface meshing was achieved using the Virtual Reality Markup Language (VRML) [73]. VRML was one of the first standard file formats proposed for 3-D Internet-based visualization. Unfortunately, the standard was never universally adopted and VRML viewers are typically limited to only a subset of the functionality and are not available on all platforms. The VRML standard is no longer being developed, and the Web3D consortium has migrated its efforts into developing a new extensible 3-D format [74] for Internet applications. The second attempt for portable geometry and mesh visualization was to use a visualization package named JGV [75], an applet implemented in pure Java. The viewer was less elaborate than the VRML viewers, but did not require special plugins and is only 100 kilobytes in size, small enough to be transmitted with the geometry over low bandwidth links such as a 56 Kbps modem.

5.5.2 Server application

The server side framework is implemented as a collection of PERL and Tcl CGI scripts that manage the transmission and storage of data and control other simulation tools on the server. To organize the information maintained on the server, the idea of a “project” is employed. A project consists of all of the files needed as input (e.g. process flow and masks), the desired output files of the simulation tools (e.g. visualization files), files that

maintain the current state of the device (e.g. if mesh generation is completed), and log files (e.g. files detailing the success or failure of a desired operation). Currently this is handled transparently by having the server track the needed files and storing them physically in a single directory for the given project. The client controls the projects via a submenu in the interface that permits the fundamental operations of creating, deleting, renaming, and copying a project. The user is constantly reminded of the current project name since it is clearly displayed in a separate frame of the web-browser (see Figure 5.3). Since the server tracks the varying states of projects, the user can change between projects or interact with multiple projects via additional windows of the web browser. The main application called by the server scripts, referred to as the Stanford Virtual Fab (**^{s}vfab**), is detailed in the following section.

5.5.3 ^{s}vfab application

The **^{s}vfab** application consists of extensions of the **Geodesic** software architecture detailed in section 2.3. **^{s}vfab** incorporates the algorithms to be discussed in sections 5.6-5.8 to create MEMS geometry of varying degrees of physical accuracy using multiple geometric simulation techniques. It has an integrated meshing layer that enables automatic tetrahedral mesh generation [76]. In addition, it can create input files automatically for a finite-element code based on the ProPHLEX finite-element kernel [77]. Using the ProPHLEX kernel allows the coupled electromechanical problem to be solved using *hp*-adaptive finite-element methods.

The input to **^{s}vfab** consists of a set of masks (defined in a CIF file) and a process flow (specified using the Composite CAD Process Definition Specification [72]). The geometry is then built in a layer by layer fashion by emulating and/or simulating the processing steps used to build the actual device (i.e. virtual fabrication). Currently only physical processing steps are supported, though these are sufficient for the devices discussed herein. When fabrication steps such as implants and diffusions alter the

electrical or mechanical behavior of the device, they can be approximated by using distinct geometric regions for regions of significantly different doping.

With the overview of the **SVfab** application complete, more detail is now provided on building geometry appropriate for microsystem simulation. As detailed below, the task of creating the geometry relies heavily on solid modeling and can involve surface evolution algorithms and advanced geometric algorithms as discussed in section 5.8.

5.6 CREATING GEOMETRY FOR MEMS SIMULATION

In the field of microsystems, numerous commercial and academic efforts are underway to develop simulation-based design systems [4, 64, 65, 66, 67, 68]. These systems use assorted methods to generate geometry for electromechanical simulation. All of these systems currently rely purely on geometric operations to create geometry. That is, there is no physical process simulation involved or used in creating the geometry of a micro-mechanical device. These tools commonly refer to their operations as "process emulation," which implies that the user must know certain characteristics of the final geometry. More specifically, the user must, at a minimum, specify layer thickness, etching masks, and a simplified process flow. Alternatively, process simulation allows the user to specify an input topology and a physics-based model will evolve the geometry based on process-specific parameters such as etch rate, selectivity, and sticking coefficients. Thus, a user must be able to characterize a processing step and feed the simulator the appropriate parameters, but characteristics of the final geometry are not specified. Dutton and Yu [78] provide a brief history of process simulation and its application.

When one considers utilizing MEMS geometry, there is an important distinction between creating geometry for visualizing the outcome of processing steps and using geometry for simulation. In the case of visualizing the geometry, a designer may be curious what his device will actually look like given the masks and processing steps he has specified.

Visual inspection may lead to the discovery of design rule violations, obvious design errors such as missing supports, inappropriate layer thicknesses based on prior knowledge of similar devices, and so forth. In the visualization of the geometry, the retention of small features in a device will have little impact on the ability of the designer to work with it. That is, even the most detailed devices that will be discussed below can comfortably be viewed interactively on a modern PC or workstation. However, in stark contrast, the retention of small features in the geometry that do not impact the global behavior of the device can significantly increase the device simulation time. For example, the retention of etch release holes that do not ultimately affect performance may not be expensive for visualization, but may make the mesh generation and the simulation of the device take CPU days instead of CPU hours to perform.

Although the issues are similar in devices for microfluidics and thermally actuated devices, the focus of this work was on generating geometry for the electromechanical simulation of microsystems. This is reasonable since there are a large number of devices that perform in this physical domain (e.g. RF switches, comb drives, etc.) and electromechanical devices also provide a rich set of devices with varying types of geometry. The goal was to develop a versatile software framework that enables the creation and utilization of varying degrees of topologic complexity and geometric physical accuracy as required for simulation.

5.6.1 Previous and current work in geometric modeling for VLSI and MEMS

Interest in automating 3-D geometry construction, using only mask and process information, for VLSI and MEMS applications has been actively pursued for nearly two decades. One important way to classify these systems is by their implementation. This lumps systems into two basic categories: those that utilize commercial solid modeling kernels and efforts based on proprietary code to represent solid models. Examples of the latter include OYSTER [79], 3DTOP [80], and MemCel [81]. Efforts using commercial solid modeling kernels include VIP3D [82], MemBuilder [83] (the original mechanism

used to build geometry in the commercial system MEMCAD [68]), and the **SUVEVfab** application developed in this work.

Another important classification of these systems is the underlying representation used to represent the geometry. **SUVEVfab**, for example, relies on polyhedral representations of geometry. VIP3D relied on a boundary representation (B-Rep) [82], a common model representation used in solid modeling. A B-Rep solid can contain polygonal faces and a polyhedron can be directly represented directly in a B-Rep. The distinguishing characteristic between these two solid model representations, however, is that a B-Rep includes complex analytic surfaces such as NURBS and swept surfaces.

5.6.2 Surface representation

An important design decision in **SUVEVfab** was to use polyhedra instead of complex analytic surface representations such as NURBS. The motivation for this was twofold. First, Boolean operations on NURBS surfaces can be problematic, particularly for the geometry typical of MEM devices. Second, all of the process simulators used in this work require and generate discretized geometry. Translation of the simulation results from polyhedra into more complex analytic representations such as NURBS can be difficult.

5.6.3 Process emulation vs. process simulation

Often for complicated structures, or depending on the versatility of the processing steps allowed in the package, additional processing steps must be introduced to be able to create geometry that mimics what is expected. For example, a common processing step in the fabrication of micro-devices is conformal deposition. However, if the geometry engine used cannot perform a conformal deposition, pseudo-processing steps of deposition, planarization, and etching with additional masks can sometimes be used to emulate a conformal deposition. Two of the examples discussed in Chapter 6 highlight the differences between process emulation and simulation. The reactive ion etching example shows that the results of emulation can be implementation specific, while the

level set example shows the ability to properly capture rounded corners and the results of select etches using simulation.

5.6.4 Geometric modeling algorithms

Since devices and desired levels of simulation accuracy vary significantly, **SVFAB** was designed as an integrated environment to enable geometric modeling of varying degrees of physical accuracy. It is important, particularly from a designer's perspective, to have a single tool capable of generating geometry from the earliest conceptual stage to final production when detailed knowledge of the fabrication processes can be modified for higher yield or better reliability. In addition to an efficient and robust method to create geometry using only solid modeling operations, **SVFAB** provides the capability to incorporate physically based 2-D and 3-D deposition and etching process simulation results into the geometry. Three key approaches are employed to achieve varying degrees of physical accuracy: a geometric-based deposition algorithm, domain decomposition, and level set physical process simulation.

First, an efficient geometric-based deposition algorithm [42] and standard solid modeling operations form the basis of the process emulation capabilities inside of **SVFAB**. The algorithm provides for surface angle dependent deposition thickness to allow for non-uniform sidewall and step coverage. In addition, the algorithm was designed to be independent of the solid modeling kernel used. A detailed discussion of the implementation can be found in section 5.7. Second, a domain decomposition algorithm discussed in section 5.8.1 (see also [84]) was created to improve the overall computational efficiency for building complex geometries for devices requiring process simulation. The device is decomposed (using the etch masks) at each deposition or etching step into regions identified as needing 1-D, 2-D, or 3-D process simulation. Different geometric and physical approaches to geometry manipulation can be arbitrarily applied among these regions. The domain decomposition algorithm is a critical enabling technology for the level set technique because it drastically reduces the processing power

and memory requirements needed to perform process simulation for a realistic size micro-mechanical device. Finally, as mentioned previously and discussed in section 5.8.2, **[s]vfab** contains a fully integrated general multi-dimensional level set kernel (detailed in [11]) that can be used for process simulation.

5.7 A SOLID MODELER INDEPENDENT IMPLEMENTATION OF THE CCPDS

The Composite CAD Process Definition Specification (CCPDS) [72] was a proposed standard for MEMS CAD vendors to specify the process flow used to create geometry (i.e. solid models) for simulation. The file format was developed as part of a Defense-Advanced-Research-Project-Agency (DARPA) project and was the result of direct interactions of the major MEMS CAD software vendors in existence at the time. The standard had several key design objectives. First, it was designed to be an ASCII text file that was "human readable." In addition, it was to include the most commonly used microfabrication processes (with particular focus on process operations which altered the physical geometry). Finally, version 1.0 was designed to be a minimal standard so that maximum compliance across vendors could be achieved.

This section describes the implementation details of the basic standard found in [72]. Here the overview of a solid modeler independent implementation is given. The CCPDS format consists of eight commands, which are organized functionally in Table 5.1:

Table 5.1: CCPDS Commands Organized by Functionality

General	**Process Emulation**	**Process Macros**
ProcessVersion ProcessUnit Wafer	Deposit Etch MechanicalPolish	ImplantDiffuse Grow

Each of the three functional categories will be discussed separately. It should be noted that the CCPDS requires only solid modeling operations to build geometry of interest. While this standard is useful for many applications, section 5.7.3 will close by discussing the limitations and ambiguity in the file format. An important topic, which is not included in the current standard but discussed in detail in this work, is utilizing process simulation to build geometry.

5.7.1 Implementation details of the CCPDS

5.7.1.1 General commands

The first two general commands listed in Table 5.1 are not specific to any particular step in the process flow. They typically should appear first in a given process flow since they affect the meaning of the other commands in the file. The following are more detailed explanations of ProcessVersion and ProcessUnit:

1. ProcessVersion: This command allows the user to specify the version number of the process flow file. It is useful since it allows for the standard to evolve in the future without invalidating existing process flows. In addition, it permits the vendor to determine if the user has embedded extra information in the file (e.g. extensions).

2. ProcessUnit: This command allows the user to specify the units to be used in creating the model. All of the operations in the basic standard have been implemented to be dimensionally independent.

The most important general command in Table 5.1 is “Wafer” and it must appear prior to any processing steps in the process flow. The wafer is specified using a box in the CIF file, and is specified such that the “bottom” of the wafer is in the plane z=0. All additional processing steps (e.g. depositions) will take place on the wafer. This allows us

to define the "overall height" of the device as the distance from the bottom of the wafer to the maximum z-coordinate in the model.

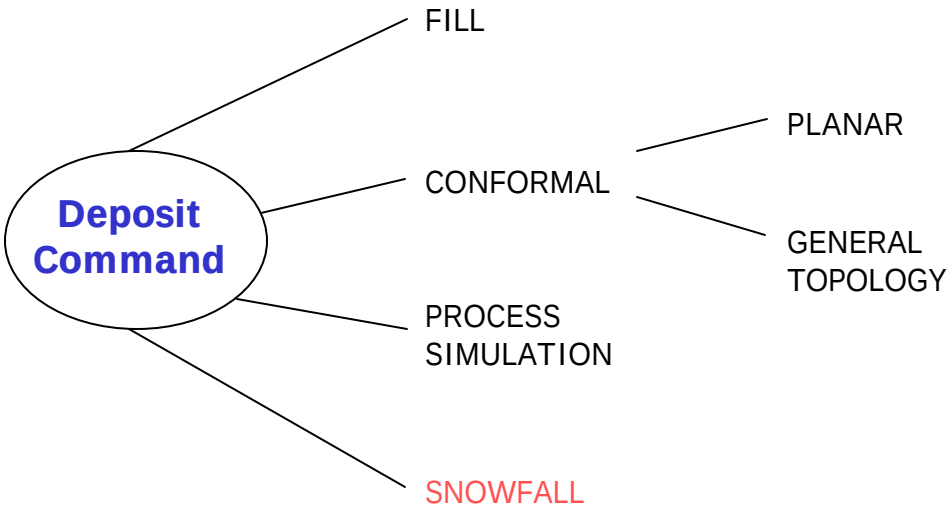


Figure 5.4: Deposition types in **[s]vfab**. The SNOWFALL deposition type was not implemented in **[s]vfab**, and the PROCESS SIMULATION type represents an extension to the CCPDS standard. For implementation purposes, the conformal deposition was subdivided into planar and general topology (see also Table 6.1).

5.7.1.2 Process emulation commands

The process emulation commands consist of methods to add material (deposition) and remove material (etch, mechanical polish) from a given wafer. Recall that these are process emulation steps in that the user must know the final outcome of the real processing step (e.g. deposition thickness).

There are three types of emulated deposition supported by the CCPDS standard: conformal, snowfall, and fill. For implementation purposes, in the present work conformal deposition was further subdivided into two categories: planar and general topology. **[s]vfab** has an additional physical deposition type that can be invoked to use process simulation. Figure 5.4 shows the organization of the deposition commands. The only CCPDS deposition type not implemented was "snowfall." This can be integrated in a straightforward manner via the process simulation deposition type by adding an

anisotropic velocity function that takes into account visibility calculations (see [85]). Figure 5.5 shows graphically the conformal and fill deposition types. Conformal deposition on a non-planar topology requires the offset solid algorithm detailed in [42].

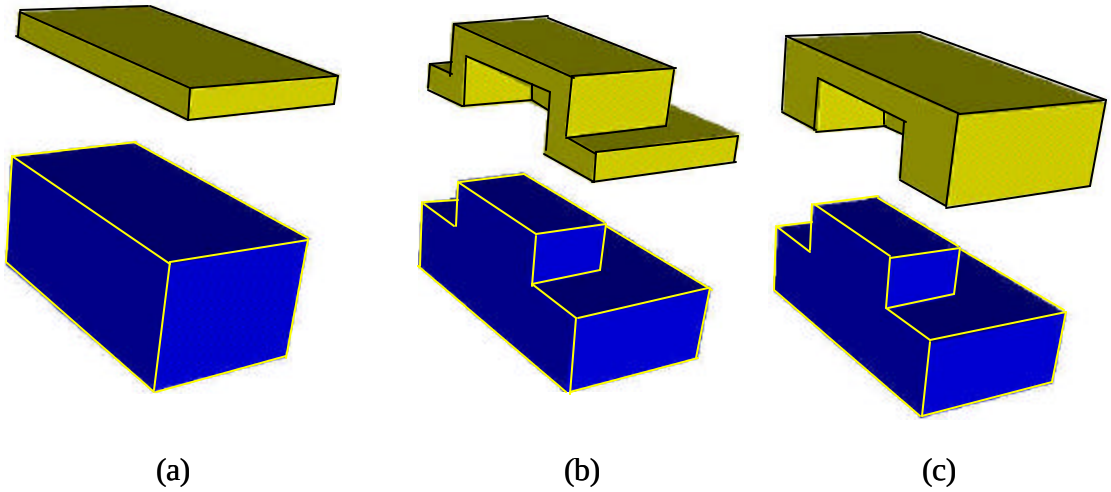


Figure 5.5: Three types of deposition. Figure (a) shows planar deposition, (b) shows conformal deposition, and (c) shows a fill operation. Figure (a) and (c) can be implemented using only solid modeling operations, while (b) requires the offset solid algorithm shown in Figure 5.6.

Figure 5.6 shows schematically the steps involved in using the offset solid algorithm to perform a conformal deposition. In a preliminary step, all of the current layers and wafer for the device are joined (boolean addition) together simplifying all internal boundaries. This leaves a solid with only exterior walls (i.e. free surfaces) as seen in Figure 5.6 (a). Conceptually, we want to generate a new solid (d) that we can subtract the original geometry (a) from and have the result be the deposited material. This requires four major steps. First, the bottom and sides of the wafer are identified (b) and fixed so they cannot move. This is so material can only be deposited on top of the wafer. Second, infinite planes representing the remaining faces are offset in the direction of the normal by the specified deposition thickness (where deposition thickness can be a function of surface normal). Third, at each vertex of the original geometry, the intersection of all infinite planes representing the faces meeting at the vertex is calculated. Assuming the topology

of the device does not change, these vertices along with the original connectivity define the new geometry shown in (d). The final step consists of doing a boolean subtraction of (a) from (d) to get the desired result.

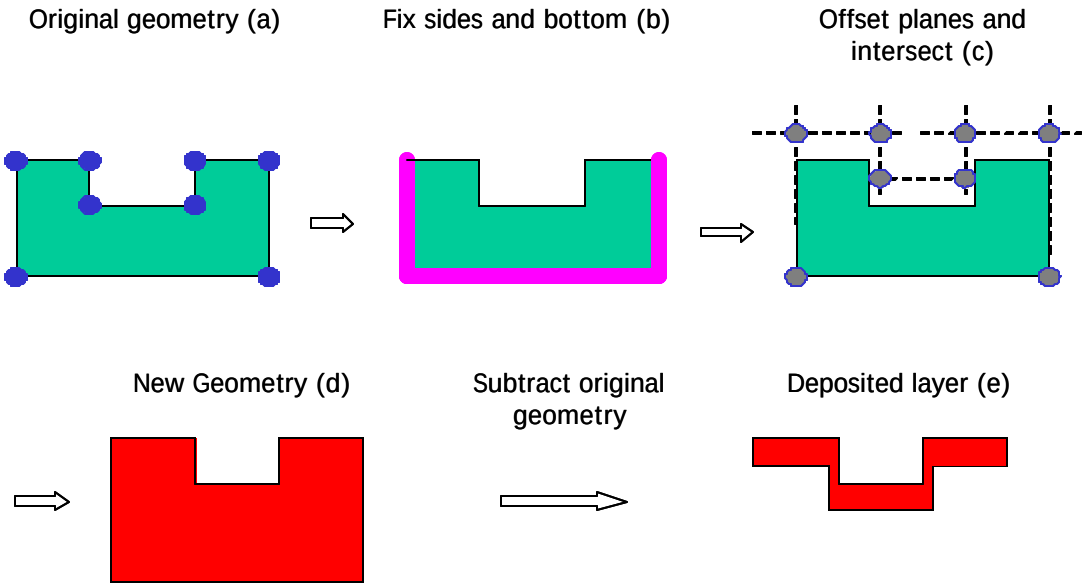


Figure 5.6: Offset solid algorithm in **[s]vfab**. The offset solid algorithm is used to emulate idealized conformal deposition. Conceptually, the goal is to create a solid (d) from which the original geometry (a) can be subtracted resulting in a solid representing the deposited material.

There are three types of etching defined in the CCPDS: sacrificial, surface, and bulk. In addition, **[s]vfab** has an additional method of etching using the level set kernel for physical process simulation. For implementation purposes, surface etching is further subdivided into two categories: ideal and undercut with angled sidewalls. The standard also subdivides surface etching into two types: inside and outside. Figure 5.7 shows the organization of the etch commands.

The sacrificial etch represents the release step commonly used in microsystems. In the present implementation, this corresponds to deleting the specified layer of material. This

is an approximation that assumes that the material is completely removed, and that the entire sacrificial layer is exposed to etchant. This is not always the case, and special procedures will have to be employed if part of the sacrificial material is to be maintained in the final device geometry (e.g. if it were completely enclosed).

The case of surface etching with vertical sidewalls and no undercut begins with extruding the mask in the z-direction by the appropriate depth of the etch. If "EtchMask=INSIDE" is specified, then this extruded solid is subtracted from each layer specified in the command step. If "EtchMask=OUTSIDE" is specified, then a two-step process is required. The first step involves creating a "negative" of the extruded mask solid by subtracting the extruded mask from a rectangular solid the size of the wafer with a thickness of the etch depth. The second step involves subtracting the negative mask solid from each layer specified in the command step.

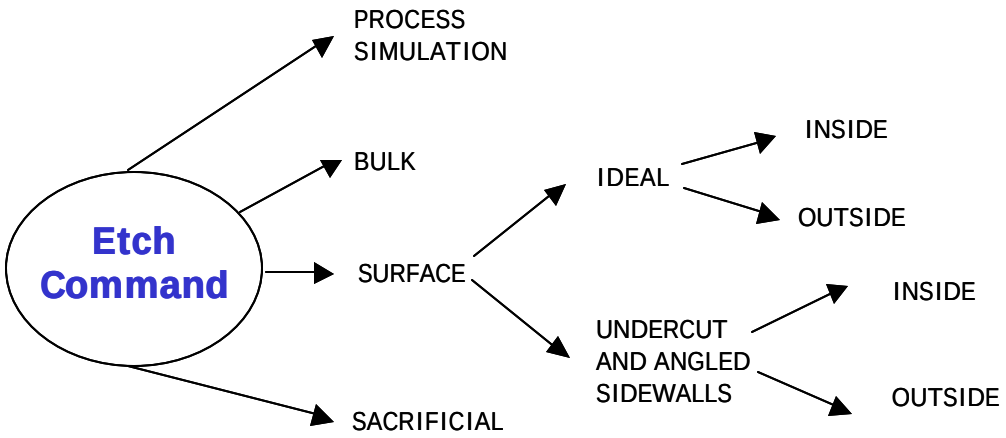


Figure 5.7: Etching types in **SVFAB**. The process simulation type represents an extension to the CCPDS standard. For implementation purposes, surface etching is divided into two subcategories.

Performing a surface etch with angled sidewalls and/or undercut is more complicated. The masks are required to consist of non-overlapping simple CIF box objects. The box is then scaled by the undercut value depending on the type of etch performed. Outside

etches require the box edges move inward by the specified undercut, while an inside etch requires the box edges move in the outward direction by the specified undercut. Using trigonometry, the image of the box is also calculated at the specified depth of the etch. The end result is eight vertices with known connectivity which can be used to create a trapezoidal solid. The aggregation of these trapezoidal solids is then used in place of the extruded etch mask, and the process of handling the "EtchMask=INSIDE" and "EtchMask=OUTSIDE" is identical to the vertical wall case. Note that only simple etch masks are handled properly with this implementation.

The final type of etching in CCPDS is anisotropic bulk etching (i.e. EtchType=BULK). The bulk etching option is a special case of surface etching with angled sidewalls. The first step is to find the minimum bounding box (MBB) of the specified etch mask. Using the resulting MBB as a pseudo mask, a surface etch with a sidewall angle of 54.7 degrees is then performed.

