Fall 2018

A primarily Eulerian means of applying left ventricle boundary conditions for the purpose of patient-specific heart valve modeling

Aaron M. Goddard
University of Iowa

A PRIMARILY EULERIAN MEANS OF APPLYING LEFT VENTRICLE BOUNDARY CONDITIONS FOR THE PURPOSE OF PATIENT-SPECIFIC HEART VALVE MODELING

by

Aaron M. Goddard

A thesis submitted in partial fulfillment
of the requirements for the Doctor of Philosophy
degree in Biomedical Engineering in the
Graduate College of
The University of Iowa

December 2018

Thesis Supervisor: Associate Professor Sarah C. Vigmostad

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

PH.D. THESIS

_____

This is to certify that the Ph.D. thesis of

Aaron M. Goddard

has been approved by the Examining Committee for
the thesis requirement for the Doctor of Philosophy degree
in Biomedical Engineering at the December 2018 graduation.

Thesis Committee: _____
Sarah C. Vigmostad, Thesis Supervisor

_____
H.S. Udaykumar

_____
Madhavan Raghavan

_____
Edward Sander

_____
Jia Lu

To my wife and family for all their love and support.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Associate Professor Sarah Vigmostad, for her guidance and opportunity to create this body of work. I would like to thank my wife, Amanda, and my family for their love and continued support during my education. I would also like to thank lab mates Nirmal, Jake, Sarah, Oishik, Liza, Moustafa, Ehsan, Mike, Sidhartha, Pratik, Kayley, and Jared for creating an enjoyable work environment.

ABSTRACT

Patient-specific multi-physics simulations have the potential to improve the diagnosis, treatment, and scientific inquiry of heart valve dynamics. It has been shown that the flow characteristics within the left ventricle are important to correctly capture the aortic and mitral valve motion and corresponding fluid dynamics, motivating the use of patient-specific imaging to describe the aortic and mitral valve geometries as well as the motion of the left ventricle (LV). The LV position can be captured at several time points in the cardiac cycle, such that its motion can be prescribed *a priori* as a Dirichlet boundary condition during a simulation. Valve leaflet motion, however, should be computed from soft-tissue models and incorporated using fully-coupled Fluid Structure Interaction (FSI) algorithms. While FSI simulations have in part or wholly been achieved by multiple groups, to date, no high-throughput models have been developed, which are needed for use in a clinical environment. This project seeks to enable patient-derived moving LV boundary conditions, and has been developed for use with a previously developed immersed boundary, fixed Cartesian grid FSI framework.

One challenge in specifying LV motion from medical images stems from the low temporal resolution available. Typical imaging modalities contain only tens of images during the cardiac cycle to describe the change in position of the left ventricle. This temporal resolution is significantly lower than the time resolution needed to capture fluid dynamics of a highly deforming heart valve, and thus an approach to describe intermediate positions of the LV is necessary.

Here, we propose a primarily Eulerian means of representing LV displacement. This is a natural extension, since an Eulerian framework is employed in the CFD model

to describe the large displacement of the heart valve leaflets. This approach to using Eulerian interface representation is accomplished by applying "morphing" techniques commonly used in the field of computer graphics. For the approach developed in the current work, morphing is adapted to the unique characteristics of a Cartesian grid flow solver which presents challenges of adaptive mesh refinement, narrow band approach, parallel domain decomposition, and the need to supply a local surface velocity to the flow solver that describes both normal and tangential motion. This is accomplished by first generating a skeleton from the Eulerian interface representation, and deforming the skeleton between image frames to determine bulk displacement. After supplying bulk displacement, local displacement is determined using the Eulerian fields. The skeletons are also utilized to automate the simulation setup to track the locations upstream and downstream where the system inflow/outflow boundary conditions are to be applied, which in the current approach, are not limited to Cartesian domain boundaries.

PUBLIC ABSTRACT

Despite major progress in the medical field, understanding, evaluating and treating heart valve disease remains a challenge. Personalized computer modeling of a patient's heart valves has the potential to improve patient evaluation and enable personalized treatments. This requires advances in both image processing and computational modeling along with the interaction between the two fields.

Computational simulation of a heart valve requires both structural and fluid mechanics models. These are typically represented using different mathematical forms, which makes combining the two very challenging. Since heart valves experience large displacement, the problem is formulated using approach for which the geometry of the solid is tracked as it passes over a fixed fluid computational domain. The position of the solid is communicated to the fluid using the level set method. This represents the solid interface by storing the nearest distance between the solid surface and fixed locations in the fluid domain.

To accurately model heart valve dynamics, the contractile motion of the left ventricle must be included. This prescribed moving boundary is obtained using a time sequence of patient images. Due to the disparate time scales of image acquisition and computational modeling, intermediate interface locations are needed. This is typically accomplished by deforming a meshed representation of the interface which is then converted to a level set for each computational time-step. We present a method of determining intermediate positions by directly deforming the level set representation of the interface and thus eliminating the need for a meshed model of the left ventricle.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Motivation

In the US 25,700 people died from heart valve disease during the year of 2015 [1]. According to the American Heart Association, an estimated 2.5 percent of the US population suffers from heart valve disease [1]. Heart valve disease can be primarily separated into two groups. Regurgitation occurs when the valve does not properly seal, resulting in backward flow. This disorder can lead to other complications such as arrhythmias and heart failure [2]. Regurgitation can be caused by a variety of sources including leaflet perforation, leaflet prolapse, and leaflet thickening. Left ventricle enlargement is yet another cause for regurgitation by which heart disease cases the ventricle walls to thicken resulting in valve annulus dilation. Valve stenosis is the other major category of valve disease. Valve stenosis is the narrowing of the valve orifice, and can occur for a variety of reasons including leaflet thickening and calcification. Based on the severity of the diseased state, a physician will decide between medical treatment, valve repair, and valve replacement, with increasing levels of invasiveness. Unfortunately, valve repair rates correlate strongly with surgeon and experience [3]. This means that sometimes less-optimal treatments are being selected for patients. Tools to help physicians to correctly diagnose and select the best treatment are desirable. These tools can come in many forms.

Computational modeling is one such tool that is gaining much attention. These simulations have the ability to aid the physicians in three important ways, such that a more informed decision can be made about whether and how to proceed with treatment. First, computational simulations can help to evaluate patient health. Accurate patient-

specific simulation has the capability of providing quantitative metrics that are typically measured invasively. An example is the services offered by Heart Flow, Inc. (Redwood City, CA), which provides non-invasive quantification of coronary fractional flow reserve based on patient CT images [4]. Patient data is sent to an off-site location for processing by Heart Flow technical experts. While this is an improvement over invasive, catheter based measurements, it has the disadvantage of being performed as an off-site service rather than an immediate in-clinic evaluation. The second potential application of patient-specific modeling is to aid in patient education. Computational models would allow the patients to visualize and better understand the current diseased state, the proposed procedure, and the expected outcome. This is demonstrated by the work of Bernhard et al. [5] which uses 3D printing to construct physical models of the patient kidney and tumor anatomy. Patients demonstrated an improved understanding kidney anatomy, tumor characteristics, and the planned surgical procedure. Finally, patient-specific modeling has the potential to provide physicians with virtual tools allowing them to better predict outcomes and choose the best treatment plan [6]–[8]. While computational modeling of the cardiovascular system has already demonstrated some exciting results, there is still significant room for improvement and expanding the scope of their application. One such application worth considering is the simulation of the left ventricle with heart valves.

1.2 Cardiovascular Anatomy

The cardiovascular system acts to continuously circulate blood throughout the entire body. Blood is the transport vehicle for distributing fuel and nutrients to the living cells. Blood is also responsible for removing certain waste products such as carbon

dioxide. Blood consists of formed elements suspended in a fluid called plasma. The formed elements are mainly red blood cells, white blood cells, and platelets. Red blood cells, erythrocytes, are primarily tasked with carrying oxygen. White blood cells, leukocytes, are responsible for immune responses and platelets have a role in healing injury. Many nutrients, proteins, and waste products are simply dissolved and carried by the fluid plasma.

