VENTRICULAR BLOOD FLOW SIMULATION AND ANALYSIS FOR CARDIOVASCULAR DIAGNOSTICS

BY SCOTT ANDREW KULP

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Dimitris Metaxas

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2015

ABSTRACT OF THE DISSERTATION

Ventricular Blood Flow Simulation and Analysis for Cardiovascular Diagnostics

by Scott Andrew Kulp

Dissertation Director: Dimitris Metaxas

The heart has long been seen as a symbol of life, due to its critical function of pumping blood throughout the body. However, despite its importance, we still do not fully understand how the heart works, due to its complex motion and structure. In particular, doctors today are very interested in learning how the heart geometry may affect cardiac blood flow. However, current imaging techniques, such as MRI or Ultrasound, provide only low-resolution views of blood flow, which do not provide the desired level of detail.

In this dissertation, We will be presenting how we are using images from high-resolution CT scans to build accurate, animated 3D models of a patient's heart, which are then used as boundary conditions in solving the Navier-Stokes equations to simulate ventricular blood flow. This way, we can visualize how the complex structures within the heart interact with the flow in both healthy and diseased hearts, which has never been seen before. We can also use similar simulation techniques with high-detail aortic valve reconstructions, to better understand how diseased-induced alterations in the blood flow pattern may promote chronic remodeling of the aortic root. Finally, we have modified the Smoothed Particle Hydrodynamics algorithm to allow for fast and effective boundary collision management to greatly speed up our simulations.

Acknowledgements

I would like to sincerely thank all those people who have helped me get to this point in my academic career. First and foremost, I would like to thank my Ph.D. adviser, Dimitris Metaxas. Dimitris has provided me with a tremendous amount of resources over the last six years, including funding, powerful computing resources, and much of his time. He has taught me how to do excellent computational science, and how to be a productive scholar. I greatly appreciate all of his ideas and thoughtful discussion that made this Ph.D. experience successful. I look forward to our continued collaboration in research and publishing.

I would also like to thank Matthew Stone and Vladimir Pavlovic for being part of my Qualifying Exam Committee and Ph.D. Committee. They both spent a great deal of time reviewing and supporting my work, as well as providing advice and feedback over the last several years. I also thank my external committee member, Xiaolei Huang, from Lehigh University, for the time she spent reviewing this disseration, providing valuable feedback, and traveling to Rutgers for my final defense.

I give great thanks to Tami Carpenter and those at the Center for Dynamic Data Analytics at Rutgers. For my first three years at Rutgers, DyDAn provided me with a very generous fellowship and funding. This allowed me to focus on my academics in those early years, so that I could finish my course requirements quickly, and be able to start my research work sooner.

I would like to acknowledge my labmates at CBIM, without whom, much of the work in this thesis would have been impossible. In particular, Mingchen Gao, Chao Chen, and Shaoting Zhang spent many hours to generate beautiful animated models of several patients' hearts. Chao Chen also worked with me a great deal in another flow analysis project. I thank Viorel Mihalef, who trained me for the first two years at

Rutgers in fluid simulation and visualization.

Much of my work and success has also been made possible by Leon Axel at New York University. Since I started on the cardiac blood flow projects, he has spent many hours discussing with me ideas on how to tackle flow analysis. He has provided very useful insight and advice in several of my papers, and I really appreciate his thoughtful input.

I would like to thank Zhen Qian and the other collaborators at Piedmont Heart Institute, who produced and shared large volumes of incredibly high-quality CT imagery, used throughout my work. Additionally, for the work in aortic valve blood flow simulations, Zhen and Piedmont also provided fantastic 3D models of the valve geometry.

I would like to give my deep thanks to April Kontostathis, my undergraduate adviser at Ursinus College. April inspired me to get into scientific research, and helped prepare me for graduate school. Even after graduating from Ursinus, she has been a great support in my PhD studies. There is no way I would have gotten to this point without her.

I would also like to acknowledge Ben Strauss, my supervisor at Climate Central, who has been extraordinarily supportive and flexible in the last couple years of working on my Ph.D. Working at Climate Central has been a fantastic experience, and I look forward to working with Ben and everyone else there over the coming years.

I also acknowledge the Department of Defense for bringing me in for four summers to work on several very interesting research projects and gaining valuable experience. I would also like to thank my mentors at the DoD, in particular, Antwan Clark, whom I worked with on an iris recognition project. We collaborated for several years and Antwan has provided great support and advice during that time.

Special thanks to my other New Jersey friends: Andrew Miller, Aryeh Lansley, Bill Tamashunas, Janet Trachy, Curtis Elliott, Josie Walentowicz, and all the rest. Going out to eat has given me something to look forward to every week, as well as a place to relax and unwind. I truly appreciate all of their support over the last six years.

Finally, I am deeply thankful to my parents, Jeff and Kim Kulp, and my siblings, Holly, Adam, Heidi, Robin, and Amber, for their love and support. Most of all, I would

like to thank my wife, Yazhi Zheng, to whom I am truly and forever grateful for her patience, sacrifice, and love. Thank you so much for joining me in this adventure!

Table of Contents

Chapter 1

Introduction

In patients who experience a heart attack, or in those who suffer from other various cardiovascular diseases, the motion of the heart walls and valves can become disturbed, leading to an abnormal blood flow pattern. If the blood is not being fully circulated within the heart and becomes stagnant, these patients are at high risk of thrombus, leading to stroke. Thus, it is very important for doctors to be able to visualize and understand a patient's cardiac blood flow. While it is possible to acquire flow data from MRI or Doppler ultrasound imaging, the relatively low quality and resolution of this data severely limits its usefulness to doctors. In particular, these imaging techniques lose nearly all detail in the apex regions of the left ventricle, where there is heavy trabeculation. Therefore, along with the rapid development of high-resolution cardiac CT, patient-specific blood flow simulation is quickly becoming one of the central goals in the study of cardiac blood flow.

However, there are very difficult problems faced when simulating blood flow through the left ventricle. Most notably, the endocardium of the LV are of extraordinary geometric complexity, due to the papillary muscles and the aforementioned heavy trabeculation. Since this complicated geometry makes simulation much more difficult, LV models are typically simplified and smoothed, such as in [1] and [2].

While simplifying these models greatly reduces computation cost, we believe that the geometry and motion of these complex structures may be critical to function and efficiency of the left ventricle. The function of the trabeculae is currently not well-understood, but we hypothesize that healthy trabeculae helps move blood within the trabeculae, preventing stagnant blood and potential thrombus. Additionally, diseased

trabeculae likely might prevent blood from fully circulating, increasing the risk of clotting. Therefore, in this thesis, we seek answers to the following questions:

- Can we simulate, visualize, and analyze full Navier-Stokes fluid simulations of the left ventricle with highly detailed models, including papillary muscles and trabeculae, and can our visualizations and analysis of these simulations be used in clinically useful discrimination between healthy and diseased hearts?
- Does there exist interaction between trabeculae motion and fluid flow?
- Are there clinically useful reasons to use and prefer the more complex models and simulations of the left ventricle over the more traditional smoothed-wall simulations?

Another, related problem is in understanding the flow through the aortic valve and aortic root. We believe that aortic valve disease-induced alterations in the blood flow pattern may promote chronic remodeling of the aortic root and the left ventricle. Although the mechanism of the aortic root remodeling is not fully understood, the hemodynamic pattern in the aortic valve vicinity is believed to play an important role. As mentioned, however, MRI and ultrasound images of the blood flow produce very low resolution image, and to our knowledge, no one has performed high-resolution flow simulations across diseased aortic root models to understand this phenomenon. So, we also propose the following question:

- Can we use blood flow simulations to understand how the blood flow pattern in the aortic valve may play an important role in the remodeling of the aortic root?

Furthermore, another problem that currently limits the practicality of patient-specific blood flow simulations is computation time. Current state-of-the-art simulators take days to run, which is impractical for use in a clinical setting. However, in recent years, there has been increasing interest in meshless methods, such as Smoothed Particle Hydrodynamics (SPH), due to their improved running times. However, we know that while SPH is much faster than more traditional fluid solvers, they also lose accuracy.

Additionally, working with complex boundary conditions is a notoriously difficult unsolved problem, and even the most recent methods of SPH boundary management are certainly incapable of handling the geometry and deforming motion of the left ventricle. So, we present two more major questions to be answered in this manuscript:

- Can we find a new method of SPH boundary management that can handle the complex geometry and motion of the left ventricular walls?
- Does SPH perform at a high enough accuracy that could still remain useful for clinical applications?

Finally, another problem remains on the topic of SPH boundary management. All boundary handling algorithms, to our knowledge, requires exceptionally thick walls. Without thick walls, nodes on one side of a wall may interact with nodes on the other side of the wall, leading to inevitable instability. If we find that SPH can perform at clinically useful levels of accuracy, we still would lose a large amount of detail. Therefore, we propose a final major question, which will be discussed later in this thesis:

- Can we derive a new algorithm to allow for Thin-Wall Smoothed Particle Hydrodynamics?

This thesis is structured as follows: Chapter 1 provides an introduction to the fields of Computation Fluid Dynamics, how fluid fields can be solved through the Navier-Stokes Equations and Smoothed Particle Hydrodynamics, and a literature review on previous work in cardiac blood flow simulations. Chapter 2 (published [3]) describes how high-resolution 4D CT imagery of the left ventricle can be used to accurately simulate and visualize blood flow through the heart, including features such as the papillary muscles. Chapter 3 (published [4]) builds upon this work by incorporating heart models built from much more sophisticated reconstruction techniques, so that we can actually see the complex interactions between ventricular trabeculae and blood flow. In Chapter 4, we seek to prove the clinical importance of using these high-detail meshes over the more standard smoothed left ventricle models. In Chapter 5 (published [5]), we use these simulation and visualization techniques on CT-reconstructed models of the aortic

valve and root. In Chapter 6 (published [6]), we seek a much faster way to perform cardiac blood flow simulations by modifying the Smoothed Particle Hydrodynamics to allow for effective and accurate complex-boundary handling. In Chapter 6, we introduce a new algorithm for Thin-Wall Smoothed Particle Hydrodynamics, which allows SPH to handle arbitrarily thin, curved boundaries, which has never been possible before.

1.1 Navier-Stokes

For the purposes of most problems in Computational Fluid Dynamics, we disregard the molecular composition of fluids and assume that all fluid volumes are a continuum. In the 1840's, Claude-Louis Navier and George Gabriel Stokes applied Newton's Second Law to the fluid continuum assumption to derive the classical Navier-Stokes (NS) equations, which describe the motion of fluid flow [7]. The first equation, which enforces conservation of momentum, was found to be as follows:

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla P + \nabla \cdot \mathbf{T} + \mathbf{f} \tag{1.1}$$

Here, ρ is the fluid density, u is the 3D velocity vector field, P is the pressure field, $\mathbf{T}$ is the shear stress tensor, and $\mathbf{f}$ is the body force, such as gravity. The $\frac{\partial \mathbf{u}}{\partial t}$ term corresponds to the temporal acceleration of the fluid at a given moment, while the $\mathbf{u} \cdot \nabla \mathbf{u}$ corresponds to the spatial (convective) acceleration at a given point in space. The convective acceleration component is an important, but somewhat unique, term in fluid flow, not normally seen in classical rigid-body dynamics. As an example, this term accounts for the pressure drop seen in steady flow through a narrowing pipe, causing the fluid to accelerate in space, though remaining unchanging in time.

The Navier-Stokes equations is often written in terms of the material derivative, $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \nabla \cdot \mathbf{T} + \mathbf{f}, \tag{1.2}$$

more clearly showing its relation to Newton's Second Law, $F = ma$.

In the case of modeling liquids, such as water or blood, we assume incompressibility, meaning that the density throughout the flow field is constant. With this assumption, the shear stress term becomes $\nabla \cdot \mathbf{T} = \mu \nabla^2 \mathbf{u}$, where μ is the fluid's coefficient of viscosity. Additionally, in this work, we note that gravity is the only body force, so $\mathbf{f} = \rho \mathbf{g}$, where g is the acceleration due to gravity. Since there will be no open surfaces the hydrostatic pressure gradient at any point would be $\nabla P_{hydrostatic} = \rho \mathbf{g}$, and so $-\nabla P_{hydrostatic} + \mathbf{f} = 0$, so we can disregard both terms. Therefore, the final conservation of momentum equation of fluids reduces to

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \mu \nabla^2 \mathbf{u}. \tag{1.3}$$

Since, in 3D Cartesian coordinates, there are 3 components of velocity, this produces three equations (one for each $\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z$). This is an underdetermined system, as we seek to solve for the three velocity components, as well as pressure. Therefore, the final equation of fluid motion enforces conservation of mass:

$$\frac{D\rho}{Dt} + \rho(\nabla \cdot \mathbf{u}) = 0, \tag{1.4}$$

or, assuming the fluid is incompressible, and ρ is constant:

$$\nabla \cdot \mathbf{u} = 0. \tag{1.5}$$

Expanded, this equation reads:

$$\frac{\partial \mathbf{u}_x}{\partial x} + \frac{\partial \mathbf{u}_y}{\partial y} + \frac{\partial \mathbf{u}_z}{\partial z} = 0. \tag{1.6}$$

This equation can be interpreted as saying, that given any volume within a fluid, the amount of fluid entering the volume must be exactly equal to the amount of fluid exiting the volume. That is, there exist no sources or sinks within any volume of the fluid, and thus the mass of fluid remains constant.

A variety of boundary conditions exist for the NS equations. For solid boundaries, the most common approach is the no-slip condition, which states that at the interface between the fluid and a solid, the velocity of the fluid is exactly equal to the velocity of

the wall (that is, both the normal and tangential components of fluid velocity are equal to the wall velocity). Less commonly used is the free-slip conditions, which allows fluid to freely flow tangential to the wall, but the normal component of the fluid velocity is still exactly equal to the normal component of the solid velocity. For viscous fluids, the no-slip condition models the true behavior of fluid at a boundary.

At the inlet and outlets, more options exist. At the most basic, inlet/outlet velocities can be set to known values. At a free surface, a pressure boundary condition is used, such that the fluid pressure at the air/liquid interface is equal the the air (atmospheric) pressure. Finally, we may use periodic boundary conditions, enforcing that fluid velocity and pressure at the outlet is exactly equal to the velocity/pressure at the inlet. For the applications described in this manuscript, periodic boundary conditions are most-often used, unless described otherwise.

1.1.1 Numerical Methods for Solving Fluid Flow

Even though the Navier-Stokes equations are among the most important and widely-used equations in existence, there currently exists no general way to solve these equations analytically, or even any proof that it is possible to find a solution to a general NS problem. While some closed-form solutions exist for very basic problems, such as flow between parallel plates, or across a cylinder [8]. For the much more complex problems described in this manuscript, there certainly exists no known exact solutions, and so we must use numerical methods for solving the Navier-Stokes equations.

Time Discretization

In most Navier-Stokes solvers, there are two major steps to time discretization. The first is the velocity prediction step, where we use the momentum equation to solve for an intermediate solution to velocity, temporarily ignoring the pressure term. Rearranging the terms in the momentum equation and removing the pressure term, we can get

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} + \frac{\mu}{\rho} \nabla^2 \mathbf{u}. \tag{1.7}$$

There are many options for solving this equation. For example, using the Euler method:

$$\mathbf{v} = \mathbf{u}_n - \mathbf{u}_n \cdot \nabla \mathbf{u}_n + \frac{\mu}{\rho}\nabla^2 \mathbf{u}_n, \tag{1.8}$$

where $\mathbf{v}$ is the intermediate velocity solution. The Euler method is very straightforward, but is only over order 1, leading to lower accuracy and greater chance of instability. An alternative is the explicit Runge-Kutta method, which is of order 2, but slightly more difficult ot implement. For the momentum equation, the Runge Kutta solution is as follows [9]:

$$\mathbf{k1} = -\mathbf{u}_n \cdot \nabla \mathbf{u}_n + \frac{\mu}{\rho}\nabla^2 \mathbf{u}_n \tag{1.9}$$

$$\mathbf{u}^* = \mathbf{u}_n + \Delta t \cdot \mathbf{k1} \tag{1.10}$$

$$\mathbf{k2} = -\mathbf{u}^* \cdot \nabla \mathbf{u}^* + \frac{\mu}{\rho}\nabla^2 \mathbf{u}^* \tag{1.11}$$

$$\mathbf{v} = \mathbf{u}_n + \frac{\mathbf{k1} + \mathbf{2k}}{2}\Delta t. \tag{1.12}$$

After acquiring the intermediate velocity $\mathbf{v}$, we must find the updated pressure field, and then the final velocity field. To do this, we use the pressure projection method [10] to solve for the pressure gradient. First, Helmholtz-Hodge Decomposition states that $\mathbf{v}$ can be decomposed into a divergence-free part $\mathbf{v}_d$ and the pressure gradient:

$$\mathbf{v} = \mathbf{v}_d + \nabla P. \tag{1.13}$$

By taking the divergence on both sides, we get

$$\nabla^2 P = \nabla \cdot \mathbf{v}, \tag{1.14}$$

which is the Pressure Poisson equation. We can solve for pressure with a variety of techniques, most commonly with some variation of the Preconditioned Conjugate Gradient method. A summary of such techniques can be found in [11].

Finally, we can use the new pressure gradient to update the velocity field, where we will have our final solution for this time step:

$$\mathbf{u}_n = \mathbf{v} - \frac{\Delta t}{\rho}\nabla P. \tag{1.15}$$

Spatial Discretization

Spatial Discretization of the Navier-Stokes equations is an incredibly broad topic that is still a highly-active field of research. This section summarizes the technique we use most often in this work, Finite Difference Method.

Finite Difference Method (FDM) is the simplest of the three categories, and is most often used when computation time is a high priority. Foster and Metaxas [12] were the first to develop a fast method of solving the NS equations for 3D graphics applications using Finite Difference Method. In FDM, we split the domain into a grid, and we seek to solve the NS equations at every cell. Spatial derivatives are discretized by finite differencing. For example, using central differencing, a velocity derivative at some grid coordinate (i,j,k) may appear as follows:

$$\frac{\partial \mathbf{u}}{\partial x}_{(i,j,k)} = \frac{\mathbf{u}_{(i+1,j,k)} - \mathbf{u}_{(i-1,j,k)}}{2h}, \tag{1.16}$$

where h is the distance between cells.

An improvement to FDM is the Marker-And-Cell Method [13]. Here, instead of solving both velocity and pressure at cell centers, velocity is solved at the cell faces, also known as a staggered grid. This helps prevent pressure-velocity decoupling and checkerboard errors that often occur in normal FDM. Additionally, particles are seeded throughout the flow that move with the fluid velocity, which helps track which cells have fluid and the location/geometry of the fluid-air interface through time.

1.1.2 Boundary Management

Enforcing the no-slip condition on solid boundaries is extremely important for both accuracy and stability of the solver. While Finite Difference Method is computationally much simpler than other methods, special consideration is required for solid boundaries. Unless the solid boundary is exceedingly simple, such as a flat plane or a cube, it is

highly unlikely for a solid to perfectly coincide with the FDM computational mesh. Earlier studies, such as [14] and [12], simply rasterized the solid onto the computational grid, thus boundaries of the new solid match the grid. While this method is quite fast, depending on grid resolution and boundary motion/complexity, it will certainly lead to inaccurate flow, or even instability. Another option is to use curvilinear FDM grids [15], which can exactly fit the solid boundaries, including solving Navier-Stokes in flows across cylinders [16] or through circular pipes [17]. While highly accurate in certain situations, these methods can only be applied in cases where the solid boundaries are nonmoving and of very simple topology.

Foster and Fedkiw [18] improved FDM boundary management (on Cartesian grids) to allow for much more accurate simulation around solid moving bodies. Here, the fluid is seeded with massless particles that move by convection. The interface between a solid/air and the fluid is determined by the particles' positions. This way, a cell can easily contain both fluid and solid, rather than either 100% fluid or 100% solid. At each time step, the incompressible Navier-Stokes equations are solved for the entire fluid domain. For cells that contain solid, the boundary condition $\mathbf{u}_{fluid} \cdot \mathbf{n} = \mathbf{u}_{solid} \cdot \mathbf{n}$ is enforced, so that no fluid particles can intersect the boundary. The no-slip condition is relaxed to allow particles to flow across the boundary tangentially, which is much easier to work with in a FDM framework. However, this method also has several limitations. For one, with this algorithm only one polygon can be used as a solid boundary in a cell, and it is less clear how to deal with cells with multiple polygons (for example, in corners). Additionally, since the boundary management is done after the Navier-Stokes equations are solved, the forces imposed by the solid wall do not guarantee conservation of mass or momentum, leading to less accurate flow, and possibly loss of fluid mass.

One of the most commonly used methods for boundary management with FDM is the Immersed Boundary Method. First developed by Peskin in the 1970's [19, 20] and refined in 2002 [21], this method allows for arbitrary moving and deforming 3D solid boundaries with very high accuracy and stability. With this algorithm, the solid boundary is modeled with massless fibers, denoted by Γ. These fibers are defined by massless points that move with the fluid velocity, and are used to compute the fibers'

parametric curves $X(s,t)$, where s is the parameter. Additionally, the forces along the fiber are defined as $F(s,t)$, which is typically a spring force that causes the fibers (and fiber particles) to snap back into their correct positions. With these fiber forces, we can then interpolate the forces onto the fluid field at any point x as follows:

$$\mathbf{f}_{fiber}(x,t) = \int_{\Gamma} F(s,t)\delta(x - X(s,t))ds, \tag{1.17}$$

where δ is the Dirac δ function. In practice, a Gaussian weighting function, or similar smoothing function, often replaces δ, in order to more The Navier-Stokes momentum equation is then modified as follows:

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla P + \nabla \cdot \mathbf{T} + \mathbf{f}_{fiber}, \tag{1.18}$$

Thus, the immersed boundary method can be incorporated into existing fluid simulation software relatively easily, by just adding these force terms.

Over the years, there have been a huge number of modifications to the Immersed Boundary Method algorithm to optimize the solutions for various problems, such as those with elastic boundaries [22], rigid boundaries [23], and many variations thereof. A detailed review of such Immersed Boundary Methods can be found in [24]. In this work, we use the method detailed in [21].

1.1.3 Validation

Proper validation of any fluid simulator is critical to our work. The Navier-Stokes solver that we use is the TOUGH2 general-purpose numerical simulatin program, which was originally developed by E.O. Lawrence Berkeley National Laboratory and University of California, Berkeley. This software package has been extensively validated with many sample problems, discussed in detail in [25]. Further validation that demonstrates very high accuracy of the boundary management methods can be found in [26].

1.2 Smoothed Particle Hydrodynamics

An attractive alternative to solving fluid flows using mesh-based methods described above are meshless methods, which are usually much faster and can handle deforming of liquid with relative ease. One of the first to use particle-based methods in graphics applications was [27], modeling effects such as fire, fireworks, and sand to great effect. However, these primitive methods did not allow for interactions between particles. More sophisticated methods, such as Smoothed Particle Hydrodynamics (SPH), can be used to model solid deformation fluid flow at high speeds and reasonable accuracy. For example, in recent years SPH has been used to simulate shallow water flow [28], lava flow [29], and melting ice [30]. The algorithm is described below:

1.2.1 Algorithm

SPH [31] is a meshless method originally developed in 1977 to model the motion of astronomical phenomenon. Later, [32] and [33] adapted it to explicitly solve the equations of motion at unconnected points, or "particles," within the domain, each storing its own mass, density, pressure, position, and velocity. Unlike Eulerian-based methods, such as FDM and FEM, no computational mesh is required, and particles are free to move in the flow. For a quantity A, and for any point $\mathbf{r}$ in the domain, we can estimate the value of A at that point by the following equation:

$$A(\mathbf{r}) =\approx \iiint_{\Omega} A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)d\mathbf{r}', \tag{1.19}$$

where W is a smoothing kernel with h radius. We discuss the smoothing kernel in more depth later in this chapter. We can discretize this function as follows:

$$A(\mathbf{r}) = \sum_{j=1}^{N} A_j \frac{m_j}{\rho_j} W(\mathbf{r}-\mathbf{r}_j,h), \tag{1.20}$$

where m_j is the mass of particle j, and ρ_j is the fluid's density at particle j.

Using integration by parts on Equation 1.19, we can also derive the equations for the gradient and Laplacian of A as follows:

$$\nabla A(\mathbf{r}) = \sum_{j=1}^{N} A_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j, h), \tag{1.21}$$

$$\nabla^2 A(\mathbf{r}) = \sum_{j=1}^{N} A_j \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r}_j, h) \tag{1.22}$$

At the beginning of a time step, for each particle i, we first search for all neighboring particles within some distance h. Subsituting rho for A in Equatin 1.20, the particle's density is computed as follows:

$$\rho_i = \sum_{j=1}^{N} m_j W(\mathbf{r}_i - \mathbf{r}_j, h), \tag{1.23}$$

where N is the number of neighboring particles, m_j is the mass of particle j, r is a particle's position, and $W(r, h)$ is a smoothing kernel of radius h. In SPH, fluid is actually assumed to be semi-compressible, and so to find we pressure, we use the constitutive equation

$$P_i = c^2(\rho_i - \rho_0), \tag{1.24}$$

where c is the speed of sound and ρ_0 is the rest density, which we set to $1050 kg/m^3$[1]. Higher values for c represent greater incompressibility, but will cause the simulation to become unstable if Δt is too high. Specifically, the Courant-Friedrichs-Lewy (CFL) condition requires the following for stability [34]:

$$\Delta t \leq \frac{h}{c}. \tag{1.25}$$

In other words, the CFL condition states that the tim steps must be small enough such that a pressure wave (travelling at speed c) can not "skip over" a particle.

Once density and pressure are computed, we can compute the forces as follows:

$$\mathbf{f}_i^{pressure} = -\sum_{j=1}^{N} \frac{m_j}{\rho_j} \frac{P_i + P_j}{2} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h), \tag{1.26}$$

$$\mathbf{f}_i^{viscosity} = \mu \sum_{j=1}^{N} \mathbf{u}_j \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h). \tag{1.27}$$

We note the $\frac{P_i + P_j}{2}$ term in the pressure gradient equation above. Normally, we would simply substitute P_j for A_j in Equation 1.21 to find ∇P. However, this could

lead to particles i and j applying different forces to each other, which is a violation of Newton's third law. So, it is common practice to use the average of P_i and P_j for this term.

Once we know the forces due to the pressure gradient and viscosity everywhere at every particle, since the mass of each particle is previously defined, it is very simple to compute the accelerations, new velocities, and new particle positions at every time step, advancing the flow.

We also note the reason why this algorithm can perform with much faster performance than the more traditional Navier-Stokes solvers. Critically, if we do not create or destroy new particles, we get conservation of mass for free with SPH. All the computations listed above are completely explicit, and so no expensive linear system solvers are needed. When using FDM, however, conservation of mass requires us to implicitly solve the Pressure-Poisson equation, which can take a very long time. However, with SPH, we cannot achieve perfect incompressibility, as that would require c to be extremely high, and so SPH solvers will never be as accurate as FDM solvers. That said, numerical experiments have shown SPH to maintain at least order 1 accuracy, compared to order 2 accuracy in using FDM [35].