5.7.1.3 Process emulation macros

Process emulation macros differ in a significant way from the other process emulation commands. Namely, while these commands represent common IC processing steps (e.g. dopant implant and diffusion), the implementation proposed in the standard requires significant approximations and consists of performing a fixed sequence of steps using other commands from the CCPDS. For completeness, the major steps involved are listed, but the reader is referred to [72] for the detailed explanation of these macros. Following is an overview of the Grow and ImplantDiffuse macros:

1. Grow: This command is a gross approximation to the growth of thermal oxide on a silicon substrate. It does not predict residual stress or other physical characteristics of this processing step. It consists of the following sequence of fundamental steps: etch silicon that would be converted during the thermal oxidation, deposit oxide, and etch oxide where growth would be inhibited by a nitride mask.

2. ImplantDiffuse: This command allows the user to create multiple solid regions from an initial layer (e.g. to assign differ material properties to a region of higher doping). It consists of the following sequence of fundamental steps: etch initial layer, deposit layer of higher doping, and etch layer of higher doping.

5.7.2 Additional notes on implementation of the CCPDS

Recall that **SVFAB** is built on top of the **Geodesic** architecture described in section 2.3. In **SVFAB**, the scripting language Tcl is utilized to process the CCPDS file (which is typically only a few hundred lines of text) while coding the computationally intensive commands in C++ with bound Tcl-interpreter commands. The scripts were written in a modeler independent way by using Tcl-bound commands that have modeler implementations "hidden" from the interpreter level. The user can toggle between solid modelers using the "solid_setKernel" command (see [11]), and the scripts remain unaltered. An extensive set of general solid modeling commands has been implemented in the **Geodesic** modeler-independent API, but only a subset of these is required to implement the basic CCPDS standard as shown in Table 5.2. In addition to the "solid_" commands which typically create objects, a SolidModel object has a set of useful methods. **Geodesic** provides more methods than required to implement the basic CCPDS standard, so Table 5.3 summarizes only the required SolidModel methods to implement the standard.

5.7.3 Summary and limitations of the CCPDS

The CCPDS standard has several important limitations. First, it was designed to be a standard that could be implemented using only solid modeling operations as detailed above. Second, the standard is predominantly written from a 2-D perspective. For example, in surface etching the angle is not clearly defined in the general 3-D setting. Third, it is possible to perform a sequence of depositions and etches which make specifying subsequent steps impossible using only variables such as undercut and angle. Finally, several of the commands are overly simplistic. For example, a sacrificial etch

just removes a layer from the model, which can lead to unphysical results if part of the sacrificial layer was not accessible to the etchant.

In summary, the CCPDS is an extremely useful file format for specifying process flows relevant for MEMS simulation. Although it has some limitations, these can be overcome by providing extensions and clarifications to the basic standard. Unfortunately, the standard has failed to receive widespread adoption by the commercial MEMS CAD vendors, which limits its ultimate practical effectiveness.

Table 5.2: Solid Modeling Commands Required to Generate Geometry*

	General	**Create**	**Booleans**
Basic CCPDS	solid_getKernel solid_setKernel solid_readNative	solid_box2d solid_box3d solid_copy solid_extrudeZ solid_poly3dSolid solid_polyPts	solid_subtract solid_union
Domain Decomposition	*no additional*	solid_poly3dSurface solid_extrude	solid_intersect
Level Set	*no additional*	*no additional*	*no additional*

* Note that only the first six rows are required for generating geometry for the standard CCPDS, while additional commands are required if domain decomposition or level set simulation is desired.

Table 5.3: SolidModel Object Methods Required for Model Generation

	General	**Transforms**	**Classifications**	**Labels***
Basic CCPDS	GetClassName GetKernel GetPolyData WriteNative	Translate	*none*	ClearLabel GetLabel GetLabelKeys SetLabel
Domain Decomposition	*no additional*	Rotate Scale	Find Centroid	*no additional*
Level Set	*no additional*	*no additional*	ClassifyPt Distance	*no additional*

* see [11]

5.8 ADVANCED GEOMETRIC ALGORITHMS FOR MEMS PROTOTYPING

5.8.1 Domain decomposition

As discussed in section 5.3, the use of 3-D physical process simulation to obtain a geometric representation of a typical MEM device is typically infeasible. However, if three-dimensional process simulation is only used in smaller areas of design interest (e.g. a support) or complex surface geometry, the problem becomes computationally tractable. For device simulation, though, the entire geometry of the device is still needed. By using the domain decomposition algorithm given in this section, 2-D and 3-D process simulation can be integrated with other geometric algorithms previously discussed to generate more physically accurate geometry.

The process of building a solid model representing a deposited layer of material has been divided into three basic steps to reduce the computational cost and enable the use of available process simulators:

1. Decompose the domain into 1-D, 2-D, and 3-D simulation regions
2. Perform the appropriate simulations in each region
3. Combine the resulting simulations together to create a final 3-D solid model

Starting with a process flow and the layout masks, a geometry is created using conventional geometric operations. When a more precise representation of a layer (such as a layer of polysilicon to be used as the main structural component for actuation) is desired, the geometry is decomposed into 1-D, 2-D, and 3-D regions of simulation. Since it would be difficult and computationally intensive to use the 3-D geometry of the device to determine the regions of simulation, the etch masks are used along with knowledge of the process flow. There are four major steps in the domain decomposition algorithm:

1. Create a uniform two-dimensional grid to cover the wafer
2. Classify each box as a 1-D, 2-D, or 3-D simulation region based on masks

3. Enlarge the classified regions using heuristics.
4. Merge similar adjacent regions

Figure 5.8 shows an example of the domain decomposition algorithm described in this section. The figure shows a conformal deposition being performed on a non-planar initial geometry.

5.8.1.1 Creating a uniform grid

First, a tensor product (rectangular) grid is created using the deposition thickness (defined as the total thickness of deposition on a flat surface of the current step) rounded up to the nearest integer as the grid size. To be precise, "grid size" actually refers to the height and width of each individual box, and thus the total number of boxes is given by:

$$N = ceil\left(\frac{l\ w}{ceil(g)^2}\right) \tag{5.1}$$

where: N = total number of boxes (integer), l = length (float), w = width (float), g = grid size (float), and *ceil(x)* is a function returning the smallest integer not less than x.

5.8.1.2 Classification of each box

The next step is to loop over the N boxes and determine the nature of the simulation required in each box (see Figure 5.9). It is important to note that in the case when multiple etch masks have been used to create the surface topography, a simple superposition of the masks is used. The assumption here is that regions requiring higher dimensional simulation (2-D or 3-D) can only occur within a given distance (ε) of the edges of the etch masks. Physically, this makes sense because the effects of a feature, e.g. a step up, will only be felt in the local area of the feature on the wafer and deposition a "reasonable" distance away will be unaffected. Based on numerical experiments and empirical results, ε is assumed to equal to the deposition thickness for the current step.

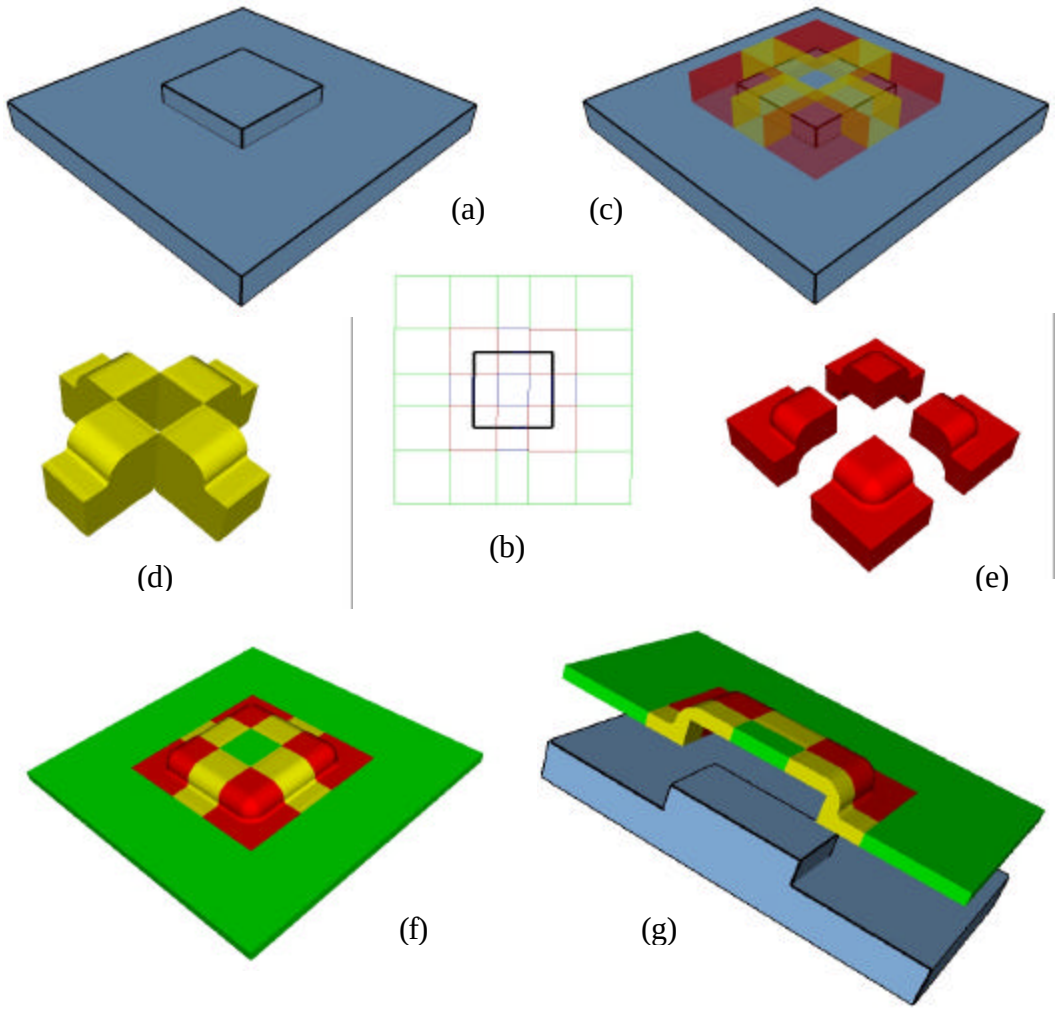


Figure 5.8: Domain decomposition algorithm for deposition. Figure (a) shows the original geometry created from a single etch mask. Figures (b) and (c) show the domain decomposition based on the etch mask. The red regions in (c) reflect corners requiring 3-D simulation, while the yellow regions reflect step ups requiring 2-D simulation (the rest of the surface can be created using 1-D simulation). Figure (d) and (e) show the intermediate solids created from the results of level set process simulation. Solid models representing the 1-D, 2-D, and 3-D regions are combined and then the original geometry (a) is subtracted to create the resulting deposited layer shown in (f) and (g).

In the case of a wafer that undergoes multiple (repeated) depositions utilizing the domain decomposition technique, ε is the sum total thickness of all of the relevant deposits. It is believed that this is a conservative estimate (i.e. upper bound) of the effects of a given feature and appears to be the case in practice.

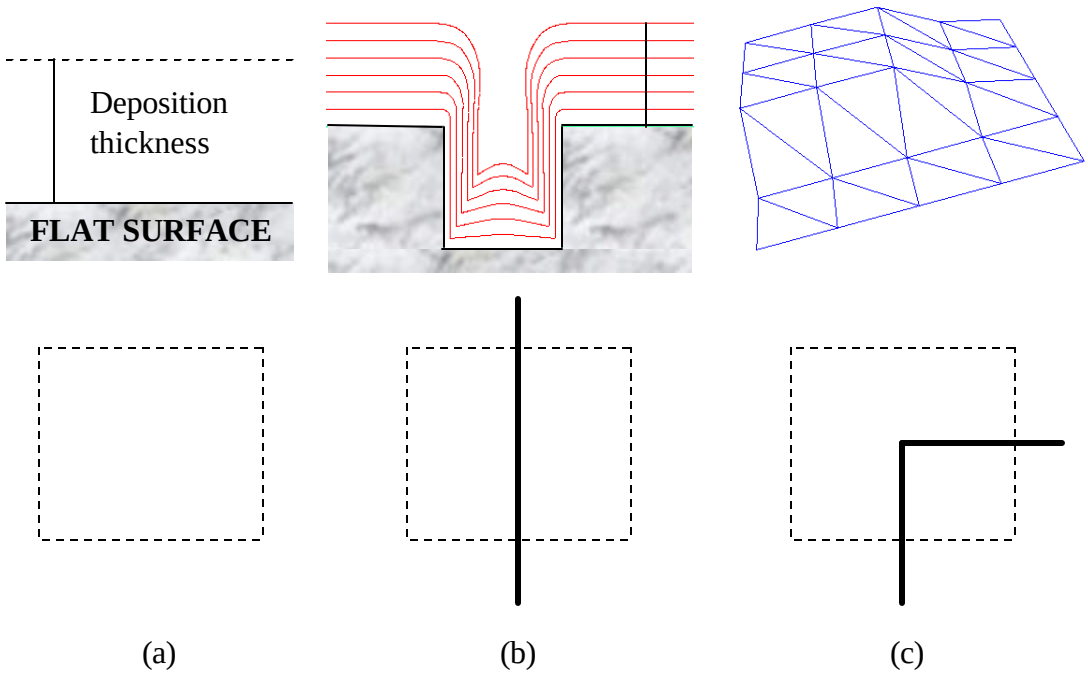


Figure 5.9: Classification of process simulation regions. The etch masks used to create the geometry (since the last planarization step) are superposed and a tensor product grid is created the size of the wafer box. Each region is then classified according to the number of mask edges contained inside of it. If the region contains no line segments (a), it is considered a 1-D region. If it contains a single mask edge (b), it is considered a 2-D region. If the region contains two or more mask edges (c), it is considered a 3-D region.

5.8.1.3 Enlargement of 2-D and 3-D regions

The next step in the process is to enlarge the 2-D and 3-D simulation regions appropriately. The best way to justify this step is by example. Take a hypothetical box from above which contains a corner and has been classified as a 3-D simulation region. Now, assume that the corner occurs at an extrema of the box, such as the lower left hand corner. By the classification procedure in section 5.8.1.2, the box may have as its left

neighbor a 1-D box (see Figure 5.10). This is incorrect because the assumption was that the feature (i.e. corner) has an effect in a region of influence given by ε. One possible course of action to handle this problem might be to query each 2-D and 3-D box to gain additional information about the exact feature contained within to determine its region of influence. The drawback is that this would require handling many special cases. A simpler procedure, which was implemented, is to create a safety zone around each 2-D and 3-D simulation region. In general, the etch mask edges may be in arbitrary directions. The current implementation handles only Manhattan style etch masks. Thus, for a 2-D box containing a "vertical" etch mask edge, its adjacent neighbor box to the left and right are converted to 2-D if either were 1-D. For a "horizontal" etch mask edge, its adjacent neighbor box above and below are converted to 2-D if either were 1-D. For 3-D boxes, all eight of the adjacent neighbors are converted to 3-D (see Figure 5.10).

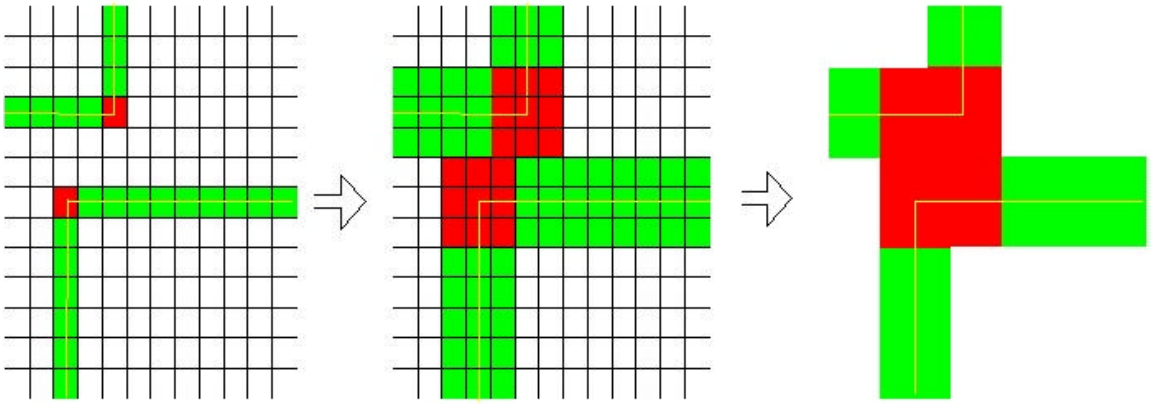


Figure 5.10: Classification, enlargement, and merging of simulation regions in the domain decomposition algorithm. An initial grid is created and the regions are classified (a) according to Figure 5.9. The regions classified as 2-D or 3-D are then enlarged (b) to ensure they are sufficiently large to satisfy the assumptions used in the domain decomposition algorithm. Regions of similar order (e.g. 3-D) adjacent to each other are then combined (c) prior to performing process simulation.

5.8.1.4 Combining adjacent 2-D and 3-D regions

Finally, adjacent boxes of the same order are combined to yield rectangular simulation regions of a given order. For the 1-D and 2-D regions this is a straightforward procedure. In the 3-D case, however, it is more complicated because initially separate 3-D regions might be in contact after the enlargement process described above. An efficient contour tracing algorithm is used to find the largest possible 3-D region (i.e. bounding box) by combining interacting or nearly interacting 3-D regions as seen in Figure 5.10. The major motivation for the rectangular 3-D simulation regions is that it typically simplifies the subsequent steps of the domain decomposition algorithm without significant cost penalty.

5.8.1.5 Performing the simulations

For the 1-D regions, the deposition thickness can be calculated using 1-D process simulation. This results in planar surfaces in the 1-D regions. In the 2-D regions, a 2-D process simulator (e.g. [69]) can be used to predict the geometry. Surfaces are created by effectively extruding the 2-D profile in the appropriate direction to yield a 3-D surface. For the 3-D regions, two simulation methods were explored. Early work involved using 2-D process simulations to create pseudo 3-D geometry [84,86,87,88]. More recent work has focused on using 3-D level set process simulation (see section 5.8.2) and this is now the preferred method.

5.8.1.6 Reconstruction algorithm

After the process simulations have been run to create the geometry for each region, the results have to be combined to create a continuous deposition surface. Currently, the method implemented is to create a solid from each simulation region and then union (boolean addition) all of the individual solids to create the final 3-D geometry. The motivation for the current implementation is its simplicity. However, robustness issues (predominantly with tolerances) within the solid modeler motivate the need for exploring alternative reconstruction techniques in the future.

5.8.1.7 Alternative decomposition algorithm

An alternative algorithm for decomposing the domain was also investigated inspired by the work in [81]. In this case, the vertices on the free edges of the polygons defining the etch masks used prior to the current step are sorted in a particular way to directly create the decomposition. Specifically, two separate lists are created where one list consists of the x-coordinates of each vertex and the other consists of the y-coordinates. For each list, each coordinate x_i is replaced by two coordinates defined explicitly as $x_i + \varepsilon$ and $x_i - \varepsilon$ where ε is given in section 5.8.1.2. Each list is then sorted by value, with coordinates less than ε from a neighbor coordinate being dropped. Along with the coordinates of the corners of the wafer mask defining the domain, these two lists define explicitly a tensor product grid where each node is given by (x_i,y_j) (where $0 \le i < M$, $0 \le j < N$, M and N are the number of coordinates in the x-list and y-list, respectively). The rectangular grid regions can then be classified as in section 5.8.1.2. The benefits of this method are that it reduces the number of box classifications required and may be easier to implement. The major drawback is that it requires a more sophisticated algorithm (which was not developed) to enlarge regions than discussed 5.8.1.3, limiting its use in the present work to devices created using simple etch masks.

5.8.2 Level set process simulation

[s]vfab contains a fully integrated general multi-dimensional level set kernel that can be used for physical process simulation. Recall from section 2.2.1 that the level set method refers to a mathematical formulation which represents the motion of a propagating hypersurface in the form of a partial differential equation (PDE) which is evolved in time. The major difficulty in physical process simulation using the level set method involves defining appropriate physics-capturing velocity functions for different deposition and etching processes. Using the currently implemented velocity functions in **[s]vfab**, only processes such as conformal deposition and wet etching can be accurately modeled. The object-oriented nature of the level set implementation inside of **Geodesic** (see [11])

facilitates the implementation of additional velocity functions. Section 6.4 provides examples of using the level set implementation available in **SVfab** for process simulation.

CHAPTER 6 MEMS APPLICATIONS

6.1 OVERVIEW

Five examples are included here to highlight the versatility and power of the **SUGAR** application and Internet-based client-server architecture outlined in the previous chapter. The examples include: geometric modeling of a RF switch test device, extending the CCPDS with an idealized reactive-ion etch implementation, conformal deposition and selective etching using level set process simulation, Internet-based computational prototyping, and an example of backend wafer processing for interconnect modeling.

First, an example of a RF switch gives a detailed look at using the CCPDS to generate the geometry of a device fabricated using the MUMPS process. Table 6.1 shows each processing step, the command specified, the resulting geometry, and implementation issues for the given step inside of **SUGAR**. In addition, the geometry of the switch was constructed using the domain decomposition algorithm and level set process simulation and the results were compared to the results generated using only process emulation.

The second example highlights the flexibility and extensibility of **SUGAR**. Although the default geometric etching algorithm is implemented as described in [72], it may be desired to more accurately account for the effects of anisotropic etching. This example highlights one way of implementing process emulation of reactive ion etching. Many extensions such as this one are possible because process emulation routines are implemented at a script level using lower level functionality in **SUGAR**.

The third example demonstrates the capabilities of the 3-D level set kernel. An example of conformal deposition and selective wet etch on a corner shows the geometric features that can be captured using process simulation instead of emulation.

Fourth, the efforts in Internet-based prototyping are summarized. It is shown that by using a standard web-browser with an Internet connection, geometry of practical interest can be generated on a remote server and displayed on a local machine.

Finally, an example of creating geometry of interest to the mainstream IC community is discussed. Backend wafer processing for interconnect modeling is shown to highlight the applicability of **[s]vfab** to other application areas of engineering interest.

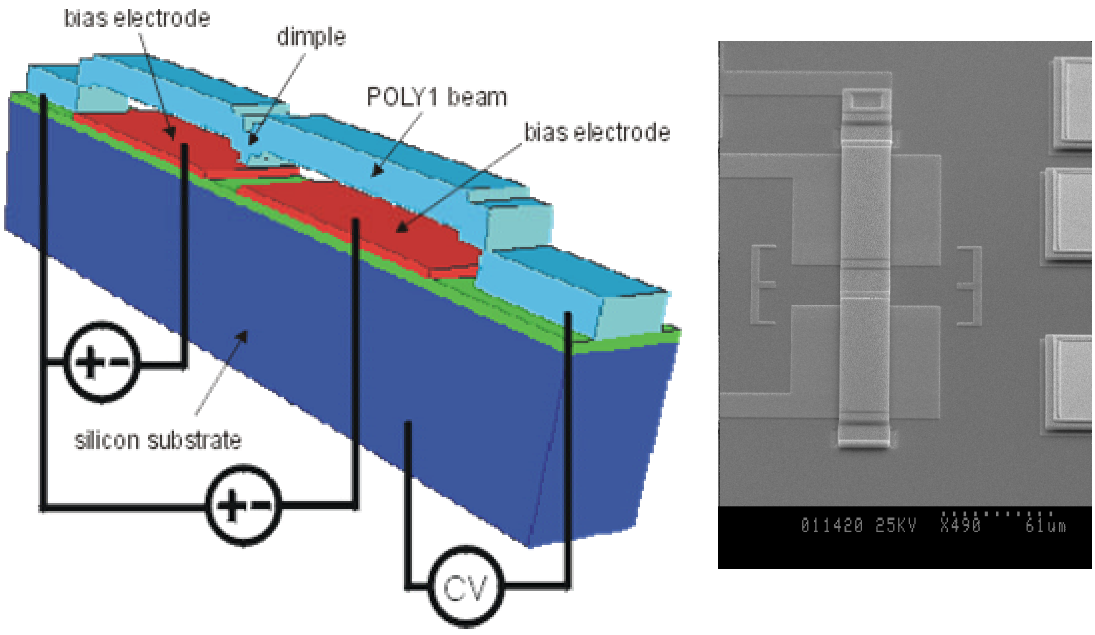


Figure 6.1: Dual electrode RF switch test structure (from [89]). The image on the left shows a schematic of the switch, while the image on the right shows a SEM of the device fabricated using the MUMPS process.