The closed loop of the cardiovascular system primarily consists of two major circuits connected end-to-end such that the flow rates of the two must match. A diagram of this arrangement is shown in Figure 1. The first circuit sends the blood to the lungs for the purpose of exchanging waste, carbon dioxide, with fuel, oxygen. The second circuit is the systemic circulation which distributes oxygen to the remainder of the body. To accomplish this, an enormous amount of vessel branching is required. As the vessels bifurcate, they get smaller in diameter causing a large disparity in diameter between the vessels near the heart, and the vessels which distribute the oxygen to cells (capillaries). In fact an adult aorta is on the order of centimeters while capillaries are on the order of micrometers. Both of these circuits are centered at the heart.

The human heart provides the force to propel blood through the circulatory system. It consists of two pumps, one for each of the two circuits. An illustration of the relevant heart anatomy is shown in Figure 2. Four check valves ensure the motion of blood flows in a single direction. Deoxygenated blood from the veins first enters the right atrium. During diastole, the heart relaxes and the pressure in the right ventricle decreases causing the tricuspid valve to open. The low pressure pulls blood from the right atrium into the right ventricle. Meanwhile the left semi-lunar valve is closed

preventing the blood upstream from the right ventricle from flowing backward into the right ventricle. As the heart begins to contract, the pressure increases in the right ventricle and causes the tricuspid valve to close. A short period of iso-volumetric contractions occurs until the pressure in right ventricle exceeds the pressure in the pulmonary artery. Blood flows through the newly opened right semilunar valve toward the lungs until contraction of the systolic phase ends. The pressure in the right ventricle again drops causing the semi lunar valve to return to the closed position. A very similar sequence is observed in left portion of the heart. During diastolic relaxation, the mitral valve opens and the low pressure causes oxygenated blood to flow from the pulmonary veins through the left atrium and into the left ventricle. At the end of the diastole, a small contraction of the superior myocardium surrounding the left atrium increases the atrial pressure and forces a little more blood into the ventricle. This is often referred to as an "atrial kick". As the left ventricle begins to contract, the increased pressure causes the closure of the mitral valve which prevents backflow of oxygenated blood toward the lungs. After a short period of iso-volumteric contraction, the pressure in the left ventricle rises above the pressure in the first systemic artery, the aorta, causing the left semi-lunar, aortic, valve to open. Systolic contraction continues forcing the blood from the left ventricle into the systemic region of circulation. Once the heart muscle begins to relax, the pressure of the ventricle falls below the systemic pressure, resulting in the closure of the aortic valve. These cycles continuously repeat resulting in the pulsatile forward flow necessary for maintaining life.

While the sequence of events occurring in the left and right portions are nearly identical, the anatomy and physics have notable differences. These difference are largely

due to the difference between the size and resistance of the respective circuits. The pulmonary circuit is quite small in comparison to that of the systemic. To match the blood flow rates between these circuits, the pressure load applied to the blood must be different. The diastolic pressure of the aorta for a healthy adult is 80mmHg, while the diastolic pressure of the pulmonary arteries is only 8mmHg [9]. Heart anatomy reflects this difference in driving force between the two circuits. The geometry of the left ventricle is prolate-spheroidal in shape with a thick muscular wall, as shown in the illustration of Figure 3. This configuration yields an optimized and powerful pumping action able to reach the 120mmHg required during systole [9]. The right ventricle is wrapped around the left with a crescent shape and thinner muscular wall. While this arrangement is much less efficient and powerful, it is sufficient to provide the 22mmHg needed during systole. This analysis makes clear that the left ventricle is the primary driving pump for cardiovascular circulation and is of much higher concern for medical treatment of diseased states.

To understand the detailed workings of the heart's pumping action, attention must be given to heart valve anatomy. The four heart valves all act to prevent backward flow of blood, but the anatomy has notable variation. The valves can be sub-divided into two groups. The aortic and pulmonary valves are known as semilunar valves. These valves both act to prevent the backflow of blood into the ventricle during diastole; each having three passive leaflets. The mitral, also known as bicuspid, and tricuspid valves are called atrio-ventricular valves. As the name indicates, they are positioned between the atrium and ventricle to prevent backward flow into the atrium during systole. The primary difference between these two are the number of leaflets. As the names suggest the

tricuspid valve of the right heart has three leaflets while the bicuspid (mitral) of the left heart has only two. Both of these valves are reinforced by chordae tendineae attached all along the leaflet edge on one end to papillary muscles on the other. The papillary muscles are located near the apex of the ventricle and contract during systole to reinforce the closed position of the atrio-ventricular valves. These features can be identified in Figure 2. All four of these valves lie in a semi-rigid transverse plane located in the superior region of the heart, Figure 4. The stiffness is provided by a fibrous skeleton structure which forms the annuli to which each of the four valve attaches. The stiffness and location of this feature will be of benefit when considering Fluid Structure Interaction (FSI) modeling of heart valves.

1.3 Clinical Need/Requirements

After review of cardiovascular anatomy, the importance of proper heart valve function is apparent. As previously stated, extending the scope of computational modeling to heart valve analysis will improve the quality of patient care. Many engineering disciplines are needed to accomplish this goal including, image processing, solid mechanics, fluid mechanics, and fluid structure interaction. Before considering specific methods to construct this computational framework, the particular constraints of the scenario must be carefully considered.

A computational framework for the clinical environment should first and foremost minimize the amount of human intervention required to perform the simulation. High levels of user intervention result in the need to have highly trained and specialized operators able to perform the modeling. This generates clinical burden due to limitations on personnel resources, any necessary training sessions, and lead times for conducting

simulations. Relying on highly specialized personnel also introduces human error and variability. The more tasks a person must complete to perform a simulation, the more likely they are to commit an error. There will also be challenges that arise due to patient-to-patient variations and similarly, operator-to-operator variations. It is clear that an ideal process would contain robust, automated algorithms. This can be achieved by selecting numerical methods which minimize the number of adjustable input parameters, which are often determined by trial-and-error or machine learning algorithms. Trial-and-error is not robust or generalizable, and the success of machine learning depends on the size and quality of the training data set. A robust computational framework should limit unnecessary layers of methods. An example is the representation of the moving interface extracted from the patient image data. Frameworks simulating large displacement will often represent the moving interface using both Lagrangian and Eularian methods. Since the image is captured on an Eulerian domain, the Lagrangian representation may be unnecessary. The Lagrangian representation often requires notable user intervention to construct [10]. Finally, a clinical framework should be fully integrated into a single computational platform. Since computational frameworks track and store data in different formats, file storage, data transfer, and conversion are required for many programs to communicate with one another. Each of these steps present the possibility of introducing error.

From this analysis, a set of constraints and guidelines can now be used to analyze computational methods that allow patient-specific heart valve modeling in the clinical environment. Methods should be highly automated, minimize user input, minimize tunable parameters, eliminate redundant layers, and be integrated into a single

computational framework. Each step of the heart valve simulation work flow must be measured against this criteria.