1.2.2 Weighting Functions

The choice of weighting function is vitally important to the accuracy and stability of Smoothed Particle Hydrodynamics. There are many different weighting functions developed for the SPH algorithm, each with their particular uses. We note that all smoothing kernels should have the follow properties [31],[36]:

$$\iiint_{\Omega} W(\mathbf{r}, h) d\mathbf{r} = 1, \tag{1.28}$$

$$\lim_{h \to 0} W(\mathbf{r}, h) = \delta(\mathbf{r}), \tag{1.29}$$

$$W(\mathbf{r}) \geq 0. \tag{1.30}$$

With these properties, the weighting function becomes a normalized averaging function for a given neighborhood of points. Also recommended [33] is that the kernel have a compact range:

$$W(\mathbf{r}, h) = 0, r \geq A \cdot h, \tag{1.31}$$

for some positive constant A. Some weighting functions do not meet this property, such as the Gaussian kernel [32]):

$$W(\mathbf{r}, h) = \frac{1}{\sqrt{\pi}h} e^{-u^2}, \tag{1.32}$$

where $u = \mathbf{r}/h$, but these techniques are far more computationally expensive to work with, with minimal benefit [37]. There are an enormous number of weighting functions developed that fulfill these requirements, though, such as the B-spline [38]:

$$W(\mathbf{r}, h) = \frac{1}{h} \begin{cases} 2/3 - u^2 + 0.5u^3 & : 0 \leq u < 1 \\ (2-u)^3/6 & : 1 \leq u < 2 \\ 0 & : u \geq 2 \end{cases} \tag{1.33}$$

the Q-spline [38]:

$$W(\mathbf{r}, h) = \frac{1}{120h} \begin{cases} (3-u)^5 - 6(2-u)^5 + 15(1-u)^5 & : 0 \leq u < 1 \\ (3-u)^5 - 6(2-u)^5 & : 1 \leq u < 2 \\ (3-u)^5 & : 2 \leq u < 3 \\ 0 & : u \geq 3 \end{cases} \tag{1.34}$$

the quadric [39]:

$$W(\mathbf{r}, h) = \frac{1}{h} \begin{cases} 3(4-u^2)/32 & : 0 \leq u < 2 \\ 0 & : u \geq 2 \end{cases} \tag{1.35}$$

the cosine [40]:

$$W(\mathbf{r}, h) = \frac{1}{h} \begin{cases} \frac{3\pi^2}{8(\pi^2+3)}(1 - \frac{u^2}{4})(1 + cos(\frac{\pi u}{2} & : 0 \leq u < 2 \\ 0 & : u \geq 2 \end{cases} \tag{1.36}$$

and the sextic: [41]:

$$W(\mathbf{r},h) = \frac{315}{(}64\pi h^9) \begin{cases} (h^2 - ||r||^2)^3 & : 0 \leq ||r|| < h \\ 0 & : ||r|| \geq 2 \end{cases} \tag{1.37}$$

Each such smoothing kernel has their own advantages and disadvantages when it comes to computational cost, sensitivity to choice of h, sensitivity to particle positions, and overall stability [37]. Studies such as [42] and [43] have shown that the Q-spline generally performs with the highest stability, due to its smooth second derivative function. We have tried all of these weighting functions, and have found Q-spline and the sextic polynomial kernels to perform the best. Most of the time, we follow [44] and use the sextic kernel, due to the much faster computational time, since we do not need to take the square root of $||\mathbf{r}||$ at any point using this formula.

However, while all of these preceding weighting functions work very well for computing the density, they have major problems in pressure gradient and viscosity computation. In all cases, the derivative of the weighting function at $\mathbf{r} = 0$ is zero. This means, as two particles become closer and close together, the pressure forces between them will approach zero, leading to particle clustering and eventually instability. A better technique would be to find a weighting function with a derivative that is maximal at $\mathbf{r} = 0$, and use this weighting function only for the pressure gradient step.

In 1996, [45] solved this problem by developing the spikey kernel:

$$W(\mathbf{r},h) = \frac{15}{(}64\pi h^6) \begin{cases} (h - ||r||)^3 & : 0 \leq ||r|| < h \\ 0 & : ||r|| \geq 2 \end{cases} \tag{1.38}$$

which has the following gradient:

$$\nabla W(\mathbf{r},h) = -\frac{45}{\pi h^6}\frac{\mathbf{r}}{||\mathbf{r}||} \begin{cases} (h - ||r||)^3 & : 0 \leq ||r|| < h \\ 0 & : ||r|| \geq 2 \end{cases} \tag{1.39}$$

This gradient has approaches a maximum as $\mathbf{r}$ approaches zero, so it will create a more realistic repulsive force as particles move closer to each other. Later, [41] validated the success of this kernel by using it in their own problem to great success.

Experimentally, we have also found that the spikey kernel works much better in the pressure gradient step, than any of the preceding weighting functions.

1.2.3 Boundary Management

The enforcement of boundary conditions is one of the most challenging problems in SPH. There are a wide variety of techniques to solve this problem in different contexts.

The simplest such method is to model the solid surface as a layer of boundary particles that apply repulsive forces to the nearby fluid particles [46],[47]. This repulsive force, PB_{ij} between fluid particle i and boundary particle j is a function of r_{ij}, the distance between both particles:

$$PB_{ij} = D\left[\left(\frac{r_0}{r_{ij}}\right)^{n_1} - \left(\frac{r_0}{r_{ij}}\right)^{n_2}\right]\frac{x_{ij}}{r_{ij}^2}, \tag{1.40}$$

where D, n_1, n_2, and r_0 are parameters. The stability of this method is very sensitive to the choices of these parameters, however. Notably, for optimal stability and accuracy, r_0 - the cutoff distance - must be kept rather small, about the average space between fluid particles [48]. Later [49] refined this method to allow for sharp corners. Monaghan et. al showed in several works that this method of boundary management can be used in a variety of applications, such as dropping boxes [50] and sinking ships [51]. In 2010, [52] further improved the efficiency of this method by performing the repulsive force computations from the boundary surface directly, rather than requiring thousands of boundary particles. However a small r_0 would lead to large instability problems in our cardiac blood flow simulations, since particles on one side of the wall would certainly interact with particles on the other side of the wall.

A second, common method of boundary management using SPH is by modeling the solid as boundary fluid particles [53]11, [54]. Here, quasi-fluid particles are used to model the entire solid. Contrast with the previous repulsive boundary particle method in that the entire solid is filled with particles, rather than just the boundary surface. Each quasi-fluid particle maintains its own mass, density, and pressure, just like a normal fluid particle. At each time step, density and pressure are updated for each

quasi-fluid particle in exactly the same way as the rest of the fluid, and the normal fluid particles interact with these solid particles no differently than other fluid particles. This way, the solid naturally provides a repulsive force against the fluid, and is not dependent on sensitive parameters like the previous method. These methods were used in [55] to successfully model simulate free surface flows. Later, [56] validated this method with several standard 2D and 3D experiments, such as the backward-facing step problem, flow across a cylinder, and flow within a pipe.

However, this method does encounter some disadvantages. First, these solids must be reasonable thick to ensure stability. If the solids are too thin, the computed density of nearby fluid particles will be too low, causing unrealistic pressure gradients and pressure oscillations [48]. Additionally, if there is fluid on either side of the wall, particles on one side of the wall may interact with particles on the other side. Later, however, [57] further improved this method by smoothing the pressure near the boundaries, greatly reducing potential oscillations, but will not help issues of interacting particles on either side of a thin wall.

In current state of the art SPH system, ghost particles, introduced by [58], are one of the most widely-used methods for very accurate and stable boundary management. Here, fluid particles on one side of the wall are reflected with the same mass, density, and pressure on the other side of the wall. This method guarantees adherence to the no-slip condition if velocities of the mirrored particles are also equal, but with opposite sign. A free slip condition can also be used by forcing equal velocities and equal signs onto the mirrored particles [59].

This algorithm works very well for flat surfaces, but curved boundaries are a more difficult problem, since reflected ghost particles may be too dense or two sparse, depending on the direction of the curvature. Recently, [43] developed a method to allow for a pseudo-ghost particle method on curved boundaries. They achieved this by generating a regular grid of psuedo-fluid particles within the solid (similar to the boundary fluid particles method described above). The density and pressure are computed in the same way as boundary fluid particles, but the velocity of the psuedo particles are

determined by interpolating reflected ghost particle velocities. More recently, [60] developed a more refined full ghost-particle method to allow for curved geometries with much greater accuracy.

However, ghost particles are also not well-suited for problems in which the solid is thin and complex. In these difficult problems, fluid particles on each side of the thin surface will produce their own ghost particles, which the fluid particles on the other side will include in its list of neighbors during density/force computation, generating instability.

Boundary management in SPH is a highly active field of research, and many other techniques exist for various problems. For example, [61] developed a method to solve highly viscous flows across boundaries, resulting in highly accurate eddy formations. However, this method also struggles with complex boundaries [62]. Later, [63] solved SPH with highly complex surfaces, but only for rigid, nonmoving, 2-manifold geometries.

1.3 Previous Work in Cardiac Blood Flow Simulations

With the rapid development of high-resolution cardiac CT, patient-specific blood flow simulation is quickly becoming one of the central goals in the study of cardiac blood flow. Earlier work in blood flow simulation used less refined models. For example, [64] was the first to extract boundaries from MRI data to perform patient-specific blood flow simulations. Later, [65] and [66] used simple models of the left side of the heart, with smooth ventricular walls, and imposed boundary conditions in the valve regions.

More recently, [67] have published a framework for simulating atrioventricular blood flow, and showed simulation results using a complete model of the left side of the heart, including the atrial venae and an aortic stub, together with modeled valve kinematics. Later, [1] used smoothed 4D CT data to simulate left ventricular blood flow, and compared the flow around through the aortic valve in a healthy heart and two diseased hearts. Our work is influenced by these papers, and we improve the heart model by using higher-quality CT scans of normal subjects and live patients with cardiovascular

disease that allow us to capture the complex details of the heart walls and trabeculae. In contrast, the geometry used in [67] was obtained from scans based on data from the Visible Human Project [68], while the kinematics was transfered to the model from MRI data, and the models derived from CT data in [1] were highly smoothed and were not useful for understanding the true interactions between the blood flow and the walls.

Our approach (similarly to the one in [67]) uses predefined motion for the valves, whose asynchronous opening and closing provides a simple geometric mechanism for taking care of those boundary conditions. This approach relies on the reasonable assumption that the left ventricle drives essentially all of the dynamics of the blood flow in the left side of the heart.

Recently, [69] implemented the approach of [70] to obtain a system that can efficiently deal with complex geometric data, such as a system of blood vessels. In 2010, [2] coupled fluid simulation with solid deformation, which is particularly useful in surgical simulations. However, performing solid mechanic simulations on trabeculae structures is computationally infeasible with modern equipment, so we only use completely predetermined solid motion in our experiments.

In the field of SPH, attempts at blood flow simulations have been comparatively few. Microscale simulations through extremely small vessels on the scale of individual blood cells is a common application, such as in [71] and [72]. In 2004, [73] developed a framework for realtime blood flow simulations through blood vessels to assist in virtual surgery. This work used FEM to model the solid dynamics of the vessel, which, while functional for problems such as these, would be completely infeasible in a cardiac flow simulation. Later, [74] improved on this framework by using repulsive boundary particles and allowed for interactions between blood flow and vessel deformation. This allows for much simpler handling of the boundary conditions, and also for more complex vessel geometries. However the solid wall geometry was still relatively simple with overall little motion compared to the left ventricle.

Chapter 2

Patient-Specific Modeling and Visualization of Blood Flow Through the Heart

Currently, valvular blood flow can be monitored using imaging techniques such as Doppler ultrasound and MRI. However, the spatial resolutions of such techniques are low, and it is not possible to observe the detailed interaction of the blood flow and the endo-cardial surface of the heart, so the formation of cardiac thrombus remains difficult to predict. If a physician were able to visualize or quantitatively measure the detailed alteration of the blood flow by altered contraction, he might be able to make a better diagnosis or treatment plan. Therefore, in this chapter, we seek to find ways to perform flow modeling and visualizations, given a 3D model of a heart.

2.1 Data Acquisition

The CT images were acquired on a 320-MSCT scanner (Toshiba Aquilion ONE, Toshiba Medical Systems Corporation). This advanced diagnostic imaging system is a dynamic volume CT scanner that captures a whole-heart scan in a single rotation, and achieves an isotropic 0.5mm volumetric resolution with less motion artifact than the conventional 64-MSCT scanners. A conventional contrast-enhanced CT angiography protocol was adapted to acquire the CT data in this work. After the intravenous injection of the contrast agent, the 3D+time CT data were acquired in a single heart beat cycle when the contrast agent was circulated to the left ventricle and aorta, so that we were able to achieve an optimal intensity difference level between the blood pool and the left ventricular myocardium. After acquisition, 3D images were reconstructed at 10 time phases in between the R-to-R waves using ECG gating. The acquired isotropic data had an in-plane dimension of 512 by 512 pixels, with an effective atrio-ventricular region

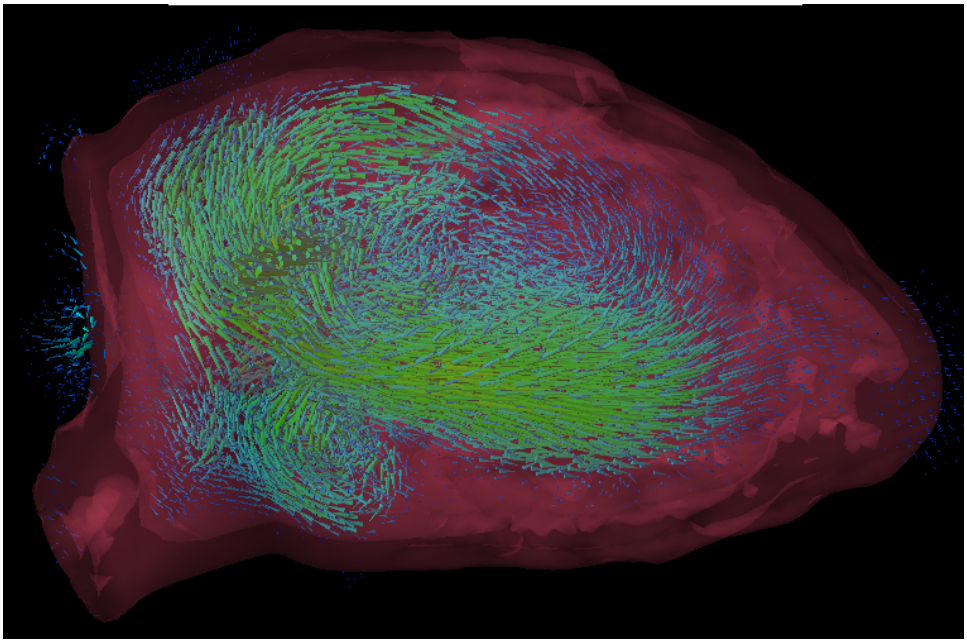


Figure 2.1: Velocity field visualization of full heart before systole

measuring about 300^3 pixels.

The left ventricle region was extracted from the dataset using an initial median filtering, followed by isosurfacing and mesh cleanup and smoothing. The surface was extracted at mid diastole, and its motion was transferred from the smooth mesh motion obtained from the same CT data. An example mesh can be seen in figure 2.2.

2.2 Simulation System

We use the same Immersed-Boundary Navier-Stokes simulation method described in the preceding chapter to simulate the flow through the aortic valve. The heart models used here are embedded in a computational mesh of 100^3 cells on which the full NS equations are solved using FDM. The blood is modeled as a Newtonian fluid, with viscosity of 4mPa· s and density of 1050kg/m^3. We use 0.005 second time steps (0.5% cardiac cycle). A simulation of a single cardiac cycle takes about 6 hours to complete on a machine with an Intel i7 processor and 16GB of RAM.

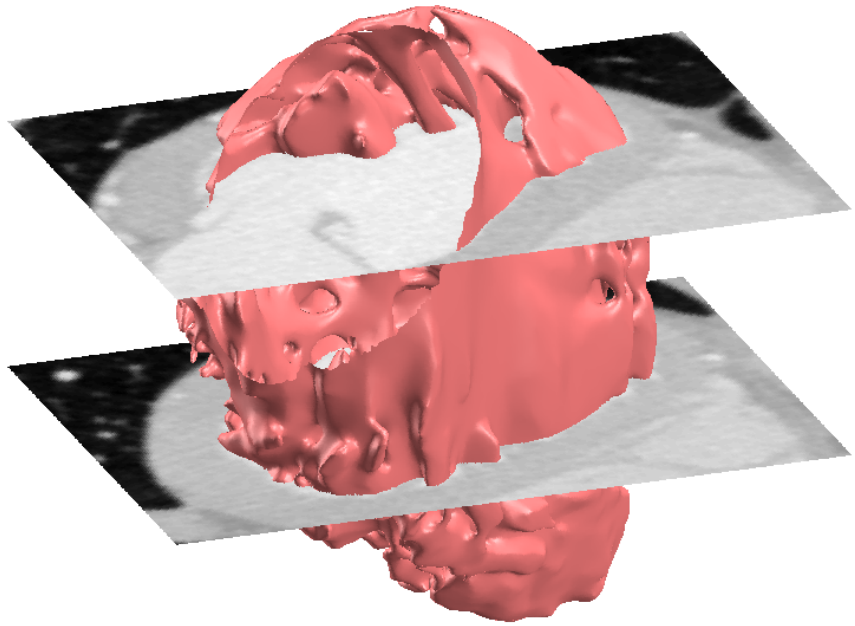


Figure 2.2: View of a detailed mesh extracted from CT data using isosurfacing. Note the complex trabeculae inside the heart.

2.2.1 Validation

We qualitatively validated the results of our simulation by the inspection and visual verification of several doctors. In addition, our simulation methods have been proven in [26] to be extremely accurate. It is possible to noninvasively obtain velocity data from MRI data, but it is prone to error and can only be retrieved in one direction. In the future, we plan to use phantoms to perform validation of these results.

2.3 Visualization

As mentioned previously, we adapted the framework described in [67] to perform the fluid simulation. With the fluid velocity fields and level sets generated for each time step, we use Paraview to visualize the simulations. We also visualized flows in a simulated diseased heart suffering from a large perfusion defect area created by reducing motion

in the anterior apical area, as would be expected from ischemia or infarction in the distribution of the left anterior descending coronary artery.

2.3.1 Full Heart

We first performed a visualization of the velocity field of the entire heart, as seen in figure 2.1. The velocity of the blood at a given point is represented by a cone pointed in the direction of the flow. A cone's size increases linearly as the magnitude of the velocity increases. Additionally, we adjust the color of a cone by first setting its hue to 160 (blue), and then linearly lowering this value to a minimum of 0 (red) as velocity increases.

We then used finite differencing to compute the flow's vorticity at every point in the field. We use the magnitudes to create a volumetric visualization of vorticity, where brighter areas are associated with higher vorticites and greater rates of rotation and circulation.

2.3.2 Cross-Section

Next, we examined cross-sections of the heart, and visualized the velocities here. This way, we have a clearer picture of how each of the structures and trabeculae affect the flow of blood. We visualized the velocity field in the same way as above, representing the velocity at each point with a colored cone. Screenshots of the visualization can be seen in figures 2.4 and 2.5.

2.3.3 Flux

In the previous two sets of visualizations, we have seen how the complex structure of the walls of the heart affects the flow of blood. We now introduce two additional methods we use to visualize the flux across a certain region. To do this, we simply inserted a plane at the desired location and orientation, interpolated to sample the velocities across the plane, and projected the velocities onto the plane's normal vector. Two examples can be seen in figure 2.6.

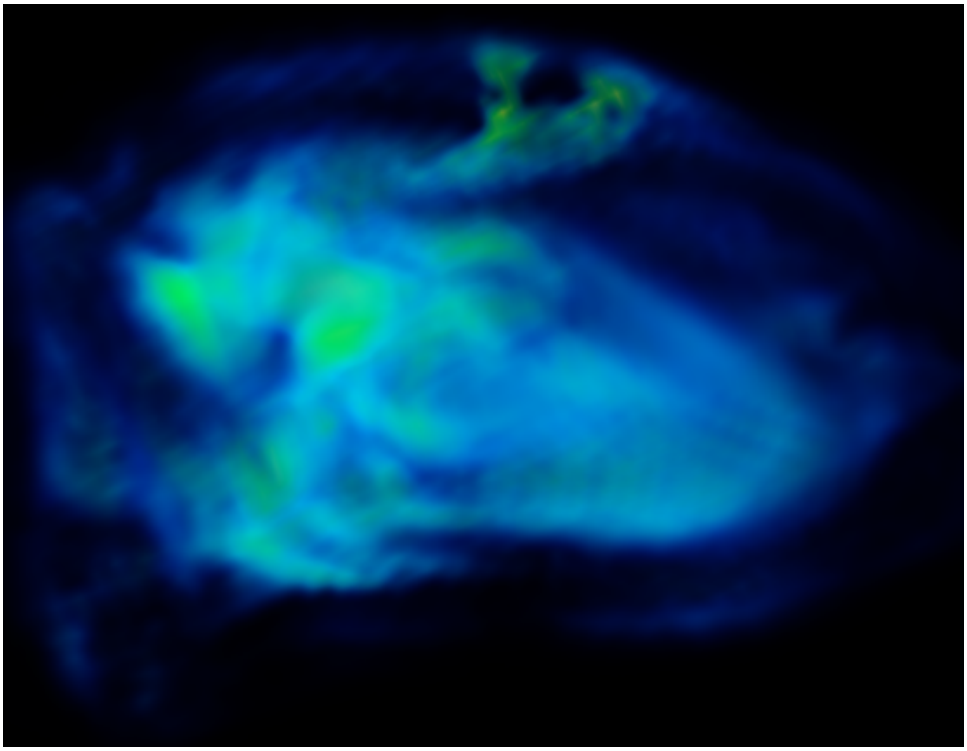


Figure 2.3: Vorticity field visualization of full heart. Green regions correspond to higher magnitudes of vorticity and thus greater rates of rotation. Compare to figure 2.1, taken at the same time step.

2.4 Results and Discussion

2.4.1 Visualizations of Healthy Heart

Full Heart

The visualizations of the velocity field and the vorticity field for the entire heart can be seen in figures 2.1 and 2.3, respectively. The images generated from visualizing the vorticity are visually stunning. We see in figure 2.3, the areas with the largest vorticities are close to the base, near the papillary muscles and the valves, which suggests that the fluid is spinning more rapidly here. Note that in figure 2.1, which is the velocity field taken at the same time step as that in figure 2.1, we see a number of large vortices in the same location.

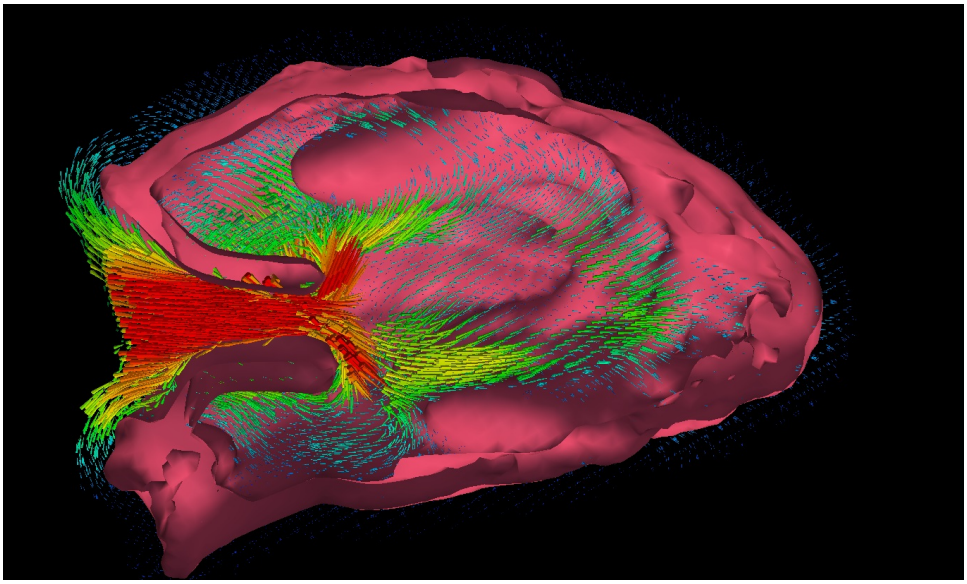


Figure 2.4: Cross-section of heart during late diastole

Cross-Sections

As displayed in figures 2.4 and 2.5, the cross-section visualization allows us to see the interactions between the blood flow and the complex heart walls, which, to the best of our knowledge, has never been possible before. In our full animation, we can see how blood moves through and around the trabeculae, the papillary muscles, and the valves during the entire cardiac cycle. This is far more accurate than earlier attempts of cardiac blood flow simulations with smooth walls, and we can use this to find problems with diseased hearts, such as stagnant fluid in the trabeculae that increases the risk of thrombosis. In a later section, we use this way of visualizing flow to compare the the blood flow of a healthy heart to that of a diseased heart, demonstrating its potential clinical use.

Flux

To test the flux visualization method, we place planes at two different locations. First, we place a flux plane near the apex of the heart and visualize the flow across it. In figure 2.6 (a), we see the flux across this region during ventricular systole. We then insert a

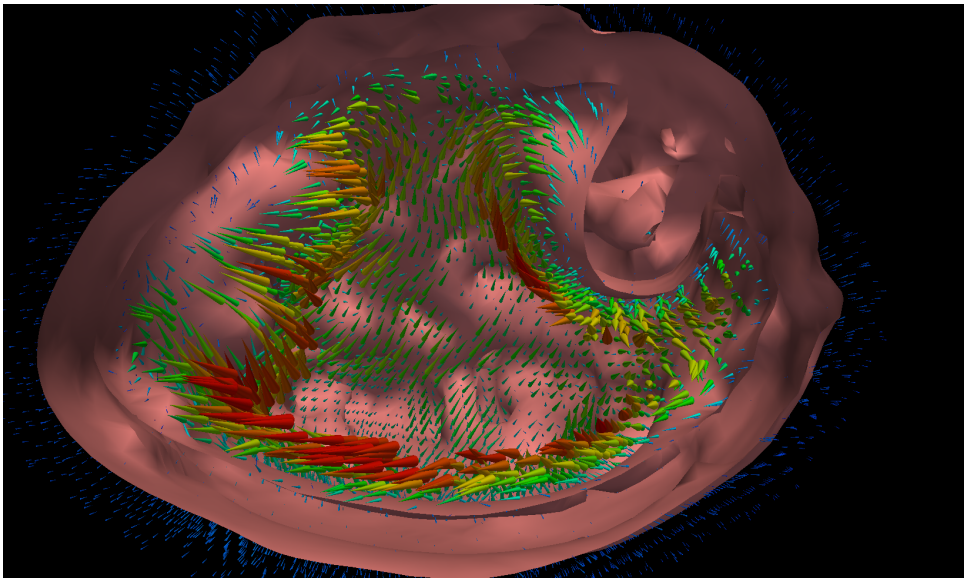


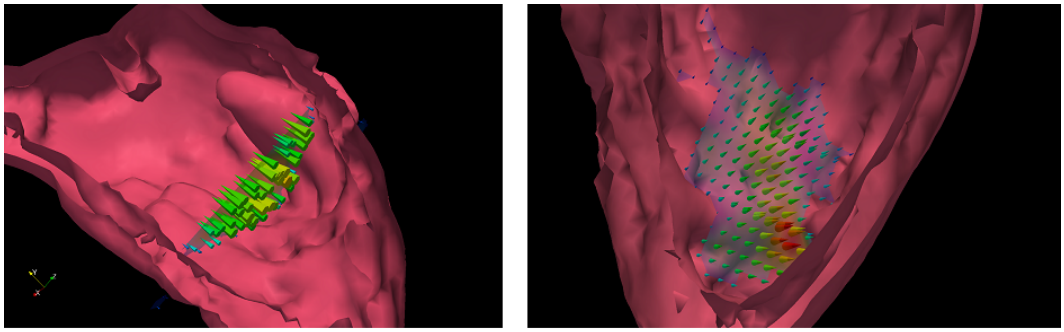
Figure 2.5: Cross-section view of apical region during systole. Only fluid velocities directly against the heart walls are displayed, in order to more clearly see the complex interactions.

flux plane directly against the trabeculae of the heart walls. From figure 2.6 (right), we can clearly see how, during ventricular systole, the trabeculae contract, expelling the blood. In the future, we plan to run additional simulations on diseased hearts whose trabeculae do not contract. We will be using these methods of visualization in order to determine if blood remains trapped, leaving the patient at greater risk of a blood clot.