6.2 RF SWITCH TEST DEVICE

Extensive characterization and simulation of electrostatically actuated RF switch test structures was detailed in [89]. Figure 6.1 shows a schematic and a scanning-electron-microscope (SEM) image of a dual electrode device which was fabricated using the standard MUMPS process. Experimental investigation determined that the sidewalls of the device were not vertical, and that the undercut was occurring during several of the

etching processing steps. In addition, simulation results determined that properly capturing these processing effects were critical in quantifying the performance of the device [89].

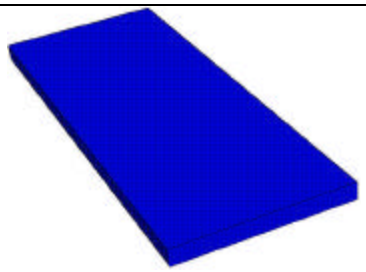
A CCPDS process flow for MUMPS (similar to that found in [72]) is given in Table 6.1. Along with the CIF file found in Figure 6.2, Table 6.1 shows the processing steps and discusses the implementation details for building a 3-D solid model representing the switch for electromechanical simulation.

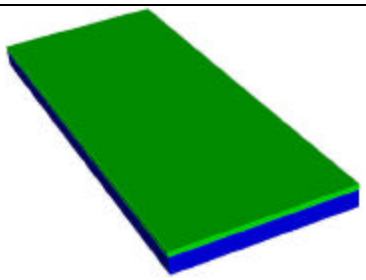
Table 6.1: Twelve Steps in the Process Flow for a RF Switch.

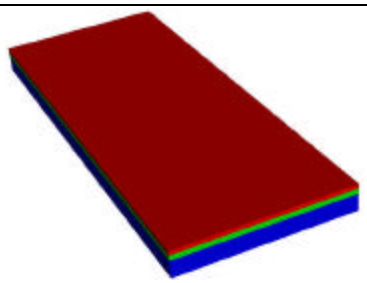
step	CCPDS command	Geometry after Command	Comments*

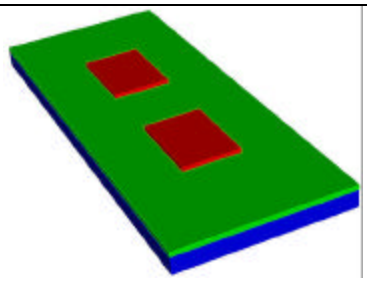
1	ProcessVersion={
	Version="1.0"
	}
			This command specifies the version number of the CCPDS file.

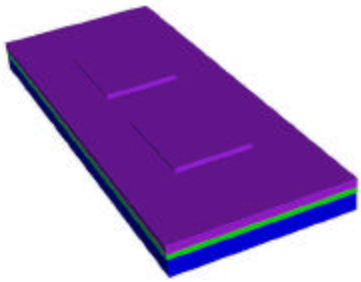
2	ProcessUnit={
	Unit=microns
	}
			This specifies the units to be used throughout the model creation process.

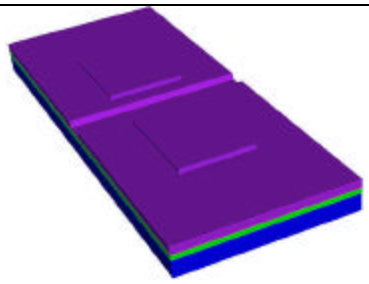
3	Wafer={
	MaskName="GND"
	Thickness=10
	Color=blue
	CIF_filename=masks.cif
	Lambda=1
	}
			Creates a 3-D rectangular solid of *Thickness* representing the wafer. It is oriented such that the bottom is in the plane z=0. The *MaskName* mask consists of a single CIF box which specifies the center, length, and width of the region of interest.

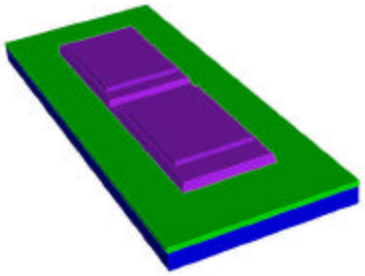
4	Deposit={
	DepositType=CONFORMAL
	Face=TOP
	LayerName="Nitride"
	Thickness=0.6
	Scf=c
	Color=green
	}
			Since the device is still planar (no etches have been performed), the deposit routine calls **vfabGeometricDeposit** which creates a rectangular solid of thickness properly oriented in space (with layer *centroid* = *overall_height* + (*Thickness* / 2)).

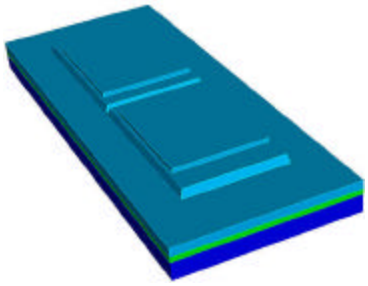
5	Deposit={
	DepositType=CONFORMAL
	Face=TOP
	LayerName="Poly0"
	Thickness=0.5
	Scf=c
	Color=red
	}
			This deposit is done the same way as step 4. All conformal depositions are performed this way until an etch is performed. Once an etch is performed, *device_is_planar* is set to FALSE for the rest of the steps (unless a MechanicalPolish or DepositType=FILL is performed).

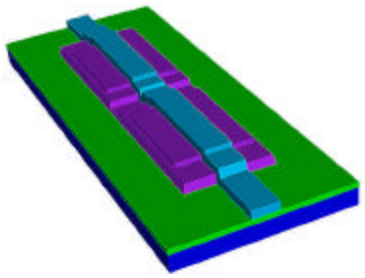
6	Etch={
	EtchType=SURFACE
	Face=TOP
	MaskName="CPZ"
	EtchMask=OUTSIDE
	Depth=3
	Angle=87
	Undercut=0.157
	EtchRemoves="Poly0"
	}
			This etch step causes the variable *device_is_planar* to be set to FALSE. Since this etch involves an undercut and results in angled sidewalls, **vfabAngledEtch** is used. Only the *Poly0* layer is affected by the etch since it appears within the EtchRemoves.

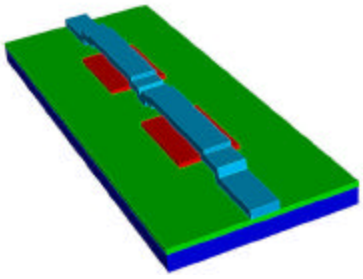
7	Deposit={
	DepositType=CONFORMAL
	Face=TOP
	LayerName="ox1"
	Thickness=2
	Scf=.01
	Color=purple
	}
			Since *device_is_planar* is FALSE, we have to use the more general **vfabOffsetDeposit** algorithm that in essence creates the deposited layer by offsetting the original geometry by *Thickness* along the surface normals.

8	Etch={
	EtchType=SURFACE
	Face=TOP
	MaskName="DIMP"
	EtchMask=INSIDE
	Depth=1.2
	Angle=75
	Undercut=0.32
	EtchRemoves="ox1"
	}
			This etch uses **vfabAngledEtch**. Note that EtchMask=INSIDE tells the geometry engine to remove the material "underneath" the mask layout. In actual fabrication this corresponds to a negative resist.

9	Etch={
	EtchType=SURFACE
	Face=TOP
	MaskName="MORE"
	EtchMask=OUTSIDE
	Depth=4
	Angle=75
	Undercut=0.32
	EtchRemoves="ox1"
	}
			This etch uses **vfabAngledEtch**. Note that EtchMask=OUTSIDE tells the geometry engine to remove the material from around the mask layout. In actual processing this corresponds to a positive resist.

10	Deposit={
	DepositType=CONFORMAL
	Face=TOP
	LayerName="Poly1"
	Thickness=2
	Scf=.01
	Color=deepskyblue
	}
			Again since *device_is_planar* is FALSE we use the general offset algorithm as in step 7. Note that it is very important that small features that would "fill in" such as dimples be removed from the masks by the user in a layout editor before using the offset solid algorithm.

11	Etch={
	EtchType=SURFACE
	Face=TOP
	MaskName="TOP"
	EtchMask=OUTSIDE
	Depth=6
	Angle=90
	Undercut=0
	EtchRemoves="Poly1"
	}
			Since there is no undercut or angle required by this etch, **vfabGeometricEtch** is used. This corresponds to simply extruding the mask in the z-direction by *Depth* and subtracting the resulting extruded-solid from the layer (Poly1) specified by the EtchRemoves token.

12	Etch={
	EtchType=SACRIFICIAL
	Face=TOP
	EtchRemoves="ox1"
	}
			The sacrificial etch removes an entire layer from the database. No visibility or etchant exposure calculations are performed.

* implementation details (italics are used for variables while bold text is used for actual procedure names found in **$vfab**)

```
(CIF file for canonical_3d.ccpds);
DS 1 1 1;
L GND;
B 60 500 0 0;
L DIMP;
B 100 10 0 0;
L MORE;
B 100 360 0 0;
L CPZ;
B 40 150 0 -90;
B 40 150 0 90;
L TOP;
B 30 500 0 0;
DF;
C 1;
E
```

Figure 6.2: CIF file for the dual electrode RF switch test device. The figure shows the layouts used to create the switch in Table 6.1.

In addition to the standard CCPDS commands, **[s]vfab** allows custom extensions in the CCPDS process flow file to utilize the advanced algorithms detailed in section 5.8. When process simulation of a conformal deposition is desired, the user sets a flag prior to the deposition step indicating to **[s]vfab** that domain decomposition and the level set method should be used. In addition, in the current implementation the user specifies a desired feature resolution (e.g. 0.1 micron) that is used to calculate the grid dimensions for the level set simulations. With the exception of setting the flag and feature resolution, the process flow remains identical to the one shown in Table 6.1. Figure 6.3 compares the geometry created using the standard CCPDS to the results obtained using domain decomposition and level set process simulation. As seen in the figure, the level set process simulation yields a more faithful representation of the geometry actually created using the MUMPS process. Figure 6.4 shows the device created in this example meshed and passed to MEMCAD for electromechanical simulation.

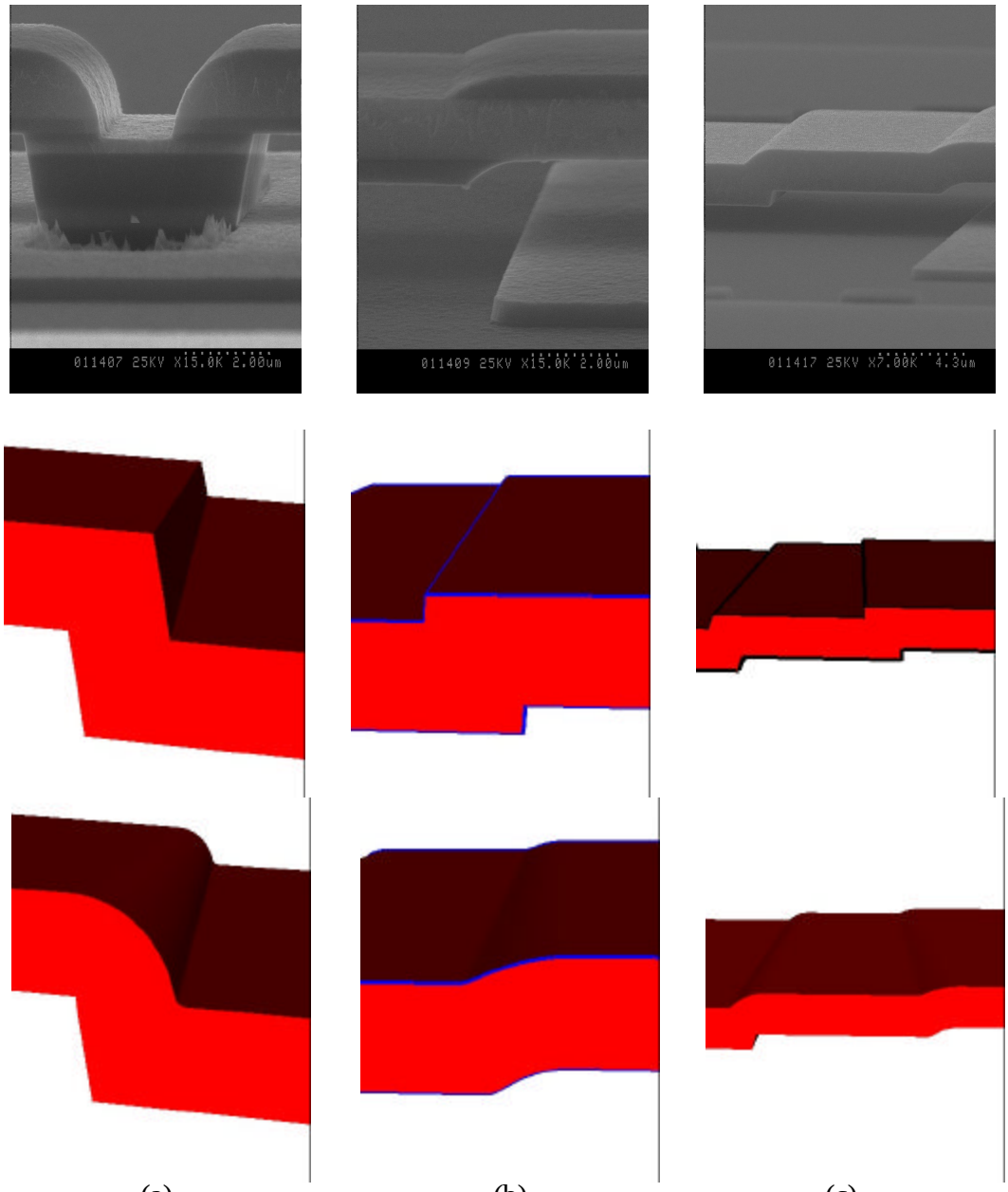


Figure 6.3: Comparison of actual RF switch geometry with models constructed using **[s]vfab**. Three different close-ups (a-c) are shown of the device from Figure 6.1. The rows of the figure show SEM images, the model constructed in Table 6.1, and the geometry created using the domain decomposition algorithm with level set process simulation, respectively.

Finally, the discussion in section 5.6.4 of utilizing multiple levels of physical accuracy in geometric modeling of MEMS is highlighted in this example. If several initial design iterations were carried out using simplified geometric models such as the one generated in Table 6.1, a more physically accurate geometric model (e.g. Figure 6.3) could be utilized to perform a detailed analysis of the final design.

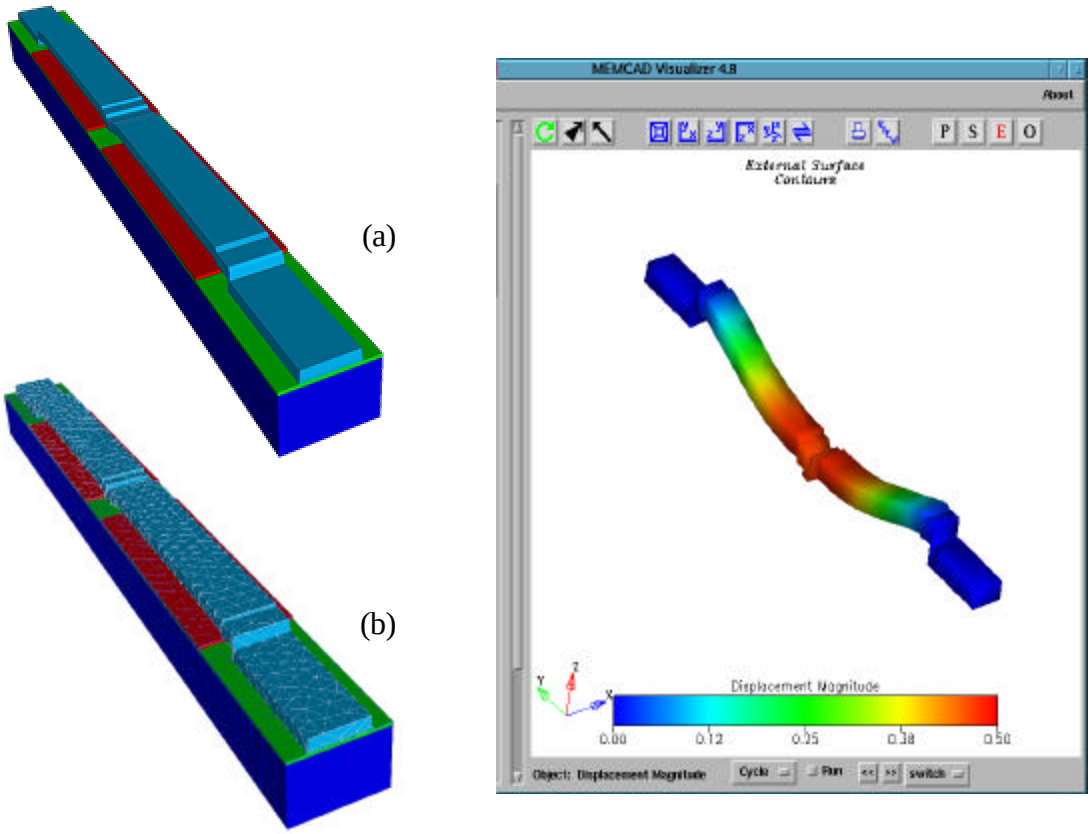


Figure 6.4: RF switch test device simulated in MEMCAD. The solid model was created using process simulation (a) and meshed (b) in **svfab** and then simulated using MEMCAD (c).

6.3 EXTENDING SVFAB – REACTIVE ION ETCHING

In the current implementation of the CCPDS within **svfab**, etching is performed by extruding the mask and subtracting the resulting solid from the layers specified by the

"EtchRemoves" option. While this emulates some physical processes well, on non-planar geometry it can be a poor approximation. Figure 6.5 shows a schematic of reactive ion etching. The yellow region on the left represents the material that should be removed by an idealized reactive ion etch of unit depth. The figure on the right shows what occurs using the standard scripts from **svfab** to perform the etch. In this case the yellow region represents material incorrectly remaining after the etch.

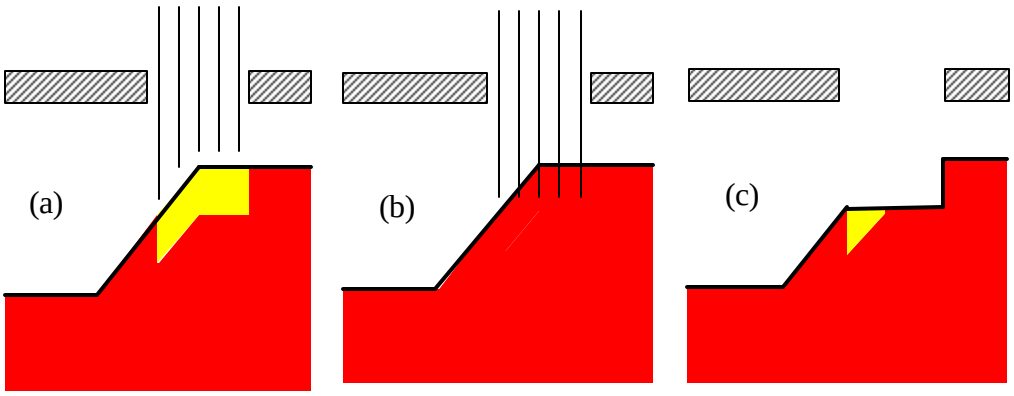


Figure 6.5: Idealized reactive-ion etch in CCPDS. Figure (a) shows in yellow the material which should be removed by an idealized reactive ion etch. Figure (b) shows the effective extrusion of the mask by a unit depth, and (c) shows the material actually removed by the etch simulated using an extruded version of the mask.

The pseudo code in Table 6.2 and Table 6.3 shows the **svfab** scripts for two different implementations of an idealized reactive-ion etch. This code assumes the following three objects exist:

1. 3-D solid model of a layer named "etchMe" to be etched by unit depth
2. 2-D polygonal solid of an etch mask named "myMask"
3. 3-D solid named "mems_device" which is the exterior of the entire device

Table 6.2 lists the pseudo code to implement the etch show in Figure 6.5. An improved idealized reactive ion etch is shown in Figure 6.6 and the corresponding pseudo code is given in Table 6.3. Figure 6.6f indicates a better approximation to reactive-ion etching on non-planar topography than the original algorithm discussed in the CCPDS standard

and shown in Figure 6.5c. This example also highlights the flexibility and extensibility of the **svfab** application developed in this work.

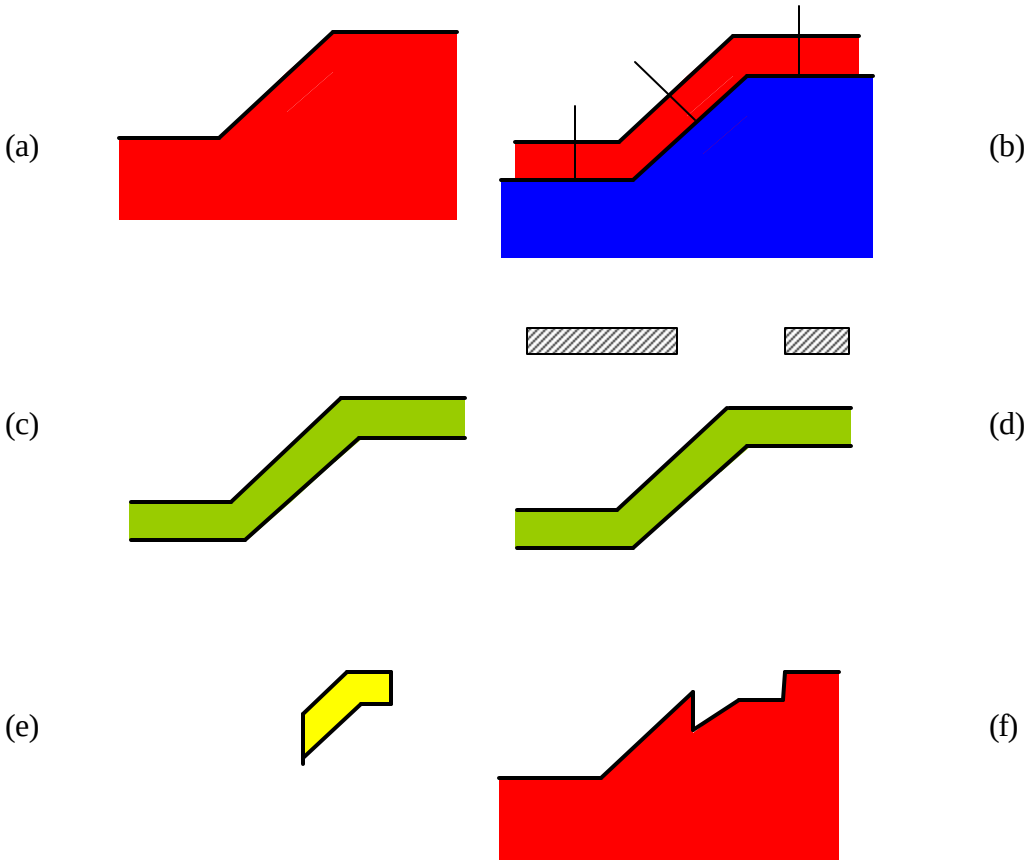


Figure 6.6: Schematic of improved reactive-ion etching emulation. Figure (a) shows the original geometry, while (b) shows the offset of the top surface in the direction of the inward normal. Figure (c) shows the result of the boolean subtraction of the solid created in (b) from the original solid in (a). This layer would correspond to a reactive-ion etch over the entire surface. Figure (d) shows the intersection of rectangular solids created based on the etch mask to represent the area of the wafer exposed by the etch mask. Finally, the resulting solid from (d) shown in (e) is subtracted from (a) to obtain the desired result.