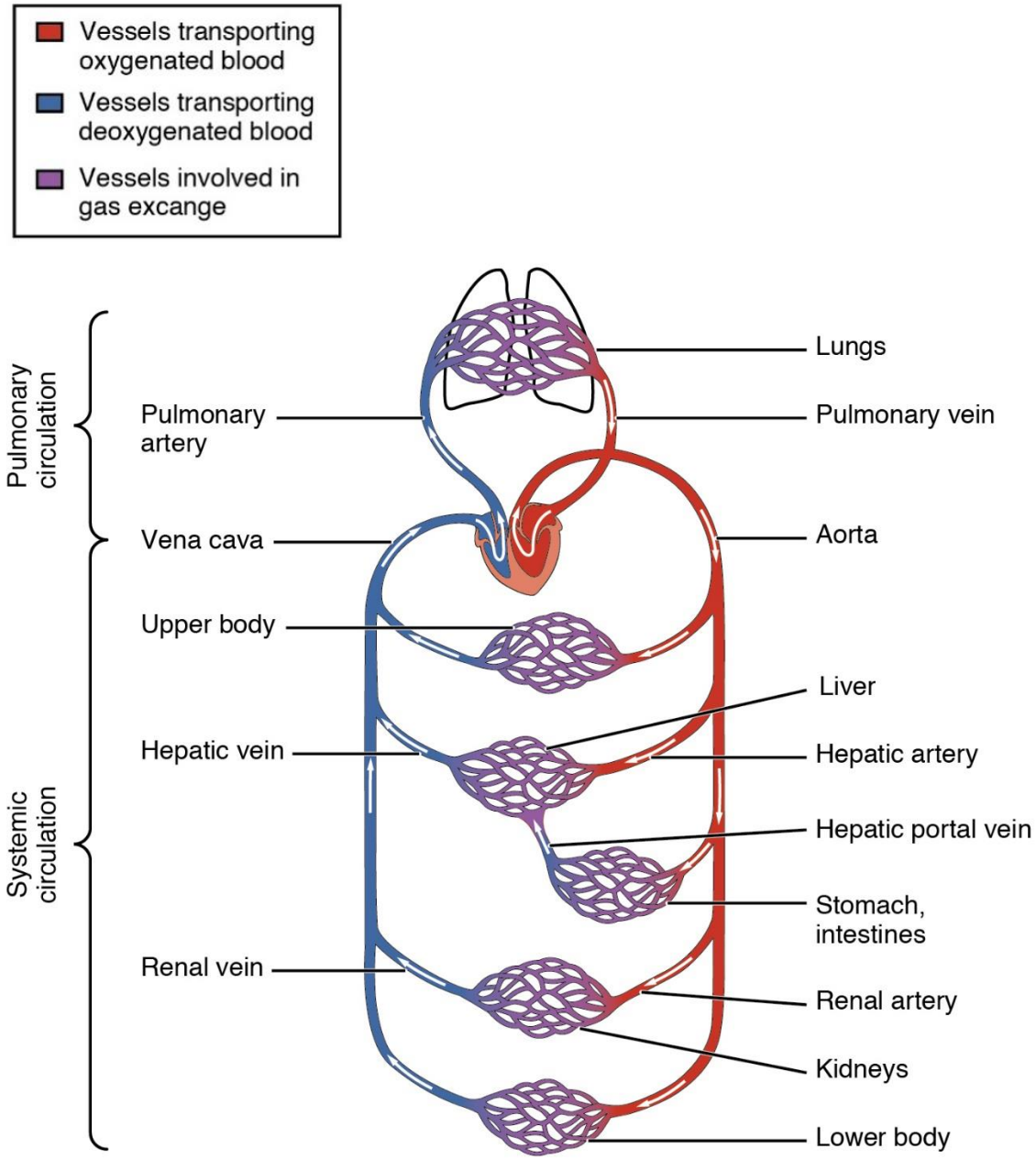


Figure 1: Cardiovascular schematic showing the systemic and pulmonary circulations.

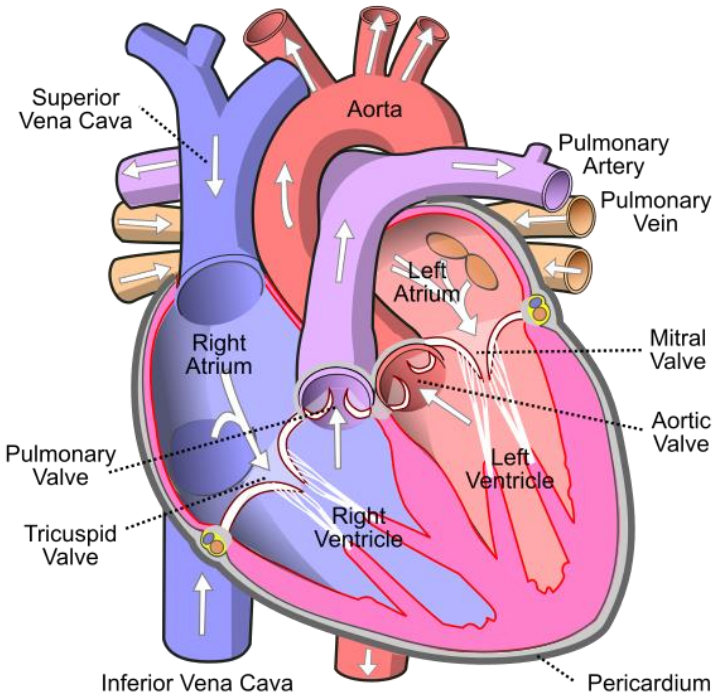


Figure 2: Diagram labeling the relevant anatomical features of the heart and indicates the direction of blood flow.

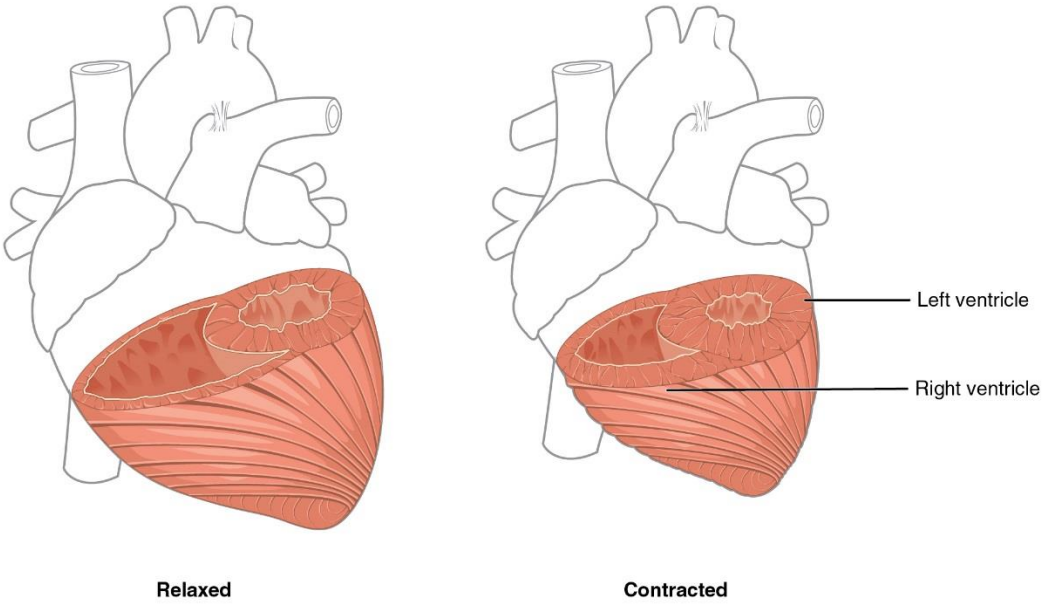


Figure 3: Image showing the difference in shape and wall thickness between the left and right ventricles.

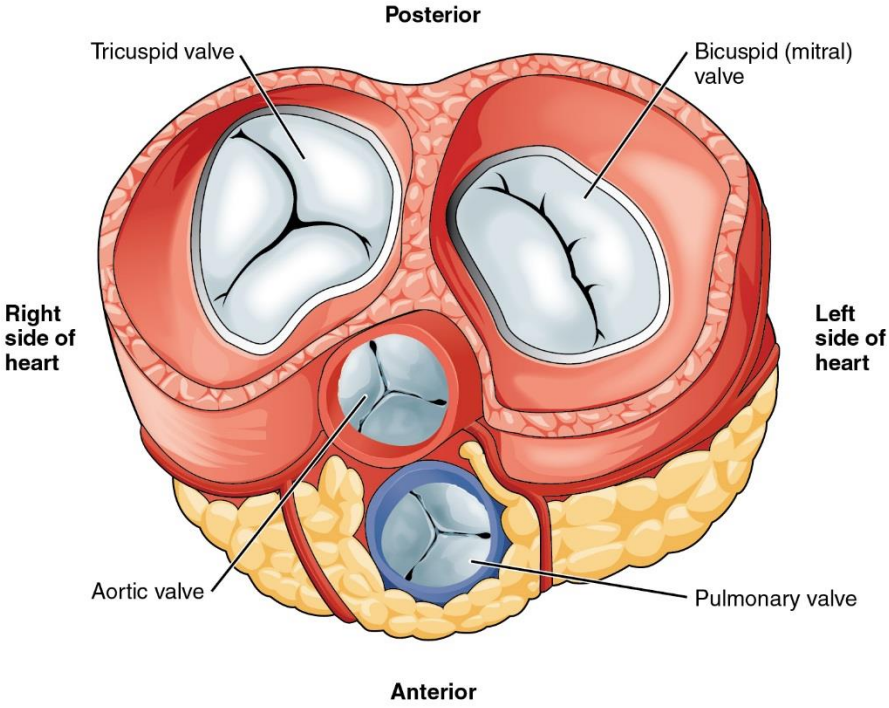


Figure 4: Transverse plane of the heart which contains all four heart valves being attached to a fibrous skeleton structure which acts as a semi-rigid framework.