2.4.2 Comparison Between Normal and Diseased Heart

We now compare the flow fields of a healthy heart to that of a diseased heart, whose apex does not contract properly during ventricular systole. In figure 2.7, we show both hearts at three different stages of the heart cycle. The top row displays a normal heart, and the bottom row displays the diseased heart. The first column shows mid diastole, the second shows late diastole, and the third shows systole.

In mid diastole, the fluid entering through the mitral valve reaches the lower regions and causes the blood to circulate (figure 2.7 (a)). During late diastole, we see that a large amount of blood enters the healthy heart at high velocity (figure 2.7 (b)), causing



(a) (b)

Figure 2.6: Left: visualization of flux through the apex of the heart during systole; Right: velocity field directly against trabeculae during systole

significant turbulence and further circulation. Finally, as the heart fully constricts during systole, the fluid velocities everywhere remain high and the blood is adequately ejected (figure 2.7 (c)).

However, we see that in the case of the diseased heart, the inflow during diastole is significantly reduced and the blood in the apex is fairly stagnant (figure 2.7 (d), (e)). Then, in systole, we see again that the flow velocities are lower than that of the healthy heart and does not fully expel the blood (figure 2.7 (f)). This result would be extremely useful to a doctor, as this lack of circulation across all stages of the heart cycle presents a clear risk of clotting to the patient. We also note that the size of the diseased heart is larger than that of the normal heart, since the apex is not properly contracting during systole.

For a quantitative comparison, we seek the velocities around the apex of the heart. To do this, we selected a small spherical region close to the trabeculae, and computed the velocity magnitudes within each cell in the sphere at each time step for both the healthy and diseased heart. We plot the results for two cardiac cycles in figure 2.8. Late diastole begins at approximately time steps 2 and 52, and systole begins at time steps 10 and 60. We see that, on average, the velocities in the diseased heart in this region are about half of those in the healthy heart. In particular, we see that during systole, the blood velocities at the apical region of the diseased heart tend not to increase as

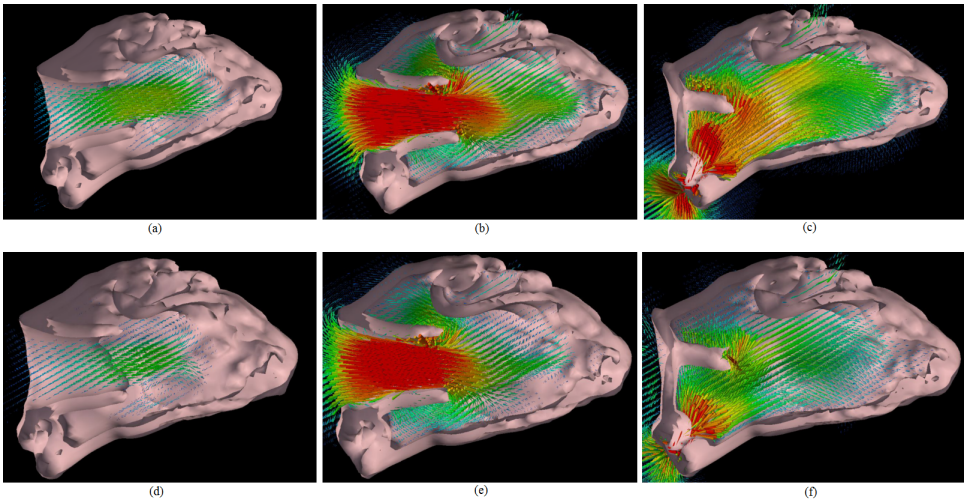


Figure 2.7: Comparison of blood flow between a normal heart (top) and a diseased heart with hypokinesis in the apical region (bottom). First column: Mid diastole; Second column: Late diastole; Third column: Systole. Note how that in all three stages, the blood in the apical region remains relatively stagnant in the diseased heart, increasing the risk of clotting. We also note that the apex of the diseased heart remains large in all stages, since it is not properly contracting.

they do in the healthy heart, suggesting that this fluid is not being effectively expelled, creating a risk of clotting.

2.5 Chapter Conclusion

We have demonstrated how we can now model blood flow through hearts extracted from high-resolution CT data that contain very complex moving boundaries. We have also described a number of ways to visualize the modeled flow of blood through the heart, which can be extended and be useful to doctors in diagnoses and treatment plans. Finally, we have shown how these visualizations can be used to compare the simulated blood flow through a normal heart and one suffering from disease to show that how the abnormal heart may be at increased thrombosis risk due to noncirculating blood. In the next chapter, we look to perform similar visualizations on other models of diseased hearts derived from patient-specific CT data, with even better model reconstruction techniques. In particular, we are looking to understand the movement of blood through trabeculae and detect risk of clots in these regions, and how stagnant fluid may affect

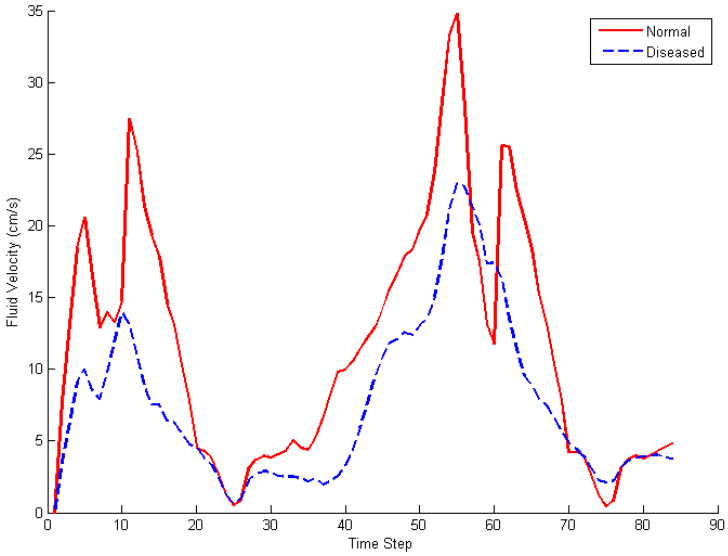


Figure 2.8: Average magnitudes of velocities in the apex of the healthy and diseased heart during two cardiac cycles. Late diastole begins at approximately time steps 2 and 52, and systole begins at time steps 10 and 60.

the flow.

Chapter 3

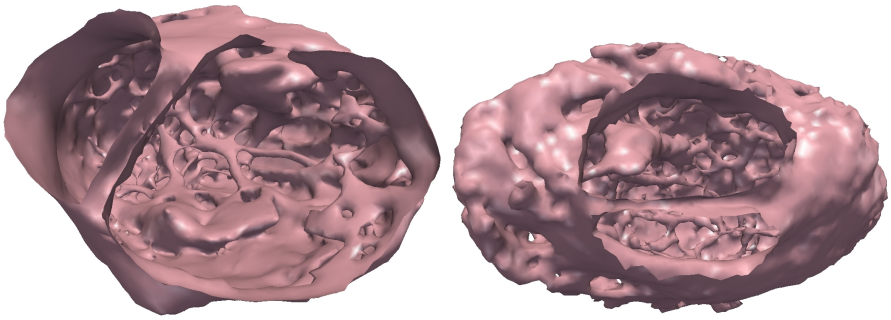
Using High Resolution CT Data to Model and Visualize Patient-Specific Interactions Between Trabeculae and Blood Flow

In the previous chapter, accurate heart models were achieved by generating a mesh from high-resolution CT data at mid-diastole. Then, motion was transferred to this model from the smooth mesh motion obtained from the same CT data to create the animation. This allowed for more realistic features to be present on the heart walls in the simulation, including the papillary muscles and some trabeculae. However, while this approach was an improvement from the smooth-wall assumption, the trabeculae were missing details and did not move accurately.

In this chapter, we use an improved method of creating the mesh to capture these smaller details and generate a more accurate simulation. To the best of our knowledge, we are able to visualize blood flow in unprecedented detail.

3.1 Data Acquisition

The CT images were acquired on a 320-MSCT scanner (Toshiba Aquilion ONE, Toshiba Medical Systems Corporation) using contrast agent. This advanced diagnostic imaging system is a dynamic volume CT scanner that captures a whole-heart scan in a single rotation, and achieves an isotropic $0.3mm$ volumetric resolution. A conventional contrast-enhanced CT angiography protocol was adapted to acquire the CT data in this work. After the intravenous injection of contrast agent, the 3D+time CT data were acquired in a single heart beat cycle when the contrast agent was circulated to the left ventricle and aorta. After acquisition, 3D images were reconstructed at 10 time phases in between the R-to-R waves using ECG gating. The acquired isotropic data had an



(a) (b)

Figure 3.1: Meshes reconstructed from CT data (valves removed). (a) Healthy heart (b) Diseased heart.

in-plane dimension of 512 by 512 pixels.

The detailed cardiac shape features can be used as the boundary conditions and incorporated in a fluid simulator to derive the hemodynamics throughout the whole heart cycle. Our goal in defining these boundary conditions is to capture the fine detail structures of the myocardium, as well as the one-to-one vertex correspondence between frames, which is required in the fluid simulation. There has been much recent work in cardiac reconstruction, such as [75], who combined high-resolution MRI images with serial histological sectioning data to build histo-anatomically detailed individualized cardiac models to investigate cardiac function. In this work, we use the techniques described in [76]. Here, snake based semi-automatic segmentation is used to acquire the initial segmentation from high resolution CT data for an initial (3D) frame of data. The initial mesh is generated as an isosurface of the segmentation, which is deformed to match the shape of the heart in each consecutive frame. Also, during the deformation, we achieve the necessary one-to-one correspondence between frames.

The aortic and mitral valves are thin and move fast, and so the CT data is not currently able to adequately capture these details. We add 3D models of the valves created from ultrasound data to each mesh in the sequence, and open and close the valves at the appropriate time steps.

Reconstruction results for a healthy and a diseased heart can be seen in Figure 3.1. Note the high level of structural detail at the apex. To the best of our knowledge, this

has never been simulated before.

3.2 Fluid Simulation

We use the same Immersed-Boundary Navier-Stokes simulation method described in the preceding chapters to simulate the flow through the aortic valve. The heart models used here are embedded in a computational mesh of 100^3 cells on which the full NS equations are solved using FDM. The blood is modeled as a Newtonian fluid, with viscosity of 4mPa· s and density of 1050kg/m^3. We use 0.005 second time steps (0.5% cardiac cycle). A simulation of a single cardiac cycle takes about 5 days to complete on a machine with an Intel i7 processor and 16GB of RAM.

3.3 Visualizations

With the fluid velocity fields and level sets generated for each time step, we use Paraview to visualize the simulations. We analyzed a healthy heart and two diseased hearts, and we describe below our visualization methods and our results.

Blood Flow Velocity

We performed a visualization of the velocity field within the heart, as seen in Figure 3.3. The velocity of the blood at a given point is represented by a cone pointed in the direction of the flow. The size of cone increases linearly as the velocity increases. We also adjust the color of a cone by first setting its hue to 160 (blue), and then linearly lowering this value to a minimum of 0 (red) as velocity increases. The magnitude of fluid velocity ranges from 0-.9 m/s.

Streamline visualizations are shown in Figure 3.2. The color at a point within a streamline is chosen in the same way as the cones described above. In order to disambiguate direction, we add cones that point in the direction of flow

3.3.1 Blood Residence Time

In addition to the blood flow velocities, we wish to visualize the residence time of blood within the heart. By doing so, we can quantitatively determine regions of the heart that are at greater risk of thrombus, as slower flows are known to be a significant factor predisposing to thrombus formation.

In order to compute the residence time of blood, we must first determine which regions in the computational domain are interior to the heart. This region changes at every time step, due to the deformation of the heart. We find this interior area by determining which cells are within concave regions of the heart mesh. For each empty (non-solid) cell in the domain at index (i, j, k), we check whether there exists a pair (l_1, l_2) such that both $l_1, l_2 > 0$, and either both cells $(i + l_1, j, k)$ and $(i - l_2, j, k)$ are solid, cells $(i, j + l_1, k)$ and $(i, j - l_2, k)$ are solid, or cells $(i, j, k + l_1)$ and $(i, j, k - l_2)$ are solid. While this method does not guarantee that all cells within concave regions are determined, our results show that it accurately determines each cell interior to the heart.

At the initial time step, ten thousand particles are generated randomly within the heart. At the beginning of each time step, new particles are generated at the valves, allowing fresh blood particles to enter the heart during diastole. Each new particle has an initial age of zero, and this age is incremented at every time step. While some particles are also generated outside the aortic valve, these never enter the heart and are completely removed during systole, and so they do not meaningfully affect the results.

At each consecutive time step, we determine a particle's velocity by interpolation, given the fluid velocities at the center of each cell. Each particle's new position is calculated using Euler time integration. Then, any particle in a cell exterior to the heart is removed from the system, and the average particle residence time within each cell can then be easily determined. We run this for four cardiac cycles and create volumetric visualizations, as seen in Figure 3.4. Here, blue represent regions in which average residency is less than 1 cardiac cycle, green-yellow represents 1-3 cardiac cycles, and red represents 3-4 cycles.

We can also take advantage of these particles in validation of our simulation, by computing an estimated ejection fraction. During systole, we know exactly how many particles there originally existed in the system, and how many are being expelled at each time step. To estimate the ejection fraction, we simply divide the total number of deleted particles by the original number of particles.

Estimated ejection fraction can be calculated using particles to validate our simulation. During systole, we know exactly how many particles there originally existed in the system, and how many are being expelled and deleted at each time step. To estimate the ejection fraction, we simply divide the total number of deleted particles by the original number of particles.

3.4 Discussion

The streamline visualizations provide detailed information on the trabeculae-blood interaction. Figure 3.2(b), taken during diastole, demonstrates how the complex surface causes the flow to move through and around the empty spaces between the trabeculae. Then, in Figure 3.2(c), during systole, we see another example of how the blood is forcefully expelled out of the spaces between the trabeculae, rather than simply flowing directly towards the aortic valve as older methods with simpler meshes have suggested.

The simulation and visualization methods are performed described above on three different hearts. The first is a healthy heart with no visible medical problems with an ejection fraction of about 50%. The second is a heart that has simulated hypokinesis, where the motion of the heart walls is decreased at the apex by a maximum of 50%. The third comes from a patient who has post tetralogy of Fallot repair. This heart is known to suffer from right ventricle hypertrophy, significant dyssynchrony in the basal-midseptum of the left ventricle, and a decreased left ventricle ejection fraction of about 30%.

The streamline visualizations provide detailed information on the trabeculae-blood interaction. Figure 3.2(b), taken during diastole, demonstrates how the complex surface causes the flow to fill the empty spaces between the trabeculae. Then, in Figure 3.2(c),

during systole, we see another example of how the blood is expelled out of the spaces between the trabeculae, rather than simply flowing directly towards the aortic valve as older methods with simpler meshes have suggested.

Validation is a difficult task, as current imaging techniques, such as PC-MRI, are not able to capture flow information at the required level of detail for useful comparison. We performed a partial validation by comparing the estimated ejection fraction to the true ejection fraction. The computed ejection fraction is approximately 45% for the healthy heart, 40% for the hypokinesis heart, and 30% for the dyssynchronous heart. These values for the healthy and dyssynchronous heart are in agreement with the true values, so we have confidence in the rest of our results. Performing similar validation techniques to a smoothed healthy heart model, we computed an ejection fraction of about 40%, slightly lower than that of our complex model. However, it may not be especially useful to compare the accuracy of different modeling methods using this approach, as the ejection fraction does not give information about the flow local to the apex, the region of primary interest.

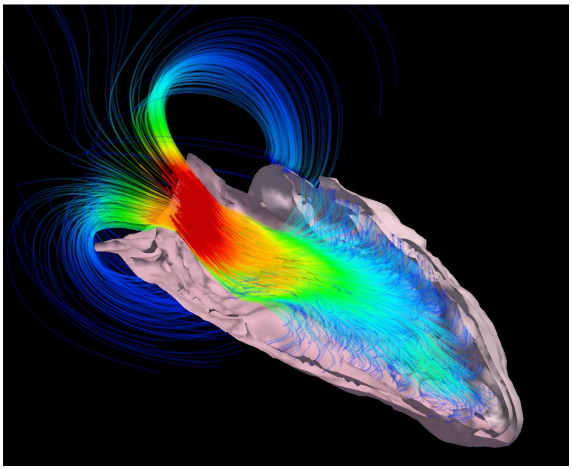
Velocity field visualizations are illustrated in Figure 3.3. We can see that in the healthy heart, the inflow during diastole is significant and fairly uniformly distributed, circulating blood throughout the heart. During systole, the velocity field throughout the heart remains high, and fluid in the apex moves toward the valves. In Figure 3.3(c), we see more detail of the interactions between blood flow and the trabeculae, as the blood is visibly expelled from these regions. However, in the heart suffering from hypokinesis, we find that the velocity field is much weaker toward the apex during both diastole and systole. In Figure 3.3(f), we also see that the trabeculae are no longer adequately expelling blood as they do in the healthy heart case. We also see in Figure 3.3(g)-(i) that the flow patterns in the heart with dyssynchronous heart wall movement appears non-normal, with overall lower velocities and even less fluid being pushed out from the trabeculae.

We then compare the visualizations of the average particle residence times for each of the three simulations, as seen in Figure 3.4. Each of these images were made at the same time step, at the start of systole, after four cardiac cycles. We find that in Figure 3.4(a),

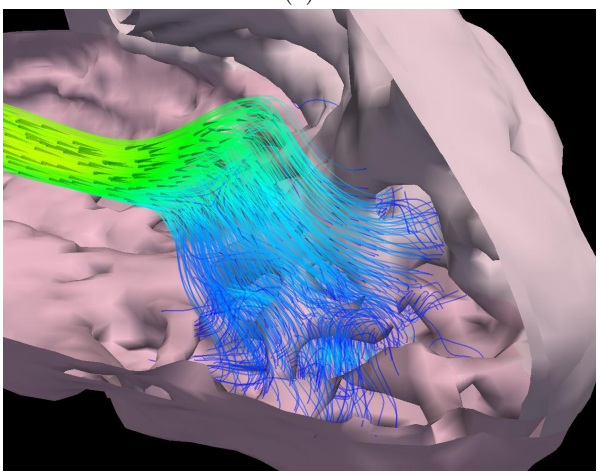
in the healthy heart, nearly the entire domain contains blood with average residence time of less than three cycles, suggesting that the blood is not remaining stagnant, and turning over well between cardiac cycles. In contrast, Figure 3.4(b) shows that in the heart suffering from hypokinesis, the average residence time is significantly higher near the walls, particularly near the hypokinetic apex. Finally, in Figure 3.4(c), we find that a very significant region of the blood has a long residence time, suggesting that due to the low ejection fraction and relatively low fluid velocities, blood is not being adequately circulated and thus is remaining stagnant near the walls, again, particularly toward the apex of the heart.

3.5 Chapter Conclusion

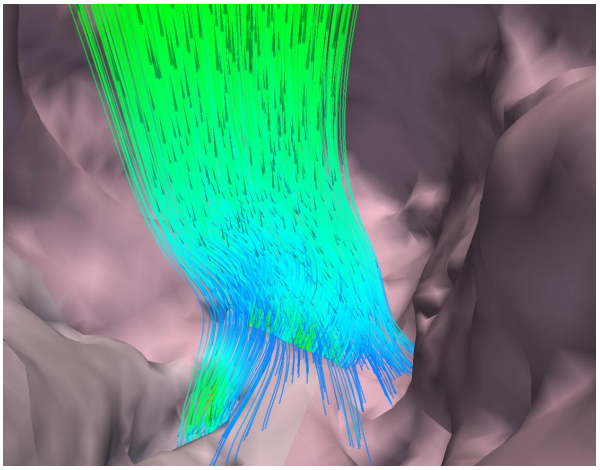
In this chapter, we have described our new framework to generate detailed mesh sequences from CT data, and used them to run patient-specific blood flow simulations. We then created several visualizations to reveal the interactions between the complex trabeculae of the heart wall and the blood, which has never been possible before, and used them to compare the flow fields between a healthy heart and two diseased hearts, which would potentially be extremely useful to doctors to help in diagnosis and treatment plans. This is the first time that intracardiac blood flow fields and their interaction with the heart wall have been investigated at this level of resolution.



(a)



(b)



(c)

Figure 3.2: Visualization of streamlines within the healthy heart. (a) Streamlines of cardiac blood flow during diastole. (b) Blood flow near apex during diastole. (c) Blood flow during systole at the apex, against the trabeculae.

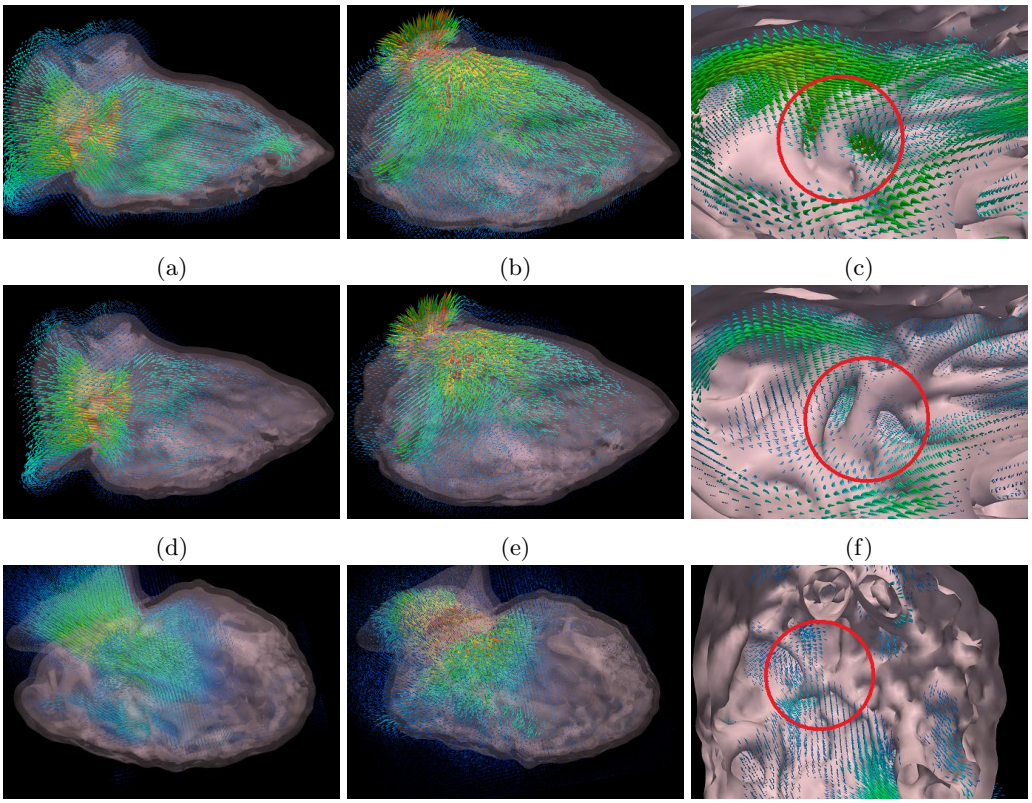
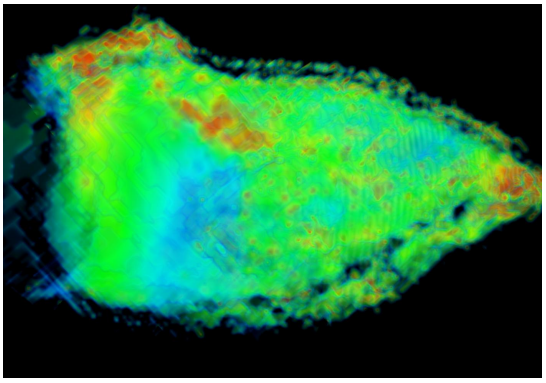
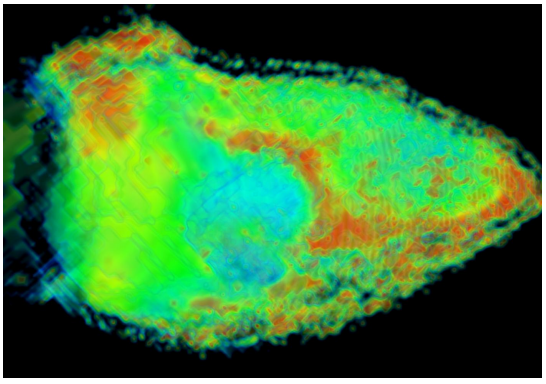


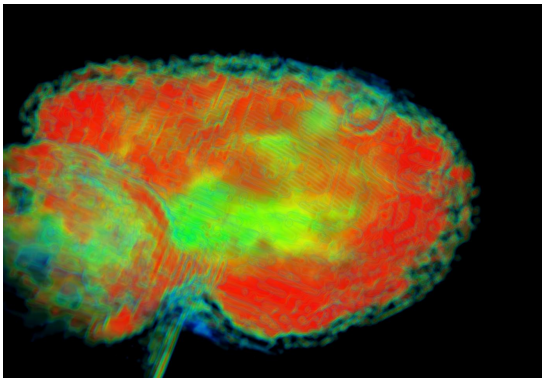
Figure 3.3: Velocity fields at various time steps for three different hearts. Top row: Healthy Heart, Middle row: Hypokinetic heart, Bottom row: Dyssynchronous heart. Left column: Diastole, Middle column: Systole, Right column: Velocity field at trabeculae during systole.



(a)



(b)



(c)

Figure 3.4: Visualization of average particle residence time. Colors closer to red represent longer average residence time. (a) Healthy Heart (b) Heart with Hypokinesis (c) Heart with dyssynchronous wall movement.

Chapter 4

The Importance of Trabeculae Structures in Cardiac Blood Flow Simulations

In the previous chapter, we used a sophisticated methods of extracting the heart and its motion from CT in order to capture the geometry of the trabeculae, and used fluid simulations to determine whether the motion and structure of the trabeculae interacts with the blood flow. While this work did show evidence of such interactions, we have not shown that these interactions are significant enough to justify the much higher cost of incorporating the trabeculae in blood flow simulations, rather than using more traditional smoothed-heart models. Thus, the purpose of this chapter is to fill this gap, by comparing the computed flow fields of simplified and complex versions of four patient-specific heart models, and visually and quantitatively show that these trabeculae structures are critical in developing the best, clinically-useful results.

4.1 Data Acquisition

The CT images were acquired on a 320-MSCT scanner (Toshiba Aquilion ONE, Toshiba Medical Systems Corporation) using contrast agent. This advanced diagnostic imaging system is a dynamic volume CT scanner that captures a whole-heart scan in a single rotation, and achieves an isotropic $0.3mm$ volumetric resolution. A conventional contrast-enhanced CT angiography protocol was adapted to acquire the CT data in this work. After the intravenous injection of contrast agent, the 3D+time CT data were acquired in a single heart beat cycle when the contrast agent was circulated to the left ventricle and aorta. After acquisition, 3D images were reconstructed at 10 time phases in between the R-to-R waves using ECG gating. The acquired isotropic data had an in-plane dimension of 512 by 512 pixels.