Table 6.2: Pseudo Code for CCPDS Idealized Etch

Step	**Figure 6.5**
# extrude the mask in the z-direction by 1 unit # note: the 2d-solid mask is in the z=0 plane solid_extrudeZ –src myMask –dst extruded_mask –dist 1.0 # translate the solid of the extruded mask so that it is # properly oriented in space extruded_mask Translate –vec {0 0 (overall_height – deposition_thickess)}	(b)
# now do a boolean subtraction of the etch mask from the layer to be etched solid_subtract –result tmp –a etchMe –b extruded_mask # now replace the existing object with the desired result solid_copy –src tmp –dst etchMe	(c)

Table 6.3: Pseudo Code for Idealized Reactive Ion Etch

Step	**Figure 6.6**
# the offset solid routine works only with polyhedra, so we extract the facets # from mems_device: mems_device GetPolyData –result device_polydata	(a)
# note: several VTK procedures are called at this point to merge # coincident points and consistently orient the triangles of device_polydata. # offset the solid for a "negative deposition." That is, we want to move the # surfaces inward along the surface normal. Since by convention inside of # Geodesic the surface normals point inwards, we actually specify # a positive displacement. vfab_OffsetSolid –src device_polydata –dst tmp_polydata –offset 1.0	(b)
# the output of the offset solid command is a vtkPolyData object, # so now we create a solid polyhedra solid_poly3dSolid –result tmp1 –src tmp_polydata –facet Union	(c)

# now we subtract the offset solid from the original solid. This effectively # leaves us with a solid which could be subtracted from the original geometry # to do a unit depth ion etch over the entire device. However, we only want # to etch the area exposed by the mask. We now want to find the material # which should be removed because it is exposed by the mask. To do this, # we intersect a solid representing the # visible area with "tmp1". solid_extrudeZ –src myMask –dst tmp2 –dist overall_height	(d)
solid_intersect –result tmp3 –a tmp1 –b tmp2	(e)
# now do a boolean subtraction of the etch mask from the layer to be etched solid_subtract –result tmp4 –a etchMe –b tmp3 # now replace the existing object with the desired result solid_copy –src tmp4 –dst etchMe	(f)

6.4 PHYSICAL PROCESS SIMULATION USING THE LEVEL SET MODULE

The use of process emulation, and in particular solid modeling based methods to build geometry such as the CCPDS, works for many cases of interest. However, there are practical situations where topologic change can occur (e.g. via or dimple filling) and the "boxy" nature of the methods created using the CCPDS standard are insufficient. In these cases one must utilize more sophisticated techniques to create the desired geometry. In this example the level set kernel inside of **SVFAB** is used to simulate wet etching and conformal deposition. To simulate conformal deposition a constant normal velocity v = 1.0 is assumed. In the case of etching the velocity function simulates selective isotropic etching by altering the normal velocity along the front based on the selectivity of the materials being etched.

The setup required for simulating conformal deposition and isotropic selective etching is very similar in **SVFAB**. However, different physical interpretations exist for the surface embedded in the scalar field ϕ. In the case of deposition, the top surface of the device is embedded in the scalar field. In etching, on the other hand, the boundary of the "air" region surrounding the device is tracked in time (i.e. the scalar field ϕ implicitly

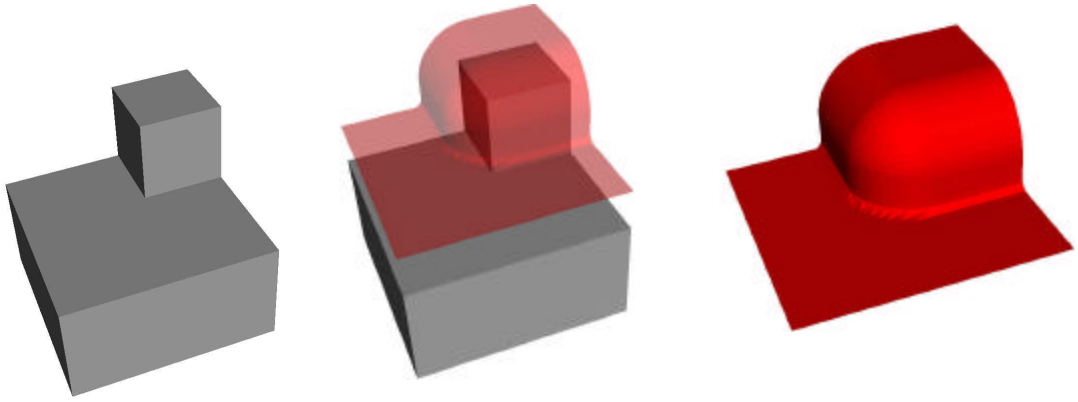
represents the surface of the air region). In the case of isotropic wet etching, this can be thought of physically as tracking the surface of the etchant. The reasoning behind the difference is as follows. In the present work, only the zero level set ($\phi = 0$) is assumed to be a material interface. In the case of conformal deposition, after a thin initial layer of material is deposited on the surface, the materials beneath the deposition surface do not affect the deposition process thus tracking the top surface of the wafer is the only requirement. However, a selective etch by definition involves multiple material regions, implying the zero level set can only be used to track the interface between the "air" region and the solid region comprised of multiple individual material regions.

Table 6.4 contains the pseudo code used in **svfab** to perform the calculations shown in Figure 6.7 and Figure 6.8. Figure 6.7 shows the results of conformal deposition on a simple corner, while Figure 6.8 shows an example of an isotropic etch of two materials with different selectivities.

Table 6.4: Pseudo Code for Level Set Process Simulation in **svfab**

```
#  define the desired deposition thickness / etch depth
thickness = 1.0

# create a solid with a raised center block in the center
solid_box3d -result box1   -dims {…} -ctr {…}
solid_box3d -result box2   -dims {…} -ctr {…}
solid_union -result input_geom -a box1 -b box2

# create air region
solid_box3d -result tmpair  -dims {…} -ctr {…}
solid_subtract -result air -a tmpair -b input_geom

# Set up grid domain
h = {0.500 0.500 0.500}
origin = …  (lower left hand corner of solution domain)
dims = …    (overall extent of simulation domain)

# Instantiate velocity function
#   deposition == constant function (setVConst)
#   etch       == selective etch function (lsetVSelEtch)

if (deposit) {
  velocity = 1.0
```

```
  setVConst vfn
  vfn SetV -v $velocity
}

if (etch)  {
 lsetVSelEtch vfn
 vfn SetRate -rate 1.0
 vfn AddRegion -obj box2 -sel 0.1
 vfn AddRegion -obj box1 -sel 1.0
}

# Compute time-step parameters to give desired deposition thickness.
# The level set kernel automatically calculates the maximum
# allowable time step based on the CFL condition.  In this case,
# since V = constant, we can directly calculate the number of times
# steps the kernel needs to perform to deposit  the desired amount of material.
numSteps  = …

# Setup level set core
lsetCore core
core SetTime -simTime -1.0 -cflFactor 1.0
core SetGrid -h $h -dim $dims -origin $origin \
                -type SparseGrid -bandExt {…}
core SetVelocity -vobj vfn
core SetTimers -flag 0

# initialize based on desired process to simulate
if (deposit) {
 core SetSolidSeed -solid input_geom
}

if (etch) {
 core SetSolidSeed -solid air
}

core Init

# the “lset_for” loop runs the level set simulations for a given number of time
# steps.
set verbose 0
lset_for core $numSteps {} $verbose 1 0

# extract a surface, which either represents the top surface of the new
# material in a deposition or the surface of the “air” region in the case of an
# etch.
core ExtractFront -out output_pd -closed 0
```



(a) (b) (c)

Figure 6.7: Conformal deposition using level set simulation. Figure (a) shows the original corner, while (b) shows a unit conformal deposition (the translucent red surface) created using level set process simulation. Figure (c) shows a magnified view of the level zero function representing the free surface of the deposited material.

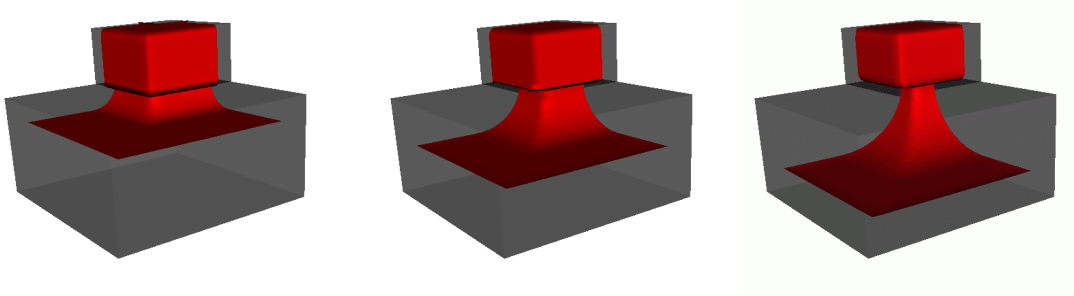


Figure 6.8: Example of a 3-D isotropic selective etch. The figure shows three time points (time increasing from left to right) in the evolution of the boundary surface, where the red surface indicates the level zero function. As can be seen, a small block of material on top of the larger block is more resistive to etch. In the case of wet etching, the level zero function physically represents the surface of the wet etchant.

6.5 INTERNET-BASED PROTOTYPING

The proposed client-server architecture for MEMS prototyping discussed in section 5.5 is demonstrated with several representative device examples. Figure 6.9 shows a representative mesh of a simple switch and a micromirror. In addition, Figure 6.10 shows a micromirror and a comb drive created using the web-based environment discussed in Chapter 5. Table 6.5 shows examples of file sizes for the various micro-electro-mechanical devices. Both VRML and OOGL (the file format used by JGV) are ASCII file formats. Most VRML browsers allow the input files to be compressed using the gzip compression algorithm. This greatly reduces the size of the actual file transmitted, as can be seen in Table 6.5. As the table indicates all of these geometries can be transmitted in a reasonable amount of time even over a low bandwidth connection (e.g. a 56 Kbps modem). Also shown are example file sizes for coarse meshes for each of the devices. Note that while a volume mesh was generated for each device on the server, only the surface mesh of each material region and the mesh quality statistics (see section 2.2.3) were transmitted back to the client for viewing. The preliminary results shown in this section indicate that Internet-based prototyping may be viable for designing microsystems in the future.

Table 6.5: File Size (in bytes) of Four Example MEMS Devices

Device	Geometry		Surface Mesh	
	VRML*	OOGL	VRML*	OOGL
Dual Electrode Switch	6116	43119	9926	61510
Comb Drive	52257	479440	87137	580514
Torsional Micromirror	4215	28743	11827	149934
Simple Switch	1429	8012	2762	30909

*compressed (with gzip) file size

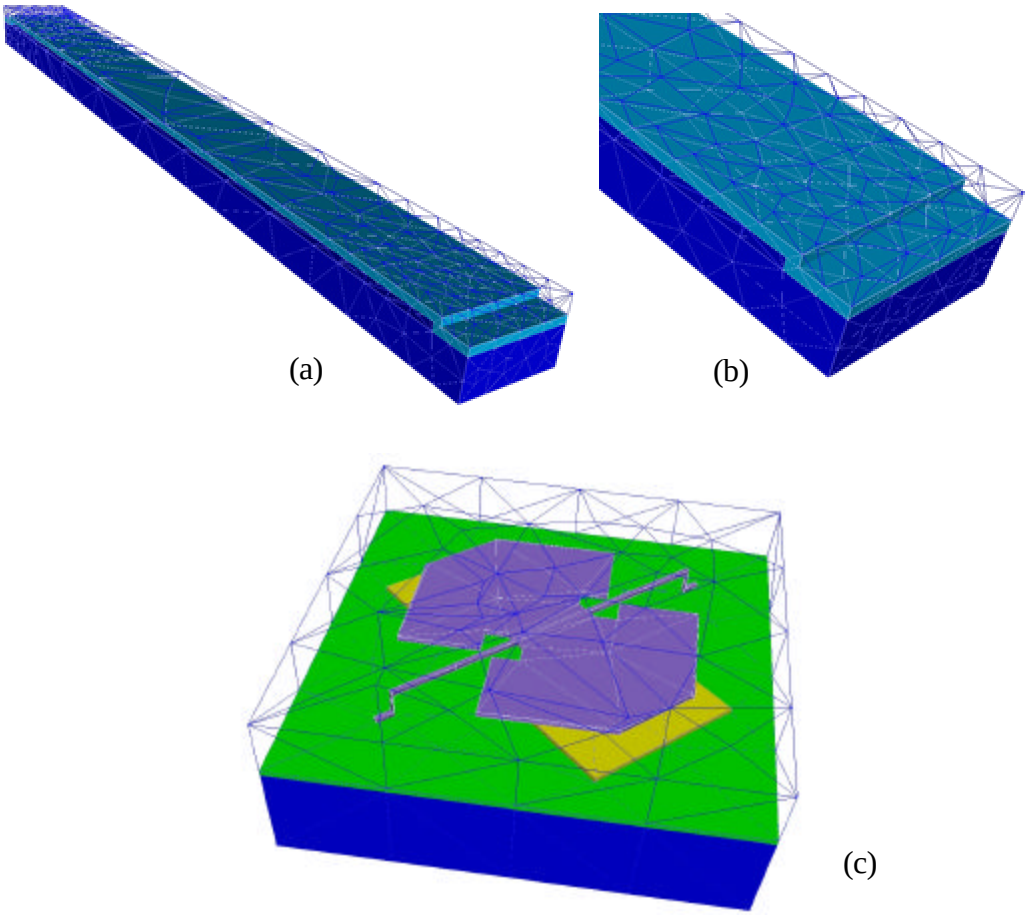


Figure 6.9: Examples of mesh generation for a simple switch and a micromirror. The simplified switch (a) and a magnified view of the step up (b) showcase automatic mesh generation as discussed in section 2.2.3 with higher mesh density near the step up. The mesh for the micromirror (c) was also created automatically. Note: All of the meshes are volume meshes with only the exterior surfaces of each layer shown for clarity. In addition, the air region around each device was meshed for finite-element electrostatic analysis.

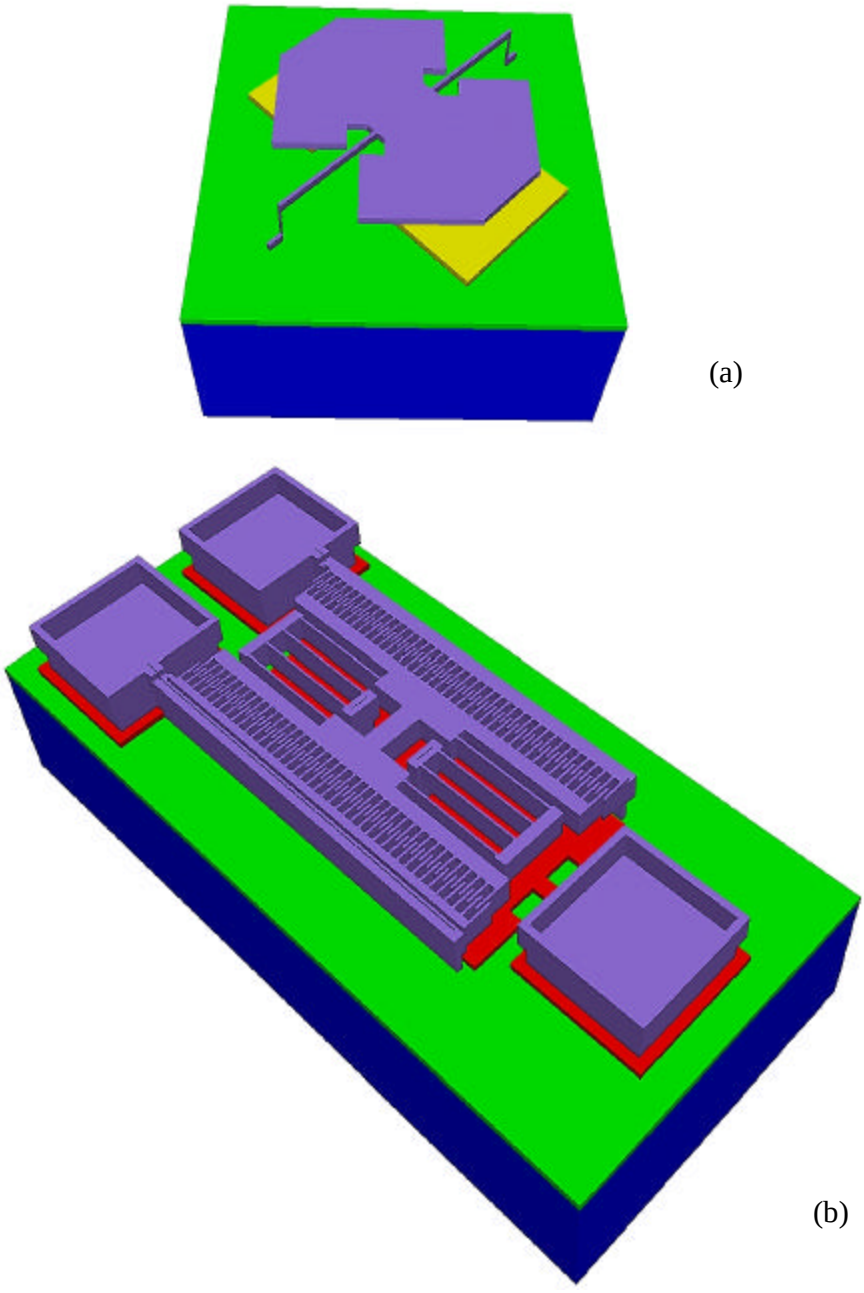


Figure 6.10: Two different MEMS geometries created using Internet-based prototyping. Figure (a) shows a micromirror while (b) shows a comb drive. The geometries shown in the figure are exaggerated in the deposition direction for visual clarity.

6.6 INTERCONNECT MODELING

This example shows the applicability of the geometric algorithms discussed in Chapter 5 to other areas of active engineering research interest including interconnect modeling. In the field of integrated circuits, "backend" processing has become increasing important. Backend processing refers to the fabrication of interconnects and the dielectrics that electrically and physically separate them. Interconnects are used to wire together active devices into circuits and connect them to the outside (see [90]). Typically interconnects are made of aluminum, although recently other materials such as copper are being investigated. Backend processing is gaining in importance because it represents a larger fraction of the total structure and process time and interconnect performance is starting to dominate the total speed of the fabricated devices.

Geometric modeling may play an important role in studying the effects of capacitance and inductance of the interconnect lines in modern IC design. Figure 6.11 shows an example of a simple integrated circuit with multiple layers of interconnects created using the **^{s}vfab** system.

6.7 SUMMARY

The applications shown in this chapter demonstrate the capabilities of the **^{s}vfab** framework, particularly for geometric modeling of micro-electro-mechanical devices. From a philosophical viewpoint, the work presented here has "come full circle." That is, backend wafer processing initially motivated the preliminary work in geometric modeling utilized here. Algorithms and a software framework were then developed for MEMS modeling as described in Chapter 5. The final example in this chapter, however, shows that the software developed in turn could be used to create geometry of interconnect structures of interest. This synergy of the two applications from a geometric modeling standpoint provides exciting potential dual use of the current and future research in interconnects and microsystems.

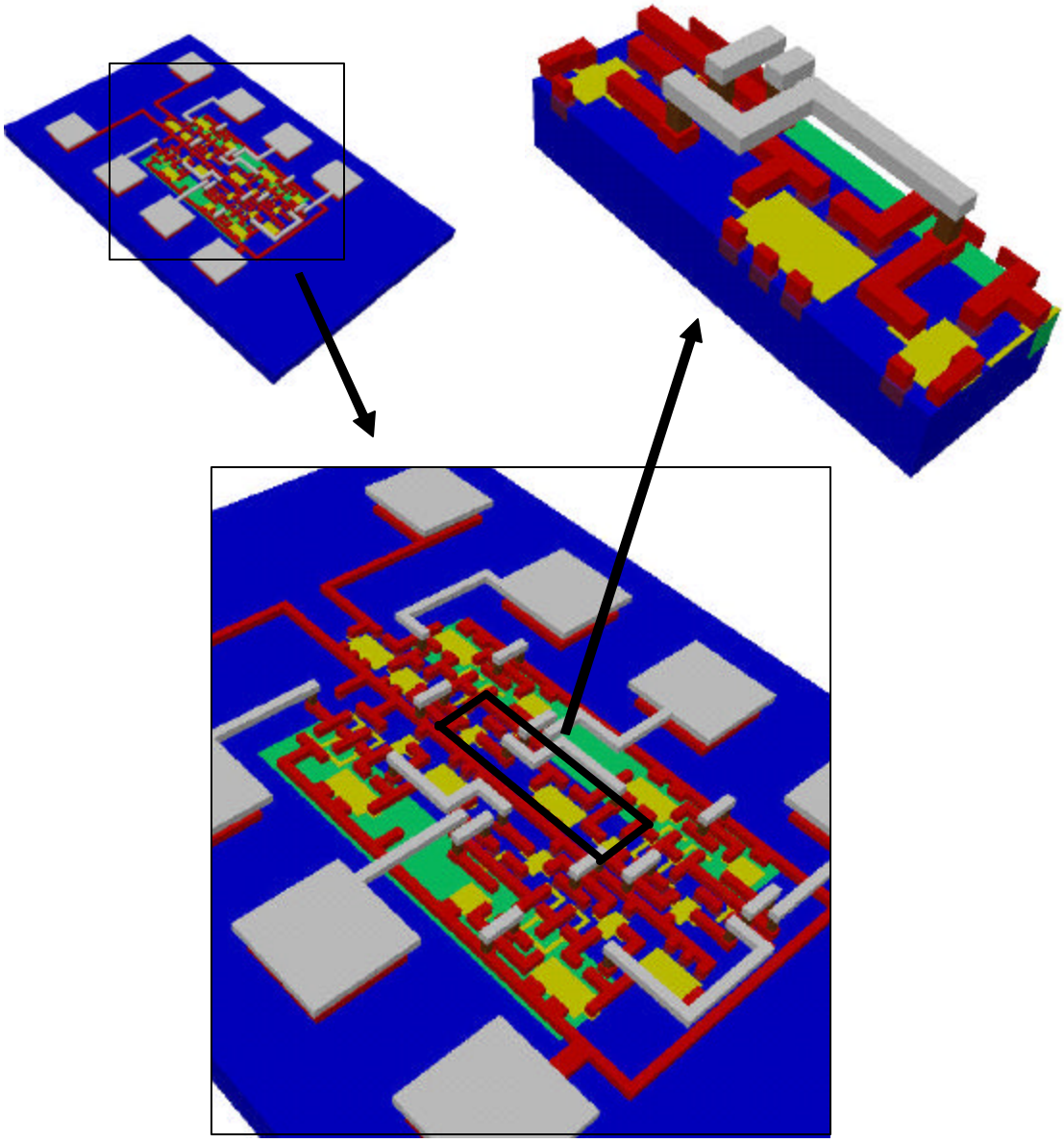


Figure 6.11: Interconnect structure created with **$vfab**. Geometric modeling may play a role in determining parasitic capacitance and inductance from interconnect lines relevant to IC design.