CHAPTER 2 MODELING WORK FLOW

Per the discussion from the previous chapter, a computational framework to perform heart valve simulations from patient-specific data has the potential to improve the quality of patient care. It is ideal to be have this capability within the clinical environment so physicians can better classify heart valve disease, improve surgical planning, and aid in patient education. Use of these simulation tools in the clinical environment places constraints on the computational methods selected to perform the task. Methods should be highly automated, minimize user input, minimize tunable parameters, eliminate redundant layers, and be integrated into a single computational framework. Since a large number of engineering disciplines are needed to perform such a simulation, each step of the heart valve modeling work flow must be measured against this criteria.

2.1 Typical Modeling Pipeline

While there is variation in work flow between different approaches to heart valve modeling from patient-specific image data, the important steps can be abstracted to lay out the typical work flow. This is demonstrated in Figure 5. A time-series of patient data is recorded using an appropriate medical imaging modality such as Computed Tomography (CT). A single frame consists of a discretized 3D domain for which each voxel contains an intensity value implying the type of tissue located in that region. The time-series of these frames provides implicit motion of the anatomical structures. Since the goal is to model the heart valve motion, leaflet geometry only needs segmentation for a single frame of the full image sequence. This is typically done in either the fully-closed or fully-open position. The remainder of the leaflet positions will be simulated by

applying a solid mechanics model to the heart valve. After the valve leaflets have been segmented, the geometry will be discretized into a polygonal mesh for use with the Finite Element Method. For the mitral valve, this includes a suitable reconstruction of the chordae tendineae. A method for doing so is described in Gade 2014 [11].

The remainder of the surrounding anatomy provides the continuously changing boundary conditions governing the loading and displacement of the heart valve, and therefore must be segmented for every frame of the image sequence. Thus, the papillary muscle attachments and the valve annuli must be identified for all frames. Methods have been developed to accomplish this feat with a varying degree of automation [12]–[14]. These locations are directly applied to the FE model of the leaflet as displacement boundary conditions.

The fluid motion causes a traction on the surfaces of the valve leaflets, resulting in the opening and closing action of the heart valves. Thus, correct fluid dynamics must be simulated so that the correct boundary condition is applied to the surface of the valve leaflets. The interface motion of the endocardial surface provides a solid-to-fluid boundary condition which induces forward flow of the contracting heart. Therefore, the endocardial surface must also be segmented for every image frame of the cardiac cycle. To simulate both valves, the left atrium, left ventricle, and ascending aorta must all be segmented. Strategies have been developed to segment all of these anatomical features [13].

As discussed, both fluid and solid mechanics models must be solved to properly simulate the motion and stresses in the valve leaflets. The mathematical representation of solids and fluids are typically of a different form which makes combining the two very

challenging. Due to the large displacement of the heart valve, the problem is typically formulated using an immersed boundary approach for which the geometry of the solid is tracked as it passes over a fixed fluid computational domain. Applying the moving boundary condition of the endocardium to the flow solver presents a challenging problem [10].

2.2 Current State of Technology

Patient-specific computational modeling has gathered much attention in the recent years with exciting possibilities of improving patient care [4], [15], [16]. Human physiology presents a very complex modeling situation where many capabilities must be integrated to produce a successful simulation. These capabilities often include image segmentation, Computational Fluid Dynamics (CFD), computational solid dynamics (i.e. FEM), and Fluid Structure Interaction (FSI). Several groups have been able to combine all of these to demonstrate the clinical potential of using multi-physics frameworks for heart valve simulations [7], [10], [17], [18].

One such example is the framework developed by Kulp et al. [6]–[8], [19]. High-resolution CT imaging was used to generate the patient data. A polygonal volume mesh of the endocardium was constructed from the segmented image data for an initial frame. The mesh was then morphed (distorted) for each subsequent image frame to match the segmented geometry. Heart valve models were constructed from ultrasound data. These valves were not simulated using a solid mechanics model. Instead, they were parametrically driven to open and close as to match the cardiac cycle at the appropriate steps. The blood was assumed to be a Newtonian fluid and the FSI was implemented using an immersed boundary approach. The inflow/outflow boundary conditions were

established by immersing the entire left ventricle geometry inside a cuboidal domain. The connecting vessels of the left atrium and aorta were not modeled. The cuboidal domain of fluid was assumed to be an analog for the resistance of the circulatory system. While this framework has produced some very interesting streamline visualizations of the flow near the endocardial wall, there are several disadvantages when considering it for use in the clinical environment. While the segmentation is able to capture the endocardial wall in very fine detail, this comes at the cost of performing high-resolution CT with contrast which increases the radiation exposure of the patient. This framework doesn't have the provisions for simulating the valve opening and closing dynamics and instead relies on information about the valve location so that valve displacement can be prescribed. The lack of a solid mechanics model for the valve leaflets also means there is no way to determine leaflet stress levels. Inflow/outflow boundary conditions are very limited due to the fluid domain assumption of cardiovascular resistance. Finally, due to the polygonal mesh construction, a great deal of user intervention is required for the pre-processing phase.

Other exciting results come from the framework developed by Le et al. [18], [20], [21]. For this framework, the left ventricle geometry was only reconstructed from image data for a single frame. Rather than segment the remaining frames and apply a morphing scheme, the interface was discretized as a polygonal surface mesh and a lumped parameter kinematic model was prescribed. While the lumped parameter model was able to include kinematic features such as left ventricle twist, it introduced a large number of parameters and scaling factors, all of which must be correctly tuned to match the *a priori* contractile motion of the left ventricle. The use of a kinematic model also had the

disadvantage of not ensuring alignment with the patient image determined location of the interface, thus deviating from simulating the actual physics observed in the patient. The FSI of the moving heart valves was simulated using a curvilinear immersed boundary method, CURVIB [22], [23]. While this method has shown success in a variety of applications, the structured hexagonal mesh must be carefully constructed to match the tortuous geometry of the cardiovascular system, which is not practical for a clinical setting. Since this approach uses a structured mesh for the flow domain, separate mesh "blocks" were constructed to partition the flow domain and allow for vessel branching. The union between these mesh "blocks" had to be carefully managed so that flow variables could be passed between the structured mesh of each "block".

A final framework of interest was developed by Mittal et al. [10], [24]. This framework used a template-based geometric model construction. The left ventricle geometry was first segmented for every image frame. Then an elastic deformation model was applied to distort a meshed template to match the segmented geometry. While this approach simplified the generation of a polygonal mesh, it depended on having a template which was sufficiently close to the patient-specific geometry. This framework also used an immersed boundary method for FSI and modeled the blood as a Newtonian fluid [25].

A common feature of all these computational frameworks was the use of a polygonal mesh throughout the cardiac cycle to represent the moving left ventricle interface. The construction and management of these polygonal surface representations has been described as the most significant bottleneck when performing left ventricle simulations from patient-specific image data [10].

2.3 Unique Contribution of this Work

The goal of this work is to develop strategies to automate and streamline simulation of patient-specific heart valve modeling in the clinical environment. This is to be furthered by focusing on the process of prescribing image-derived solid-to-fluid boundary conditions of the moving endocardial wall. As previously mentioned, this step has been described as the major bottleneck for performing heart valve simulations. This particular step of the modeling work flow is identified with the red arrow in Figure 5. To be able to do this in an automated manner, it is also necessary to develop a method of isolating an appropriate region of the segmented geometry and applying inflow/outflow boundary conditions at specific locations. The scope of this work does not include image segmentation or solid mechanic simulation of the valve leaflets, which have been extensively studied and developed elsewhere [13], [20].