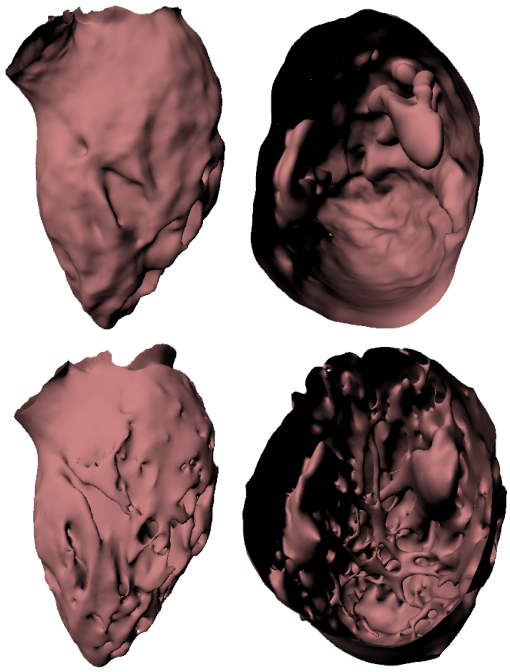


Figure 4.1: 3D meshes generated from high-resolution CT imagery. Patient 1 (Normal) Row 1: Smoothed, Row 2: Complex; Column 1: Outside, Column 2: Apex

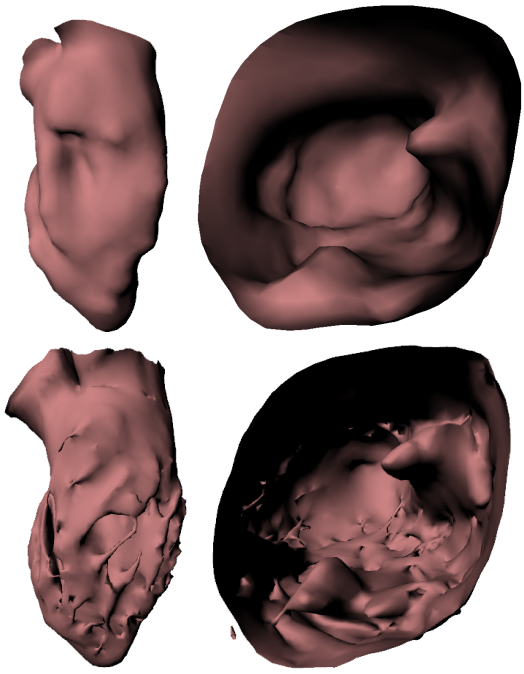


Figure 4.2: 3D meshes generated from high-resolution CT imagery. Patient 2 (Nonobstructive CAD) Row 1: Smoothed, Row 2: Complex; Column 1: Outside, Column 2: Apex

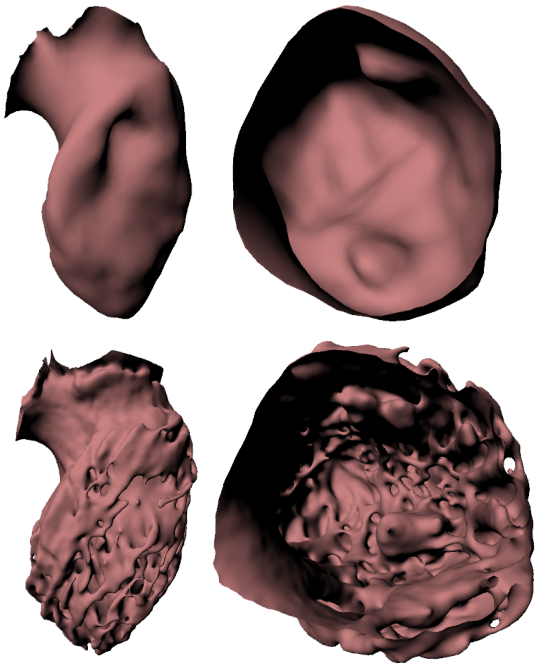


Figure 4.3: 3D meshes generated from high-resolution CT imagery. Patient 3 (Obstructive CAD) Row 1: Smoothed, Row 2: Complex; Column 1: Outside, Column 2: Apex

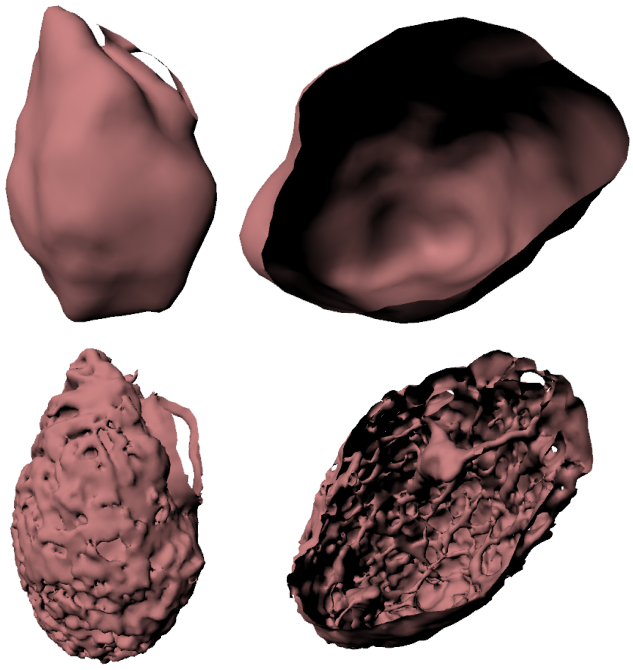


Figure 4.4: 3D meshes generated from high-resolution CT imagery. Patient 4 (Dyssynchrony) Row 1: Smoothed, Row 2: Complex; Column 1: Outside, Column 2: Apex

CT images were acquired from four patients. Patient 1's heart is healthy and functions normally. Patient 2 is 49 year old female with nonobstructive coronary artery disease. Patient 3 is a 62 year old patient with obstructive coronary artery disease. Finally, Patient 4 suffers from dysynchronous cardiac function. For each patient, we construct both "smoothed" (less trabeculae) and "complex" (more trabeculae) 4D models of their heart, described below.

4.1.1 Mesh Generation

In order to study the impact of trabeculae structures on patient-specific cardiac blood flow simulations, a high quality segmentation which captures these fine structures is required. However, accurately segmenting the complex structures is a challenging task due to their complexity and thin nature. The trabeculae structures, which are attached to the heart wall at their ends, and can be freely moved in the middle, form the topological structure called handle. Gao et al. [77] proposed and demonstrated an algorithm to segment 3D high resolution CT data by explicitly restoring topological handles. The location and geometry of these handles are suggested by a tool from computational topology, namely, persistent homology. Intuitively speaking, a handle will be detected if it goes to relatively high intensity in the CT image. The method has been proved to be accurate both topologically and geometrically. The segmentation results are reconstructed to 3D mesh to build a patient-specific cardiac endocardial surface model.

After the model was reconstructed at the end-diastole frame, which is where those structures are most expanded and cleared captured in the CT images, we deformed the model to other frames of the data in order to recover the LV motion [78]. The deformation is constrained by both the imaging information and the model geometrical information, such that to look for a balance between the model consistent between frames and model fitting to images at different frames. Since we were using the same model to segment all frames of the data, we were able to find one-to-one correspondence automatically through the segmentation. The one-to-one correspondence provides the velocity of the LV motion and is used to drive the blood flow simulation.

To compare the impact the trabeculae structures, we also reconstructed the smoothed models with trabeculae. The registration between smoothed models directly would not be reliable because of the limited anatomical information. We first build the correspondence between the smoothed models and complex models. Then using the registration of the complex models between frames, the smoothed models also captures the motion of the LV endocardial surface.

The aortic and mitral valves are thin and move fast, and so the CT data is not currently able to adequately capture these details. We add 3D models of the valves created from ultrasound data to each mesh in the sequence, and open and close the valves at the appropriate time steps.

4.2 Fluid Simulation

We use the same Immersed-Boundary Navier-Stokes simulation method described in the preceding chapters to simulate the flow through the aortic valve. The heart models used here are embedded in a computational mesh of 100^3 cells on which the full NS equations are solved using FDM. The blood is modeled as a Newtonian fluid, with viscosity of 4mPa$\cdot$ s and density of 1050kg/m^3. We use 0.005 second time steps (0.5% cardiac cycle). A simulation of a single cardiac cycle takes about 5 days to complete on a machine with an Intel i7 processor and 16GB of RAM.

4.2.1 Blood Residence Time

In addition to the blood flow velocities, we wish to visualize the residence time of blood within the heart. By doing so, we can quantitatively determine regions of the heart that are at greater risk of thrombus, as slower flows are known to be a significant factor predisposing to thrombus formation.

In order to compute the residence time of blood, we must first determine which regions in the computational domain are interior to the heart. This region changes at every time step, due to the deformation of the heart. We find this interior area by determining which cells are within concave regions of the heart mesh. For each empty

(non-solid) cell in the domain at index (i, j, k), we check whether there exists a pair (l_1, l_2) such that both $l_1, l_2 > 0$, and either both cells $(i + l_1, j, k)$ and $(i - l_2, j, k)$ are solid, cells $(i, j + l_1, k)$ and $(i, j - l_2, k)$ are solid, or cells $(i, j, k + l_1)$ and $(i, j, k - l_2)$ are solid. While this method does not guarantee that all cells within concave regions are determined, our results show that it accurately determines each cell interior to the heart.

At the initial time step, ten thousand particles are generated randomly within the heart. At the beginning of each time step, new particles are generated at the valves, allowing fresh blood particles to enter the heart during diastole. Each new particle has an initial age of zero, and this age is incremented at every time step. While some particles are also generated outside the aortic valve, these never enter the heart and are completely removed during systole, and so they do not meaningfully affect the results.

At each consecutive time step, we determine a particle's velocity by interpolation, given the fluid velocities at the center of each cell. Each particle's new position is calculated using Euler time integration. Then, any particle in a cell exterior to the heart is removed from the system, and the average particle residence time within each cell can then be easily determined. We run this for four cardiac cycles and create volumetric visualizations, as seen in Figure 4.5. Here, blue represent regions in which average residency is less than 1 cardiac cycle, green-yellow represents 1-3 cardiac cycles, and red represents 3-4 cycles.

4.3 Results and Discussion

Visualizations of average residency time for each patient after four cardiac cycles can be seen in Figure 4.5. Figures 4.5(a-b) are of Patient 1, smoothed and complex respectively, (c-d) are of Patient 2, (e-f) are of Patient 3, and (g-h) are of Patient 4. Each image was taken during diastole, and both images within a pair were taken from the same angle. We immediately notice large differences between the smooth and complex experiments for all four patients. In Patients 1 and 2, in particular the latter, we see that blood in the complex models have a significantly lower residency times than in the smoothed models.

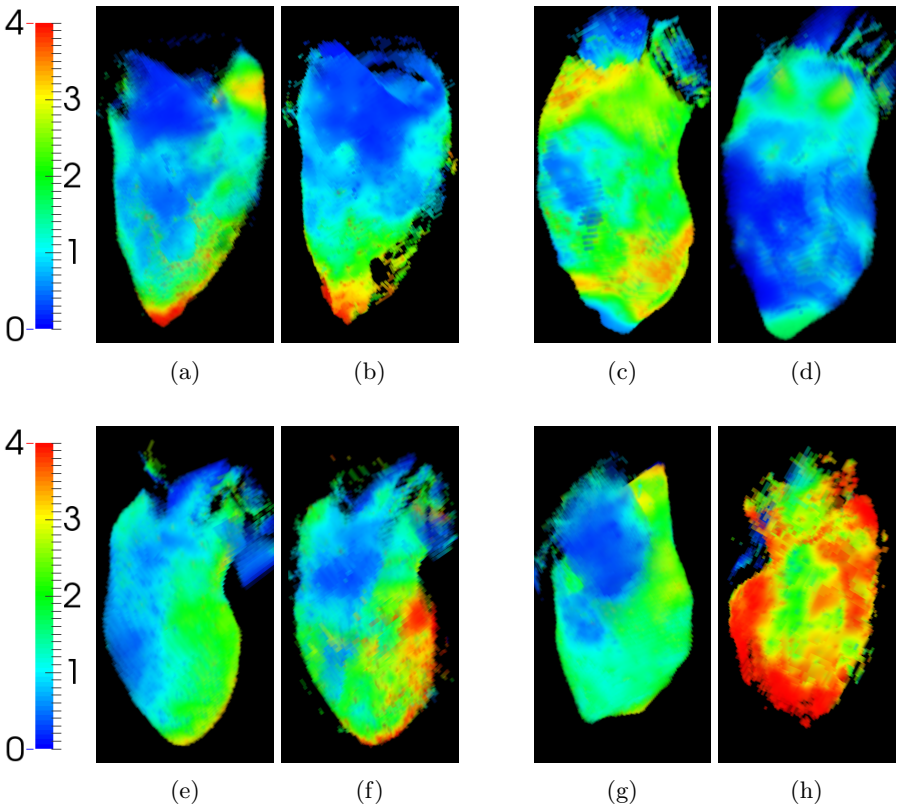


Figure 4.5: Average residency time visualizations for four blood flow simulations after four cardiac cycles. Blue regions represent areas of fresh blood, red regions represent blood that has been in the left ventricle for about four cycles. (a-b) Patient 1 [Normal] - Smoothed and Complex meshes, respectively; (c-d) Patient 2 [Nonobstructive CAD] - Smoothed/Complex; (e-f) Patient 3 [Obstructive CAD] - Smoothed/Complex; (g-h) Patient 4 [Dyssynchrony] - Smoothed/Complex. We note that in Patients 1 and 2, average residency time appears higher in the smoothed version, but for Patients 3 and 4, residency time is higher in the complex version.

	Mean Age (Cycles)	Standard Deviation	T-test p-value
Patient 1 (Smoothed)	1.425	0.93	6.1743e-04
Patient 1 (Complex)	1.353	0.94	
Patient 2 (Smoothed)	1.438	0.75	7.2427e-14
Patient 2 (Complex)	0.708	0.44	
Patient 3 (Smoothed)	1.078	0.76	1.7572e-78
Patient 3 (Complex)	1.486	1.03	
Patient 4 (Smoothed)	1.775	0.79	1.0817e-102
Patient 4 (Complex)	2.496	0.97	

Table 4.1: Mean and Standard Deviation of particle ages after four cardiac cycles.

This suggests relatively healthy trabeculae motion and fast blood turnover rates. While Patient 2 does suffer from non-obstructive CAD, hearts with non-obstructive CAD can still function normally when at rest. Therefore, these results are consistent with our expectations.

We then see the opposite effect in Patients 3 and 4. In both cases, the average residency time is much higher in the complex version, suggesting poorly-functioning trabeculae that is severely dampening the intraventricular flow. Since Patient 3 has obstructive CAD, her cardiac function is impaired, even at rest. Even more importantly, we note that in the smoothed cases, Patient 2 and Patient 3 appear to have very similar average residency times. However, in the complex cases, Patient 3 is clearly much worse. If just the simplified models are used in cardiac blood flow simulations, these clinically significant problems would be completely invisible to the physician.

Quantitative analysis corroborates our visual inspection of the particle age fields. In Table 1, we see the mean and standard devation of the average particle ages at the end of four cardiac cycles. We note that in both Patients 1 and 2, there is a significant drop in median residency time as we move from smoothed to complex models. In Patients 3 and 4, we see that the opposite: median residency time sharply increases in the complex case. This, also, suggests that healthy trabeculae helps to circulate residual blood within the ventricle so as to speed up the turnover rate, while poorly-functioning trabeculae of the older/obstructive CAD (Patient 3) or dyssynchronous (Patient 4) heart by acting as a cushion or trap, slowing the blood turnover rate.

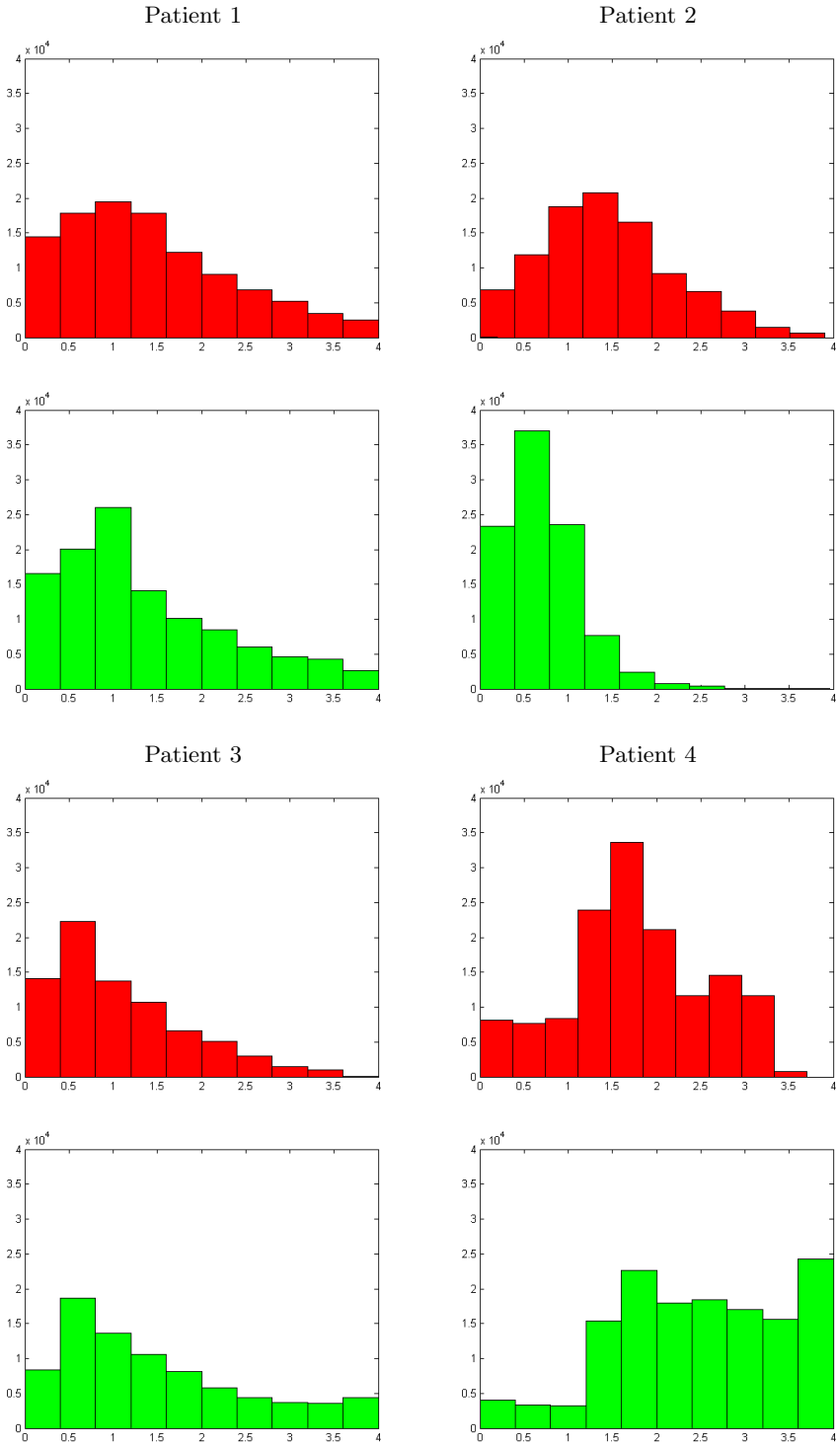


Figure 4.6: Histograms showing distribution of average particle age within cells. X-axis: Average Particle Age, Y-axis: Number of cells. Red: Smoothed, Green: Complex

Finally, in Figure 4.6, we plot histograms showing the distribution of residency time within each heart. The x-axis represents residency time in cardiac cycles, while the y-axis represents the number of cells whose average particle age falls in a given bin. While Patient 1 and Patient 2 have interest, we note that in Patient 2's case, the significantly smaller variance in the complex model . Additionally, we note that the clearly bimodal distribution in Patient 3's and Patient 4's complex cases reflect their poor blood turnover rate, and with Patient 3, this bimodal image is not visible in the smoothed model.

4.4 Chapter Conclusion

In this paper, we have described our method of deterimining the impact trabeculae geometry, or lack thereof, can have on the computed flow fields in a cardiac blood flow simulation. For example, in the cases of Patients 2 and 3, while they appear to maintain similar average residency times in simplified heart models, we have seen that by adding the trabeculae to the models, these residency times can significantly change. It is clear that these structures provide an important role in intraventricular hemodynamics, and thus their inclusion in future simulations is critical for the most clinically useful results.

Chapter 5

Patient-Specific Aortic Valve Blood Flow Simulations

Aortic valve disease-induced alterations in the blood flow pattern may promote chronic remodeling of the aortic root and the left ventricle. Although the mechanism of the aortic root remodeling is not fully understood, the hemodynamic pattern in the aortic valve vicinity is believed to play an important role. Color Doppler echocardiography and MR phase-contrast flow imaging are two clinical techniques used in quantifying valvular blood flow [79, 80]. However, both techniques are limited in imaging the blood flow in the aortic valve vicinity. Color Doppler echocardiography is not suitable for 3D flow velocity quantification because of the one dimensional imaging angle. MR flow imaging has a limited 3D spatial resolution that makes it difficult to quantify a high-resolution flow pattern in a small structure of the aortic root. Furthermore, the blood flow in the aortic vicinity exhibits high rates of acceleration and a turbulent pattern with high vorticity, which compromises the flow quantification accuracies of Doppler-based echocardiography and phase-contrast MRI. In [81] , it is found that the aortic flow quantification discrepancy found in color Doppler echocardiography and phase-contrast MRI mainly came from the flow vorticity.

In this chapter, we propose a framework to study the blood flow within the aortic valve vicinity using the computational fluid simulation and patient-specific aortic root models. Computational fluid simulation has been employed in the studies of the intra-ventricular flow and aortic flow. In 2011, Kulp et al [4] described a method of extracting the heart and its motion from CT to capture the geometry of the trabeculae and papillary muscles and show evidence of interactions between these structures and the blood flow. On the other hand, CT has become an important preprocedural imaging tool for assessing the aortic root anatomy in transcatheter aortic valve implantation patients,

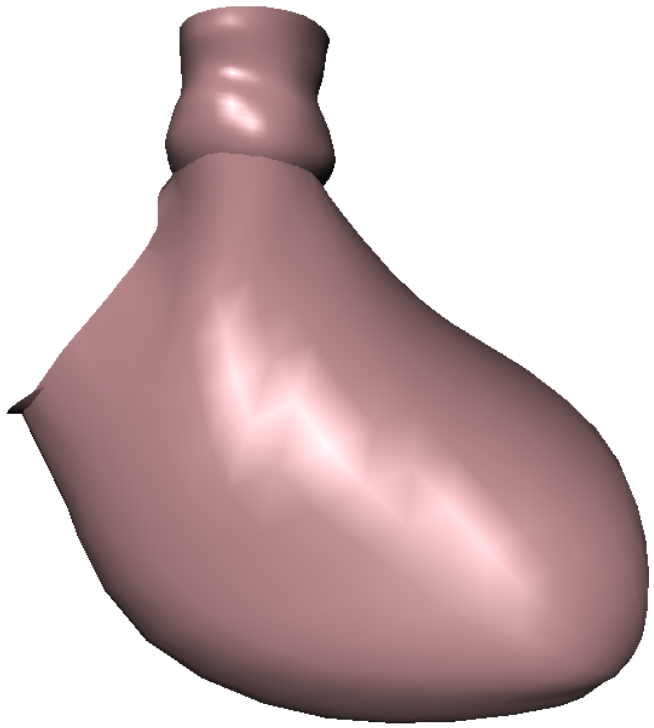


Figure 5.1: Our aortic root model is attached to the left ventricle model segmented from Visible Human Project data to form a complete system of the left ventricle, the left ventricle outflow tract, and the aortic root.

because of its high spatial resolution and high reproducibility [82]. In this study, we utilized CT as the imaging modality for accurate aortic valve modeling.

5.1 Data Acquisition

The aortic root models were reconstructed from contrast-enhanced 3D CT images that were acquired using a retrospective full R-R ECG gating protocol. In each heart cycle, 10 phases of 3D CT volumes were reconstructed at every 10% interval. The patient-specific aortic models were semi-automatically reconstructed using a valve segmentation research software (Siemens Cooperate Research). The aortic models of the 10 temporal phases were further aligned and registered to generate a smooth aortic root motion model. Images of the stenotic valve can be seen in Figure 5.2.

In addition to the geometry and motion of the valve, inflow boundary condition

plays a critical role in determining the accuracy of our simulation results. However, the inflow boundary condition is largely determined by the left ventricular function. We cannot use patient-specific left ventricle geometries in this work, as we are focused solely on the impact of valvular geometry and motion on the aortic flow patterns, independent of changes in LV function. Therefore, we attach our aortic root models to a standard 3D model of a human left ventricle, which was reconstructed from the high-resolution Visible Human Project datasets, as discussed by Hurmusiadis et al [83]. Ventricular motion was derived from a fiber-based deformation model, also described in [83]. Models generated using this method were used in [26] to simulate flow through the left ventricle with high accuracy, so we are confident that these models provide acceptable inflow velocity boundary conditions to the aortic root. We removed the aortic root structures generated from the Visible Human Project data, and manually attached our own reconstructed aortic root models, as seen in Figure 5.1. We then used this full, animated model as the solid boundary conditions in our fluid simulator, as described below.

5.2 Fluid Simulation

We use the same Immersed-Boundary Navier-Stokes simulation method described in the preceding chapters to simulate the flow through the aortic valve. The heart models used here are embedded in a computational mesh of 100^3 cells on which the full NS equations are solved using FDM. The blood is modeled as a Newtonian fluid, with viscosity of 4mPa$\cdot$ s and density of 1050kg/m^3. Since the aortic valve moves much faster than the rest of the left ventricle, we use much smaller time steps than normal, at 0.00005 seconds. A simulation of a single cardiac cycle takes about 5 days to complete on a machine with an Intel i7 processor and 16GB of RAM.

5.3 Visualizations

After solving the flow fields for both aortic root models, we then use Paraview to visualize the results. We are primarily focused on differences in flow velocity and

vorticity between the healthy and diseased valves. Our methods for visualization are described below.

Figure 5.3 gives visualizations of the velocity field of both valve models at two different time steps. Fluid velocity is represented by cones pointing in the direction of the flow and we change the hue of a cone depending on the magnitude of velocity at its location. At hue=160 (blue), velocity is approximately 3.5cm/s, and at hue=0 (red), velocity is approximately 190cm/s. Areas of flow velocities less than 3.5cm/s are not shown in our renders, as they tend to be noisy and less interesting in our analysis. Figures 5.3 (a) and (b) show the flow field through the healthy aortic valve in early and late systole, respectively, while figures 5.3 (c) and (d) give the flow field through the diseased aortic valve, also in early and late systole, respectively. To provide a more useful comparison, Figures 5.3 (a) and (c) were taken exactly the same time step and at the same angle, as were Figures 5.3 (b) and (d).

In Figure 5.4, we have visualizations of the flow vorticity at several time steps. Since vorticity is a scalar field, we render it volumetrically, rather than in the glyph-based manner of the velocity field. Similar to Figure 5.3, the color of a particular region represents the magnitude of the vorticity at that point. At hue=160 (blue), vorticity is nearly $0s^{-1}$, and at hue=0 (red), vorticity is approximately $3s^{-1}$. Like Figure 5.3, Figures 5.4 (a) and (b) are of the healthy aortic valve, while (c) and (d) are of the diseased valve. Each image in Figure 5.4 was taken at the same time step and at the same angle as their respective images in Figure 5.3.