CHAPTER 7 CONCLUSIONS

7.1 SUMMARY OF THESIS

This thesis introduced the concepts and methods of computational prototyping, simulation-based design, and simulation-based medical planning and discussed their application in problems of engineering and medical interest. The first part of this thesis focused on the major component technology (geometric modeling, mesh generation, and simulation) involved in computational prototyping. Each of these components was introduced individually along with concepts from software engineering on a general software framework designed to facilitate simulation-based design.

The second part of this work discussed in detail a medical application of computational prototyping, namely simulation based medical planning. In the context of the present work, the idea of SBMP is to utilize tools from computer aided engineering to enable surgeons to preoperatively evaluate different surgical procedures in a patient-specific fashion prior to surgery. While utilizing the tools introduced in Chapter 2, methods specific to this application were developed and discussed in detail in Chapter 3. The application of the software developed in this work was then demonstrated in Chapter 4 for two retrospective case studies of treatment for aortoiliac occlusive disease. An example of flow in an abdominal aortic aneurysm was also shown.

A second application of computational prototyping for the simulation-based design of micro-electro-mechanical systems (MEMS) was presented. Chapter 5 introduced methods specific to MEMS simulation, with emphasis on geometric modeling of microfabrication. Chapter 6 demonstrated application of the methods developed in Chapter 5 for an RF switch test structure and Internet-based geometric modeling.

The rest of this chapter summarizes the specific contributions of this work and discusses future directions for additional research. It is worth repeating that computational prototyping provides exciting opportunities to improve the design process in traditional engineering applications and potentially revolutionize the field of medical planning.

7.2 SUMMARY OF CONTRIBUTIONS

This thesis detailed contributions in software architecture and geometric modeling with emphasis on vascular surgery and MEMS applications. A general computational prototyping software architecture was developed and described in detail that provided the foundation upon which two applications were demonstrated. The modular, open architecture of the system enabled the use of best-in-class component technology. In addition, multiple levels of programmatic access to the functionality provided by each module enabled rapid prototyping of new application-specific algorithms.

The framework was extended to include methods specific to simulation-based medical planning (SBMP). This system, named **ASPIRE^2**, was then used to reduce the time to construct preoperative geometric models from medical imaging data from months to less than a day. In addition, the system enabled a vascular surgeon, for the first time, to preoperatively create models representing alternative surgical procedures for a patient in approximately one hour per plan. This was the first demonstration of the geometric modeling necessary for SBMP in a clinically relevant time frame. This was achieved by streamlining the overall process as opposed to optimizing individual component technology.

In the area of MEMS, geometric algorithms were developed enabling a designer to go from mask and process flow information to 3-D geometry of varying degrees of physical accuracy for computational prototyping. These algorithms enabled the integration of idealized process emulation with level set based process simulation in the same software framework. In particular, a domain decomposition algorithm was developed that

significantly reduced the computational expense in performing process simulation for devices of the size and aspect ratio typical of micro-electro-mechanical devices.

7.3 SUGGESTIONS FOR FUTURE RESEARCH

With respect to simulation-based medical planning, this work provides motivation both by strengthening an emerging application and highlighting the technical challenges that remain for clinical application. A major contribution of this work was the demonstration of geometric model construction and operative planning in a clinically relevant time frame. In this sense, this work provides application-pull for improvements in medical imaging. Existing medical imaging technology may be sufficient for many diagnostic purposes. For quantitative applications such as planning surgical interventions, however, the examples demonstrated in this work highlight the need for improved and novel imaging techniques for quantification of volumetric flow and velocity patterns (particularly in small vessels). In addition, this work provides application-push technology where the ability to process imaging data and perform numerical flow simulations within a reasonable amount of time can be utilized to improve and optimize pulse sequences for MRA and PCMRI data acquisition.

This work also motivates research into better understanding the hemodynamic conditions involved in vascular disease progression. That is, a more quantitative understanding of the role of wall shear stress, particle residence time, and areas of complex flow phenomenon carry more clinical significance if the surgeon can preoperatively design and evaluate different procedures to control these quantities as enabled by this work.

The limitations and major sources of error discussed in section 4.2 provide insight into specific future research directions relevant to simulation-based medical planning. First, geometric inaccuracies in the preoperative model construction result primarily due to limitations in the data acquisition, current techniques used to construct models, and lack of validation. Improvements in medical imaging were discussed above, but additional

future work could include investigation of three-dimensional image segmentation techniques (e.g. 3-D level set segmentation). Selective use of 3-D image segmentation in conjunction with the 2-D techniques currently employed may simplify aspects of the preoperative model construction process and enable more accurate capturing of inherently 3-D geometric features such as bifurcations.

The geometric inaccuracies introduced in the operative planning are due to a combination of software limitations and a need for a better understanding of the *in vivo* behavior of implanted medical devices. Potential areas of research from a software perspective include volume rendering and virtual reality to help create environments to enable a surgeon to quickly and intuitively create different surgical procedures.

In the area of vascular modeling, further research directions include removal of the simplifying assumptions of rigid vessel walls and improving boundary conditions. Allowing deformability of the vessel walls, possibly through solid-fluid coupling, will yield more realistic pressure solutions and eliminate much of the need to manipulate experimentally determined volumetric outflow rates as described in Chapter 4. To evaluate the efficacy of aorto-femoral reconstructions, boundary conditions capable of representing states of exercise in addition to rest are necessary. That is, the case studies presented in this work do not indicate an increase in infra-renal flow during resting conditions postoperatively. Thus, to evaluate the ability of a given procedure to alleviate claudication symptoms will require simulations under exercise conditions.

The future research in the field of microsystems will likely maintain the current emphasis on finding new applications that can take advantage of the unique properties of microscale sensors and actuators. The role of computational prototyping will be to reduce the time to market for new devices, which in turn will expand the possible range of applications. Future work in developing an open-source toolkit (possibly modeled after the successful Visualization Toolkit) for MEMS geometric modeling and simulation would significantly increase the role of computational prototyping in the field. In

addition, nano-electro-mechanical systems will provide research opportunities as quantum effects begin to play a role in device fabrication and performance.

APPENDIX A GEODESIC AND ASPIRE^2 IMPLEMENTATION DETAILS

A.1 OVERVIEW

This Appendix gives a brief introduction to the implementation details of the **Geodesic** framework and the GUI of the **ASPIRE^2** system. Explicit documentation of all of the functionality embedded in the framework is not provided (see [11] for partial documentation). First, issues relating to software engineering will be introduced. Then selected GUI's from the **ASPIRE^2** system are shown.

A.2 SOFTWARE ENGINEERING

Software engineering plays a critical role in simulation-based design and simulation-based medical planning. The integration of discretization, simulation, and visualization requires a significant amount of custom application-specific code. In addition, the software architecture must enable the use of best-in-class component technology.

As an illustration of the complexity that can be encountered, consider the **Geodesic** framework and associated applications discussed in this thesis. The code at the time of this writing consists of: 102 C++ source code files, 106 C++ header files, and 118 Tcl script files. There are approximately 50,996 non-comment lines of C++ source code (out of a total of 88,005). In addition, there are approximately 53,222 lines of Tcl script code (out of a total of 85,509). There are currently 1,403 Tcl procedures, of which approximately 457 are "general" functions (i.e. not-explicitly associated with a particular widget or GUI window).

Numerous practical considerations (e.g. number of developers, available computational resources, likely future trends in computation) must be considered when developing a

software strategy. For example, to allow for multiple developers, the source code in this work was controlled using the concurrent versions system (cvs). Cvs allows multiple users to check in / out changes from a common source repository and will merge in the changes as appropriate.

The selection process for component technology in this work focused on the key issues of performance, platform-independence, and extensibility. In the case of visualization and the integration language, open-source code projects were selected. By definition, open-source projects provide free access to the entire source code for the project. Having the source code provides considerable flexibility, and with the advent of the World-Wide-Web, newsgroups often provide significant free technical support. Commercial implementations for mesh generation, solid modeling, and simulation were utilized because of their superior performance compared to freely available software.

A scripting language, the Tool Control Language (Tcl), was selected to facilitate integration. Tcl provides the ability to integrate low-level code (i.e. C/C++) into a framework while still providing the ease of use of a scripting language. Tcl has been ported to nearly all common platforms, so Tcl places no restrictions on the computer hardware or software utilized. In addition, a companion library to Tcl named Tk provides a robust, platform-independent toolkit to create GUI's necessary for simulation-based medical planning. All of the GUI's shown in the next section are pure Tcl/Tk.

Geodesic was originally developed on a Solaris platform. However, leveraging the multi-platform nature of VTK and Tcl/Tk greatly simplified the port of the software to a non-UNIX the environment (i.e. Microsoft Windows). To simplify the build process, the same makefiles are used to compile under both UNIX and Windows with appropriate variables and parameters being selected depending on the platform.

To test for continuing cross-platform support, custom scripts were created and the source code was automatically compiled nightly on Sun, SGI, HP, and Windows platforms. The

system can be configured to email the developers when nightly compilation fails, indicating that platform-dependent code has been checked into the source code repository. The next logical step would be to develop a test suite that could run nightly to test for unexpected functionality changes.

Documentation of the source code is critical in the longevity of many software endeavors. In the case of **Geodesic**, extensive HTML documentation is generated using a freeware Tcl program named Tkautodoc. By including specially formatted comment lines, HTML documentation can be generated automatically from the Tcl scripts. For the C++ code, similar results can be achieved utilizing Doxygen. In addition to developer information, end-user documentation (particularly of the GUI's described in the next section) must be provided.

The following section provides a brief overview of selected GUI windows from the **ASPIRE**^{2} system. Most of the underlying functionality behind the GUI's was discussed in Chapter 3.

A.3 ASPIRE[2] GUI

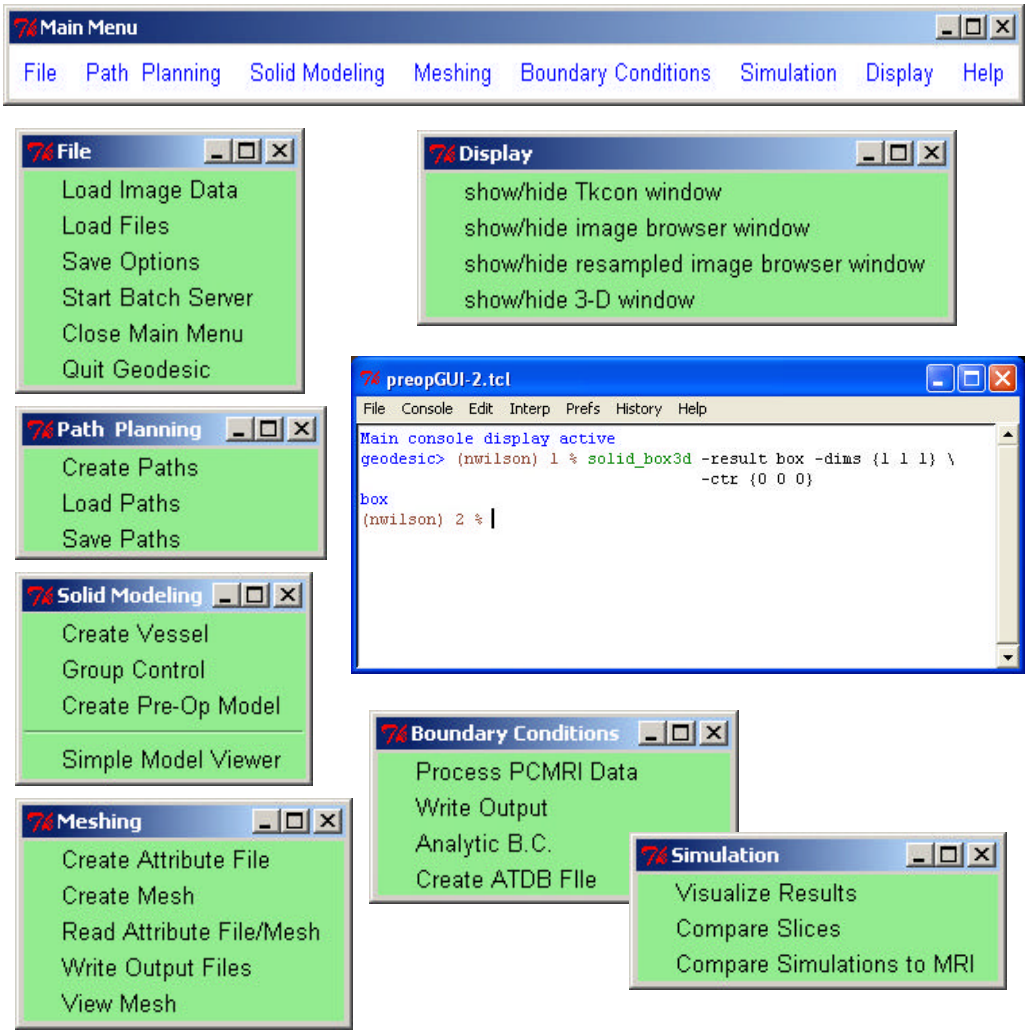


Figure A.1: Main menu. The main toolbar (top) guides the user through the common steps to go from medical imaging data to simulation results. Also shown are the tear-off pull down menus (File, Path Planning, Solid Modeling, Meshing, Boundary Conditions, Simulation, and Display) from the toolbar. In addition, in the center of the figure the console is shown where the user can enter Tcl-bound **Geodesic** commands.

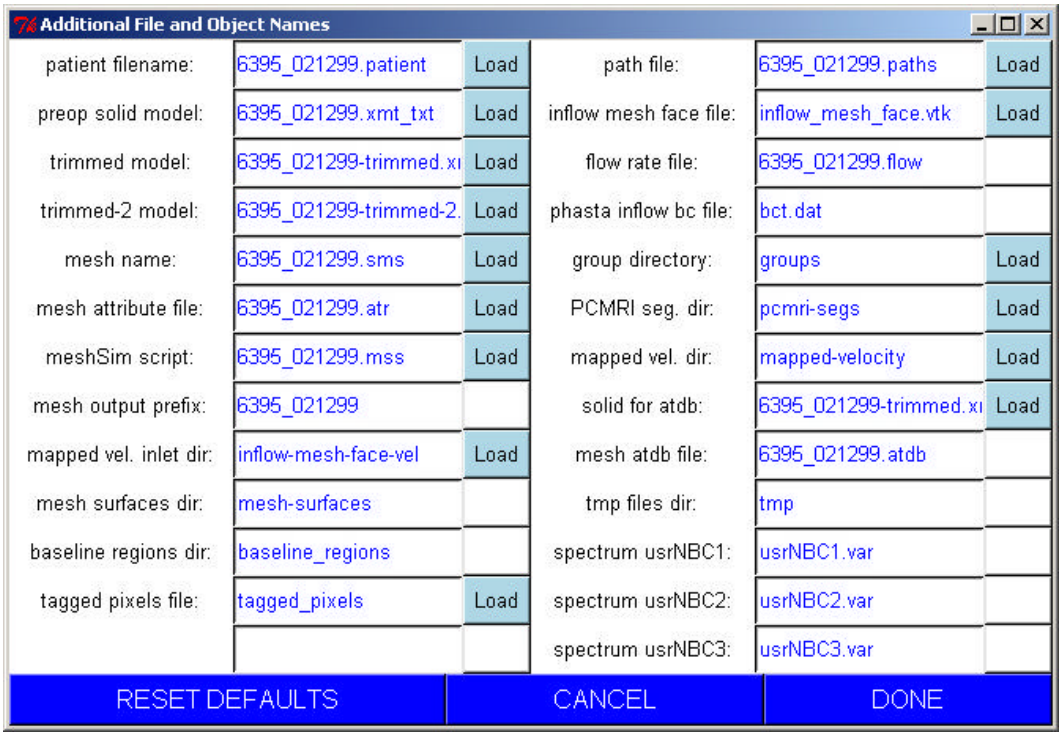


Figure A.2: File control menu. There are numerous files involved in the SBMP process, so the file control menu helps organize the files for a given patient.

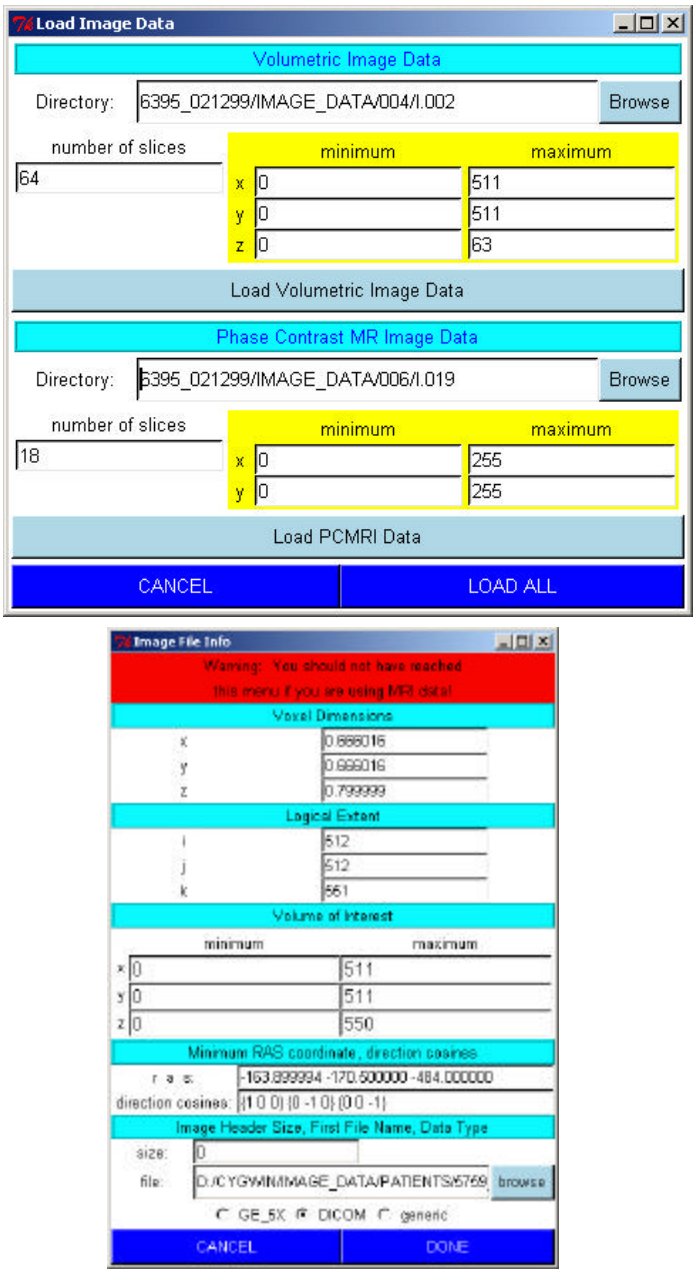


Figure A.3: Load image data menus. These menus allow the user to load image data (by default in a proprietary GE format). If the image data contains no header information, the user can specify the pixel spacing and extent. DICOM data is supported by first running the data through an external program to extract the appropriate header information and create flat image files which can then be read in by the system.

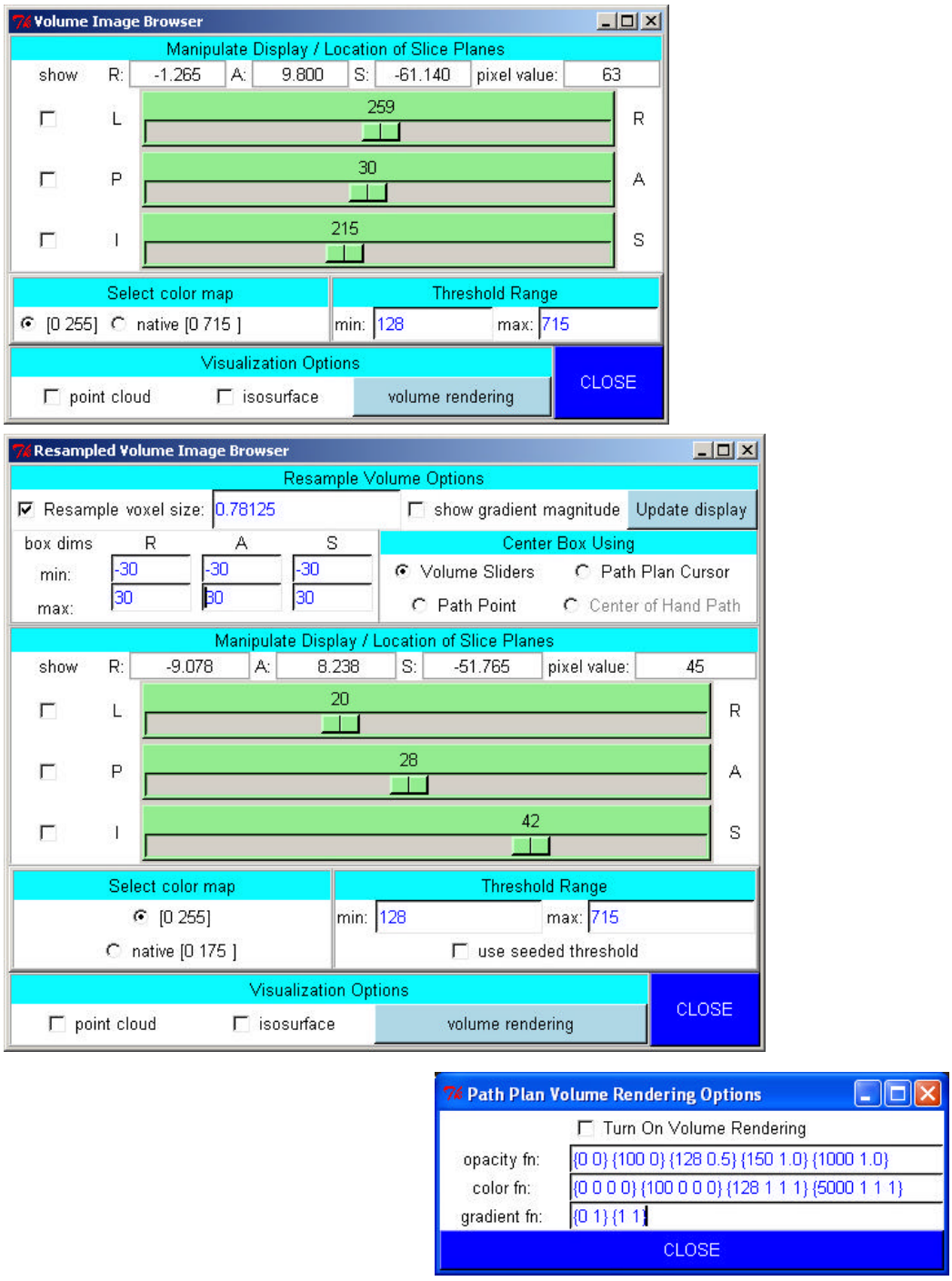


Figure A.4: Image visualization menus. These GUI's control the image visualization as discussed in section 3.5.1.