2.4 Existing Framework Used for Implementation

The selection and implementation of the computational methods must be tailored to each specific computational framework. The selected framework is pELAFINT3D [26]. This is a multi-physics framework developed for FSI simulation of incompressible fluid flows. An immersed boundary approach is used to apply the interface locations of the geometry to the governing fluid flow equations via a sharp interface approach by directly incorporating the boundary conditions into the discretization operators [27]. Multiple sharp interface methods have been developed. The current approach utilizes the Ghost Fluid Method (GFM) [25], [28], [29]. A partitioned approach is taken to alternate between solving the fluid and solid equations in a sub-iterative process during every time-step until convergence is achieved. The flow solver uses a four-step pressure-

correction scheme to solve the incompressible Navier-Stokes equations [30]–[32]. pELAFINT3D is massively parallel with the partitioned domain being load balanced at regular intervals [26]. Mesh pruning has been implemented to remove cells far from the region of interest for computational efficiency. Automatic Local Mesh Refinement (LMR) is used to increase the simulation accuracy while minimizing the computational burden [26], [33]. The immersed geometry is represented via level set functions.

This computational framework has already demonstrated success in modeling cardiovascular flow. One such example is the work of Dillard et al. [34]. In this work the computational framework was used to segment an anterior communicating artery aneurysm from CT data. Simulations were performed to evaluate the flow patterns and characteristics of the aneurysm. This was accomplished without the need for a polygonal surface mesh to represent the segmented geometry. A second example of success for this computational framework is the work of Govindarajan et al. [35]. In this work, the opening dynamics of normal and bicuspid aortic valves were investigated. The valves were positioned inside a rigid aortic arch model. An inlet velocity profile was prescribed to mimic the flow generated by the contraction of the left ventricle. These two examples demonstrate the framework's capability to perform fluid structure interaction of heart valve motion and patient-specific image-to-flow simulations. Both of these are crucial elements in the modeling pipeline previously described. Thus, pELAFINT3D is an excellent choice for investigating the stated goals of this work.

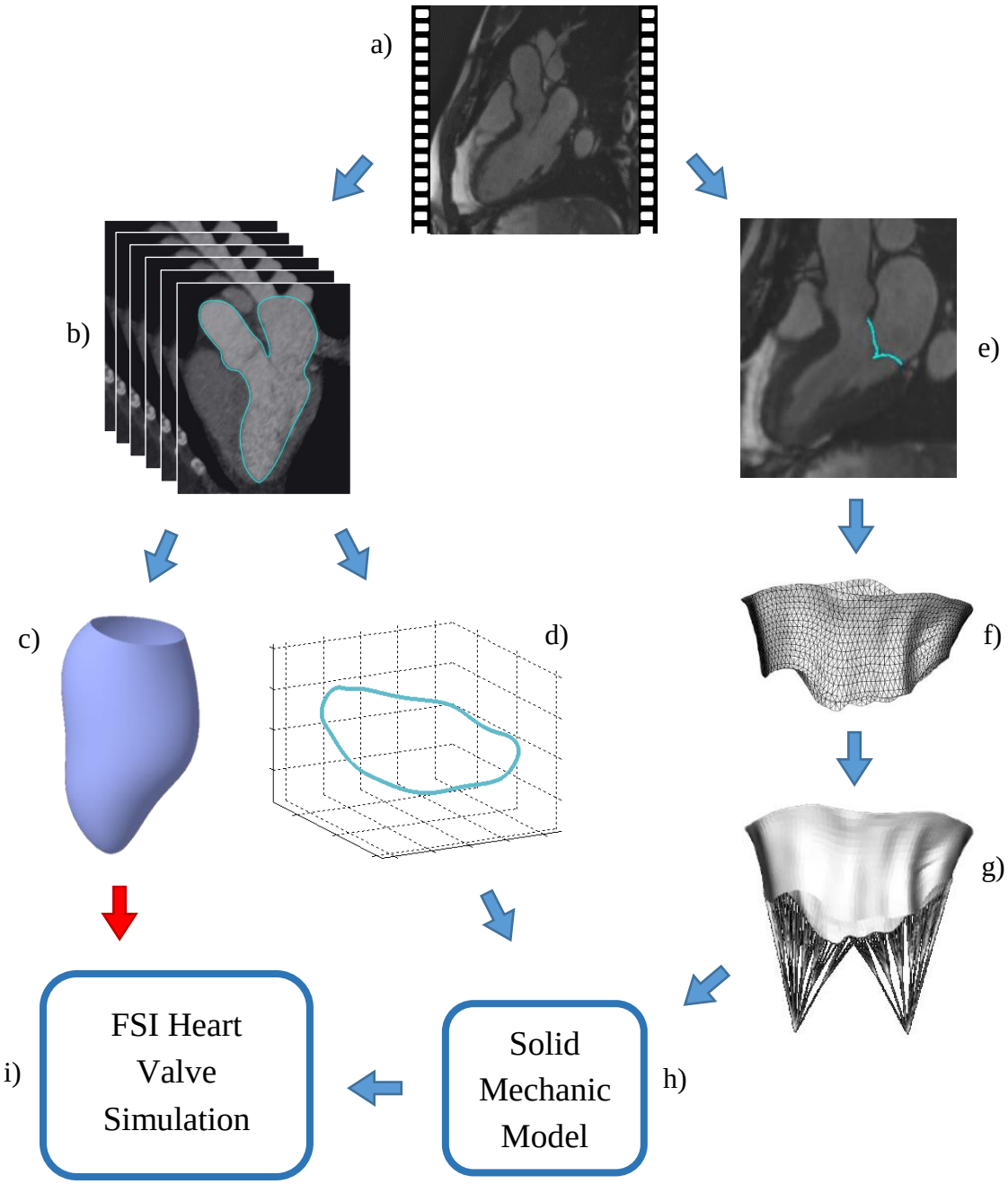


Figure 5: Typical work flow for computational modeling of left ventricle and the associated heart valves from of patient-specific images, a) raw patient-specific image data, b) segmentation of every image frame, c) resulting endocardial surface for every frame, d) annulus geometry segmented for every frame[14], e) segmentation of a single frame of the image sequence, f) polygonal surface mesh of mitral valve[11], g) chordae tendineae reconstruction[11], h) solid mechanic model of valve leaflets incorporating the reconstructed geometry and the displacement boundary conditions from the valve annulus and papillary muscles, and i) FSI heart valve simulation performed using the valve solid mechanics model and the moving boundary condition from the segmented endocardial surface.

CHAPTER 3 GEOMETRY MORPHING

3.1 Need for Geometry Morphing

Fluid flow simulations are performed by marching a solution forward in time. Time-step size is constrained by a CFL (Courant-Freidrichs-Lewy) type condition. This relationship ensures numerical accuracy and stability by relating the maximum time-step size to the maximum fluid velocity and grid resolution. This relationship is shown in the equation below with U representing the maximum fluid velocity in the domain and Δx representing the grid spacing.

$$CFL = \frac{U\Delta t}{\Delta x} \leq 1 \qquad 3.1$$

Clearly, for high resolution calculations (i.e. small Δx), the time-step size is limited to maintain stability. When applied to a left ventricle simulation, the required fluid time-step is much smaller than the frame-rate of clinical image modalities which is often around 20 frames to complete a cardiac cycle. At a resting heart rate of 72 beats per minute, the frame duration is approximately 42ms. A visual representation of this can be seen below in Figure 6. This timescale discrepancy necessitates the estimation of intermediate interface locations. The process of generating these intermediate interface locations is commonly referred to as *volume metamorphosis* or *morphing*. Morphing is the attempt to smoothly deform an interface from one known configuration to another over a finite number of intermediate steps. For this application, the primary need for morphing is due to the time-step requirements of the fluid flow solver, so the morphing method should be selected from that perspective.