Finally, Figure 5.5 include images of plotted streamlines in late systole through the healthy and diseased valves. Colors represent velocity magnitude at a given point in the streamline, and are scaled in precisely the same way as in Figure 5.3. This image was taken at the same timestep as in Figures 5.3 (b)/(d) and Figures 5.4 (b)/(d).

5.4 Results and Discussion

From Figures 5.3-5.5, we can see that this simulation framework provides a very clear view of the flow fields in both of our experiments. In our streamline visualization,

Figure 5.5, we see the formation of vortices in detail. Vortices are also easily visible in the velocity fields of Figure 5.3, especially in the diseased valve case, and in late systole. In Figure 5.6 (a), we compute the average velocities through both the diseased and healthy valves at the sino-tubular junction. The time steps plotted were temporally equally spaced, and in every time step, both valves were open. We can see that in early systole, flow through the aorta in both simulations had very similar velocities. However, in mid-late systole, flow in the aorta appears to drop more rapidly in our diseased valve simulation, showing the decreased efficiency of the stenotic aortic valve.

In Figure 5.4, the vorticity fields strongly agree with our findings in the previous images. In the sinus region, we note the high rotational energy against the stenotic valve, as compared to the healthy valve, which is consistent with the vortices visible in the streamlines and velocity field. We also note that regions of elevated vorticity are visible in both simulations at the regions above the sinus, especially in late systole. In Figure 5.6 (b) we have plotted mean vorticities during systole at the sinus and above-sinus regions. Only those points at least one full grid cell away from the wall were included in the mean vorticity computation. We see that the flow through the stenotic valve consistently produces mean vorticities significantly higher than the healthy valve in both regions. Interestingly, in the diseased case, vorticities seem to slightly grow through time in the region above the sinus, and decrease in the sinus region. This is possibly due to the effects of flow separations and shedding vorticies. This is also seen more dramatically with the healthy valve in the final plotted time step, with voriticites in the segment above the sinus sharply increasing, while decreasing at the sinus. Again, we hypothesize that this is due to vortex shedding caused by the valve beginning to close.

5.5 Chapter Conclusion

In this chapter, we have presented a new framework to simulate and visualize blood flow through patient-specific aortic root models. Our results provide a clear, high-resolution view of flow patterns, which have been previously not visible in traditional flow imaging techniques, such as Doppler-based echocardiography and phase-contrast

MR. We clearly can see differences in flow patterns through normal and abnormal aortic valve geometries and motion, which could potentially become a highly useful tool for doctors and scientists to assist in diagnosis and understanding valvular diseases.

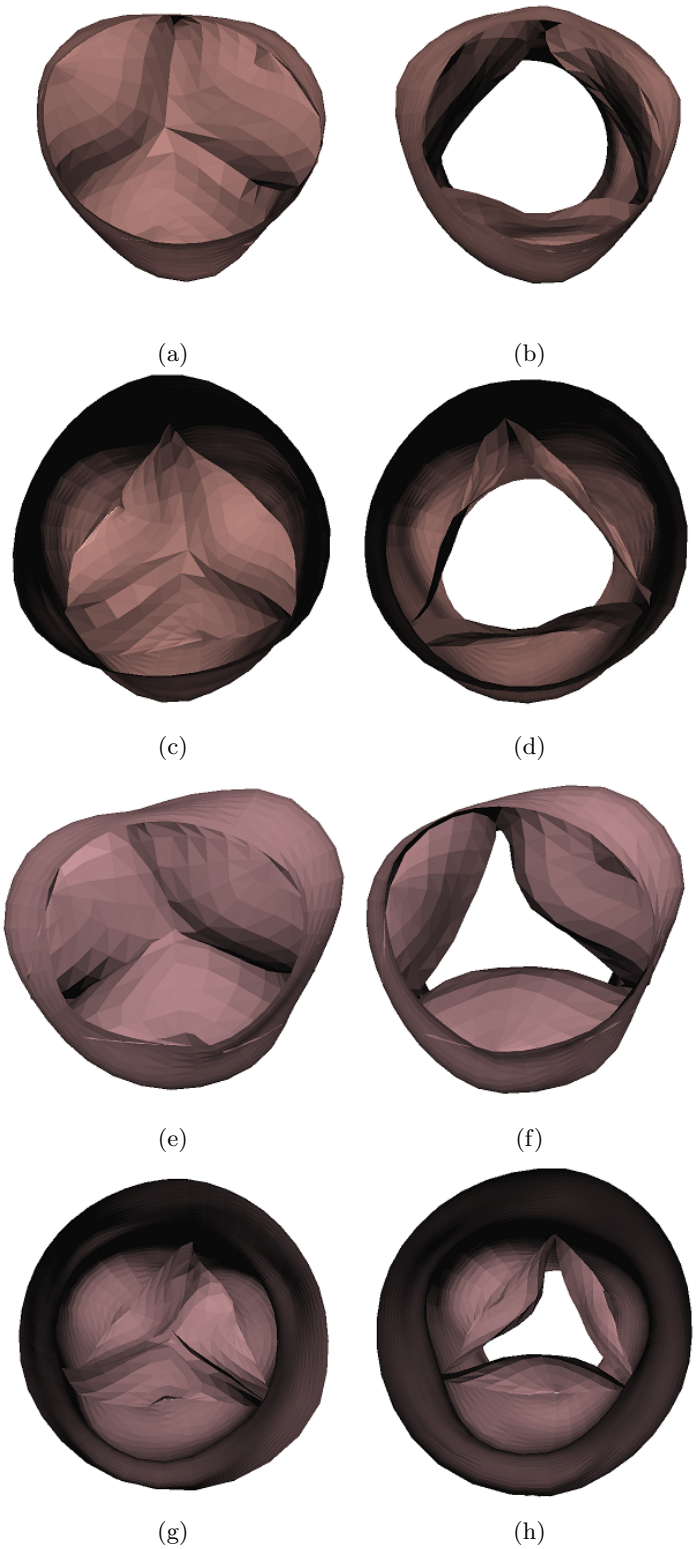


Figure 5.2: Four views each of both our healthy (top) and stenotic (bottom) aortic root models reconstructed from CT data.

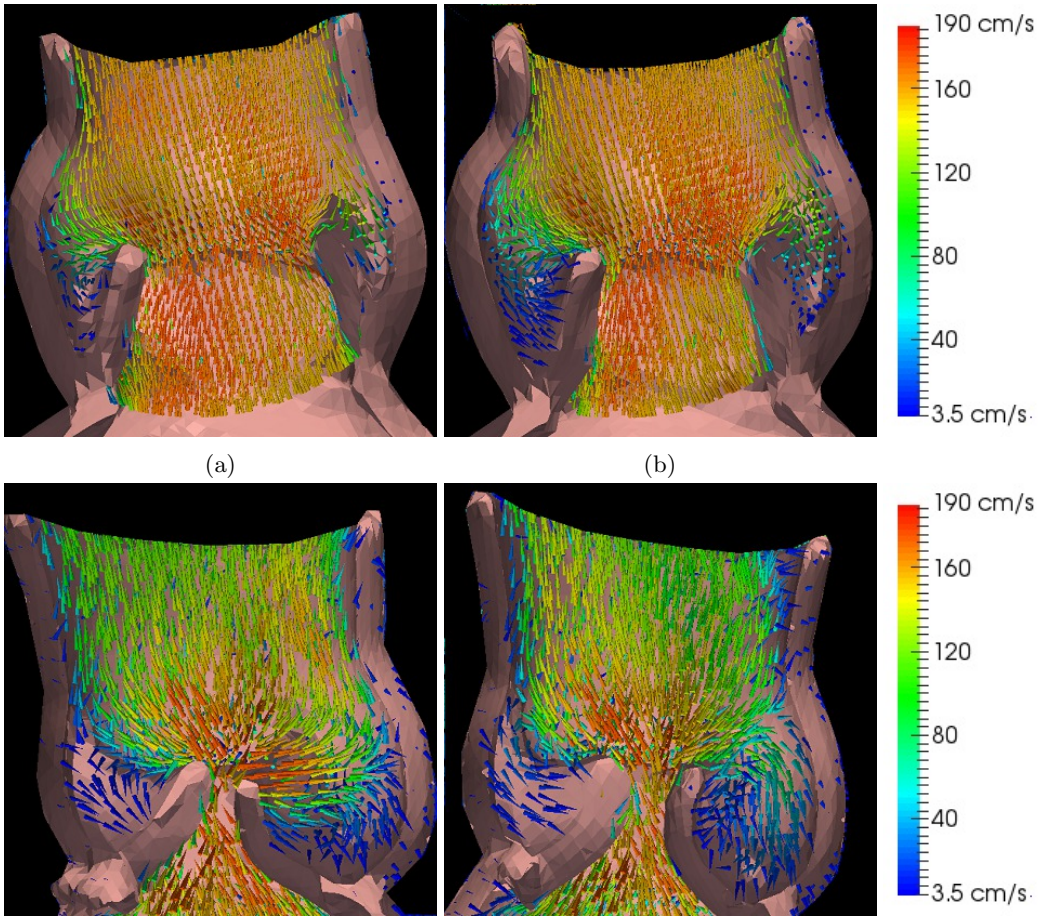


Figure 5.3: Velocity field visualizations. (a) Early systole, healthy valve; (b) Late systole, healthy valve; (c) Early systole, diseased valve; (d) Late systole, diseased valve. Both (a) and (c) were taken at the same time step, as were (b) and (d)

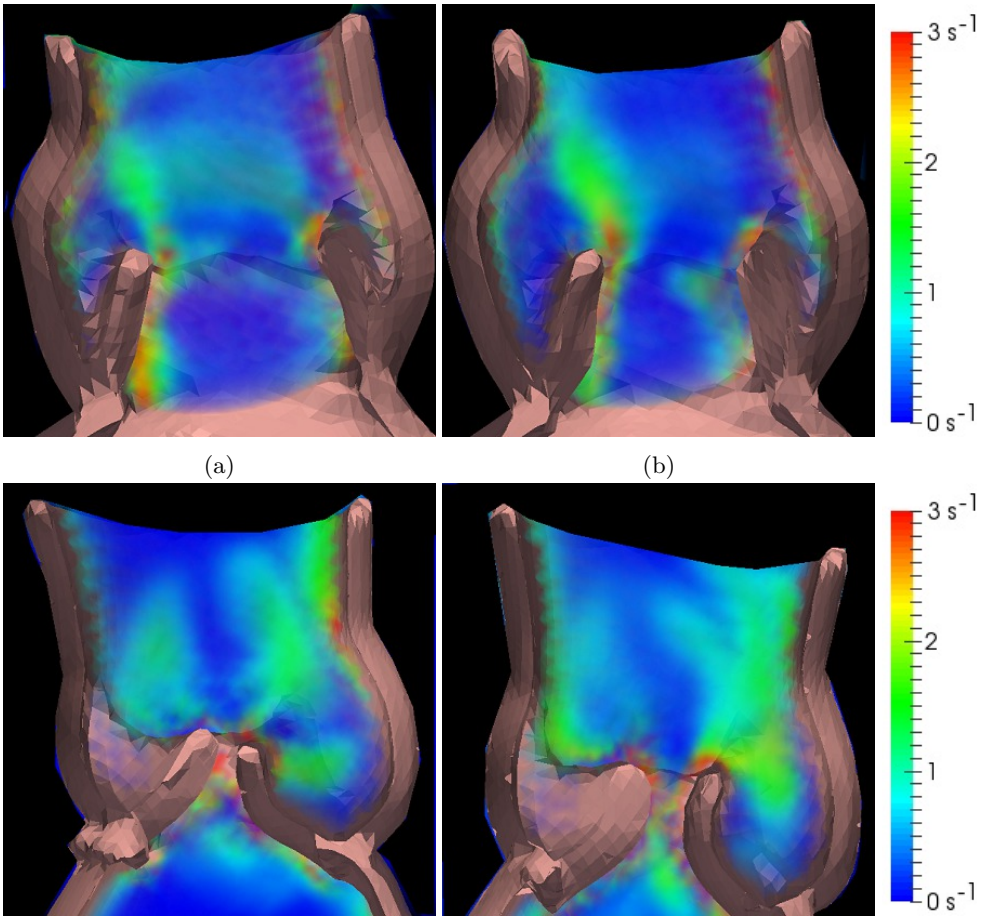
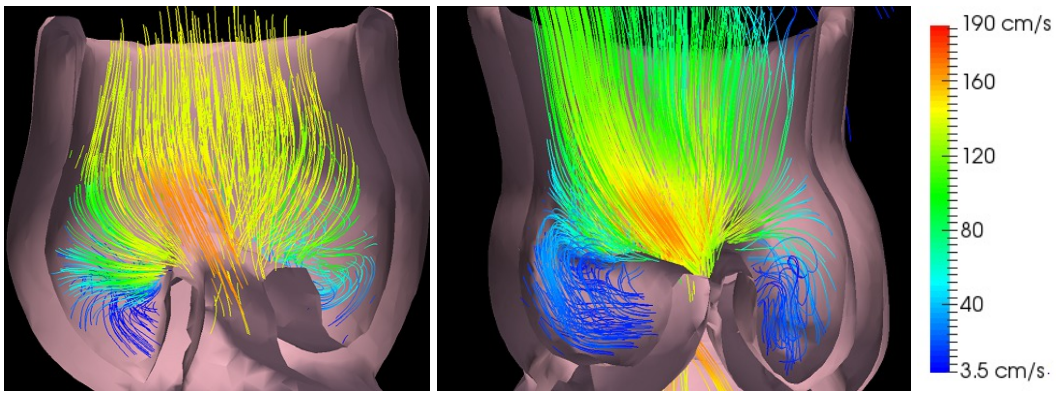
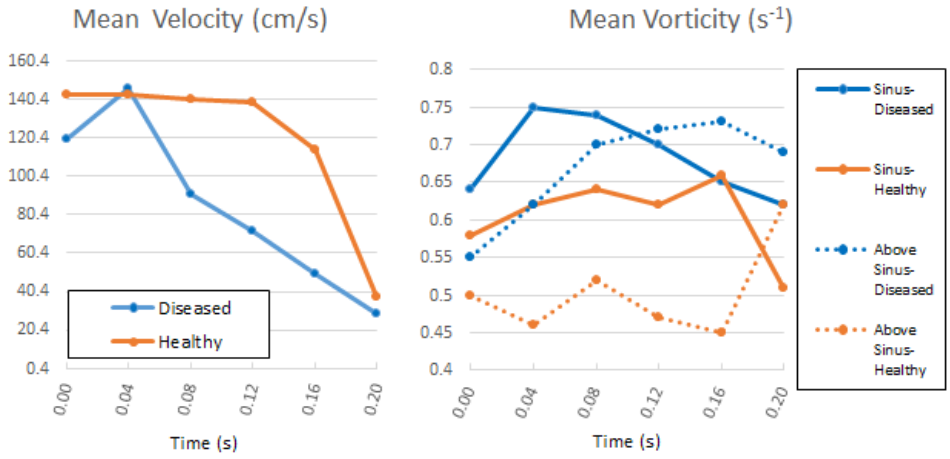


Figure 5.4: Vorticity field visualizations. (a) Early systole, healthy valve; (b) Late systole, healthy valve; (c) Early systole, diseased valve; (d) Late systole, diseased valve. Each image was taken at the same time step as their corresponding images in Figure 5.3



(a) (b)

Figure 5.5: Streamline visualizations of (a) healthy and (b) diseased aortic valve, late systole. Flow vortices are clearly visible.



(a) (b)

Figure 5.6: Analysis of blood flow velocity and vorticity. (a) Mean velocity at the sino-tubular junction (b) Mean vorticity in sinus and above-sinus regions

Chapter 6

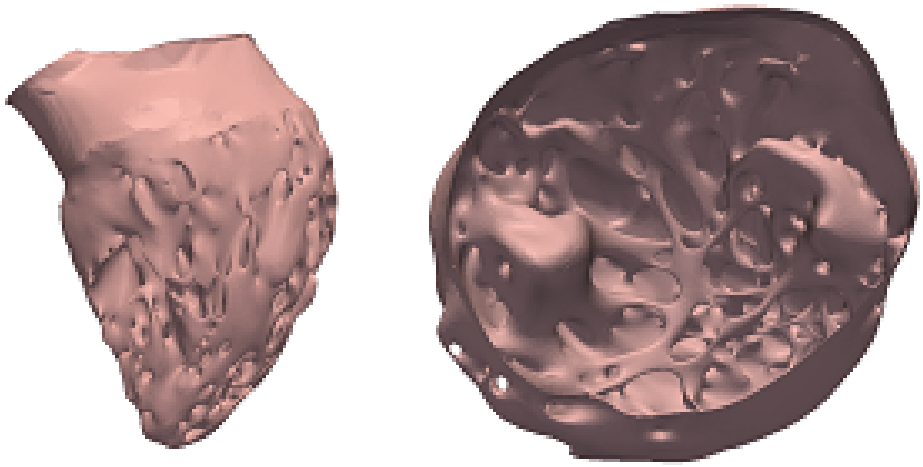
CUDA-Accelerated Particle-Based Blood Flow Simulations

As the geometry and motion of the heart wall models become more realistic and complex, the computation time for running these simulations becomes extremely high; as described in earlier chapter, single simulations can take nearly a full week to complete. In a clinical setting, these long waits for one result would be unacceptable. However, in recent years, there has been increasing interest in meshless methods, such as Smoothed Particle Hydrodynamics (SPH), due to their improved running times. However, while these algorithms are fast, working with complex boundary conditions is a notoriously difficult unsolved problem. While SPH has been successfully used for blood flow simulations before [84], these studies have focused entirely on the flow through blood vessels, which is far simpler than that of the heart. In this chapter, we present three major contributions: 1) A method to manage the highly complex boundary conditions of the heart using SPH by thickening the walls and treating the boundaries as particles, 2) A very fast and effective collision detection method optimized for a GPU implementation of SPH, and 3) Analysis of the SPH results that clearly show that these methods are practical and accurate enough in a clinical setting.

6.1 Data Acquisition

The CT images we used to generate the 3D heart mesh data were created on a Toshiba Aquilion ONE 320-MSCT scanner, which produces 10 images of the whole heart in a single cardiac cycle at a volumetric resolution of 0.3mm, and an in-plane resolution of 512x512.

For this application, we do not require such sophisticated methods to reconstruct



(a) (b)

Figure 6.1: Meshes reconstructed from CT data. (a) Outside heart (b) Apex.

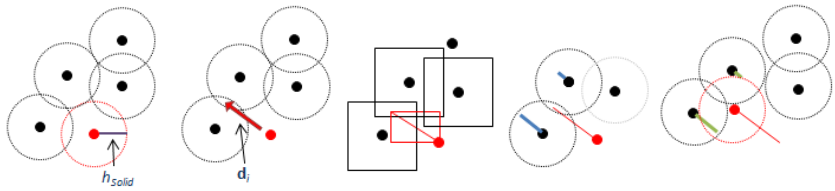
the heart model, since detail will be lost in a wall-thickening process, described in a later section. To build the mesh animation, we first apply a smoothing filter to the 3D image that corresponds to the beginning of diastole, and then extract a mesh through isosurfacing. Following manual cleanup with a 3D modeling tool, we transfer motion data, acquired from the same CT scan, to the mesh to generate a total of 10 frames. Since the valve motion is difficult to extract through CT imagery, we add models of the mitral and aortic valves generated from ultrasound data. Finally, we use cubic spline interpolation to create a final, smooth animation of 50 3D meshes, appropriate for use by the simulator. As can be seen in Figure 6.1, the reconstructed results are highly detailed, clearly showing the papillary muscles and trabeculae.

6.2 Simulation System

In this chapter, we use the SPH algorithm, described in Chapter 2.

6.2.1 Boundary Management

The enforcement of boundary conditions is one of the most challenging problems in SPH. To the best of our knowledge, no other group has attempted to adapt SPH to a



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

Figure 6.2: Collision detection on CUDA. (a) Initial State - Boundary (red) and fluid (black), sphere of radius h_{solid} surrounds each particle; (b) Boundary moves, new location too close to fluid; (*c*) Pass 1: Bounding box collision to detect danger pairs; (d) Pass 2: For all danger pairs, determine if line segment intersects sphere. If so, push fluid particles forward in the same direction as the boundary; (e) New positions: Boundary is no longer too close to fluid particles

problem of such complex geometry and movement as the left ventricle. As mentioned earlier, the most common techniques include either using fluid particles to model the solid boundaries, or using ghost particles. Ghost particles generally perform quite accurately, but they are not well-suited for problems in which the solid is thin and complex. In these difficult problems, fluid particles on each side of the thin surface will produce their own ghost particles, which the fluid particles on the other side will include in its list of neighbors during density/force computation, generating instability.

Most techniques that use fluid particles as boundaries either keep the boundary pressure constant, or raise it slightly to discourage particles from entering. However, we found that in our problem, these methods causes significant instabilities in the flow, due to the complex nature of the geometry and wall movement. We note that as the left ventricle expands during diastole, the pressure within the left ventricle drops, which allows fluid from the left atrium to enter. If the boundary particles at the walls maintain a constant pressure as the fluid particles within encounter an lower pressure, the fluid will unrealistically be repelled from the wall and cause instabilities. Similarly, we found that low pressure at the walls during systole will also become unstable. To overcome this problem, we used a technique that allows the boundary particles to naturally change in pressure with the rest of the fluid.

First, to generate the boundary particles, an implicit function computing the distances to the mesh is rasterized onto a 100^3 grid. Then, at each grid point where the

value of the implicit function is less than some distance ϵ, we label the particle generated at this position as a boundary. All other particles at a distance greater than ϵ are labeled as normal fluid particles. We then perform a search for the k closest mesh vertices, and the boundary particle's velocity is set to the inverse-distance weighted average of the velocities of its neighbors. Note that ϵ must be thick enough to prevent particles within the heart near the boundaries from including particles outside the heart during the neighborhood search. We found that setting $\epsilon \geq h/2$ and $k = 5$ produces the most stable results.

At each time step, each boundary particle's density and pressure is computed in the same manner as a normal fluid particle, but its velocity is forced to match its corresponding heart mesh vertices' velocities. As such, during diastole, the pressure at boundary particles drops as they move slightly farther apart, and the opposite occurs in systole. We found this method to be remarkably effective, and consistently produced stable and accurate results, as we discuss later.

6.3 CUDA Implementation

To further improve performance, we implemented and optimized our simulator using CUDA, allowing it to take advantage of highly-parallelizable GPUs. A framework for implementing SPH on CUDA, including the neighborhood search, density/pressure gradient computation, etc, is described in [85].

Collision detection is another open problem in SPH. Most methods focus on polygon-sphere collision, such as [86], who recently developed a method for continuous collision detection optimized for GPUs. Our problem requires sphere-sphere collision detection, so we devised a new GPU-optimized algorithm for this task. First, we set h_{solid} to be the minimum distance a fluid particle must be from a boundary particle (Figure 6.2 (a)). At the beginning of each time step, the boundary particles will advance forward in time. Let d_i be the line segment connecting particle i's starting and end positions (Figure 6.2 (b)). In the first pass, we make a list of all fluid-boundary particle pairs that are in danger of colliding by performing a bounding box test between d_i and

each neighbor (Figure 6.2 (c)). Each time a potential "danger pair" is encountered, the particle pair indices are saved in global RAM. When done, we have a full list of all potential collisions. We then execute a second CUDA kernel, where each thread performs a sphere-line segment collision test on a single pair (Figure 6.2 (d)). If a collision is detected, we know where on the line segment the intersection took place, and move the fluid particle in the direction of the boundary's motion such that the collision is resolved. The reverse procedure is done after the fluid particles move, to prevent them from moving through the boundary.

6.4 Results

As mentioned previously, the models used in this simulation were generated from CT imagery from a healthy patient's heart. The simulation was run three times, with different settings of c and Δt for each run. Each experiment was initialized with 100^3 particles evenly-spaced throughout the domain, and the smoothing radius h was set to 2.5x the initial distance between particles. In run 1, we set $\Delta t = 0{:}001$s, and $c = 10$m/s. In run 2, we set $\Delta t = 0{:}0005$s, and $c = 20$m/s. Finally, in run 3, $\Delta t = 0{:}00025$s, and $c = 30$m/s. As c increases, the fluid becomes less compressible, and so we expect accuracy to improve. All simulations were performed on an Nvidia Geforce GTX 590. The running time of the simulations scaled linearly as Δt dropped. The total computation time for run 1 was 30 minutes, the time for run 2 was 62 minutes, and the time for run 3 was 126 minutes. All of these running times are orders of magnitude better than those described in other methods. Each time step took 2.5-3 seconds to complete. The force computation was the most expensive step, taking an average of 1.5 seconds per iteration. The density computation took, on average, 0.5 seconds each per time step. The rest of the time in each iteration was spread across the remaining CUDA kernels, including the collisions; compared to the density/force/velocity correction functions, the others' individual running times were negligible.

Visualizations of the blood flow can be seen in Figure 3. Columns 1, 2, and 3 correspond to Runs 1, 2, and 3, respectively. Frames in row 1 were taken during mid-diastole, and frames in row 2 were taking during mid-systole. All frames in a row were

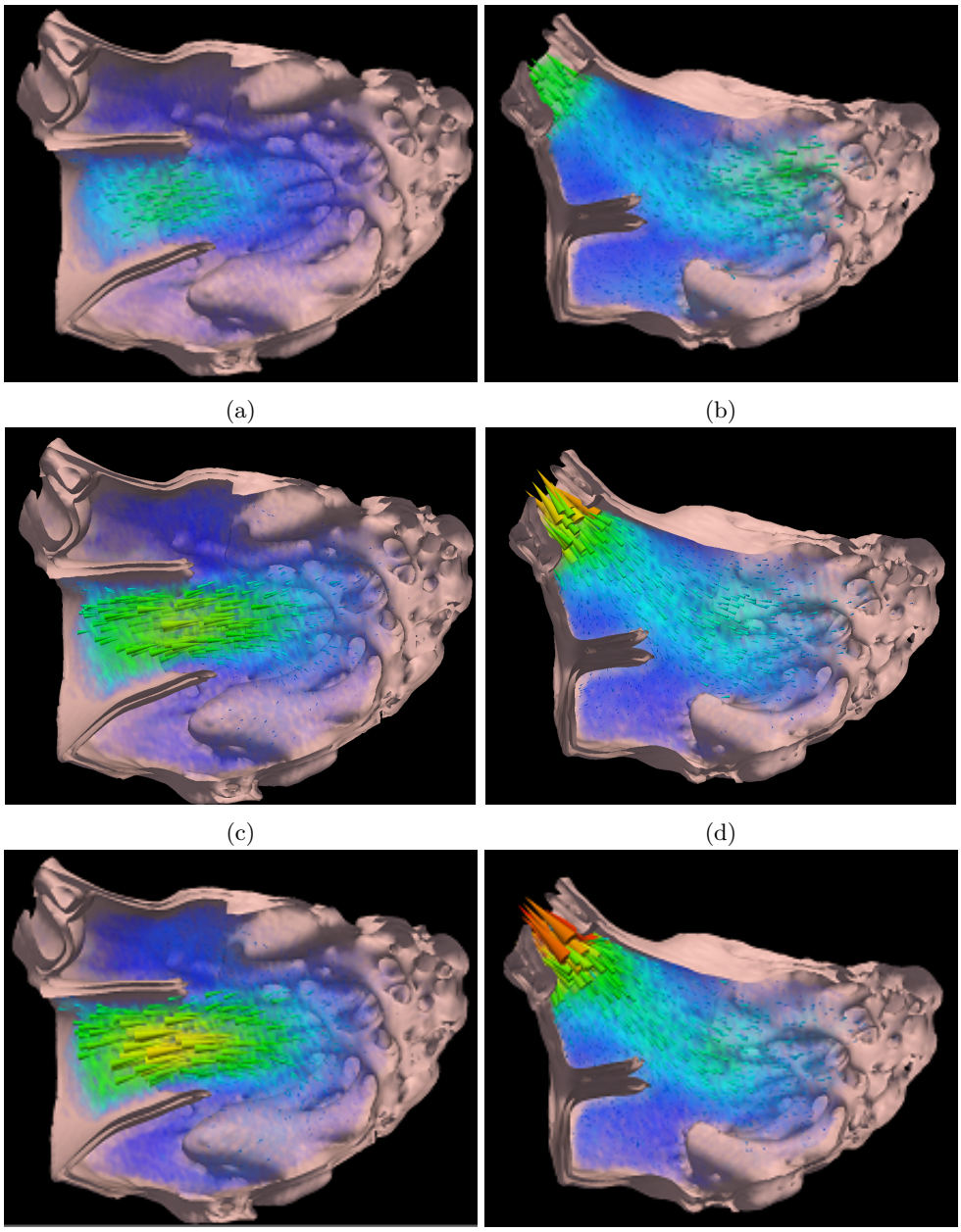


Figure 6.3: Velocity fields for simulations with different Δt. Left Column: Diastole, Right Column: Systole, Row 1: Δt=0.001s, Row 2: Δt=0.0005s, Row 3: Δt=0.00025s.

taken at equivalent time steps. The direction of the flow is seen in the direction of the embedded cones, and the velocity magnitude is shown by color, where blue regions represent velocities approaching zero, and red regions represent velocities approaching 1 m/s. We can see that as Δt decreases and compressibility goes down, the computed velocities within the heart go up and approach more accurate values. We note, however, that by thickening the walls, interactions between blood flow and trabeculae are not clearly visible.