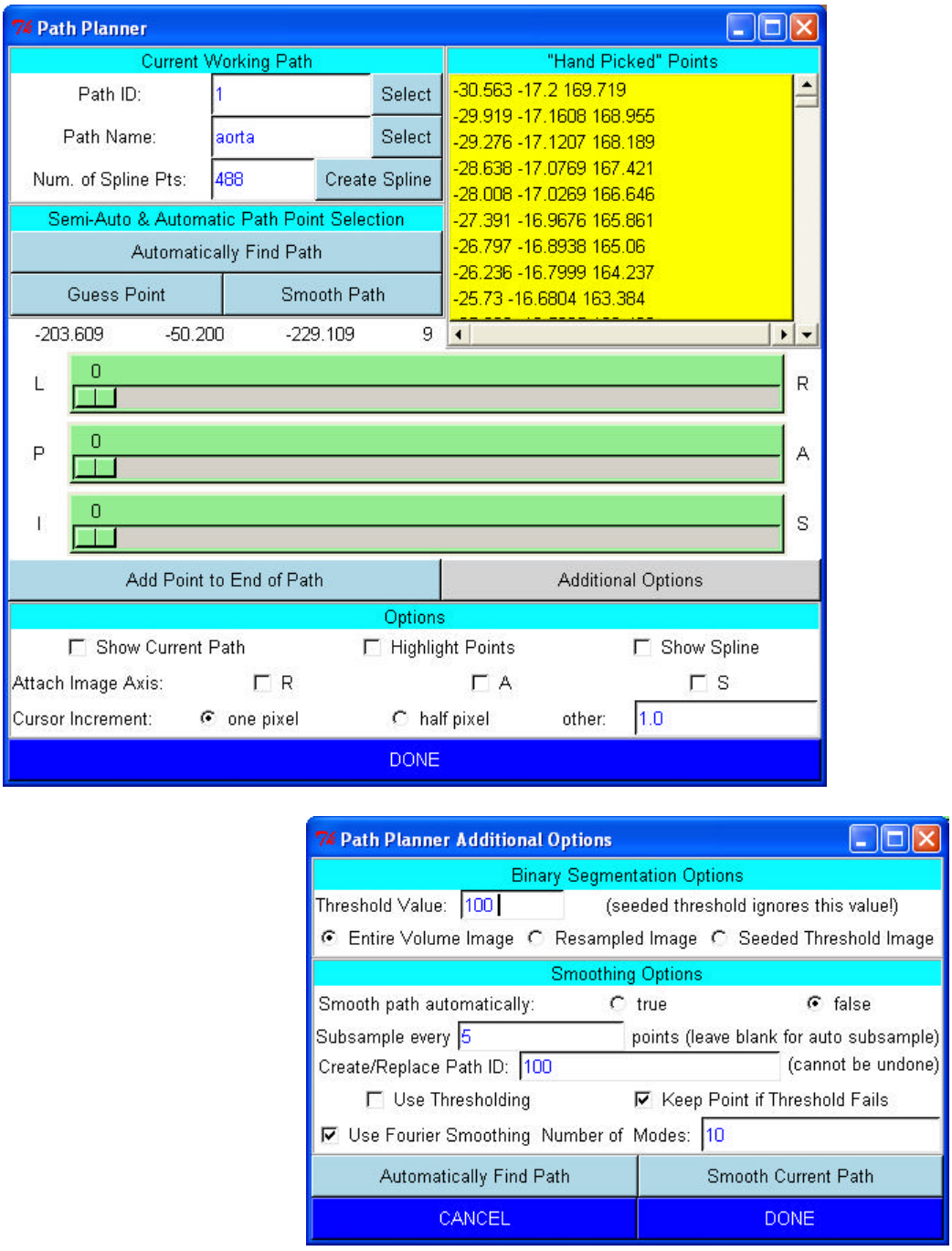


Figure A.5: Vessel path planning menus. These GUI's are used to create vessel paths as discussed in section 3.5.2.

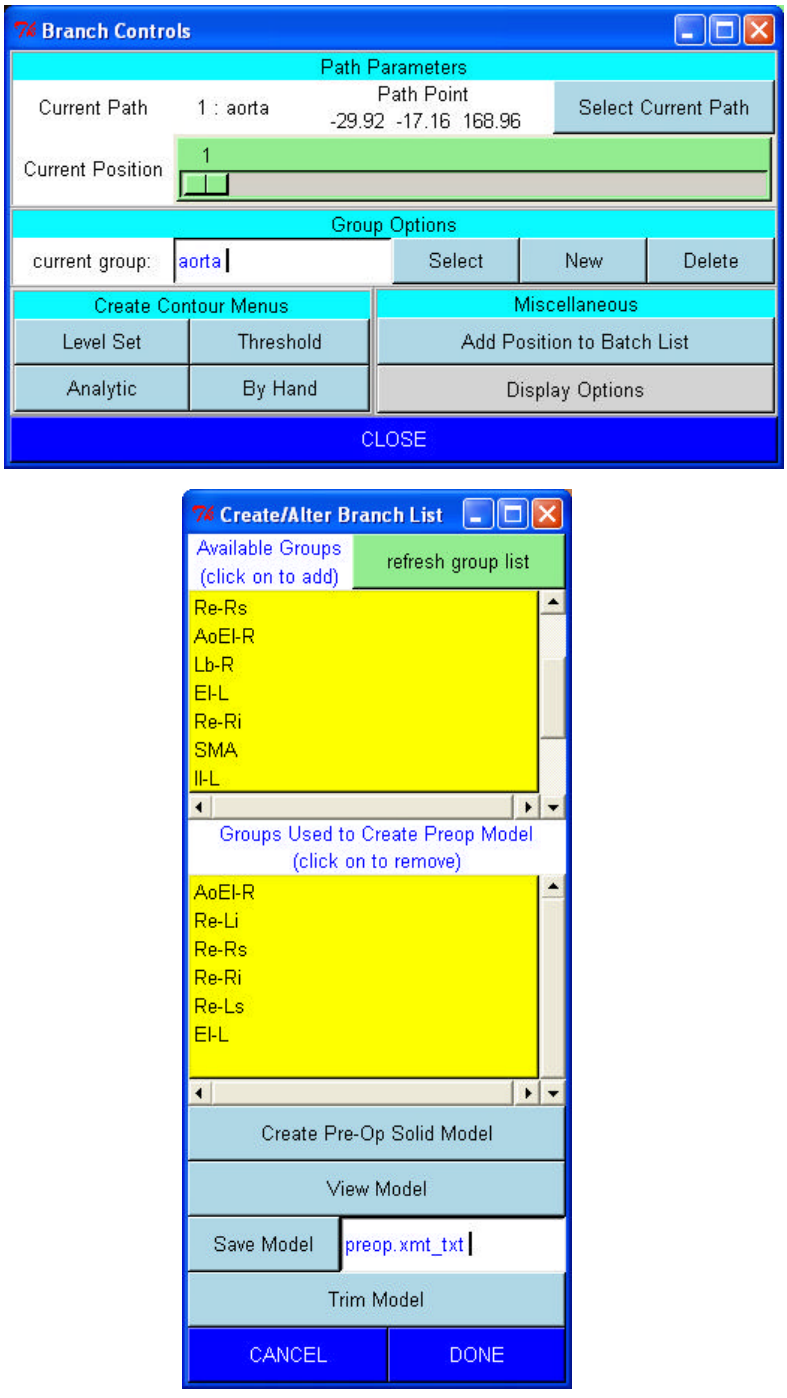


Figure A.6: Preoperative geometric model construction menus. These menus are associated with the functionality described in section 3.5.3 and 3.5.4.

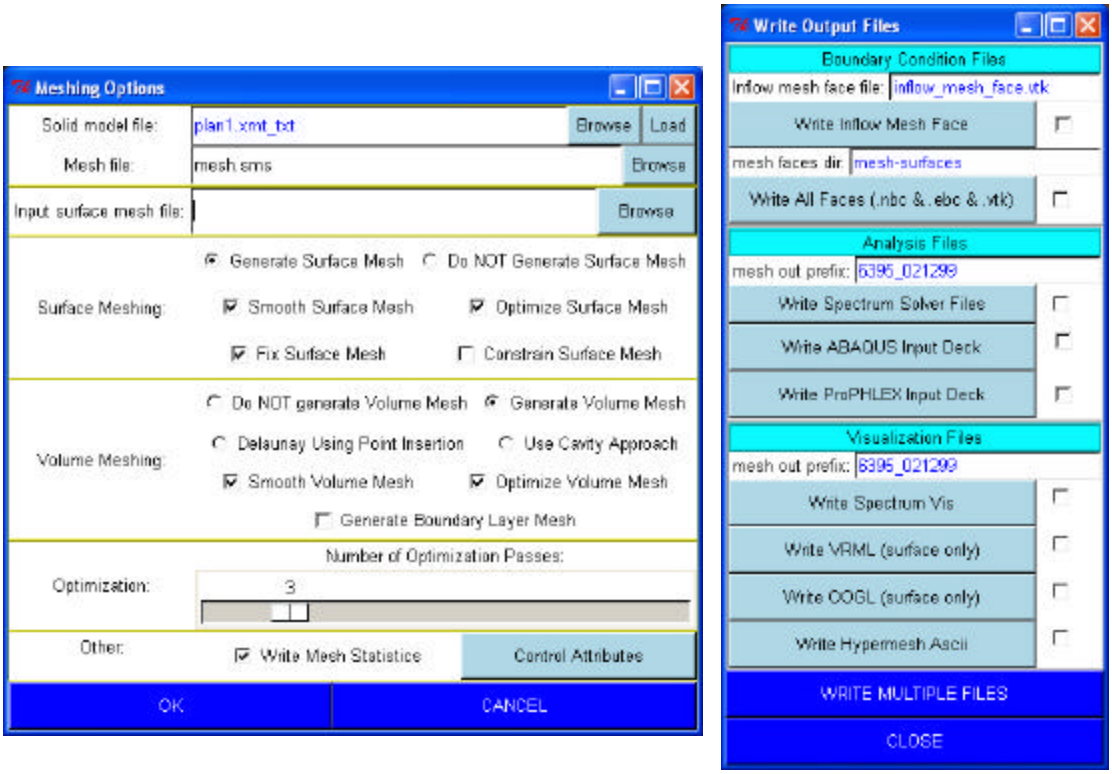


Figure A.7: Meshing control and output menus. The menu on the left is used to specify parameters for the automatic mesh generator. The menu on the right is used to write out node and element information of interest for boundary condition specification.

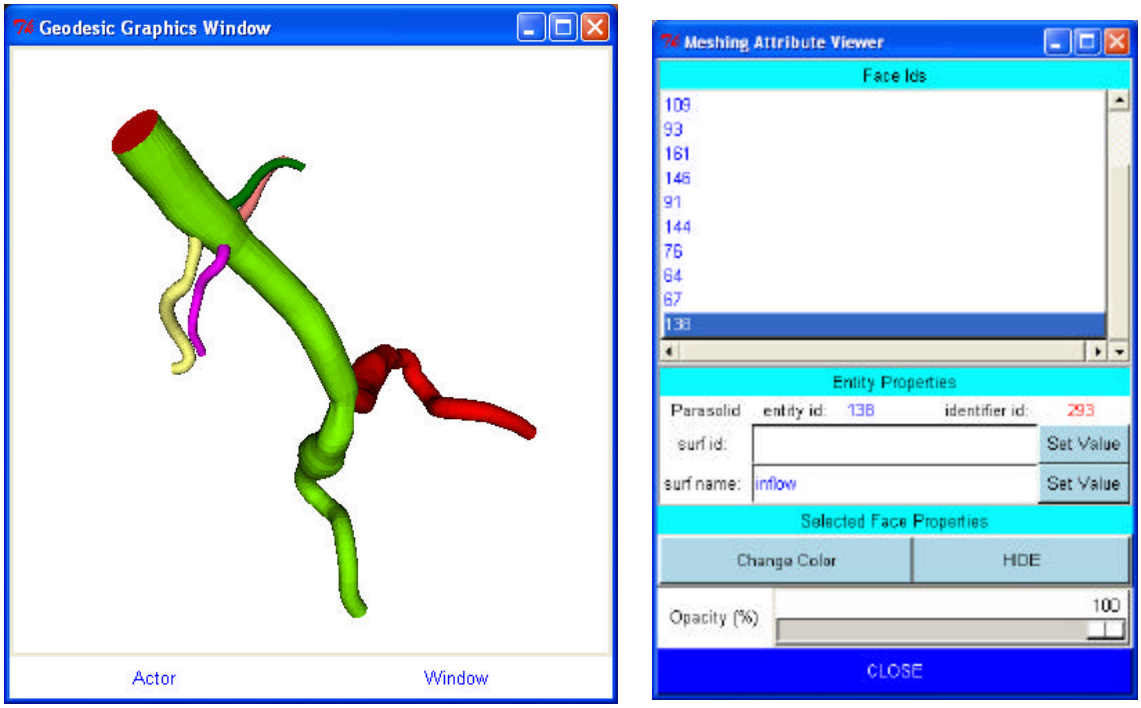


Figure A.8: Solid model viewing menu. The user can interactively select faces on a solid model shown on the left. This is often used to add or remove tags used by the **ASPIRE**[2] system to generate boundary condition files (see section 3.5.4).

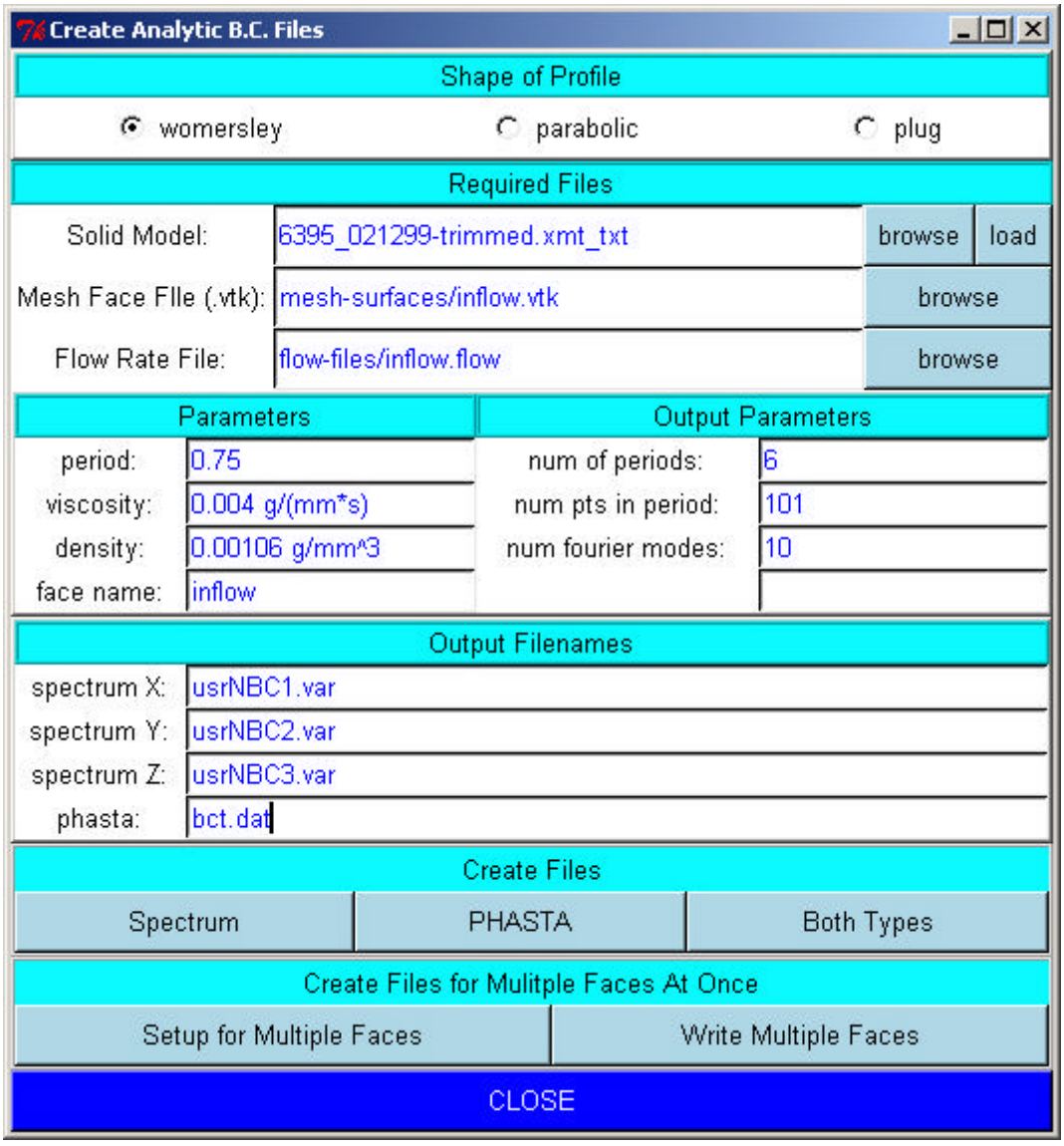


Figure A.9: Analytic boundary condition menu. This menu is used to prescribe an analytic velocity profile as a boundary condition on a mesh face.

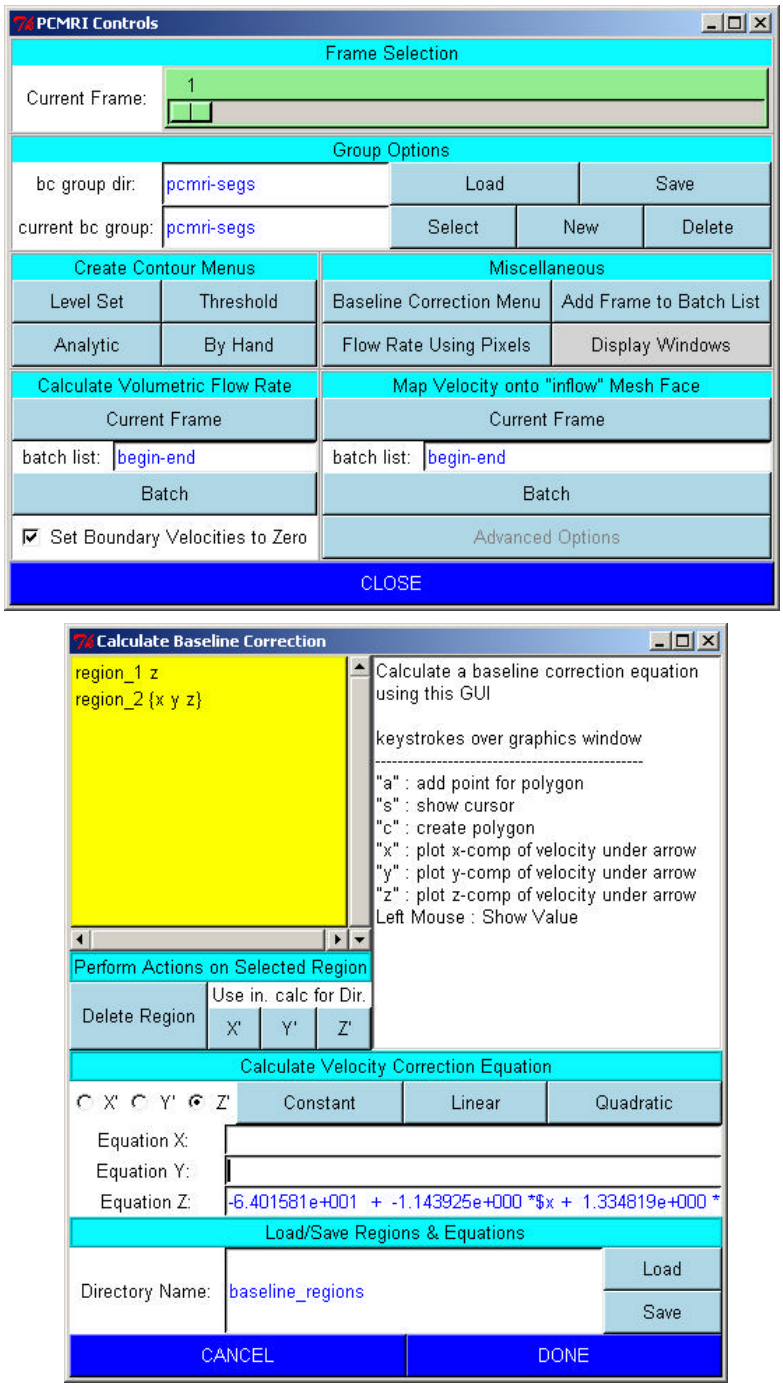


Figure A.10: PCMRI data processing menu. The top figure shows the main menu used to select a frame of PCMRI data. The bottom of the figure shows a menu that can be used to perform baseline correction if desired.

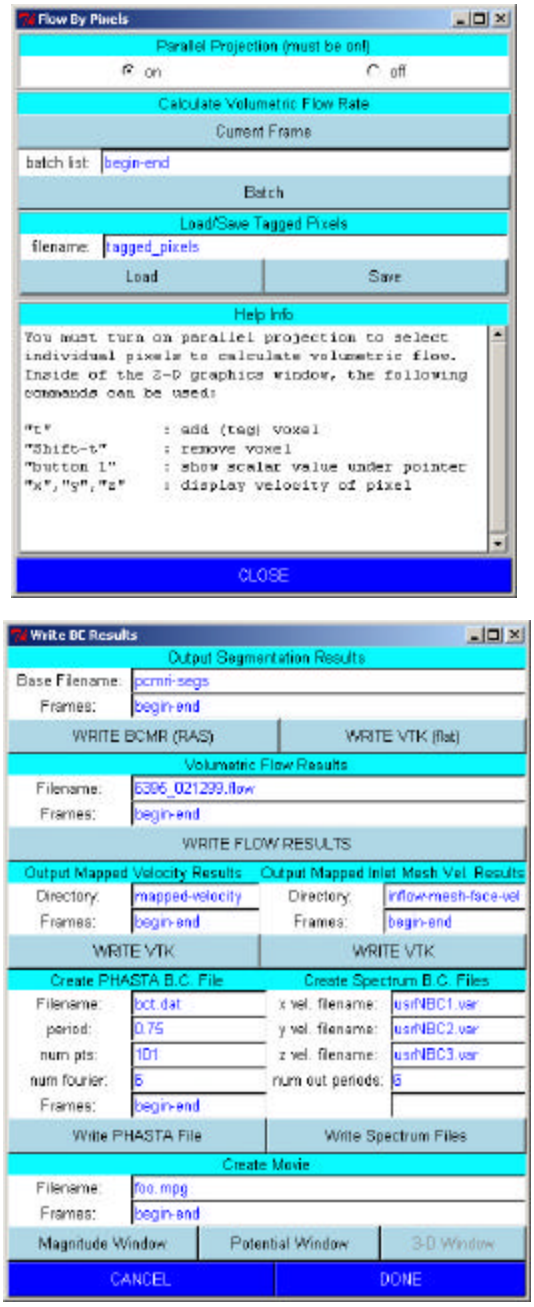


Figure A.11: Additional PCMRI-related menus. The top figure shows a menu used to calculate volumetric flow by selecting individual pixels. The bottom menu is used to create boundary condition files for the flow solver from PCMRI data.