Several factors must be weighed when considering and selecting an interface morphing approach. Most CFD solvers utilize adaptive time-stepping, causing the time-step size to vary throughout the simulation to maintain the CFL constraint. In other words, as velocities increase, the time-step size will necessarily decrease if the grid resolution is maintained. Therefore, the method must be capable of being performed during simulation run-time rather that prior to the start of the simulation using *a priori* information. In working toward closer-to-real-time simulations for the clinical environment, it is also desirable for the method to minimize the computational burden. Since the flow solver is fully parallel through domain decomposition, it is critical that the morphing scheme be compatible with this partition strategy where each partition owns (and contains information about) only a small portion of the whole fluid domain. A ghost layer of cells on adjacent partitions are copied to provide values needed to complete computational stencils of cells at the edge of the domain partition. The values of the ghost layers must be copied or exchanged continually throughout the flow simulation. Since the interface provides the no-slip solid-to-fluid boundary condition, it is important that the boundary velocity be physically realistic. The interface is to be mathematically defined using the level set method, which has sub-grid accuracy; therefore, the morphing method should be selected or developed to avoid compromising the sub-grid accuracy. Since the long term goal of this work is to develop a framework and methods that have the capability of performing heart/heart valve simulations in the clinical environment, methods should be selected or developed for 3D space that perform the morphing process in an automated manner requiring little or no user intervention.

3.2 Review of Commonly Used Morphing Approaches

Morphing has largely been motivated and advanced by computer graphics and animation needs. It should be noted that computer graphics is primarily concerned with the interface position and achieving visually smooth or aesthetically appealing transformation [36]. Morphing for the purposes of FSI has additional constraints beyond those required for computer graphics, such as maintaining a continuous and physically realistic interface velocity. Each group of morphing approaches must be considered to determine the most suitable strategy for this work, keeping in mind that morphing can be performed on the raw images, a boundary representation, or on a level set representation of the interface.

Many methods have been developed for morphing based on the raw pixel/voxel representations of the images. These methods operate on a variety of principles, such as field morphing [37], [38], energy minimization [39], or work minimization [40], [41]. While all of these methods have been shown to be successful in their particular application, because they operate on the raw pixel/voxel domain, they are not an ideal solution for patient-specific heart valve modeling. Performing the morph on the raw images would require *a priori* knowledge of the number of intermediate locations required for the simulation, which is dependent upon time-step size. As previously mentioned, with adaptive time-stepping, it is not possible to know the number of intermediate interface locations needed prior to the simulation. Since the FSI framework considered here uses a sub-grid level set representation, the grid level resolution of these methods is not sufficient. For both of these reasons, a pixel/voxel-based interface morph strategy is not a suitable strategy for this work.

Morph strategies using a boundary representations are very popular not only in the 3D computer graphics field [42]–[46], but also in previous efforts to morph the contracting left ventricle geometry [10]. The boundary representation is typically constructed using a polygonal discretization scheme, such as a surface mesh, or as a parameterized surface, such as splines. To use these approaches, the images must first be segmented and converted to the Lagrangian representation of the interface. As described by Mittal et al., the process of polygonal mesh generation is difficult to automate and remains the largest bottleneck for patient-specific left ventricle modeling [10]. This is largely due to the limitations of mesh morphing which typically require consistent topology and connectivity [36]. To generate a physiologically accurate morph trajectory of the interface, a material model is often applied to the meshed representation. The positions are then computed by minimizing the total energy of the system as implemented through the tissue model [47], [48]. In addition to the already stated challenges of this approach, the mesh deformation must be carefully managed as to not create poorly shaped or inverted elements. Interface morphing using a boundary representation is clearly not a wise choice for the application of patient-specific left ventricle modeling.

Level set approaches were first established in the field of computer graphics as an effective means of smoothly performing morph operation [49]–[51]. Level set morphing alleviates the topology and consistency issues inherent to the boundary representations previously mentioned. Since the immersed boundary approach employs a level set to represent the interface, this is a convenient approach for our application. These approaches are valid in 3D space and are compatible with the current domain partitioning strategy. It is important to note that these approaches have not been developed to satisfy

one of the primary constraints inherent to the interface requirements imparted by CFD solver, accurately capturing the motion of the interface in the tangential direction. Despite lacking tangential displacement information, a level-set-based morphing strategy similar in nature to the one developed by Breen et al. [49] will be considered as a starting point for the current algorithm's development.

3.3 The Level Set Method

3.3.1 Level Set: Mathematical Definition and Motion

A level set is a signed distance function which defines the location of an interface with respect to a fixed set of references. For this implementation, the fixed set of references exist as nodal points on a Cartesian grid. A level set can be mathematically represented as a front propagating away from the interface in pseudo-time according to the following partial differential equation.

$$\frac{\partial \phi}{\partial t} = \text{sign}(\phi)(1 - |\nabla \phi|) \qquad 3.2$$

For computational efficiency, a *narrow band* approach is utilized where the values are only stored for cells within a certain distance from the interface, often a radius of six computational cells [52], [53]. Once created, the interface can be moved (advected) by solving Equation 3.3 below. The speed function, F, determines how the interface will move and deform. A uniform speed function will result in rigid body motion of the interface while a spatially varying speed function will cause shape change to the interface. Thus, a spatially varying speed function will provide a means to implement level set morphing.

$$\frac{\partial \phi}{\partial t} + F|\nabla \phi| = 0 \qquad 3.3$$

3.3.2 Level Set: Means of Construction for Use with the Flow Solver

A level set representation of an interface can be generated in a variety of ways. For example, an analytical function can be evaluated at every grid point to compute each point's signed distance to the interface. This approach is limited to simple shapes that can be represented by an analytical equation over the entire domain. To construct level set fields of more complex geometries, one can start with a Lagrangian representation of the interface, typically a surface mesh. The mesh is placed in the domain to determine the nearest distance between the interface and each Cartesian grid point. A very fine resolution of the surface mesh is required. If the element size of the surface mesh is larger than the spacing of the fluid grid, the level set representation will reflect the discretized (non-smooth) profile of the surface mesh. Alternatively, one can leverage the use of level sets as a segmentation tool, so that the flow-based level set field can be generated by sampling the results of segmentation. One such example of an existing level set field is the end results from image segmentation using Active Contours. The Active Contours method is a common technique used to segment medical images [54]–[56]. This is typically accomplished by initiating a small circular level set field (seed) in the region-of-interest and growing the interface until a specific intensity gradient is encountered, indicating the presence of an interface.

When constructing an interface from image data, it's important to consider the limitations of image segmentation. Due to the discrete nature of both the image domain and intensity values used to describe the image, regardless of the method, image

segmentation will introduce some amount of error when the location of the segmented interface is described [57]. The significance and impact of the error depends on the application and quality of the images. In addition to position, left ventricle FSI simulations also require that an accurate estimate of interface velocity be obtained from the image segmentation, since the myocardial tissue provides the fluid with a moving boundary condition as it contracts during the cardiac cycle. Importantly, the positional error from image segmentation is compounded when making any estimate of the interface velocity. Methods of image processing have been developed to minimize the positional error of segmentation by utilizing the time history of preceding frames to influence the subsequent frames [56], [58], [59].

3.4 Challenges Inherent to the Level Set Representation

To implement an interface morphing strategy based on level sets, it is important to consider the challenges associated with an Eulerian representation of the interface. For a moving object represented by a level set field, the distance stored at any particular location on the Cartesian grid will project to a different location on the interface as the interface crosses the computational domain. A cell in a level set field always stores information about its distance to the nearest location on the interface. As the interface moves, a cell's nearest interface location continually changes. This is demonstrated below in Figure 7, where nodes are drawn on the interface to illustrate the changing location for which the level set vector of a particular cell intersects the interface.

This continuously changing reference location highlights an inherent challenge with using level sets to represent moving interfaces. The level set field does not measure the exact displacement of any point on the interface; it can only be inferred. Hence, the

interface motion may not accurately capture the true interface velocity, especially in the direction tangent to the interface. For simulations where the interface motion is primarily in the normal direction, such as a radially contracting circle or deflating balloon, the tangent velocity can be considered negligible and no further treatment is necessary. However, the contracting left ventricle is not as simple. While the left ventricle is prolate spheroidal in shape, with contraction primarily in the radial direction, it also experiences axial displacement as well as a twisting motion. The twisting motion is generated during contraction due to the geodesic arrangement of the myocardial fibers. The twist velocities of the myocardium are on the order of cm/s [60], while the jet velocity through the aortic valve is on the order of m/s [61]. Comparison of these velocities shows two orders difference in magnitude, suggesting that it is reasonable to neglect the twist velocities of the myocardium. On the other hand, it has been shown that vortices influence the filling dynamics of the left ventricle [62]. Fluid vorticity can only be generated at the solid-fluid boundary with the magnitude being a function of tangential interface motion.