Validation of cardiac blood flow simulations is difficult. In the future, we plan to acquire both CT and MRI images of the patient's heart, and use the MRI flow data to compare and validate. Here, we compute the ejection fraction by counting the number of particles within the heart at the end of systole and the end of diastole. We found that for run 1, the ejection fraction was about 0.42, for run 2, the ejection fraction was about 0.48, and for run 3, the ejection fraction was 0.50. Again, the increase in accuracy as Δt is decreased is expected, and gives the doctor a scalable option.

6.5 Chapter Conclusion

In this chapter, we have described an adaptation of SPH to simulate blood flow through the left ventricle quickly and accurately. By simply scaling Δt and c, doctors can choose an appropriate level of accuracy, while maintaining faster speeds than previous methods allowed. To the best of our knowledge, this is the fastest that patient-specific cardiac blood flow simulations has been solved.

Chapter 7

Thin-Wall Smoothed Particle Hydrodynamics

In the previous chapter, we have shown that Smoothed Particle Hydrodynamics can provide clinically useful fluid simulations at a fraction of the computational cost of full Navier-Stokes solvers. However, in that approach, we had to thicken the walls considerably to prevent particles on either side of the boundary from interacting with each other. In addition to not allowing for flows through small structures, which become filled in after wall thickening, this method could cause other undesirable inaccuracies. For example, by thickening the walls of the heart, we are also decreasing the volume of fluid within the chamber, and shrinking the openings in the valves. In an effort to improve these results, this chapter describes a new framework that allows for SPH simulations against arbitrarily thin walls.

7.1 Correcting Densities

The thin-wall problem can be seen in Figure 7.1. In this image, the boundary is flat for illustrative purposes, but there is no flat surface requirement. Let p_i denote the particle of interest, near the boundary. Let $+(S_i)$ be the side of the domain containing p_i, and $-(S_i)$ be the side of the domain opposite of p. Additionally, $\rho^{+(S_i)}$ is the density on side $+(S_i)$, while $\rho^{-(S_i)}$ is the density on side $-(S_i)$, and in this example, $\rho^{-(S_i)} > \rho^{-(S_i)}$. Therefore, since p_i is very close to the boundary (distance less than h), particles on side $-(S)$ will incorrectly exert their higher-density influence on p_i, causing the density computed at p_i to be significantly higher than it truly should be. We seek to compute correct this error.

We note that before any correction, the density computed at any near-boundary particle p_i is approximately equal to the following (in integral form):

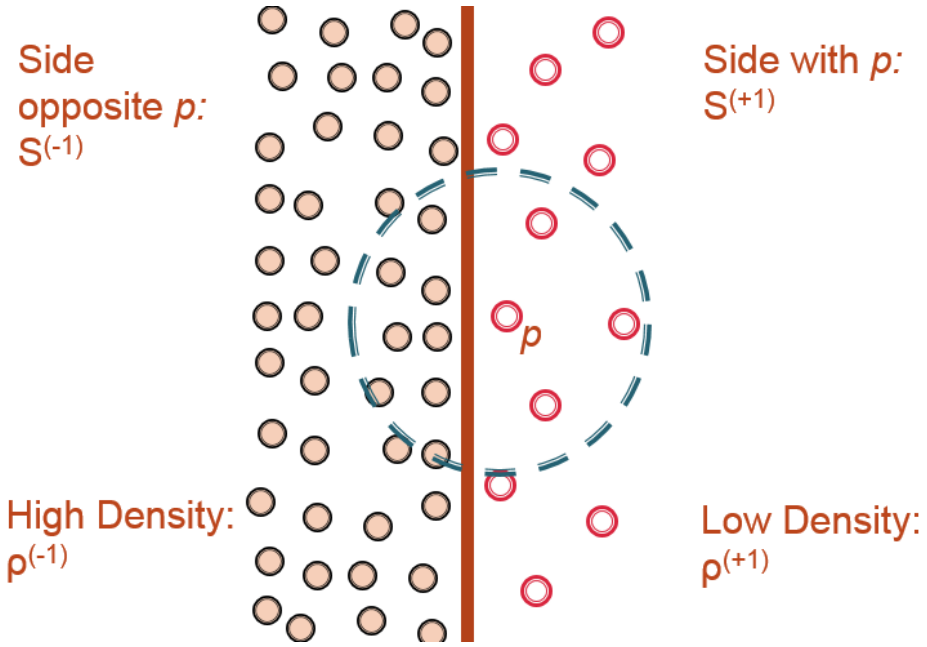


Figure 7.1: Image of the thin-wall SPH problem. Note that in the density computation step, particle p near the boundary will interact with particles on the opposite side of the wall.

$$\rho_i \approx \iiint\limits_{\Omega^{+(S_i)}} \rho^{+(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega + \iiint\limits_{\Omega^{-(S_i)}} \rho^{-(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega, \tag{7.1}$$

where Ω is the entire domain of interest, $\Omega^{+(S_i)}$ is the domain on side $+(S_i)$, and $\Omega^{-(S_i)}$ is the domain on side $-(S_i)$. Thus, the term $\iiint\limits_{\Omega^{+(S_i)}} \rho^{+(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega$ approximates the density influence side $-(S_i)$ has on particle p_i. It follows, then, that the error e_i can be found as follows:

$$e_i = \iiint\limits_{\Omega^{-(S_i)}} \rho^{-(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega - \iiint\limits_{\Omega^{-(S_i)}} \rho^{+(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega. \tag{7.2}$$

To perform the error correction, we need to compute both terms in this equation.

Now, let b_i be the point on the boundary very close to particle p_i. Taking a first order Taylor series about b_p tangent to the wall, and zeroth order normal to the wall, we find

$$\iiint_{\Omega^{-(S_i)}} \rho^{-(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega = \iiint_{\Omega^{-(S_i)}} \rho^{-(S_i)}(b_p) W(\mathbf{r}, h) \mathrm{d}\Omega + \iiint_{\Omega^{(-S_i)}} (\nabla \rho^{-(S)}(b_p) \cdot \Delta \mathbf{x}_t) W(\mathbf{r}, h) \mathrm{d}\Omega, \tag{7.3}$$

where $\rho^{-(S_i)}(b_p)$ is the side $-(S)$ density at point b_i, and $\Delta \mathbf{x}_t$ is the displacement from point b_i tangent to the wall.

Note that relative to point b_i, $(\nabla \rho^{-(S)}(b_p) \cdot \Delta \mathbf{x}_t)$ is odd, and $W(\mathbf{r}, h)$ is even, and so the product of these terms is odd. The integral of an odd function over a symmetric interval is zero, and so the entire integral term vanishes, leaving

$$\iiint_{\Omega^{-(S_i)}} \rho^{-(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega = \iiint_{\Omega^{-(S_i)}} \rho^{-(S_i)}(b_p) W(\mathbf{r}, h) \mathrm{d}\Omega. \tag{7.4}$$

Using the same technique, we can find

$$\iiint_{\Omega^{+(S_i)}} \rho^{+(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega = \iiint_{\Omega^{+(S_i)}} \rho^{+(S_i)}(b_p) W(\mathbf{r}, h) \mathrm{d}\Omega. \tag{7.5}$$

Since $\rho^{-(S_i)}(b_p)$ and $\rho^{+(S_i)}(b_p)$ are both constant, we see the error term can be computed as follows:

$$e_i = (\rho^{-(S_i)}(b_p) - \rho^{+(S_i)}(b_p)) \iiint_{\Omega^{+(S_i)}} W(\mathbf{r}, h) \mathrm{d} \tag{7.6}$$

If h is constant, the term $\iiint_{\Omega^{+(S_i)}} W(\mathbf{r}, h) \mathrm{d}$ can easily be precomputed for many values of $\mathbf{r}$, stored in a lookup table, and interpolated at runtime for very good results. So, if we can find $\rho^{-(S_i)}(b_p)$ and $\rho^{+(S_i)}(b_p)$, we can therefore compute an approximation to this error term and correct the density at particle p_i.

Since all near-boundary particles will require this correction, we will seed the entire boundary with a high number of boundary particles. To find the density for either side (S) at any boundary point b_k, $\rho_k^{(}S)$, we can use the standard discrete SPH formulation:

$$\rho_k^{(S)} = \sum_{j \in (S)}^{N} m_j W^{boundary}(\mathbf{r}_{jk}, h). \tag{7.7}$$

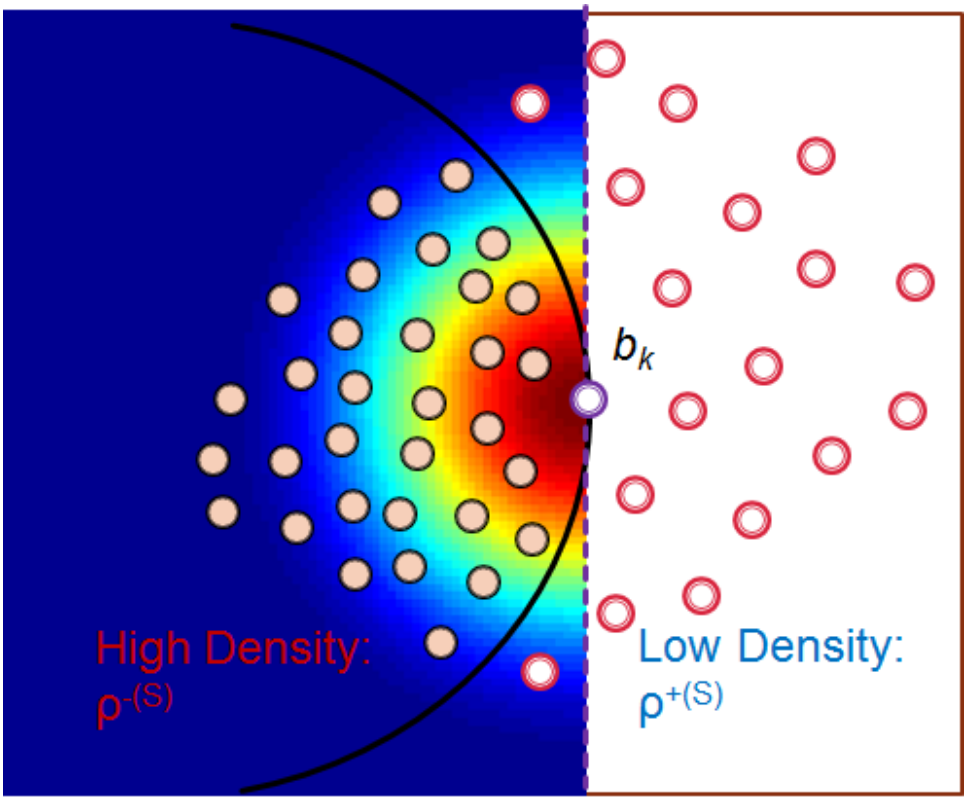


Figure 7.2: Curved wall problem: particles on low density side could "sneak" into the density computation for the opposite side.

This, however, requires some choice of weighting function. Since each region, $+(S)$ and $-(S)$, is one hemisphere on either side of b_k, it is perhaps intuitive to use

$$W^{boundary}(\mathbf{r}_{jk}, h) = 2W^{(*)}(\mathbf{r}_{jk}, h), \tag{7.8}$$

Where $W^{(*)}$ is any commonly-used weighting function described in previous chapters. This choice could work fine for flat surfaces, but has a severe problem in curved surfaces. For illustration, see Figure 7.2. Here, the solid line is the boundary, the dashed line is tangent to the boundary, and the colored region represent the relative values of $W^{(*)}(\mathbf{r}_{jk}, h)$ given this scheme.

We can see that the problem is that particles from side $+(S)$ can, in fact, be on the wrong side of the tangent wall and be counted in the density computation for $\rho^{-(S_i)}(b_p)$. If particles are very close to b_k, this problem can cause dramatic instabilities, since small

changes in particle positions close to the tangent wall - that is, moving between either side of the tangent wall - can cause huge changes computed density at b_k. Instead, we would rather find a weighting function such that small changes in particle position causes small changes in $\rho^{-(S_i)}(b_p)$. That is, the weighting function and its derivative should be equal to zero at the tangent wall. Consider the following alternative equation:

$$W^{boundary}(\mathbf{r}, h) = 2 \cdot W^* \left(\zeta(\mathbf{r}, h, \mathbf{n}, \mathbf{t_1}, \mathbf{t_2}), h\right), \tag{7.9}$$

where

$$\zeta(\mathbf{r}, h, \mathbf{n}, \mathbf{t_1}, \mathbf{t_2}) = \| \left(\mathbf{r} \cdot \mathbf{n} + \frac{h}{2}\right)^2 + (\mathbf{r} \cdot \mathbf{t_1})^2 + (\mathbf{r} \cdot \mathbf{t_2})^2 \| \tag{7.10}$$

This weighting function produces the image seen in Figure 7.3. This function is clearly much more stable at the boundary. However, it does have the disadvantage that the region of highest influence is some distance away from the wall. Empirically, we have found this weighting function to perform far better than the previous.

With this, we can now compute $\rho_k^{-(S)}$ and $\rho_k^{+(S)}$ for any boundary point b_k. As mentioned earlier, in our implementation, we seed many boundary points across the surface. We then either assign each fluid particle p_i to its nearest boundary particle b_i, or apply an inverse-distance weighting scheme to interpolate the boundary values of ρ between the closest boundary points. If the density of boundary points is high enough, both methods are nearly equivalent. In practice, we have found the most stable results using the latter (interpolated) method to determine the final ρ boundary values corresponding to each fluid particle.

7.2 Correcting Pressure Gradient

Correcting the pressure gradients at particles near the boundary takes a very similar approach. Using SPH integral form, we know the following:

$$\nabla P_i \approx \iiint_{\Omega^{+(S_i)}} \nabla P^{+(S_i)} W(\mathbf{r}, h) \mathrm{d}\Omega + \iiint_{\Omega^{-(S_i)}} \nabla P^{-(S_i)} \nabla W(\mathbf{r}, h) \mathrm{d}\Omega \tag{7.11}$$

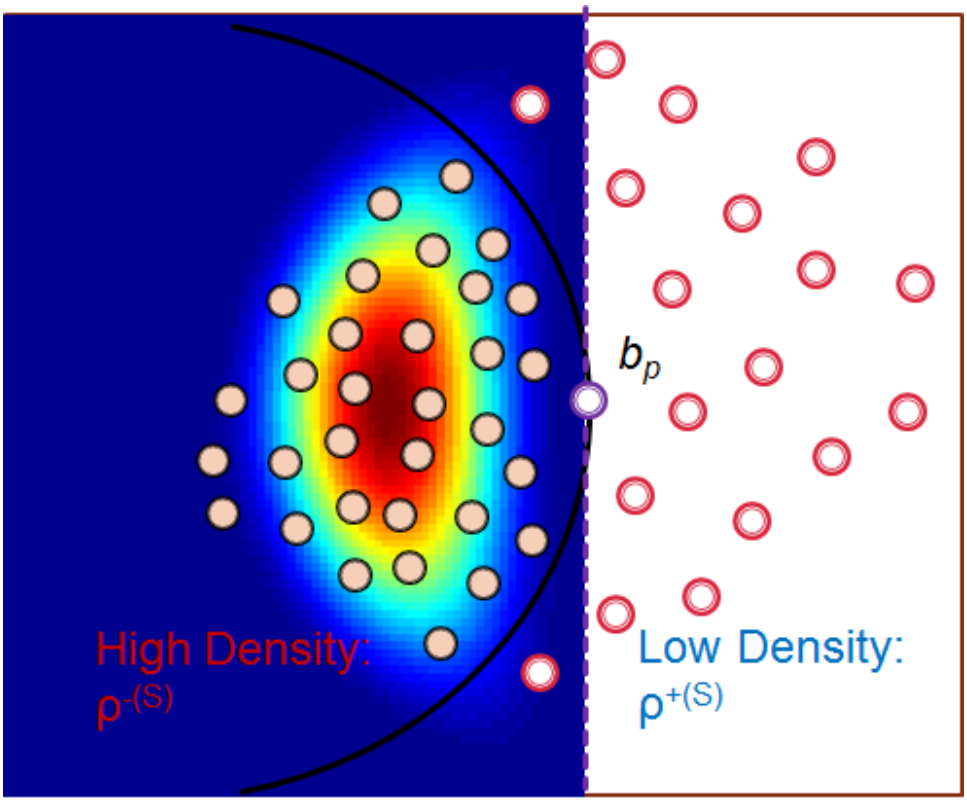


Figure 7.3: New weighting function stable at tangent wall

At the moment, we are concerned only with the pressure gradient in the tangential direction(s). Given $\mathbf{t}$ is in a direction tangent to boundary at boundary point b_i (near particle p_i), we can reformulate this equation as

$$\frac{\partial P_i}{\partial \mathbf{t}} \approx \iiint\limits_{\Omega^{(+S_i)}} \frac{\partial P_i}{\partial \mathbf{t}} W(\mathbf{r}, h) \mathrm{d}\Omega + \iiint\limits_{\Omega^{(-S_i)}} \frac{\partial P_i}{\partial \mathbf{t}} \nabla W(\mathbf{r}, h) \mathrm{d}\Omega \tag{7.12}$$

As mentioned earlier, we are taking a first-order approximation of density, and so the density gradient (and therefore also the pressure gradient) is constant in the tangential directions. Using a similar technique as in the density correction step, we can find that the pressure gradient error e_i in the direction of $\mathbf{t}$ is

$$e_i = \left(\frac{\partial P^{-(S_i)}}{\partial \mathbf{t}} - \frac{\partial P^{+(S_i)}}{\partial \mathbf{t}_n}\right) \iiint\limits_{\Omega^{(-S_i)}} W(\mathbf{r}, h) \mathrm{d}\Omega \tag{7.13}$$

We then only need to find $\frac{\partial P^{+(S_i)}}{\partial \mathbf{t}_n}$ and $\frac{\partial P^{-(S_i)}}{\partial \mathbf{t}_n}$ at the boundary point b_i to compute the error.

We seek these values for any boundary point b_k. We can use discrete gradient SPH formulation to derive the following:

$$\frac{\partial P_k^{(S)}}{\partial \mathbf{t}} = -\sum_{j \in (S_k)}^{N} \frac{m_j}{\rho_j} \frac{\partial W^{boundary}}{\partial \mathbf{t}} \left(\zeta(\mathbf{r}, h, \mathbf{n}, \mathbf{t_1}, \mathbf{t_2}), h\right) \tag{7.14}$$

This equation provides us with $\frac{\partial P^{+(S_i)}}{\partial \mathbf{t}_n}$ and $\frac{\partial P^{-(S_i)}}{\partial \mathbf{t}_n}$, and so we can use this to compute and correct the error e_i.

7.3 Correcting Velocity

The formulation in the preceding sections will provide for us corrections for the fluid pressure gradient in the direction tangent to the surface. However, we have empirically found that it is impossible to use such techniques to accurately compute the normal pressure gradient correction, since there is much less information available to the boundary particle in the normal direction.

We note, however that in the normal direction, fluid motion is predominately governed by the no-penetration condition:

$$(u - u_b) \cdot \mathbf{n} = 0, \tag{7.15}$$

where u is the fluid velocity, u_b is the velocity of the solid boundary, and $\mathbf{n}$ is the unit vector normal to the wall. Therefore, at particles near the wall, we simply set the fluid particle's final velocity $u^{(f)}$ to

$$u^{(f)} = u - (u - u_b) \cdot \mathbf{n} \tag{7.16}$$

This way, computing the corrected pressure gradient normal to the wall is completely unnecessary, since it is completely driven by the wall motion. Meanwhile, forces and motion tangential to the wall is still being accurately corrected.

7.4 Experiments

To test the accuracy of our Thin-Wall SPH method, we have devised two experiments.

The first is a static (single-frame) experiment where are testing the accuracy of the computed density fields and density gradient fields on a curved surface, both on the boundary surface and the corrected solutions projected onto the fluid. Since the pressure gradient depends on the choice of C (the speed of sound), and pressure gradient is computed as a scalar multiple of the density gradient, it allows more a more intuitive comparison to test the density gradient, rather than the pressure gradient.

Here, our boundary is a sphere. Outside the sphere, the fluid has constant density (1060) and zero density gradient. Within the sphere, we assign particle masses such that

$$\rho(\theta, \phi) = 1060 - 8\theta - 8\phi \tag{7.17}$$

The density gradient with respect to θ and ϕ can be easily computed from this formula. We then run the Thin-Wall SPH algorithm for a single frame, and compare to the true values.

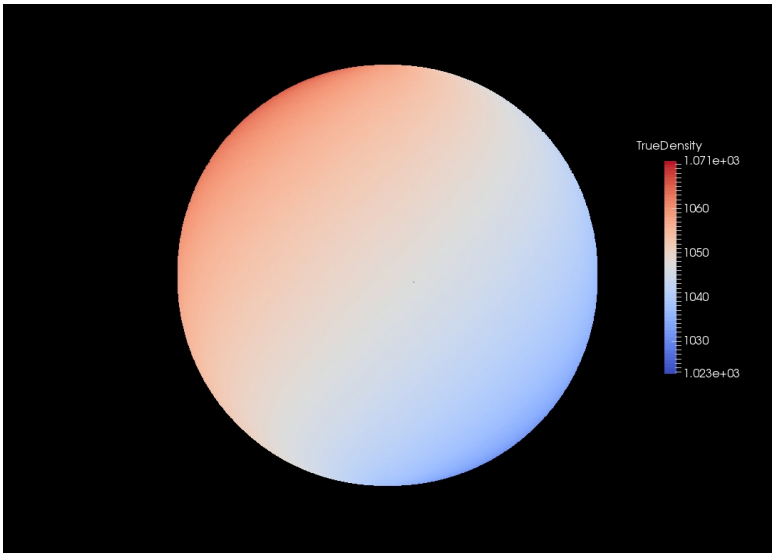


Figure 7.4: Experiment 1: True inner density at boundary

The second experiment tests this algorithm's ability to handle rapid deformation and motion. Similar to the first experiment, we start with fluid with an embedded sphere boundary. The fluid's density starts constant (1000) everywhere in the domain. The sphere then shrinks to half its original radius in one-half of a second. Since the sphere's final volume is one-eight that of its starting volume, we know that the final density of the fluid within the sphere should be 8 times higher than its starting density.

7.5 Results

The results of the first experiment can be seen in Figures 7.4-7.13. In Figure 7.4, we have the true inner density at the boundary, computed from Equation 7.17. In Figure 7.5, we have the computed density at the boundary. To the eye, these two images are nearly indistinguishable. Similarly, we see Figure 7.6, which shows the computed density in the fluid particles immediately adjacent to the boundary surface. Note that the distribution of fluid particles is more sparse than the distribution of boundary particles. Again, though, we see that visually, the computed density in the fluid particles looks nearly identical to the true density field.