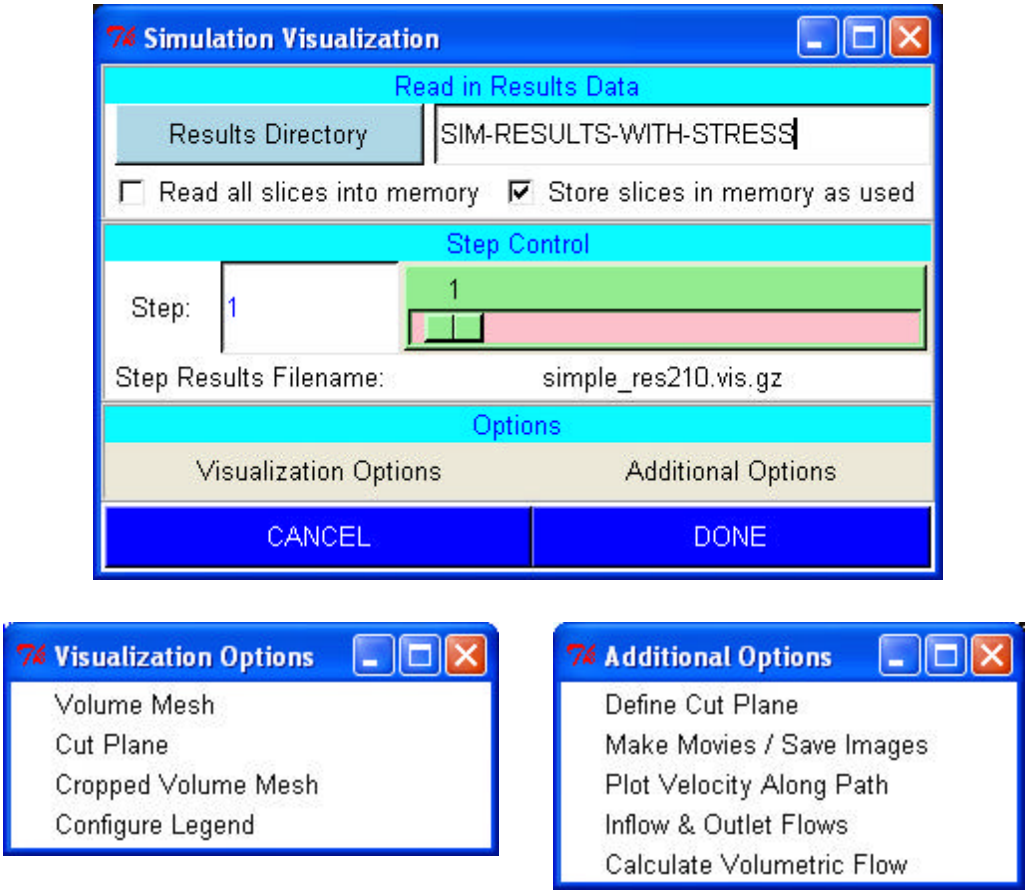


Figure A.12: Main post processing control menu. The location of the desired analysis results is specified and various visualization methods can be utilized to interact with the data as shown in Figure 3.35.

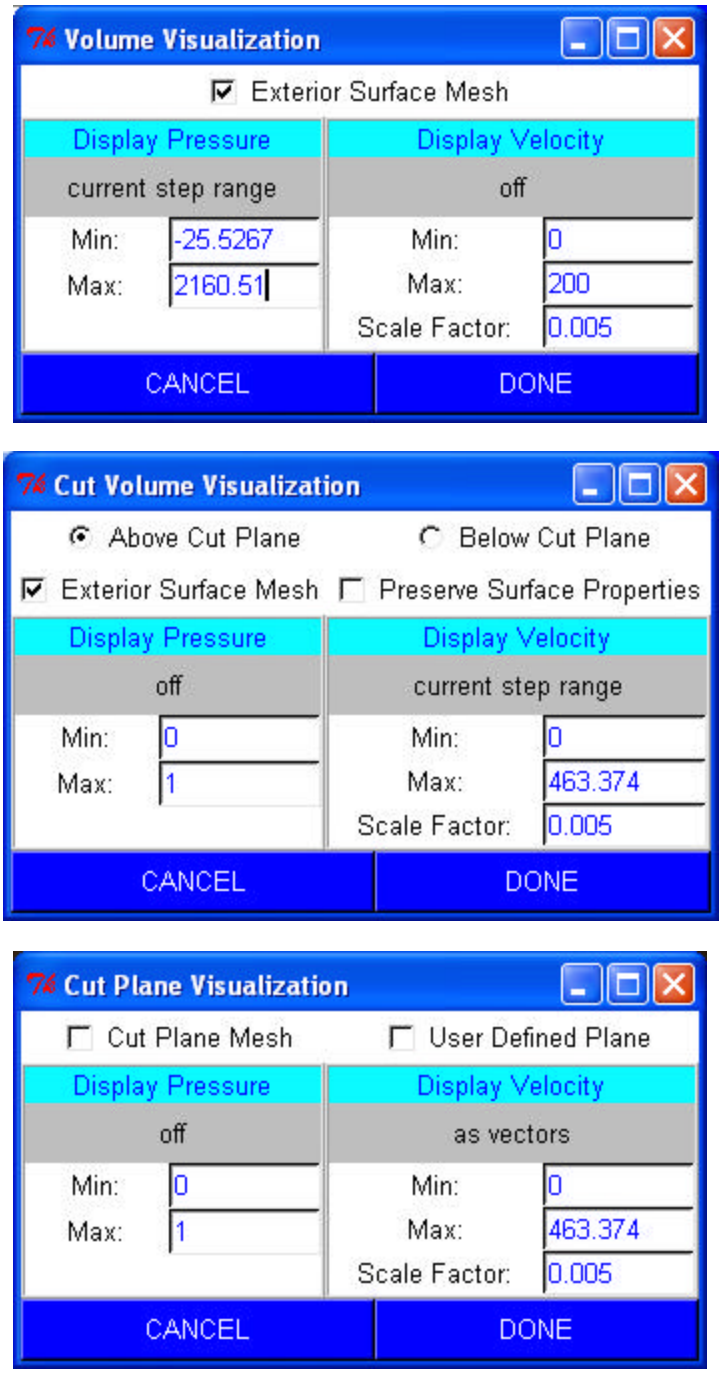


Figure A.13: Additional post processing menus. The user can display the analysis results for the volume, cut volume, and/or the cut plane.

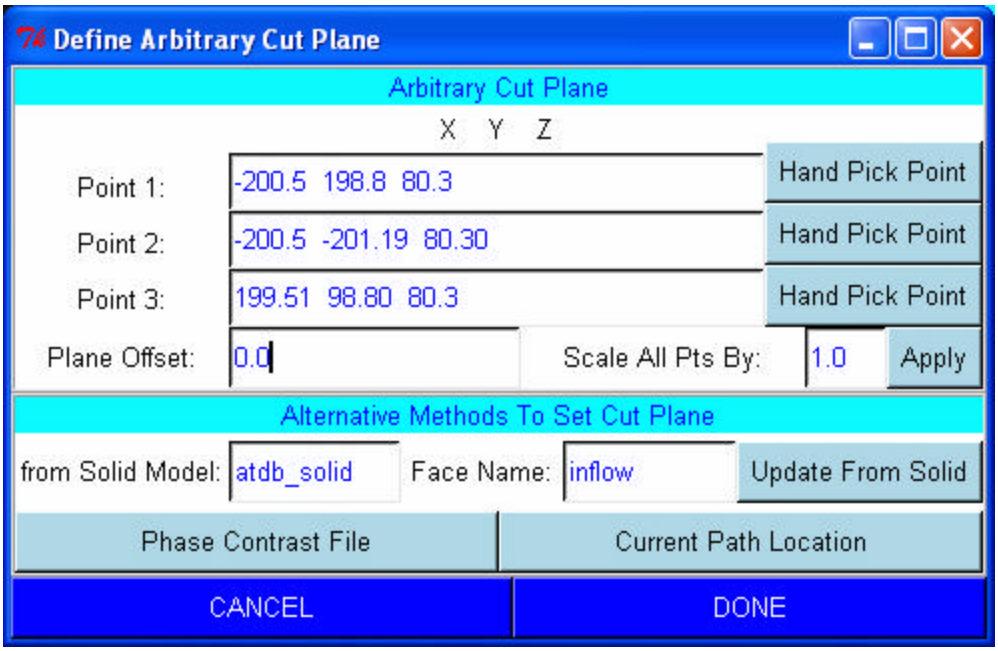


Figure A.14: Menu to define the cut plane for post processing. The user can define the plane used to slice the analysis results in various useful ways.

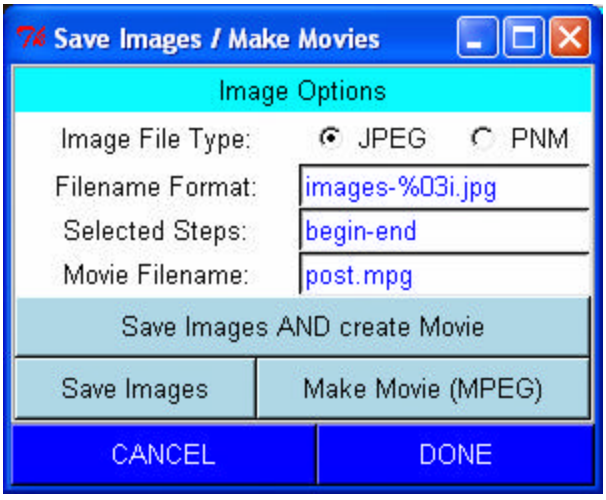
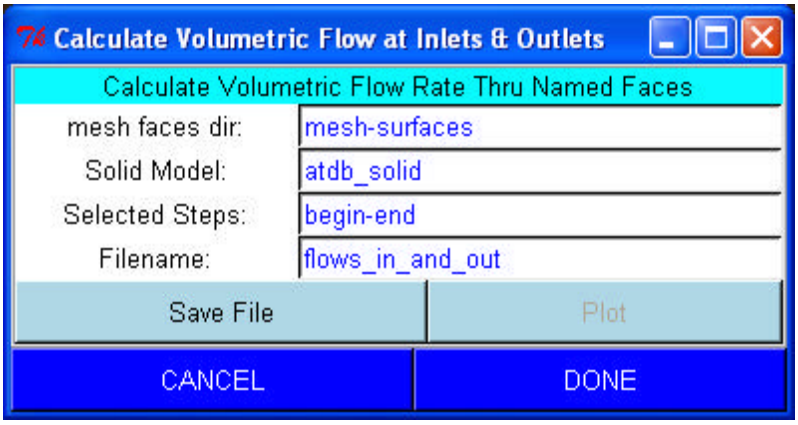


Figure A.15: Create movie menu. It is often useful to create movies of analysis results, particularly for large datasets.



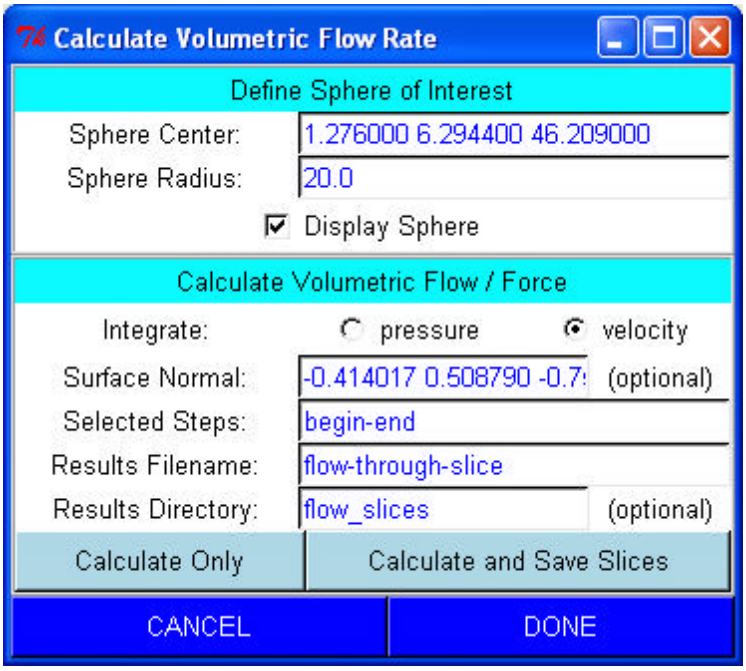


Figure A.16: Calculating volumetric flow rate menus. The menu in the top of the figure is used to calculate volumetric flow in and out of named surfaces on the exterior surface of the mesh from the analysis results. The bottom menu slices the analysis results based on a user-specified cut plane location, and then ignores information from outside a user-specified sphere of interest. This enables the user to calculate volumetric flow of vessels in close proximity to other vessels.

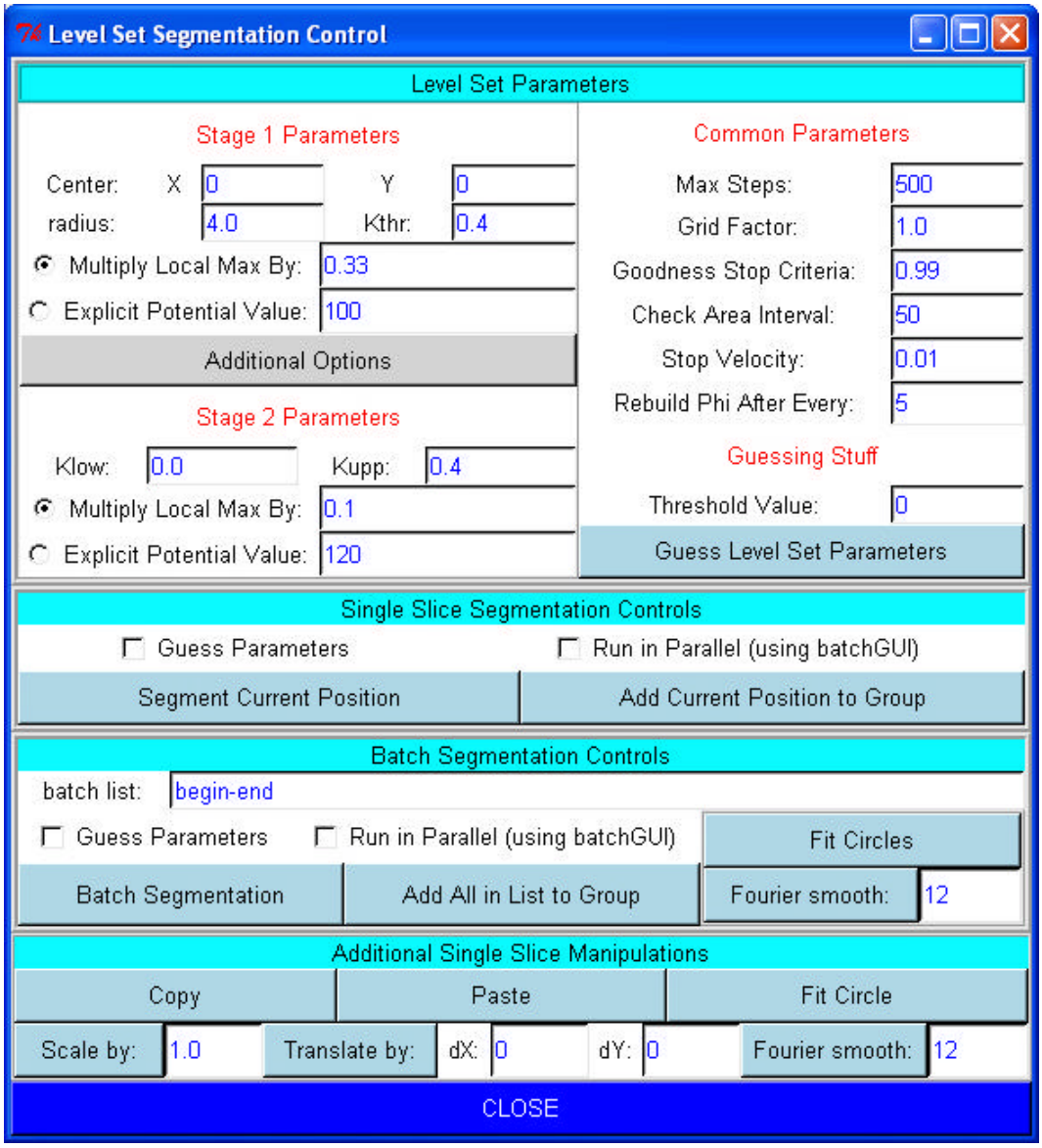


Figure A.17: 2-D level set segmentation control menu. See section 3.5.3.

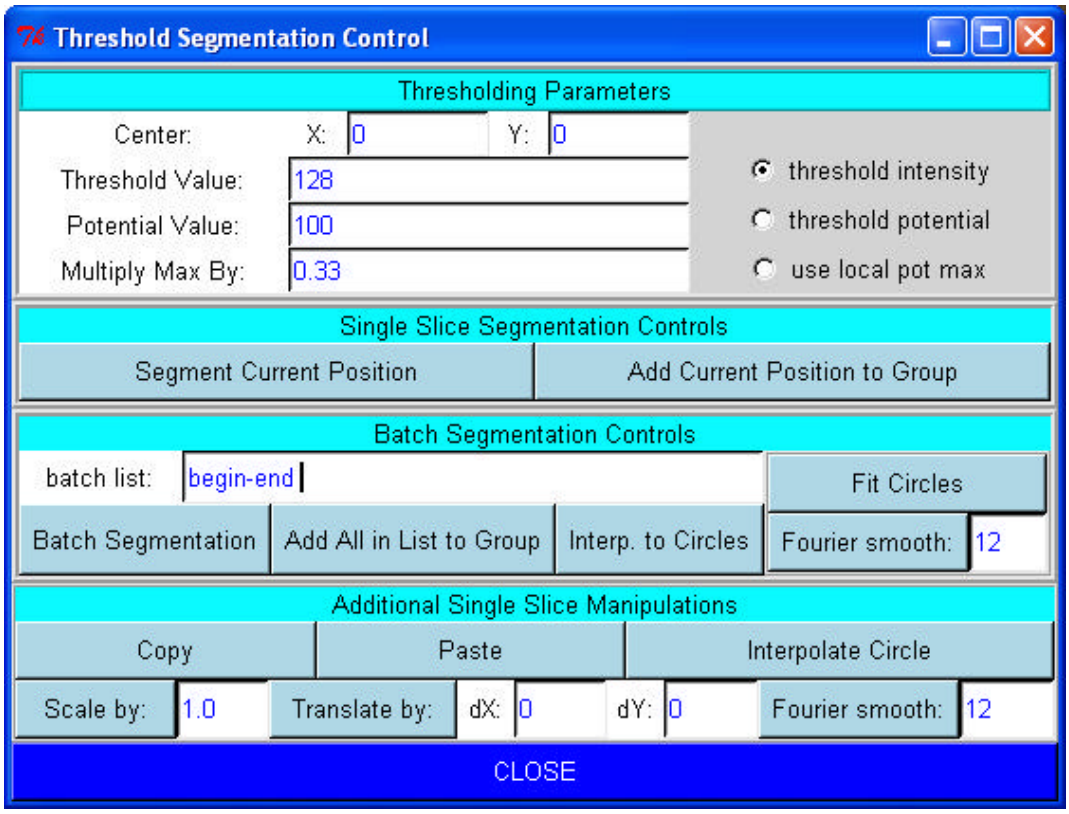


Figure A.18: 2-D threshold segmentation control menu. See section 3.5.3.

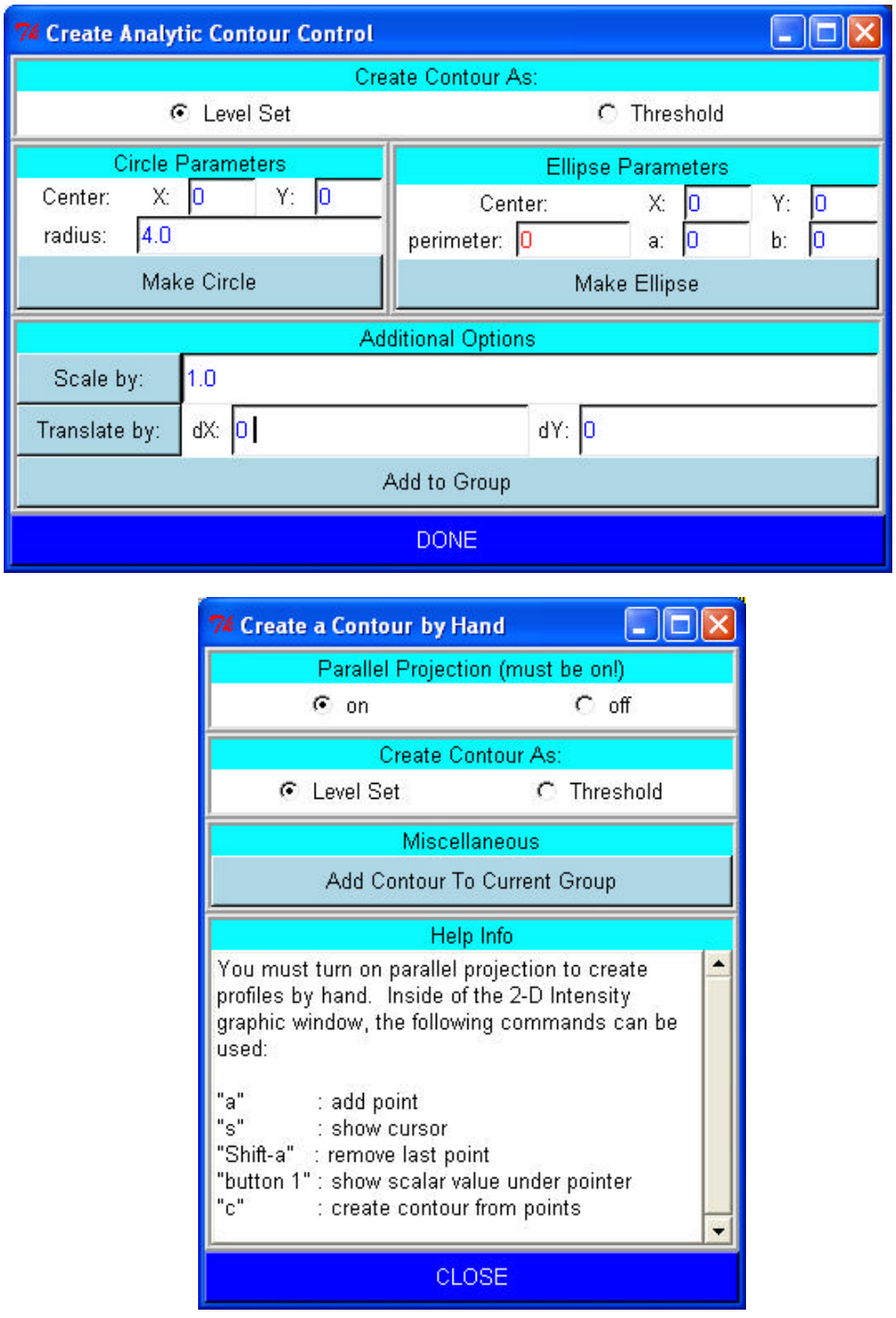


Figure A.19: Menus to create analytic and hand drawn segmentations. The top figure shows the menu used to create analytic (i.e. circles and ellipses) curves while the bottom menu is used when creating contours by hand. See section 3.5.3.

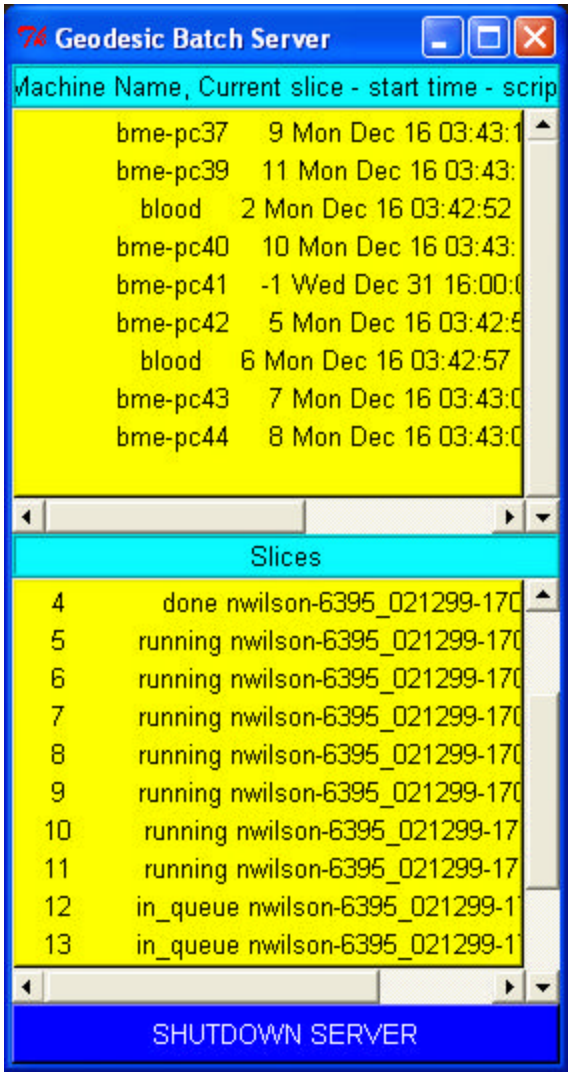


Figure A.20: Distributed segmentation server control window. See Figure 3.24 for additional information.