3.5 Previously Developed Level Set Morphing Approach

Dillard et al. 2014 [63] developed a novel framework for morphing an image-derived fluid interface represented only by a level set field. This approach directly moved the level set representation of the interface, thus eliminating the need for a Lagrangian representation (polygonal mesh) of the interface. The primary test case was that of an American swimming eel, and an optical flow approach was used to supply the level set advection velocity.

Optical flow was first introduced by Horn and Schunck [64]. This technique operates on an image by assuming the intensity field of the image is conserved and that the velocity field must be smooth due to the no-slip boundary condition at the fluid-solid interface. The constant intensity assumption implies the sequence of images has a uniform fluid image intensity with a conservation of fluid volume. This assumption is reasonable when an incompressible object is moving and deforming while immersed in external flow, such as the swimming eel case mentioned. This, however, is not the case for a contracting left ventricle. During the contraction of the left ventricle, the fluid volume decreases. This is reflected in a decrease in the number of voxels/pixels with the intensity value corresponding to blood. The intensity values can also vary as a result of the surrounding anatomy entering or exiting the medical image frame during the cardiac cycle.

In addition to the limitations offered by the assumption of conserved intensity, the optical flow approach only provides a mean advection velocity between frames, it does not provide the intermediate interface locations needed for the fluid flow solver. To achieve intermediate interface locations, Dillard et al. implemented the level set image blending algorithm developed by Whitaker [50]. This image blending algorithm was performed by constructing a level set field at the image-level of resolution. The level set field was used as a distance metric that penalizes regions of the interface between sequential frame pairs. The entire image was treated as a continuous function of image intensity. The initial frame and the subsequent frame were formulated as independent functions which are marched toward each other in pseudo-time to achieve the final blend. This approach has been noted to result in approximate, but non-physiologic morph

trajectories, since tangential displacements are not accurately represented due to forcing function metric being derived from the level set [63]. For the eel simulation, the motion resulted in a shortening and then lengthening of the tail during each frame. Since this approach only considered the initial and target frame, the first derivative of interface location was not continuous between frames. This effectively resulted in a velocity discontinuity of the interface from one frame-interval to the next. Additionally, the blend operation didn't enforce the relationship between the interface velocity and position. Since the position was obtained by a blend operation and the velocity independently determined by the optical flow algorithm, the interface was over-constrained. This likely caused issues of applying the immersed boundary treatment by providing conflicting information of position and velocity to the fluid.

Another drawback of this approach was the coarse resolution of the video image to which this strategy was applied. Since this algorithm operated at the pixel level, the advection velocity vectors and intermediate interface locations generated were at the same pixel level resolution. Interpolation schemes were used to sample the data to the much finer mesh of the fluid flow solver. While this approach showed that a level set representation can be used to perform image-to-flow simulations, it doesn't apply well to the goals set forth in this work. An alternative method of morphing the level should be considered.

3.6 Level Set Morphing Method

3.6.1 Level Set Morphing: Establishing the Interface Trajectory

Volumetric morphing is often broken down into three separate steps. First, rigid body motion, followed by a warp operation acts to change the general shape or topology

of the object. Finally, a blend operation is required which fine tunes the remaining deviation of the surface.

Figure 8 shows an example of how this progression may be performed with respect to the constraints inherent to an Eulerian representation of the interface. Figure 8a shows the starting location of the morphing object as a red stadium shape with the target location (i.e. the next interface location obtained from the subsequent frame) for this morph shows as a larger green stadium with a centroid positioned upward and to the right. Simply comparing the level set values of these two interfaces would provide a blend-only morph path as shown in Figure 8b. A stadium with uniform material properties would evenly distribute the horizontal displacement or stretch; however, the displacement of the horizontal portion of the stadium in Figure 8b is clearly unrealistic as all of the tangential motion or stretch is concentrated at the right end of the stadium. Since interface velocity is derived from the interface position, the error in tangential displacement will cause an error in the tangential velocity. This example of an interface velocity error causes incorrect boundary conditions when applying the no-slip boundary condition between the solid interface and the fluid. To achieve a physiologically realistic simulation, the warp step is of critical importance when applying this morph approach to cardiovascular simulations. The challenge is to determine a metric that can be used to first indicate the topological change in position and shape needed to perform the warp operation. Since the cardiovascular system is a network of vessels, it is reasonable to expect that a centerline representation of the network is a sufficient low-order approximation that can be used to guide the warp step of volumetric morphing. For the stadiums used in the current example, centerlines are easily and intuitively constructed.

These are shown as dashed lines for each of the interfaces in Figure 8c. A linear mapping based on arc length is used to compute the displacement of each point along the centerline. This displacement represents the large scale deformation attributed to the warp operation as shown in Figure 8d. The warp displacement and deformation of the centerline is then mapped to the interface of the morphing object to achieve an intermediate warp position of the interface shown in yellow of Figure 8e. Now the level set values can be compared between the target interface (green) and the intermediate (warp-only) interface position computed from the warp operation (yellow). Figure 8f includes the trajectory of the interface when the blend step is subsequently applied to the intermediate interface. Figure 8g shows the summation of the warp and blend vectors. The final morph trajectories of Figure 8h can be compared with those of Figure 8b to see the benefit of adding the warp operation to the morph process. Per the desire of mimicking a linear elastic interface, the use of the warp operation, results in a uniform distribution of stretch along the interface.

3.6.2 Level Set Morphing: Domain Modifications

Now that a reasonable morph strategy has been identified, it is important to determine its appropriate implementation within the current computational framework. This is best considered by continuing with the same growing stadium example illustrated in Figure 8. Consider the image frame representing the initial interface (corresponding to the red stadium) shown in Figure 9a. Once this image is segmented, a level set field is constructed in the computational domain to mathematically represent the interface, shown in Figure 9b. An iso-contour of this field at the zero level set value will be the same as the red oval starting position of Figure 8a. The same operation is performed on

the second frame of the image sequence (corresponding to the green stadium), as shown in Figure 9c-d. This second frame is the morph target for the interface to pass through for which the iso-contour of the zero level set value corresponds to the green stadium in Figure 8a. As previously mentioned, level set values are only stored in a narrow band around the interface to reduce the computational burden. Inspection of Figure 9b and d, shows there are many locations where the narrow bands around the respective interfaces do not overlap, and thus would normally not be assigned a level set value. If the target position is outside of the initial interface's narrow band, there will be a significant number of computation grid points that would only have information for the distance to either the initial interface or the target interface, but not both. This is can be visualized through the local mesh refinement shown in Figure 9e. Clearly, the narrow tube of the morph target must be expanded so that the initial interface has a contiguous field of level set values to cross. The region over which the interface must pass to reach the target will be referred to as the *morph zone*. This morph zone can be computed by independently solving a Laplace equation, shown below, for the initial object and every morph target with λ being the field result and i indicating the frame number.

$$\nabla^2 \lambda_i = 0 \qquad \lambda_i|_b = i \tag{3.4}$$

Dirichlet boundary conditions equal to the frame number are applied to cells adjacent to, and inside the interface, while cells adjacent to, but outside the interface, are given a value of zero. Solving this equation results with a contiguous region inside the interface being assigned an integer representing the frame number, as shown in red for

the starting interface location, while outside the interface, the cells are given a value of zero.

An *exclusive-or* Boolean operation is applied to the results of the Laplace equation as shown below, where δ_i indicates a binary morph zone value with 1 signifying the cell belongs to the ith morph zone. For simulations containing more than two input images, this operation is repeated for each successive pair of frames using the results of the Laplace equation above. This operation identifies cells that lie between the two interfaces. This region is the morph zone for which the interface must cross to get from the starting position to the target position as shown in Figure 9g.

$$\delta_i = MAX(\lambda_i, 1) \oplus MAX(\lambda_{i+1}, 1) \qquad 3.5$$

Once identified, these morph zone cells are refined to the smallest level allowed and added to the level set tube of the target interface, Figure 9h. This extra refinement and enlarged level set tube size increases the computational burden of the simulation, but for image sequences with a large number of frames, the interface displacement will be small between frames and thus will limit the extra computational burden. As seen in Figure 9i, the level set field of the target is reinitialized according to the equation below, which computes level set values for all the newly added cells of the morph zone, where ϕ is the level set value and *t* is a pseudo-time used to perform the calculation.

$$\frac{\partial \phi}{\partial t} + \text{sign}(\phi)(1 - |\nabla \phi|) \qquad 3.6$$

Figure 9j superimposes the starting and target interfaces on top of the target level set field to clearly show the cells adjacent to the morphing interface will have level set values from both fields during the entire trajectory of the morph operation.