In Figure 7.7, we have the percent error of the computed density on the inner

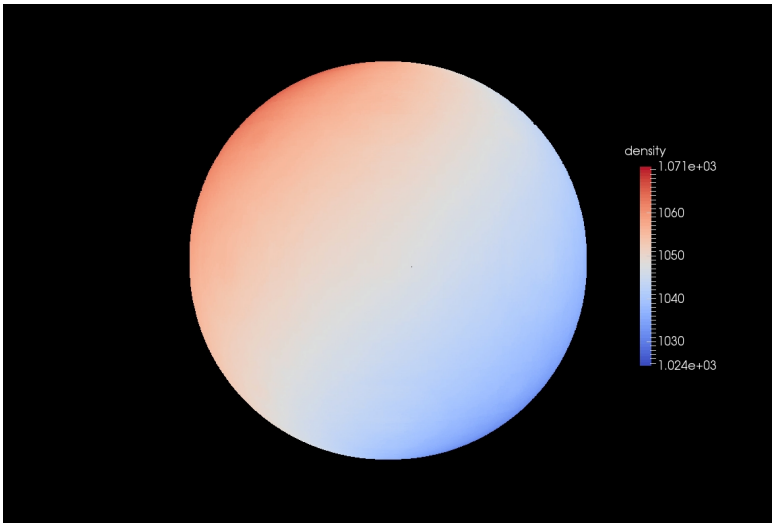


Figure 7.5: Experiment 1: Computed inner density at boundary

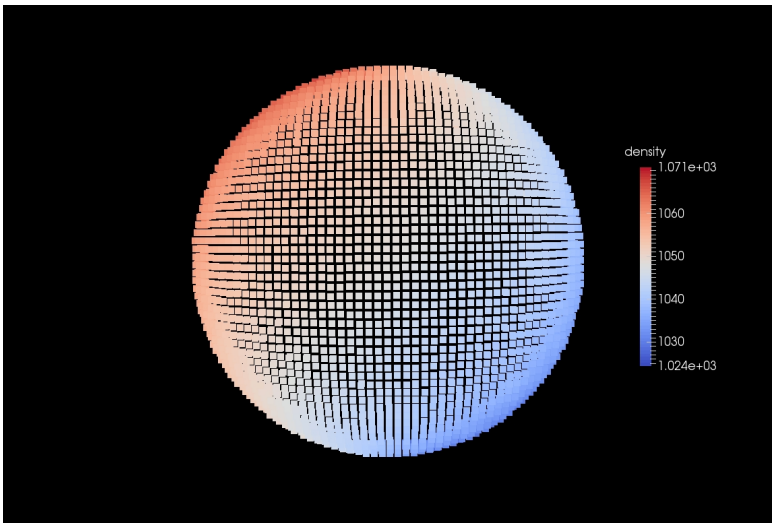


Figure 7.6: Experiment 1: Computed inner density at fluid

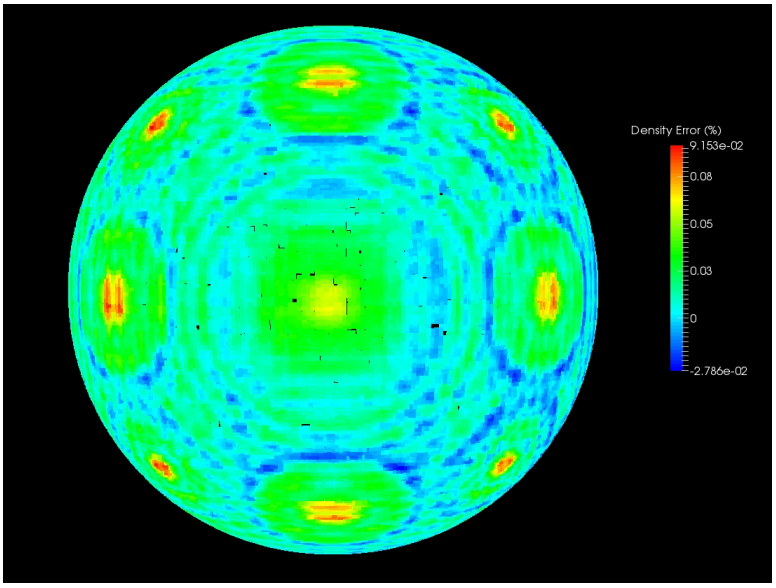


Figure 7.7: Experiment 1: Boundary Inner Density Percent Error $\leq 0.092\%$

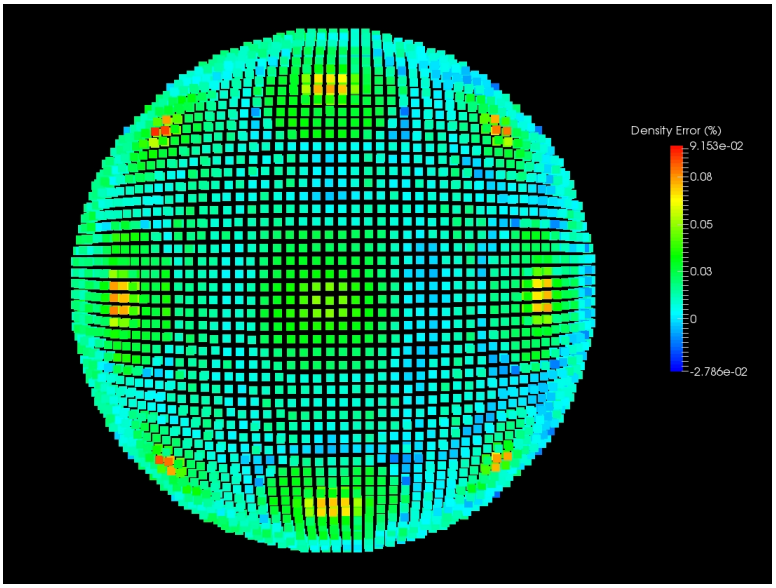


Figure 7.8: Experiment 1: Fluid Inner Density Percent Error $\leq 0.092\%$

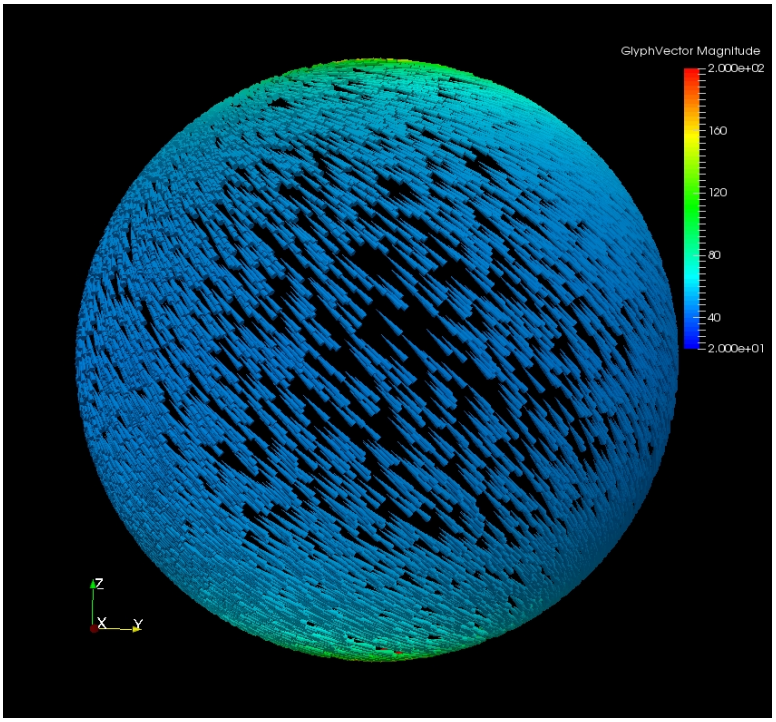


Figure 7.9: Experiment 1: True Inner Density Gradient at Boundary

boundary. The errors range between -0.028% and 0.092%, which is better than we had even expected. We note that the regions of highest error are in very small, localized regions across the sphere. The error variance is due to the changing fluid particle distributions in in the neighborhood of the boundary across the surface. In Figure 7.8, we have the percent error of the computed density of the fluid inside the sphere. Using the same color scale as in the Figure 7.7, we can see that the errors are actually noticeably less than those at the boundary. At these small errors, the computed density would be at most ± 1 kg/m^3 from the truth, which is easily on-par with errors from computing density within the fluid, without any solid boundary. Therefore, we feel that the density correction step of the thin-wall SPH algorithm works exceedingly well.

In Figure 7.9, we see the true density gradient at the boundary, represented by cone glyphs. The color of the glyphs corresponds to their relative magnitude. Note that we are using density gradient, rather than pressure gradient here. This is because, in SPH, pressure and pressure gradient can change somewhat arbitrarily depending on the

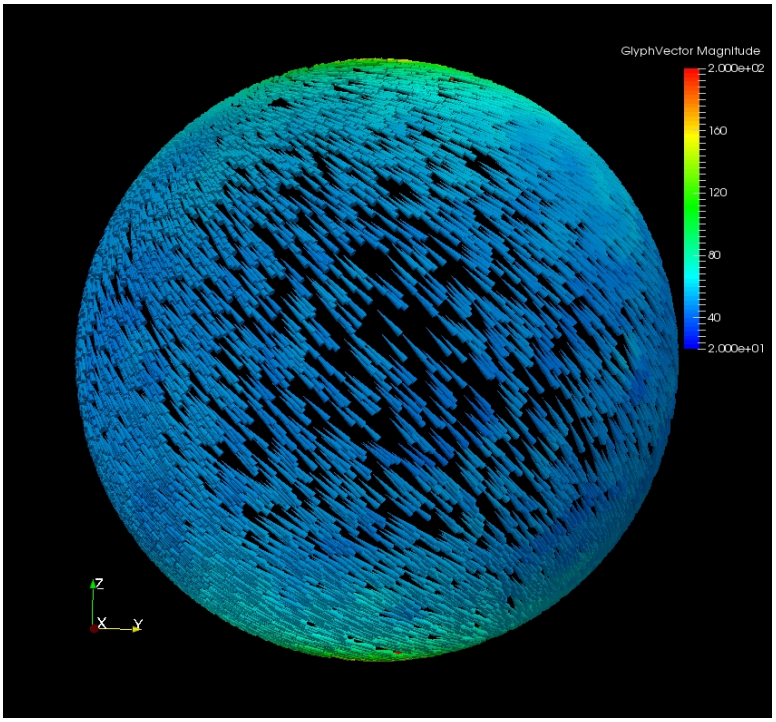


Figure 7.10: Experiment 1: Computed Inner Density Gradient at Boundary

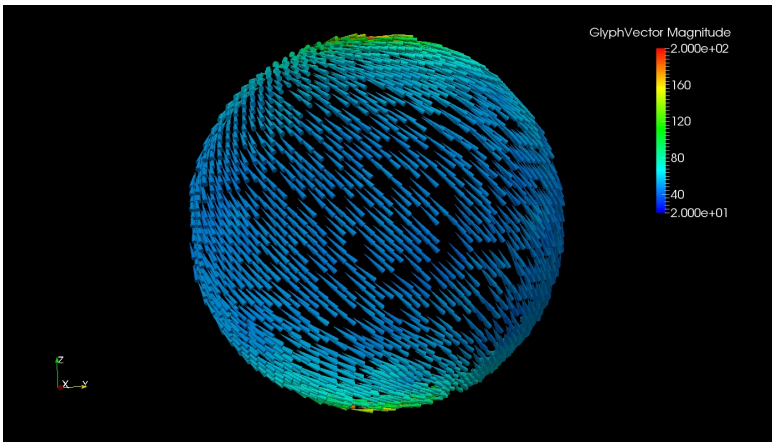


Figure 7.11: Experiment 1: Computed Inner Density Gradient at Fluid

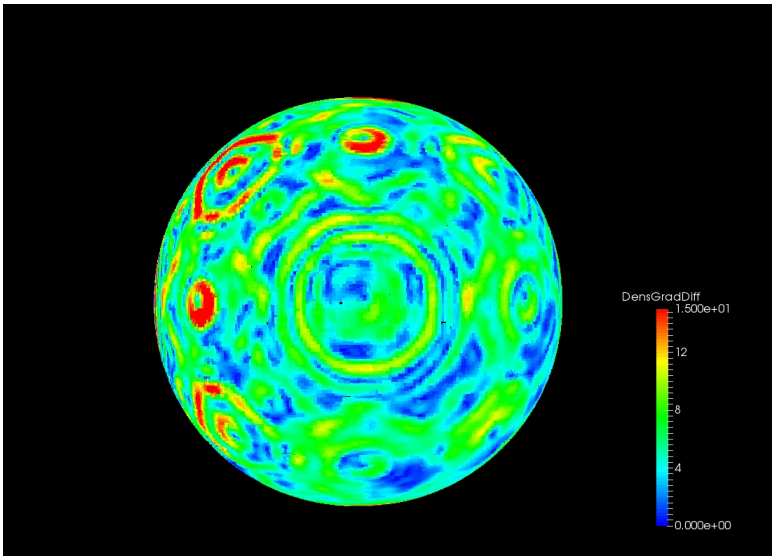


Figure 7.12: Experiment 1: Inner Density Gradient Percent Error at Boundary

choice of the speed of sound, c, and so these values are less interesting than the density gradient.

In Figures 7.10 and 7.11, we have the computed density gradient fields at the boundary and at the fluid, respectively. We note that in both cases, the density gradient fields are nearly identical to the true values in terms of their directions. However, in both cases, the magnitudes of the gradients are slightly higher here, than in Figure 7.9.

This difference is more clear in Figures 7.12 and 7.13, which show the percent error of the density gradient magnitude at the boundary and fluid, respectively. In the case of the boundary density gradient, we see that error generally is less than 10%, with some small regions where error is above 15%. The errors at the fluid are, again, considerably less, in most regions less than 8%, with some very small areas with 15% error. Again, though, from Figures 7.10 and 7.11, we know that the gradient directions are correct, so these small magnitude errors are quite acceptable, and so we believe the pressure/density gradient correction step of the thin-wall SPH algorithm also works successfully.

We now bring our attention to the second experiment, of the shrinking sphere. Again, this experiment tests the effectiveness and stability of the thin-wall SPH system

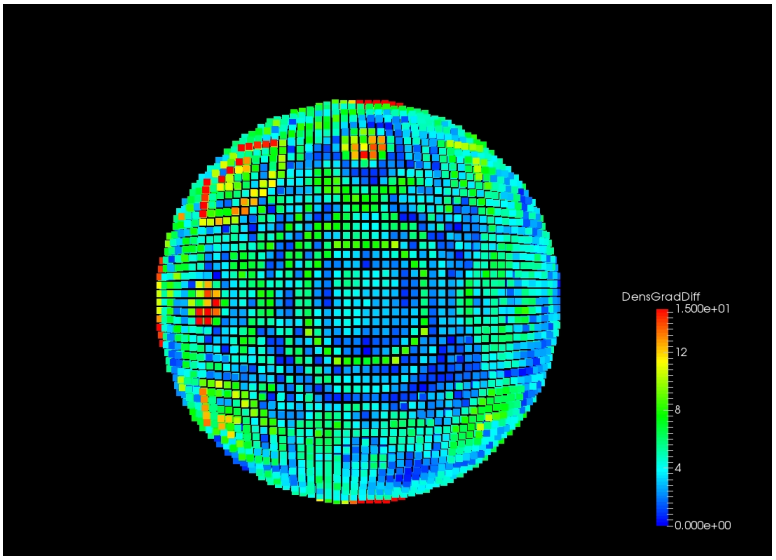


Figure 7.13: Experiment 1: Inner Density Gradient Percent Error at Fluid

with fast, deforming geometry. Figures 7.14 and 7.15 show the computed density of the fluid particles within the sphere before and after the shrink, respectively. We see that the density of the fluid started at 1 kg/m^3. As mentioned earlier, this experiment was designed such that the sphere ends with half the radius, and thus one-eighth the volume. As we can see from Figure 7.15, the simulation remains stable during the entire deformation, and the fluid density increases to 8 kg/m^3, as expected.

Finally, in Figures 7.16 and 7.17, we have the percent error of the density at the boundary and fluid, respectively. We see that errors are slightly greater than in Experiment 1, between 0.52% and -0.64% at the boundary. In the fluid, again, errors are smaller, at around $\pm$ 0.3%. The slightly increased error is certainly due to the high velocities experienced by the fluid particles. However, at over 99.5% accuracy and high stability, we feel that this experiment shows that the thin-wall SPH algorithm works exceptionally well, even in these strenuous tests.

7.6 Chapter Conclusion

In this chapter, we have described a new method to solve Smoothed Particle Hydrodynamics with an arbitrarily thin, curved, and moving solid wall boundary. We have

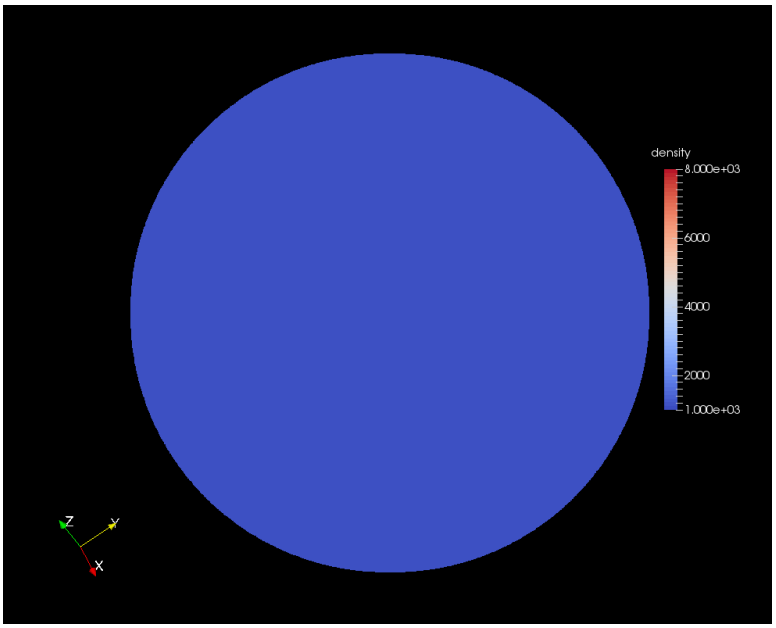


Figure 7.14: Experiment 2: Density before sphere shrinks

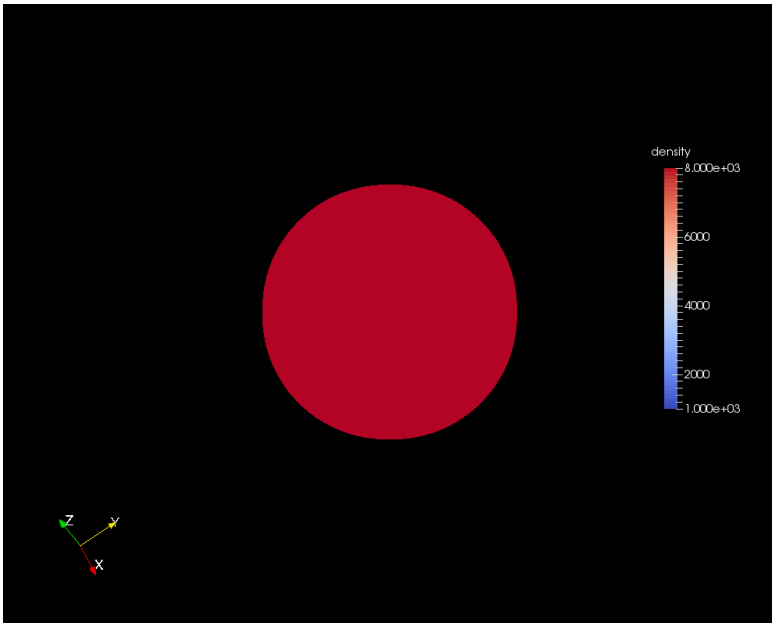


Figure 7.15: Experiment 2: Density after sphere shrinks

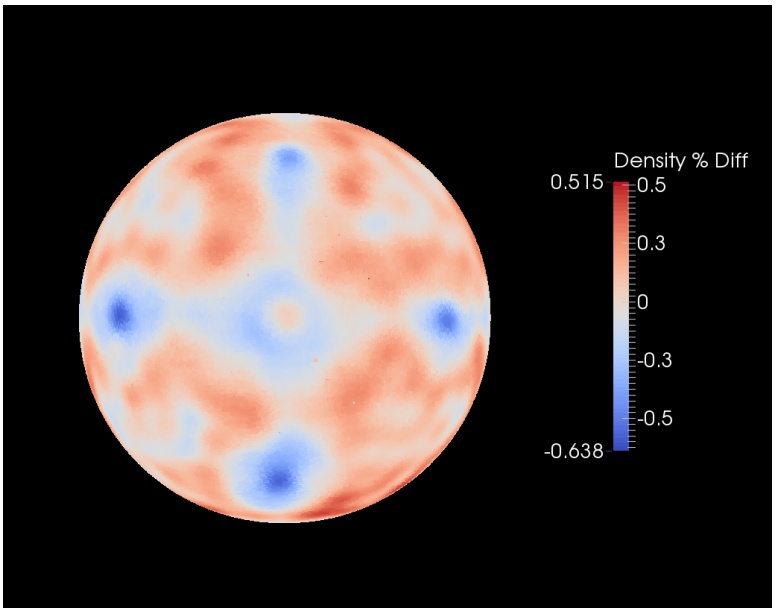


Figure 7.16: Experiment 2: Boundary density percent error after sphere shrinks

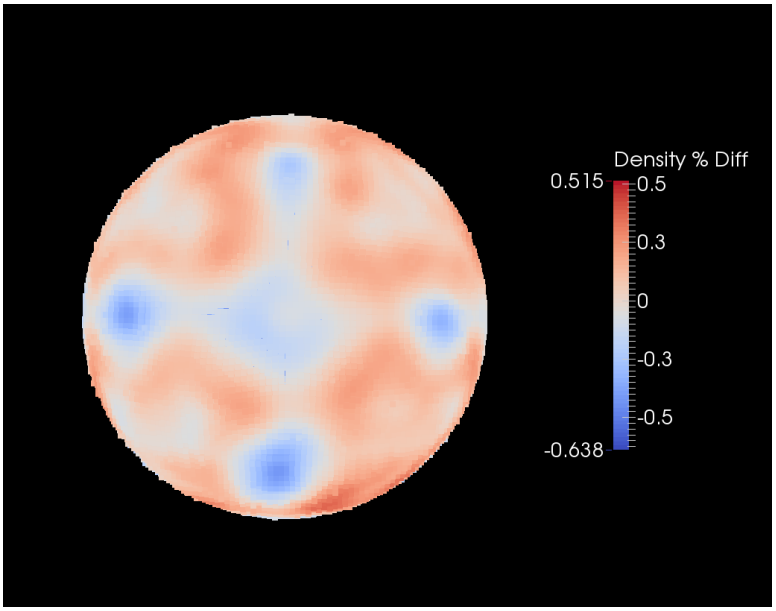


Figure 7.17: Experiment 2: Fluid density percent error after sphere shrinks

shown that this method works on two difficult test cases, including a spatially changing density field across a static spherical surface, as well as in a quickly deforming and shrinking sphere. While this method currently has limitations to clinical problems, described in more detail in the next chapter, we believe that this algorithm is an important step in solving solid wall treatment in SPH.

Chapter 8

Conclusions

In this thesis, we have described a number of difficult problems related to ventricular blood simulation and analysis. We conclude this dissertation by first providing a short summary of all the work presented here, followed by answers to the questions posed in Chapter 1. We will then describe my specific contributions to the research work performed here. Finally, we discuss limitations and future work of all methods and results presented in this dissertation.

8.1 Summary of Work

In this section, we provide a very short summary of each of the preceding chapters, in order to briefly review the methods used and results found.

8.1.1 Chapter 2 Summary

In Chapter 2, we developed ways to perform flow modeling and visualizations through the left ventricle, reconstructed from CT images, that could be used to help doctors differentiate flow between normal and diseased hearts.

For flow simulation, we use the Immersed-Boundary Navier-Stokes simulation method described in Chapter 1. With the fluid velocity fields and level sets generated for each time step, we developed visualizations of the flow fields in both a healthy heart and a simulated diseased heart suffering from a large perfusion defect area created by reducing motion in the anterior apical area. These visualizations proved to effectively and clearly differentiate between the normal and abnormal models.

We also performed a quantitative comparison of the velocities around the apex of both heart models. We found that, on average, the velocities in the diseased heart

in this region are about half of those in the healthy heart. In particular, we see that during systole, the blood velocities at the apical region of the diseased heart tend not to increase as they do in the healthy heart, suggesting that this fluid is not being effectively expelled, creating a risk of clotting.

8.1.2 Chapter 3 Summary

In Chapter 3, we use an improved method of creating the mesh to capture these smaller details and generate a more accurate simulation. To the best of our knowledge, we are able to visualize blood flow in unprecedented detail. We used models generated from a healthy patient, a model with simulated hypokinesis, as well as a patient suffering from dyssynchronous heart wall motion.

After simulation, we then analyze the blood flow field through velocity and streamline visualizations. These images, taken during diastole, demonstrates how the complex surface causes the flow to move through and around the empty spaces between the trabeculae. Also, during systole, we see another example of how the blood is forcefully expelled out of the spaces between the trabeculae, rather than simply flowing directly towards the aortic valve as older methods with simpler meshes have suggested. To the best of our knowledge, this level of detail of blood flow-trabeculae interaction has never been seen before.

Blood residency time is a new form of flow analysis, with which we can quantitatively determine regions of the heart that are at greater risk of thrombus, as slower flows are known to be a significant factor predisposing to thrombus formation. Comparing the blood residency times between each of the three heart models, we find that in the healthy heart, blood turnover is high and residency time stays rather low. In a heart model suffering from hypokinesis, the average residence time is significantly higher near the walls, particularly near the hypokinetic apex. Finally, in the dyssynchronous heart patient, we find that a very significant region of the blood has a long residence time, suggesting that due to the low ejection fraction and relatively low fluid velocities, blood is not being adequately circulated and thus is remaining stagnant near the walls, again, particularly toward the apex of the heart.

8.1.3 Chapter 4 Summary

In the previous chapter, we used a sophisticated methods of extracting the heart and its motion from CT in order to capture the geometry of the trabeculae, and used fluid simulations to determine whether the motion and structure of the trabeculae interacts with the blood flow. The purpose of Chapter 4 was to determine the importance of using these detailed models, and whether they are worth the increased computational cost. We did this by comparing the computed flow fields of simplified and complex versions of four patient-specific heart models, and we visually and quantitatively showed that these trabeculae structures are critical in developing the best, clinically-useful results.

In this Chapter, CT images were acquired from four patients. Including a healthy patient, two patients with coronary artery disease, and a patient suffering from dyssynchronous cardiac function. For each patient, we construct both "smoothed" (less trabeculae) and "complex" (more trabeculae) 4D models of their heart, described below. We performed the fluid simulation and blood residence time analysis, similarly to the previous chapter.

We found that in patients with healthy or nonobstructive CAD, blood flow in the complex models have a significantly lower residency times than in the smoothed models. This suggests relatively healthy trabeculae motion and fast blood turnover rates, which are consistent with our expectations. The opposite effect was seen in the patients with highly diseased hearts, where the average residency time is much higher in the complex version, suggesting poorly-functioning trabeculae that is severely dampening the intraventricular flow. Quantitative analysis further determined that the simulations with the complex models were significantly different than those with the smoothed models.

8.1.4 Chapter 5 Summary

In Chapter 5, we proposed a framework to study the blood flow within the aortic valve vicinity using the computational fluid simulation and patient-specific aortic root models. For this work, instead of using the high-resolution patient-specific models, we

used the more standard LV data from the Visible Human Project datasets, since we are currently focused solely on the impact of valvular geometry and motion on the aortic flow patterns, independent of changes in LV function.

We used the same Navier-Stokes simulation method described in the preceding chapters to simulate the flow through the aortic valve and produced very interesting images of the velocity and vorticity fields during systole. From these images, we clearly can see differences in flow patterns through normal and abnormal aortic valve geometries and motion.

8.1.5 Chapter 6 Summary

In Chapter 6, we described a method to improve running times by using the Smoothed Particle Hydrodynamics algorithm with NVIDIA CUDA. These algorithms are very fast, but working with complex boundary conditions is a difficult problem. We solve this problem using two new techniques. First, we presented an updated method of using fluid particles as boundaries, in order keep pressure at the walls stable. With this method, during diastole, the pressure at boundary particles drops as they move slightly farther apart, and the opposite occurs in systole. We found this method to be remarkably effective, and consistently produced stable and accurate results. Additionally, we developed a new method to perform collision detection between fluid and boundary particles.

The simulation was run three times, with different settings of speed of sound, c, and time step length, Δt, for each run. Depending on the choice of Δt, total run times were between 30 and 126 minutes. All of these running times are orders of magnitude better than those described in previous chapters. We also found that as Δt decreases and compressibility goes down, the computed velocities within the heart go up and approach more accurate values. By changing c and Δt, we found that the computed ejection fraction could range from 42% to 50%, on par with the Navier-Stokes methods.

8.1.6 Chapter 7 Summary

In Chapter 7, we described a new framework that allows for SPH simulations against arbitrarily thin walls.

We accomplish this in three steps. The first step, density correction, was performed by deriving the effect and error of computed density particles on one side of the wall will encounter due to interactions with particles on the other side of the wall. Similarly, in the second step, pressure gradient correction, we find and correct the adverse effects on the computed tangential pressure gradients due to particle interactions on either side of the wall. Finally, to enforce the no-penetration condition of particles through the solid wall, we force the normal velocity of particles near the wall to match the normal velocity of the wall itself.

We devised two experiments to evaluate the performance of the Thin-Wall SPH algorithm. The first is a static (single-frame) experiment where are testing the accuracy of the computed density fields and density gradient fields on a curved surface, both on the boundary surface and the corrected solutions projected onto the fluid.We then ran the Thin-Wall SPH algorithm for a single frame, and compared to the true values. The second experiment tests this algorithm's ability to handle rapid deformation and motion. Similar to the first experiment, we start with fluid with an embedded sphere boundary. The fluid's density starts constant everywhere in the domain. The sphere then shrinks to half its original radius in one-half of a second. The results for both experiments were highly promising, with density error values less than 0.5%, and density gradient errors of less than 15%.