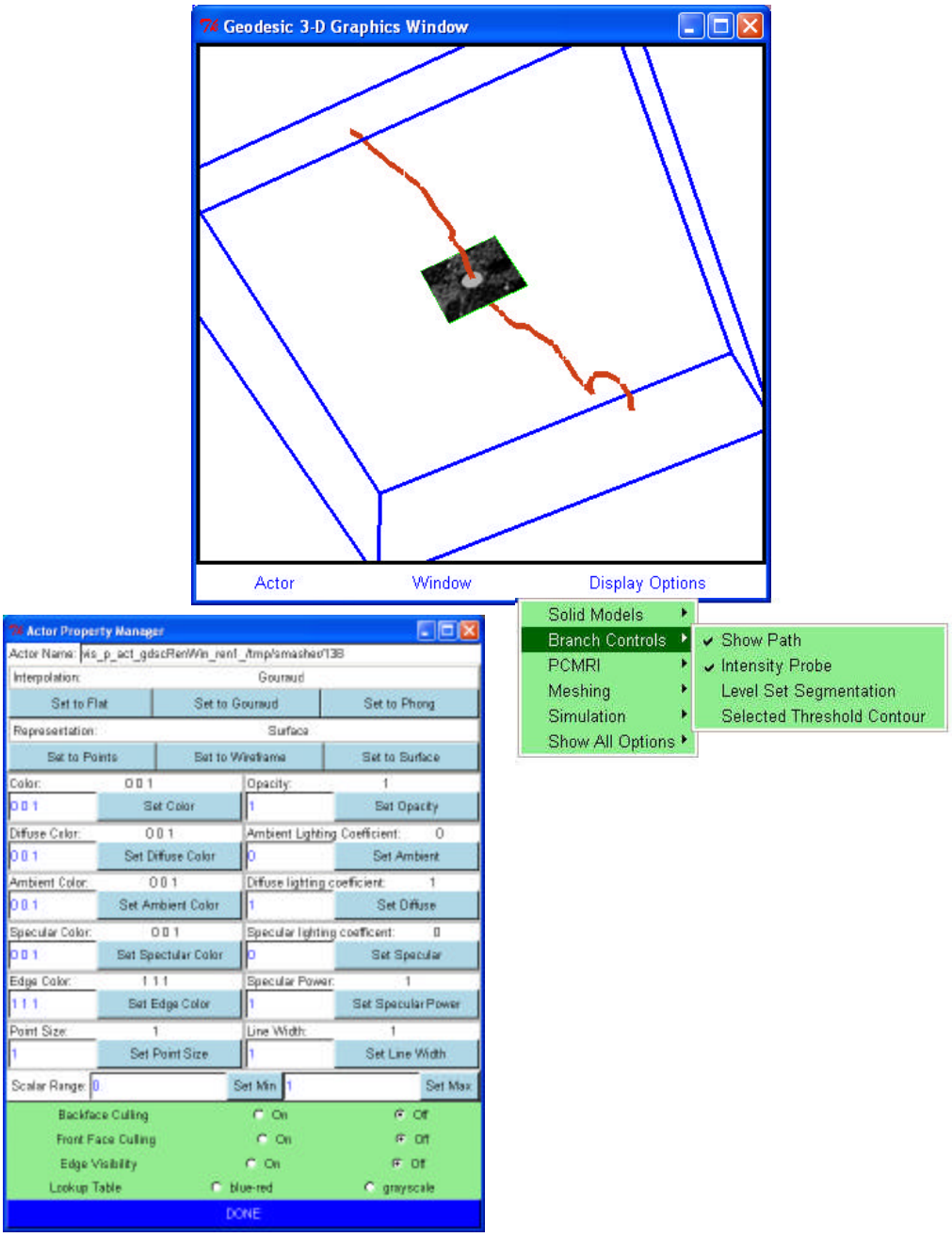


Figure A.21: Three-dimensional graphics window and menus. The user controls display quantities with the pull-down menus along the bottom of the graphics window. In addition, the user can interactively select an actor (displayed object) and explicitly change its associated properties (e.g. color, transparency).

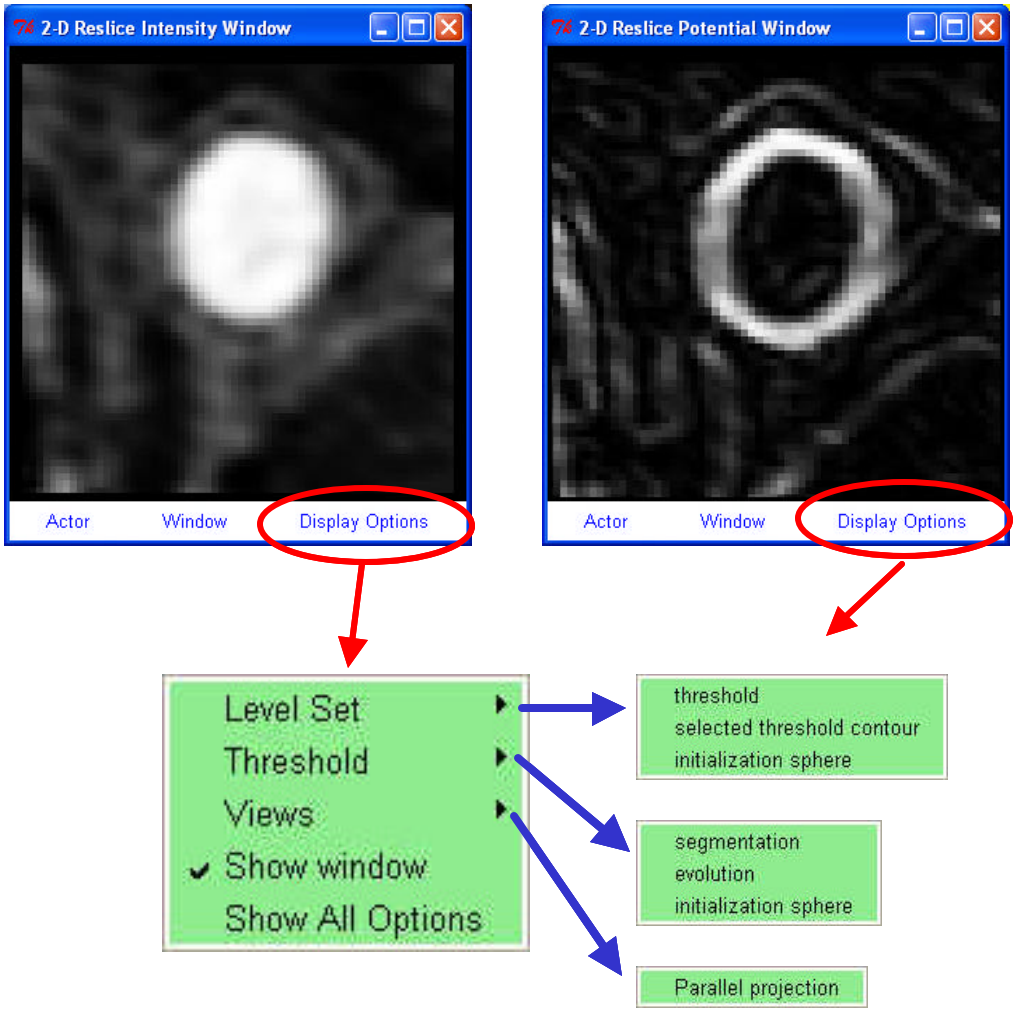


Figure A.22: Two-dimensional reslice graphics windows. The user controls the entities to be displayed.

BIBLIOGRAPHY

[1] J.E. Shigley and L.D. Mitchell, Mechanical Engineering Design, McGraw-Hill, New York, 1993.

[2] AML from TechnoSoft Inc., Cincinnati, OH, USA.

[3] C.A. Taylor, M.T. Draney, J.P. Ku, D. Parker, B.N. Steele, K. Wang, and C.K. Zarins, "Predictive Medicine: Computational Techniques in Therapeutic Decision-Making," Computer Aided Surgery, Volume 4, pages 231-247, 1999.

[4] S.D. Senturia, R.M. Harris, B.P. Johnson, S. Kim, K.N. Nabors, M.A. Shulman, and J.K. White, "A Computer-Aided Design System for Microelectromechanical Systems (MEMCAD)," Journal of Microelectromechanical Systems, Volume 1, Number 1, pages 3-13, 1992.

[5] C.A. Taylor, "A Computational Framework for Investigating Hemodynamic Factors in Vascular Adaptation and Disease," PhD Dissertation, Department of Mechanical Engineering, Stanford University, Stanford, CA, USA, August 1996.

[6] Pro/ENGINEER from PTC, Inc., Boston, MA, USA.

[7] I-deas from EDS, Inc., Plano, TX, USA.

[8] Z. K. Hsiau, "Boundary Movement in Semiconductor Etching and Deposition Simulation," PhD Dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA, USA, June 1997.

[9] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambrige University Press, Cambridge, 1999.

[10] S. Osher and J.A. Sethian, “Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations,” Journal of Computational Physics, Volume 79, pages 12-49, 1988.

[11] K. Wang, “Level Set Methods for Computational Prototyping with Application to Hemodynamic Modeling,” PhD Dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA, USA, August 2001.

[12] M. Mantyla, An Introduction to Solid Modeling, Computer Science Press, Rockville, Maryland, 1988.

[13] Shapes from XOX, Inc., St. Paul, MN, USA.

[14] Parasolid from EDS, Inc., Plano, TX, USA.

[15] ACIS from Spatial Technology, Westminster, CO, USA.

[16] S.J. Owen, "A Survey of Unstructured Mesh Generation Technology," Proceedings 7th International Meshing Roundtable, October, 1998.

[17] C. K. Lee and S.H. Lo, “Automatic Adaptive 3-D Finite Element Refinement Using Different-Order Tetrahedral Elements,” International Journal for Numerical Methods in Engineering, Volume 40, pages 2195-2226, 1997.

[18] M.S. Shephard, “Meshing environment for geometry-based analysis,” International Journal for Numerical Methods in Engineering, Volume 47, pages 169-190, 2000.

[19] MeshSim from Simmetrix, Inc., Clifton Park, NY, USA.

[20] B. Joe and A. Liu, “Relationship Between Tetrahedral Shape Measures,” BIT, Volume 34, pages 268-287, 1994.

[21] M. Berzins, “A Solution-Based Triangular and Tetrahedral Mesh Quality Indicator,” SIAM Journal of Scientific Computing, Volume 19, pages 2051-2060, 1998.

[22] T.J.R. Hughes, The Finite Element Method, Prentice-Hall, New Jersey, 1987.

[23] K.J. Bathe, Finite Element Procedures, Prentice-Hall, New Jersey, 1996.

[24] B.H. McCormick, T.A. DeFanti, and M. D. Brown, “Visualization in Scientific Computing,” Report of the NSF Advisory Panel on Graphics, Image Processing and Workstations, 1987.

[25] W. Schroeder, K. Martin, and W. Lorensen, The Visualization Toolkit, Prentice-Hall, New Jersey, 1998.

[26] VolumePro 1000 from TeraRecon, Inc., San Mateo, CA, USA.

[27] H. Hoppe, "Progressive Meshes," Proceedings of SIGGRAPH, pages 99-108, 1996.

[28] J. Popovic and H. Hoppe, "Progressive Simplicial Complexes," Proceedings of SIGGRAPH, pages 217-224, 1997.

[29] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen, "Decimation of triangle meshes," Proceedings of SIGGRAPH, pages 65-70, 1992.

[30] H. L. de Cougny, "Refinement and Coarsening of Surface Meshes," Engineering with Computers, Volume 14, pages 214-222, 1998.

[31] J.K. Ousterhout, Tcl and the Tk Toolkit, Addison-Wesley Publishing, Reading, MA, 1994.

[32] Standard for the Exchange of Product Model Data (STEP), "Industrial automation systems and integration -- Product data representation and exchange -- Part 1: Overview and fundamental principles," ISO 10303-1:1994.

[33] IRIT from Gershon Elber, Israel Institute of Technology, Israel.

[34] D. Shreiner (editor), OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.2 (3rd Edition), Addison-Wesley, Reading, MA, 2000.

[35] American Heart Association, "2002 Heart and Stroke Statistical Update," Dallas, TX: American Heart Association, 2001.

[36] R.B. Rutherford (editor), Vascular Surgical Procedures, W.B. Saunders Company, Philadelphia, PA, 2000.

[37] C. K. Zarins and B. L. Gewertz, Atlas of Vascular Surgery, Churchill Livingstone, New York, 1989.

[38] E. Silverberg, C.C. Boring, and T.S. Squires, "Cancer statistics, 1990," CA: A Cancer Journal for Clinicians, Volume 40, pages 9-26, 1990.

[39] K.W. Johnston, R.B. Rutherford, M.D. Tilson, D. M. Shah, L. Hollier, and J.C. Stanley, "Suggested standards for reporting on arterial aneurysms," Journal of Vascular Surgery, Volume 12, pages 444-450, 1991.

[40] D.A. Vorp, M.L. Raghavan, and M.W. Webster, "Mechanical wall stress in abdominal aortic aneurysm: Influence of diameter and asymmetry," Journal of Vascular Surgery, Volume 27, Number 4, pages 632-639, 1998.

[41] N.M. Wilson, K. Wang, R.W. Dutton, and C.A. Taylor, “A Software Framework for Creating Patient Specific Geometric Models from Medical Imaging Data for Simulation Based Medical Planning of Vascular Surgery, “ Proceedings of Medical Image Computing and Computer-Assisted Intervention, pages 449-456, 2001.

[42] N.M. Wilson, K. Wang, D. Yergeau, and R.W. Dutton, “GEODESIC: A New and Extensible Geometry Tool and Framework with Application to MEMS,” Proceedings of Modeling and Simulation of Microsystems, pages 716-719, 2000.

[43] D.S. Paik, C.F. Beaulieu, R.B. Jeffrey, G.D. Rubin, and S. Napel, “Automated flight path planning for virtual endoscopy,” Medical Physics, Volume 25, Number 5, pages 629-637, 1998.

[44] R. Malladi, R. Kimmel, D. Adalsteinsson, G. Sapiro, V. Caselles, and J.A. Sethian, “A Geometric Approach to Segmentation and Analysis of 3D Medical Images,” Proceedings of the Workshop on Mathematical Methods in Biomedical Image Analysis, pages 244-252, 1996.

[45] Cygwin by Redhat, Inc., Raleigh, NC, USA.

[46] N.J. Pelc, R.J. Herkens, A. Shimakawa, and D.R. Enzmann, “Phase Contrast Cine Magnetic Resonance Imaging,” Magnetic Resonance Quarterly, Volume 7, Number 4, pages 229-254, 1991.

[47] N.J. Pelc, F.G. Sommer, K.C.P. Li, T.J. Brosnan, R.J. Herfkens, and D.R. Enzmann, “Quantitative Magnetic Resonance Flow Imaging,” Magnetic Resonance Quarterly, Volume 10, Number 3, pages 125-147, 1994.

[48] Centric Engineering Systems, Inc. (now part of Ansys, Inc., Cannonsburg, PA, USA).

[49] A.N. Brooks and T.J.R. Hughes, "Streamline Upwind Petrov-Galerkin Formulation for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier Stokes Equations," Computer Methods in Applied Mechanics and Engineering, Volume 32, pages 199-259, 1981.

[50] Z. Johan, T.J.R. Hughes, and F. Shakib, "A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids," Computer Methods in Applied Mechanics and Engineering, Volume 87, pages 281-304, 1991.

[51] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," SIAM Journal on Scientific Computing, Volume 20, Number 1, 1999.

[52] Signa, General Electric Medical Systems, Milwaukee, WI, USA.

[53] G.H. Glover, N.J. Pelc, U.S. Patent #4,591,789, granted May 27, 1986.

[54] M.T. Draney, M.T. Alley, B.T. Tang, N.M. Wilson, R.J. Herfkens, and C.A. Taylor, "Importance of 3D Nonlinear Gradient Corrections for Quantitative Analysis of 3D MR Angiographic Data," Proceedings of the 10th Annual International Society for Magnetic Resonance in Medicine Conference, 2002.

[55] Personal communication with General Electric Medical Systems, Milwaukee, WI, USA.

[56] J.P. Ku, M.T. Draney, F.R. Arko, W.A. Lee, F.P. Chan, N.J. Pelc, C.K. Zarins, and C.A. Taylor, "*In Vivo* Validation of Numerical Predication of Blood Flow in Arterial Bypass Grafts," Annals of Biomedical Engineering, Volume 30, pages 743-752, 2002.

[57] V. Favier and C.A. Taylor, “Modeling blood flow in a porcine aorta bypass graft: realization of physiological conditions,” Center for Turbulence Research, Stanford University, Annual Research Briefs, 2001.

[58] C.P. Cheng, D. Parker, and C.A. Taylor, “Quantification of Wall Shear Stress in Large Blood Vessels Using Lagrangian interpolation functions with cine PC-MRI,” Annals of Biomedical Engineering, Volume 30, Number 8, pages 1020-1032, 2002.

[59] C.A. Taylor, C.P. Cheng, L.A. Espinosa, B.T. Tang, D. Parker, and R.J. Herfkens, “In Vivo Quantification of Blood Flow and Wall Shear Stress in the Human Abdominal Aorta during Lower Limb Exercise,” Annals of Biomedical Engineering, Volume 30, Number 3, pages 402-408, 2002.

[60] C.A. Taylor and T.J.R. Hughes, “A Multiscale Finite Element Method for Blood Flow in Deformable Vessels,” Proceedings of the World Congress of Biomechanics, 1998.

[61] EE Times, April 17, 1998.

[62] E. Peeters, “Challenges in Commercializing MEMS,” IEEE Computational Science & Engineering, pages 44-47, Jan.-Mar., 1997.

[63] S. D. Senturia, N. Aluru, and J. White, “Simulating the Behavior of MEMS Devices: Computational Methods and Needs,” IEEE Computational Science & Engineering, pages 30-43, Jan-Mar 1997.

[64] IntelliCAD, IntelliSense Corporation, Wilmington, MA, USA.

[65] J. Funk, “Modeling and Simulation of IMEMS,” D. Sc. Technical Thesis, Swiss Federal Institute of Technology, Zurich, 1995.

[66] P. Ljung, "Efficient MEMS CAD Tools,” Journal of Micromachining, June 1998.

[67] CFD-ACE+ from CFD Research Corporation, Huntsville, AL, USA.

[68] MEMCAD by Conventor, Inc., Cary, NC, USA.

[69] J.P. McVittie, J. C. Rey, L.-Y. Cheng, A. Bariya, S. Ravi and K. C. Saraswat, ``SPEEDIE: A Profile Simulator for Etching and Deposition," Extended Abstracts, SRC Techcon '90, pages 16-19, 1990.

[70] Personal communication with designer at Maxim Integrated Products, Inc., Sunnyvale, CA, USA.

[71] N.M. Wilson, D. Yergeau, and R.W. Dutton, "Internet Based Modeling of Micro-Electro-Mechanical Systems," International Conference of Computational Engineering & Sciences, pages 1330-1334, 2000.

[72] “Composite CAD Process Definition Specification version 1.0,” Defense Advanced Research Projects Agency internal document.

[73] VRML, “Information technology -- Computer graphics and image processing -- The Virtual Reality Modeling Language -- Part 1: Functional specification and UTF-8 encoding,” ISO/IEC 14772-1:1997.

[74] X3D, “Information technology — Computer graphics and image processing — Extensible 3D (X3D),” draft standard ISO/IEC 19775:200x.

[75] “JGV: 3D Viewing in Java,” http://www.geom.umn.edu/java/JGV/.

[76] N.M. Wilson, P.M. Pinsky, and R.W. Dutton, "Investigation of Tetrahedral Automatic Mesh Generation for Finite-Element Simulation of Micro-Electro-Mechanical Switches", Proceedings of Modeling and Simulation of Microsystems, pages 305-308, 1999.

[77] ProPHLEX from Altair Engineering, Inc., Austin, TX, USA.

[78] R.W. Dutton and Z. Yu, Technology CAD - Computer Simulation of IC Processes and Devices, Kluwer Academic Publishers, 1993.

[79] G. M. Koppelman, “OYSTER, a three-dimensional structural simulator for microelectromechanical design,” Sensors and Actuators, Volume 20, pages 179-185, 1989.

[80] J.P. Elliott, G.A. Allan, and A.J. Walton, “The Automatic Generation of Conformal 3D Data for Interconnect Capacitance Simulation,” Proc. ESSDERC, pages 405-408, 1995.

[81] M. Emmenegger, J. G. Korvink, and H. Baltes, “MemCel – An Inexpensive and Efficient Tool for 3D MEMS Prototyping,” International Mechanical Engineering Congress and Exhibition, pages 559-563, 1998.

[82] K. Wang, H.S. Park, Z. Yu., and R.W. Dutton, “3D Solid Modeling of IC Structures Using Simulated Surface Topography,” Proceedings of SISPAD, pages 131-132, 1996.

[83] P.M. Osterberg and S.D. Senturia, “MemBuilder: An Automated 3D Solid Model Construction Porgram for Microelectromechanical Structures,” Proceedings of TRANSDUCERS-95, Volume 2, pages 21-24, 1995.

[84] N.M. Wilson, S. Liang, P.M. Pinsky, and R.W. Dutton, "A Novel Method to Utilize Existing TCAD Tools to Build Accurate Geometry Required for MEMS Simulation," Proceedings of Modeling and Simulation of Microsystems, pages 120-123, 1999.

[85] J.A. Sethian and D. Adalsteinsson, "An overview of level set methods for etching, deposition, and lithography development," IEEE Trans Semiconductor Manufacturing, Volume 10, Number 1, pages 167-184, 1997.

[86] N.M. Wilson, R.W. Dutton, and P.M. Pinsky, "Utilizing Existing TCAD Simulation Tools to Create Solid Models for the Simulation Based Design of MEMS Devices", Proceedings of International Mechanical Engineering Conference and Exposition, pages 565-570, 1998.

[87] N.M. Wilson, Z.K. Hsiau, R.W. Dutton, and P.M. Pinsky, "A Heterogeneous Environment for Computational Prototyping and Simulation Based Design of MEMS Devices", Proceedings of SISPAD, pages 153-156, 1998.

[88] R.W. Dutton, E.K. Chan, N.M. Wilson, Z.K. Hsiau, and S. Shen, "Challenges in Process Modeling for MEMS", Proceedings of Modeling and Simulation of Microsystems, pages 1-4, 1998.

[89] E.K. Chan, “Modeling and Simulation of Electrostatically Actuated Micromechanical Devices,” PhD Dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA, USA, November 1999.

[90] P.B. Griffin, J.D. Plummer, and M.D. Deal, Silicon VLSI Technology: Fundamentals, Practice, and Modeling, Prentice-Hall, New Jersey, 1999.