8.2 Answers to Questions

In Chapter 1, we presented a number of major questions, which have been fulfilled over the course of this thesis. In this section, we again list the questions, with our concise answers.

- **Can we simulate, visualize, and analyze full Navier-Stokes fluid simulations of the left ventricle with highly detailed models, including papillary muscles and trabeculae, and can our visualizations and analysis of these simulations be used in clinically useful discrimination between healthy and diseased hearts?**: Yes. In Chapters 2 and 3, we successfully simulated blood flow through highly detailed heart models. With our velocity and vorticity visualizations, as well as the particle residency time visualizations, we demonstrated very clear visual differences between healthy and diseased hearts, which could have great practical use in a clinical setting.

- **Does there exist interaction between trabeculae motion and fluid flow?**: Yes. In Chapter 3, for the first time, we have shown clear interactions between trabeculae and blood flow. In a healthy patient, during diastole, trabeculae allow blood to enter. During systole, blood is squeezed out of these smalls paces, helping to circulate blood within the heart. In a diseased patient, there is much less of such interaction, and blood can remain stagnant within the trabeculae, which may lead to potential thrombus.

- **Are there clinically useful reasons to use and prefer the more complex models and simulations of the left ventricle over the more traditional smoothed-wall simulations?**: Yes. In Chapter 4, we show that there are clear differences between blood flow simulations through smoothed and complex models of several patients' hearts. These differences are both visually and quantitatively very noticeable and important, both both healthy and diseased hearts.

- **Can we use blood flow simulations to understand how the blood flow pattern in the aortic valve may play an important role in the remodeling of the aortic root?**: Yes. In Chapter 5, we show our framework provides a clear, high-resolution view of flow patterns. We clearly can see differences in flow patterns through normal and abnormal aortic valve geometries and motion, which could potentially become very useful in a clinical setting.

- **Can we find a new method of SPH boundary management that can**

handle the complex geometry and motion of the left ventricular walls?: Yes. In Chapter 6, we presented new (CUDA-enabled) methods for both treating boundaries as particle to keep the flow simulation at the solid wall stable, as well as boundary particle/fluid particle collision detection and treatment. These methods were highly effective in driving the SPH fluid flow in this difficult, high-complexity, high-deformation solid wall motion.

- **Does SPH perform at a high enough accuracy that could still remain useful for clinical applications?**: Yes. In Chapter 6, we showed that our SPH simulations of a healthy heart provided us with ejection fractions very similar to that from the more traditional Navier-Stokes simulation. However, we cannot currently capture flow through the trabeculae using SPH.

- **Can we derive a new algorithm to allow for Thin-Wall Smoothed Particle Hydrodynamics?**: Yes. In Chapter 7, we have derived a Thin-Wall SPH algorithm and tested it against two intensive experiments. This algorithm performed very well in both experiments, with density values near the boundary no less than 99.5% accurate, and density gradient values no less than 85% accurate. However, as discussed in a later section, we cannot currently use this method for ventricular blood flow simulations and analysis.

8.3 Overview of Main Contributions

In this section, we discuss, in concise terms, my main individual contributions to the work described in this thesis.

In Chapters 2 through 5, I developed all of the visualization methods, designed all of the experiments, and performed all of the analysis. However, I did not perform any of the data acquisition in any of these chapters. Details are listed below.

In Chapter 2, I designed all the visualization techniques and experiments. My analysis and visualizations showed details in the flow that were previously impossible to see, especially around the papilary muscles and some trabeculae structures. Also, the analysis I performed demonstrated clinically useful difference between normal and

simulated diseased heart models.

In Chapter 3, I designed and implemented the "blood residency time" visualization algorithm, which was used to clearly differentiate between healthy and diseased hearts. Most importantly, my visualizations and deep analysis of the cardiac blood flow fields proved the interactions the LV trabeculae and blood flow patterns, which has never been seen before, and is a completely new discovery.

In Chapter 4, I designed and developed the entire set of experiments that further prove the trabeculae-blood flow interactions described in previous chapters. These experiments and my statistical analysis also showed the clear difference and importance of using high-complexity LV models over the more traditional smoothed-wall models in blood flow simulations.

In Chapter 5, my visualizations and analysis were also used to show the clear difference in flow patterns in diseased and healthy aortic valves. Previously, it had been uncertain if and how aortic valve diseases could cause changes in blood flow pattern, which in term causes futher changes and problems in the shape, structure, and performance of the aortic root. My analysis was the first to clearly demonstrate the altered flow patterns in diseased valves, and explain how they could potentially lead to further pathological development.

In Chapter 6, the new method to use particles as boundaries in a deforming solid (LV) was of my design and implementation. I also developed The CUDA-implemented boundary particle/fluid particle collision system. I designed the experiments to test these new algorithms, and I performed the analysis that show that these methods are practical and accurate enough in a clinical setting.

In Chapter 7, I recognized the initial problem (Thin-Wall SPH), and derived all of the necessary equations and developed the entire algorithm independently. I designed experiments and performed the analysis to show that the thin-wall SPH algorithm works, and is a promising new technology for future SPH systems.

8.4 Limitations

8.4.1 Cardiac Blood Flow Analysis

The greatest limitations of the cardiac blood flow analysis for Chapters 2 to 5 is the lack of data. The complex models from high-resoluation CT scans have proven to be very difficult to acquire. Additionally, higher-quality validation is currently impossible. While the ejection fraction results have been very promising, and the simulator itself has been thoroughly validated, we would ideally like to validate the cardiac analysis by comparing computed blood flow velocities with true velocities. For this, we would need a set of patients who would go through both a CT scan to capture images of their heart, as well as MRI or ultrasound to acquire low-resolution valvular flow velocities. This data currently does not exist, and therefore limits our current ability to validate.

8.4.2 Thin-Wall SPH

As mentioned previously, the work in Chapter 6 is limited to effectively smoothed-wall heart models. In addition, since the walls are thickened, flow velocities through the valve tend to be slightly higher than they should normally be. However, the visualizations still compare well with the Navier-Stokes solver results, as well as the ejection fraction.

The Thin-Wall SPH algorithm has several limitations. First, this system does not work well with areas of high curvature. In these regions, this algorithm cannot correctly compute the correct density and pressure gradient on both sides of the wall, leading to inevitable instability. This also means geometries with sharp corners (effectively infinite curvature) perform poorly with the thin-wall SPH method. However, this problem with high curvature and sharp corners is an issue with nearly all SPH boundary management methods. Also, we have found that this method does not currently perform well with geometries that abruptly end or are cut off. For example, the ends of the mitral valve leaflets of the left ventricle cause problems. So, the experiments in this work only used closed spheres to test the algorithm. For this reason, the Thin-Wall SPH method, as it has been described in this thesis, cannot currently be used effectively for ventricular blood flow simulations. That said, we believe that this algorithm is an important first

step in solving the very difficult thin-wall problem.

8.5 Future Work

There is still a large amount of future work possible on this topic. As mentioned in the last section, more thorough validation is the most important next step for this work. However, this will not be possible until the data for such validation exists.

A more immediately approachable future extension of this work is in feature extraction of the fluid flow. In particular, with the very large datasets the fluid simulators produce, we need new tools need to assist domain experts in quick identification of critical patterns. At the time of this writing, we have already submitted a paper for peer review on a method using topological data analysis tools to analyze simulated ventricular blood flow, and automatically detect interesting topological features within the flow. Our method can automatically extract eddies created from vortex shedding across the mitral valve, but we believe that such tools could be extended to find other clinically-useful features of the blood flow.

Another long-term project would be in the simulation of the heart walls themselves. Currently, we drive the fluid flow by forcing the solid to move by a predetermined animation, generated from the CT scan images. An increasingly important topic of research has been using the material properties of the heart muscles to use physical simulations to drive the solid motion. Recent such work has mostly focused on the valves, especially after virtual surgeries [87], but similar techniques might also be used to acquire more accurate motion of the trabeculae.

For the work on the aortic valve blood flow simulations, there are other potential ways to improve this framework. Currently, while LV motion can be adjusted to replicate certain defects, such as hypokinesis, it would be optimal to use models of the patient's true LV by segmenting CT/MRI images of their full heart. This way, our framework would be fully patient-specific, including inflow boundary conditions to the aortic root.

For the Thin-Wall SPH algorithm, there could still be more work to be done, in order

to solve the limitations listed in the previous section. The high-curvature problem could be an important first step. The current method uses the flat wall tangent to the surface to act as the boundary for density/pressure gradient computation on either side. So, the higher the curvature, the more the true surface diverges from the tangent wall. One possible solution would be to find a higher-order tangent surface that fits the shape of the wall more closely. Thus, the particles used to determine density/pressure gradient are more likely to be on their 'correct side" on the correction step.

List of Figures

List of Tables

References

[1] V. Mihalef, R. Ionasec, Y. Wang, Y. Zheng, B. Georgescu, and D. Comaniciu, "Patient-specific modeling of left heart anatomy, dynamics and hemodynamics from high resolution 4d CT," in *ISBI* (W. Niessen and E. Meijering, eds.), pp. 504–507, 2010.

[2] S. Krittian, U. Janoske, H. Oertel, and T. Bhlke, "Partitioned fluid-solid coupling for cardiovascular blood flow," *Annals of Biomedical Engineering*, vol. 38, pp. 1426–1441, 2010.

[3] S. Kulp, D. Metaxas, Z. Qian, S. Voros, L. Axel, and V. Mihalef, "Patient-specific modeling and visualization of blood flow through the heart," in *ISBI* (S. Wright, X. Pan, and M. Liebling, eds.), 2011.

[4] S. Kulp, M. Gao, S. Zhang, Z. Qian, S. Voros, D. N. Metaxas, and L. Axel, "Using high resolution cardiac ct data to model and visualize patient-specific interactions between trabeculae and blood flow.," in *MICCAI 2011*, pp. 468–475, 2011.

[5] S. Kulp, Z. Qian, M. Vannan, S. Rinehart, and D. Metaxas, "Patient-specific aortic valve blood flow simulations," in *IEEE 11th International Symposium on Biomedical Imaging, ISBI 2014, April 29 - May 2, 2014, Beijing, Chin, Beijing, China*, pp. 939–942, 2014.

[6] S. Kulp, M. Gao, S. Zhang, Z. Qian, S. Voros, D. N. Metaxas, and L. Axel, "Practical patient-specific cardiac blood flow simulations using SPH," in *ISBI*, pp. 832–835, 2013.

[7] C. L. M. H. Navier, "Mémoire sur les lois du mouvement des fluids," *Mem. Acad. Sci. Inst. Fr.*, vol. 6, pp. 389–416, 1823.

[8] B. R. Munson, D. F. Young, and T. H. Okiishi, *Fundamentals of fluid mechanics*. New York, 1990.

[9] B. Sanderse and B. Koren, "Runge-kutta methods for the incompressible navier-stokes equations," *21ST AIAA COMPUTATIONAL FLUID DYNAMICS CONFERENCE*, 2013.

[10] A. J. Chorin, "Numerical solution of the navier-stokes equations," *Mathematics of computation*, vol. 22, no. 104, pp. 745–762, 1968.

[11] R. Aubry, F. Mut, R. Löhner, and J. R. Cebral, "Deflated preconditioned conjugate gradient solvers for the pressure-poisson equation," *J. Comput. Phys.*, vol. 227, pp. 10196–10208, Dec. 2008.

[12] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graph. Models Image Process.*, vol. 58, pp. 471–483, September 1996.

[13] F. H. Harlow, J. E. Welch, *et al.*, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," *Physics of fluids*, vol. 8, no. 12, p. 2182, 1965.

[14] J. X. Chen and N. d. V. Lobo, "Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations," *Graph. Models Image Process.*, vol. 57, pp. 107–116, Mar. 1995.

[15] W. W. Tworzydlo, "The fdm in arbitrary curvilinear co-ordinatesformulation, numerical approach and applications," *International journal for numerical methods in engineering*, vol. 28, no. 2, pp. 261–277, 1989.

[16] D. Edwards Jr, "A finite difference method for cylindrically symmetric electrostatics having curvilinear boundaries," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 2, 2013.

[17] N. Nikitin, "Finite-difference method for incompressible navier–stokes equations in arbitrary orthogonal curvilinear coordinates," *Journal of Computational Physics*, vol. 217, no. 2, pp. 759–781, 2006.

[18] N. Foster and R. Fedkiw, "Practical animation of liquids," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 23–30, ACM, 2001.

[19] C. S. Peskin, *Flow patterns around heart valves: a digital computer method for solving the equations of motion.* PhD thesis, Sue Golding Graduate Division of Medical Sciences, Albert Einstein College of Medicine, Yeshiva University, 1972.

[20] C. S. Peskin, "Numerical analysis of blood flow in the heart," *Journal of computational physics*, vol. 25, no. 3, pp. 220–252, 1977.

[21] C. S. Peskin, "The immersed boundary method," *Acta numerica*, vol. 11, pp. 479–517, 2002.

[22] L. Zhu and C. S. Peskin, "Interaction of two flapping filaments in a flowing soap film," *Physics of Fluids (1994-present)*, vol. 15, no. 7, pp. 1954–1960, 2003.

[23] M.-C. Lai and C. S. Peskin, "An immersed boundary method with formal second-order accuracy and reduced numerical viscosity," *Journal of Computational Physics*, vol. 160, no. 2, pp. 705–719, 2000.

[24] R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annu. Rev. Fluid Mech.*, vol. 37, pp. 239–261, 2005.

[25] K. Pruess, C. Oldenburg, and G. Moridis, "Tough2 user's guide version 2," *Lawrence Berkeley National Laboratory*, 1999.

[26] V. Mihalef, D. Metaxas, and M. Sussman, "Textured liquids based on the marker level set.," *Comput. Graph. Forum*, vol. 26, no. 3, pp. 457–466, 2007.

[27] W. T. Reeves, "Particle systems—a technique for modeling a class of fuzzy objects," *ACM Trans. Graph.*, vol. 2, pp. 91–108, Apr. 1983.

[28] B. Solenthaler, P. Bucher, N. Chentanez, M. Müller, and M. Gross, "Sph based shallow water simulation," in *Workshop in Virtual Reality Interactions and Physical Simulation*, pp. 39–46, The Eurographics Association, 2011.

[29] D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret, and J.-D. Gascuel, "Animating lava flows," in *IN GRAPHICS INTERFACE*, pp. 203–210, 1999.

[30] M. Carlson, P. J. Mucha, R. B. Van Horn, III, and G. Turk, "Melting and flowing," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, (New York, NY, USA), pp. 167–174, ACM, 2002.

[31] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics - Theory and application to non-spherical stars," *Monthly Notices of the Royal Astronomical Society*, vol. 181, pp. 375–389, Nov. 1977.

[32] J. Monaghan and R. Gingold, "Shock simulation by the particle method {SPH}," *Journal of Computational Physics*, vol. 52, no. 2, pp. 374 – 389, 1983.

[33] J. J. Monaghan, "Smoothed particle hydrodynamics," *Annual Review of Astronomy and Astrophysics*, vol. 30, pp. 546 – 574, 1992.

[34] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen der mathematischen physik," *Mathematische Annalen*, vol. 100, no. 1, pp. 32–74, 1928.

[35] J. Fang, A. Parriaux, M. Rentschler, and C. Ancey, "Improved sph methods for simulating free surface flows of viscous fluids," *Applied Numerical Mathematics*, vol. 59, no. 2, pp. 251–271, 2009.

[36] K. Erleben, J. Sporring, K. Henriksen, and K. Dohlman, *Physics-based Animation (Graphics Series)*. Rockland, MA, USA: Charles River Media, Inc., 2005.

[37] J. Hongbin and D. Xin, "On criterions for smoothed particle hydrodynamics kernels in stable field," *Journal of Computational Physics*, vol. 202, no. 2, pp. 699 – 709, 2005.

[38] J. Monaghan, "Extrapolating b splines for interpolation," *Journal of Computational Physics*, vol. 60, no. 2, pp. 253 – 262, 1985.

[39] R. Gingold and J. Monaghan, "Kernel estimates as a basis for general particle methods in hydrodynamics," *Journal of Computational Physics*, vol. 46, no. 3, pp. 429 – 453, 1982.

[40] D. A. Fulk and D. W. Quinn, "An analysis of 1-d smoothed particle hydrodynamics kernels," *Journal of Computational Physics*, vol. 126, no. 1, pp. 165 – 180, 1996.

[41] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, (Aire-la-Ville, Switzerland, Switzerland), pp. 154–159, Eurographics Association, 2003.

[42] J. Chen and J. Beraun, "A generalized smoothed particle hydrodynamics method for nonlinear dynamic problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 12, pp. 225 – 239, 2000.

[43] J. P. Morris, P. J. Fox, and Y. Zhu, "Modeling low reynolds number incompressible flows using {SPH}," *Journal of Computational Physics*, vol. 136, no. 1, pp. 214 – 226, 1997.

[44] M. Kelager, "Lagrangian fluid dynamics using smoothed particle hydrodynamics," 2006.

[45] M. Desbrun and M.-P. Gascuel, "Smoothed particles: A new paradigm for animating highly deformable bodies," in *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*, (New York, NY, USA), pp. 61–76, Springer-Verlag New York, Inc., 1996.

[46] J. J. Monaghan, "Simulating free surface flows with sph," *J. Comput. Phys.*, vol. 110, pp. 399–406, February 1994.

[47] J. J. Monaghan, "Smoothed particle hydrodynamics," *Reports on progress in physics*, vol. 68, no. 8, p. 1703, 2005.

[48] M. Liu, J. Shao, and J. Chang, "On the treatment of solid boundary in smoothed particle hydrodynamics," *Science China Technological Sciences*, vol. 55, no. 1, pp. 244–254, 2012.

[49] J. Monaghan, A. Kos, and N. Issa, "Fluid motion generated by impact," *Journal of waterway, port, coastal, and ocean engineering*, vol. 129, no. 6, pp. 250–259, 2003.

[50] J. Monaghan and A. Kos, "Scott russells wave generator," *Physics of Fluids (1994-present)*, vol. 12, no. 3, pp. 622–630, 2000.

[51] D. May and J. Monaghan, "Can a single bubble sink a ship?," *American Journal of Physics*, vol. 71, no. 9, pp. 842–849, 2003.

[52] B. Song and L. Dong, "A new boundary treatment method for sph and application in fluid simulation," in *Proceedings of the 2010 Third International Conference on Information and Computing-Volume 04*, pp. 82–85, IEEE Computer Society, 2010.

[53] G.-R. Liu and M. B. Liu, *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific, 2003.

[54] A. Valizadeh, M. Shafieefar, J. Monaghan, and S. Neyshaboori, "Modeling two-phase flows using sph method.," *Journal of Applied Sciences*, vol. 8, no. 21, 2008.

[55] M. Gómez-Gesteira and R. A. Dalrymple, "Using a three-dimensional smoothed particle hydrodynamics method for wave impact on a tall structure," *Journal of waterway, port, coastal, and ocean engineering*, vol. 130, no. 2, pp. 63–69, 2004.

[56] S. Adami, X. Hu, and N. Adams, "A generalized wall boundary condition for smoothed particle hydrodynamics," *Journal of Computational Physics*, vol. 231, no. 21, pp. 7057 – 7075, 2012.

[57] K. GONG, H. LIU, and B. long WANG, "Water entry of a wedge based on {SPH} model with an improved boundary treatment," *Journal of Hydrodynamics, Ser. B*, vol. 21, no. 6, pp. 750 – 757, 2009.

[58] P. Randles and L. Libersky, "Smoothed particle hydrodynamics: Some recent improvements and applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 139, no. 14, pp. 375 – 408, 1996.

[59] A. Colagrossi and M. Landrini, "Numerical simulation of interfacial flows by smoothed particle hydrodynamics," *Journal of Computational Physics*, vol. 191, no. 2, pp. 448 – 475, 2003.

[60] D. J. Barker, P. Brito-Parada, and S. J. Neethling, "Application of b-splines and curved geometries to boundaries in sph," *International Journal for Numerical Methods in Fluids*, vol. 76, no. 1, pp. 51–68, 2014.

[61] H. Takeda, S. M. Miyama, and M. Sekiya, "Numerical simulation of viscous flow by smoothed particle hydrodynamics," *Progress of Theoretical Physics*, vol. 92, no. 5, pp. 939–960, 1994.

[62] M. Liu and J. Shao, "A new solid boundary treatment algorithm for smoothed particle hydrodynamics," in *RECENT PROGRESSES IN FLUID DYNAMICS RESEARCH: Proceeding of the Sixth International Conference on Fluid Mechanics*, vol. 1376, pp. 335–337, AIP Publishing, 2011.

[63] T. Harada, S. Koshizuka, and Y. Kawaguchi, "Smoothed particle hydrodynamics in complex shapes," in *Proceedings of the 23rd Spring Conference on Computer Graphics*, SCCG '07, (New York, NY, USA), pp. 191–197, ACM, 2007.

[64] T. Jones, T. N. Jones, and D. N. Metaxas, "Patient-specific analysis of left ventricular blood flow," in *MICCAI* (W. M. Wells, A. C. F. Colchester, and S. L. Delp, eds.), pp. 156–166, 1998.

[65] Q. Long, R. Merrifield, G. Z. Yang, X. Y. Xu, P. J. Kilner, and D. N. Firmin, "The influence of inflow boundary conditions on intra left ventricle flow predictions," *Journal of Biomechanical Engineering*, vol. 125, no. 6, pp. 922–927, 2003.

[66] N. R. Saber, N. B. Wood, A. D. Gosman, R. D. Merrifield, G.-Z. Yang, C. L. Charrier, P. D. Gatehouse, and D. N. Firmin, "Progress towards patient-specific computational flow modeling of the left heart via combination of magnetic resonance imaging with computational fluid dynamics," *Annals of Biomedical Engineering*, vol. 31, pp. 42–52, 2003.

[67] V. Mihalef, D. Metaxas, M. Sussman, V. Hurmusiadis, and L. Axel, "Atrioventricular blood flow simulation based on patient-specific data," in *FIMH '09: Proceedings of the 5th International Conference on Functional Imaging and Modeling of the Heart*, (Berlin, Heidelberg), pp. 386–395, Springer-Verlag, 2009.

[68] "The visible human project." http://www.nlm.nih.gov.

[69] D. de Zelicourt, L. Ge, C. Wang, F. Sotiropoulos, A. Gilmanov, and A. Yoganathan, "Flow simulations in arbitrarily complex cardiovascular anatomies - an unstructured cartesian grid approach," *Computers & Fluids*, vol. 38, no. 9, pp. 1749 – 1762, 2009.

[70] A. Gilmanov and F. Sotiropoulos, "A hybrid cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies," *J. Comput. Phys.*, vol. 207, no. 2, pp. 457–492, 2005.

[71] P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, J. Périaux, and D. Knörzer, "Microscale simulation of blood flow in microcirculation using sph method," in *Proceedings of the 2004 European Congres on Computational Methods in Applied Sciences and Engineering*, ECCOMAS, 2004.

[72] T. Yamaguchi, T. Ishikawa, Y. Imai, N. Matsuki, M. Xenos, Y. Deng, and D. Bluestein, "Particle-based methods for multiscale modeling of blood flow in the circulation and in devices: challenges and future directions," *Annals of biomedical engineering*, vol. 38, no. 3, pp. 1225–1235, 2010.

[73] M. Müller, S. Schirm, and M. Teschner, "Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics," *Technol. Health Care*, vol. 12, pp. 25–31, Feb. 2004.

[74] J. Qin, W.-M. Pang, B. P. Nguyen, D. Ni, and C.-K. Chui, "Particle-based simulation of blood flow and vessel wall interactions in virtual surgery," in *Proceedings of the 2010 Symposium on Information and Communication Technology*, SoICT '10, (New York, NY, USA), pp. 128–133, ACM, 2010.

[75] G. Plank, R. Burton, P. Hales, M. Bishop, T. Mansoori, M. Burton, A. Garny, A. Prassl, C. Bollensdorff, F. Mason, F. Mahmood, B. Rodriguez, V. Grau, J. Schneider, D. Gavaghan, and P. Kohl, "Generation of histo-anatomically representative models of the individual heart: tools and application.," *Phil Trans Roy Soc*, 2009.

[76] M. Gao, J. Huang, S. Zhang, Z. Qian, S. Voros, D. Metaxas, and L. Axel, "4D cardiac reconstruction using high resolution ct images," in *FIMH* (D. Metaxas and L. Axel, eds.), 2011.

[77] M. Gao, C. Chen, S. Zhang, Z. Qian, D. Metaxas, and L. Axel, "Segmenting the papillary muscles and the trabeculae from high resolution cardiac CT through restoration of topological handles," in *IPMI*, vol. 7917, pp. 184–195, Springer, 2013.

[78] M. Gao, J. Huang, S. Zhang, Z. Qian, S. Voros, D. N. Metaxas, and L. Axel, "4D cardiac reconstruction using high resolution CT images," in *FIMH*, pp. 153–160, 2011.

[79] R. Bonow, B. Carabello, K. Chatterjee, and et al, "2008 Focused update incorporated into the ACC/AHA 2006 guidelines for the management of patients with valvular heart disease: a report of the American College of Cardiology/American Heart Association task force on practice guidelines," *Circulation*, vol. 118, pp. e523–e661, 2008.

[80] S. Caruthers, S. Lin, P. Brown, M. Watkins, T. Williams, K. Lehr, and S. Wickline, "Practical value of cardiac magnetic resonance imaging for clinical quantification of aortic valve stenosis: comparison with echocardiography," *Circulation*, vol. 108, pp. 2236–2243, 2003.

[81] J. Garcia, R. Capoulade, F. Le Ven, E. Gaillard, L. Kadem, P. Pibarot, and E. Larose, "Discrepancies between cardiovascular magnetic resonance and Doppler echocardiography in the measurement of transvalvular gradient in aortic stenosis: the effect of flow vorticity," *J Cardiovasc Magn Reson.*, vol. 15, pp. 84, Epub ahead of print, Sept. 2013.

[82] R. Gurvitch, J. Webb, R. Yuan, M. Johnson, C. Hague, A. Willson, S. Toggweiler, D. Wood, J. Ye, R. Moss, C. Thompson, S. Achenbach, J. Min, T. Labounty, R. Cury, and J. Leipsic, "Aortic annulus diameter determination by multidetector computed tomography: reproducibility, applicability, and implications for transcatheter aortic valve implantation," *JACC Cardiovasc Interv.*, vol. 4, pp. 1235–45, Nov. 2011.

[83] V. Hurmusiadis and C. Briscoe, "A functional heart model for medical education," in *Functional Imaging and Modeling of the Heart* (A. Frangi, P. Radeva, A. Santos, and M. Hernandez, eds.), vol. 3504 of *Lecture Notes in Computer Science*, pp. 85–91, Springer Berlin Heidelberg, 2005.

[84] M. Müller, S. Schirm, and M. Teschner, "Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics," *Technol. Health Care*, vol. 12, pp. 25–31, Feb. 2004.

[85] P. Goswami, P. Schlegel, B. Solenthaler, and R. Pajarola, "Interactive sph simulation and rendering on the gpu," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, (Aire-la-Ville, Switzerland, Switzerland), pp. 55–64, Eurographics Association, 2010.

[86] M. Tang, D. Manocha, J. Lin, and R. Tong, "Collision-streams: Fast GPU-based collision detection for deformable models," in *I3D '11: Proceedings of the 2011 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 63–70, 2011.

[87] N. A. Tenenholtz, P. E. Hammer, R. J. Schneider, N. V. Vasilyev, and R. Howe, "On the design of an interactive, patient-specific surgical simulator for mitral valve repair," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1327–1332, Sept 2011.