3.6.3 Level Set Morphing: Computing Steady Morph Advection Velocity

With the previously mentioned domain modifications for each frame increment, attention can be turned to computing the velocity needed to move the interface to the target location. The first step is to compute the steady morph advection velocity for each interval. This is a position dependent velocity field that when applied to the interface for the full frame duration, would morph the starting interface to the target interface location. Computation of this field is demonstrated using the same interface geometry from Figure 9. Consider a cell, *ic*, located on the starting interface with a domain coordinate $\mathbf{x}_{ic}$ as shown in Figure 10a. Since the warp operation is performed with respect to the centerline representation, cells in the level set narrow band must be mapped (associated) to a suitable location on the centerline representation. For the stadium geometry in Figure 8a, the mapping is straightforward due to symmetry, as shown in Figure 10b. Here, a linear mapping can be employed, with the percent shown in the figure indicating the arc length position with respect to the total length of the skeleton segment. The mapping is used to match the corresponding location on both centerlines as shown in Figure 10c, with $\mathbf{x}_{sk1}$ locating the base of the arrow and $\mathbf{x}_{sk2}$ locating the arrowhead. These locations are then used to compute the warp offset distance, as indicated by Figure 8d, which is then added to the cell position to determine probe location for the blend operation as shown in the equation below. The probe location for this particular cell is shown as the blue x in Figure 10d.

$$\mathbf{x}_{probe} = \mathbf{x}_{ic} + (\mathbf{x}_{sk2} - \mathbf{x}_{sk1}) \quad 3.7$$

The probe location is necessary to compare level set values between the initial and target interface to compute the blend offset. Since the probe location will lie in an arbitrary position in the computational domain, it is necessary to identify the group of cells bounding this probe location (a box of 4 cells in 2D and a cube of 8 cells in 3D). From here, bi/tri-linear interpolation can be performed using the target field level set values of the bounding cells in order to compute the target field level set value of the probe location, $\phi_{probe,2,}$ which will be compared to the corresponding level set value on the interface of the initial field, $\phi_{ic,1}$. Since the computational framework utilizes domain decomposition for parallel computation, interface adjacent cells near the partition boundaries will likely have probe locations which lie on other domain partitions. The greater the warp displacement, the larger the number of cells for which this will occur. Thus, each probe location must be tested to see if it is contained in the same domain partition as the cell. A list of off-partition probe locations is distributed to all processors so the probe containing partitions can be identified, evaluated, and communicated back to the partition which owns the initial cell. The level set value at the probe location can then be used to determine the blend displacement. This is computed for the current interface cell using the starting field level set value, $\phi_{ic,1}$, of the cell and the level set normal direction, $\mathbf{n}_{ic}$.

$$\mathbf{x}_{blend} = (\phi_{probe,2} - \phi_{ic,1})\mathbf{n}_{ic} \quad 3.8$$

At each interface cell, the warp displacement ($\mathbf{x}_{probe}$) and blend displacement ($\mathbf{x}_{blend}$) are summed to compute the total displacement for that cell. From here, the steady advection velocity ($\bar{\mathbf{u}}_{ic}$) is calculated by dividing the total displacement by the duration of the frame interval ($t_{frame\_duration}$), as shown in Equation 3.9.

$$\bar{\mathbf{u}}_{ic} = \frac{\mathbf{x}_{blend} + \mathbf{x}_{probe}}{t_{frame\_duration}} \tag{3.9}$$

After computing the mean advection velocity at the interface, this information must be extended through the thickness of the level set tube, so that the advection equation has an appropriate velocity value for each incremental interface location as it moves from the initial to the target position. Figure 10e shows the steady morph velocity vectors are extended through the entire level set field. This process is repeated for every frame increment of the image sequence, such that at increment 2, the previous target serves as the new initial frame, and the next frame in the image sequence becomes the new target interface.

3.6.4 Level Set Morphing: Using Interpolation to Achieve Smooth Advection

Simply applying the steady morph advection velocity would result in a piecewise advection velocity that would be constant during each frame interval. While the interface position would match the target very well, the interface velocity would experience a discontinuity at the end of every frame increment. A physiologic representation should instead provide a continuous interface velocity, which in turn applies a continuous boundary condition to the fluid through the no-slip wall assumption used for laminar flow. To achieve a continuous interface advection velocity, the steady morph advection

velocity is interpolated between the current frame increment (initial interface to target interface) and the following frame increment (target interface to the following target interface). Equation 3.10 shows this computation, where *i* indicates the current frame increment and *i+1* denoting the following frame increment. The duration of the current frame increment is denoted as t_{fdur}, and *t* indicates the current simulation time.

$$\mathbf{u}_{ic} = \left(1.0 - MOD\left(t, t_{fdur}\right)\right)\bar{\mathbf{u}}_{ic,i} + MOD\left(t, t_{fdur}\right)\bar{\mathbf{u}}_{ic,i+1} \qquad 3.10$$

This approach provides a continuous interface advection velocity as shown in the example of Figure 11; however, the interpolation interface is out of phase from the frame interval by half a frame as shown in the figure. For the particular example shown in Figure 11, the frame duration is 1.0 units of time. Thus the first morph increment is only 0.5 units in time and linearly interpolates from an advection velocity of zero to the first steady morph advection velocity. Then from simulation time 0.5 to 1.5 the advection velocities are interpolated between the first steady advection velocity field and the second steady advection velocity field. This staggered shift is repeated throughout the simulation.

While this approach provides a continuous interface velocity, it does nothing to counteract the accumulation of positional error over the course of the simulation. This is because the movement of the level set field is only a function of velocity and not position. While there is a differentiable relationship between position and velocity, this scheme doesn't strictly enforce a constraint on interface position. If the simulation were performed over several cardiac cycles, the later cycles would accumulate additive positional error and deviate more from the targets. The error is most significant in

regions where the interface normal direction changes rapidly. To address the accumulated positional error, the interface is provided a supplemental position correction velocity. This is computed during runtime only on time-steps for which the interface has arrived at the expected target frame. The level set values of cells adjacent to the morphing interface are directly compared to the level set values of the target frame for determining the positional error. This error is divided by the duration of the next frame to compute a mean position correction velocity, $\mathbf{u}_{ic,cv}$.

$$\mathbf{u}_{ic,cv} = \frac{\left(\phi_{ic,1} - \phi_{ic,ft}\right)\mathbf{n}_{ic}}{t_{frame\_duration}} \tag{3.11}$$

Once this value is computed for the entire interface, it too is extended throughout the entire thickness of the level set narrow band. This value is added to the advection velocity until the next frame is encountered modifying the morph advection velocity equation as shown. The correction velocity is set to zero prior to reaching the first frame target.

$$\mathbf{u}_{ic} = \left(1.0 - MOD\left(t, t_{fdur}\right)\right)\bar{\mathbf{u}}_{ic,i} + MOD\left(t, t_{fdur}\right)\bar{\mathbf{u}}_{ic,i+1} + \mathbf{u}_{ic,cv} \tag{3.12}$$

3.6.5 Level Set Morphing: Mass Balance for Volume Changing Interface

The morphing strategy has been developed to provide interface position and interface velocity as to properly interact with the immersed boundary method. The flow solver should be carefully studied to identify any other information the morphing strategy must provide. One such example is the volumetric change of the interface and how it impacts the inflow/outflow boundary conditions. The selected framework utilizes a mass

conserving outlet boundary condition as is common with CFD frameworks simulating internal incompressible flows. A mass conserving Dirichlet condition via convective outflow boundary condition is applied to the outlet as shown below were where $\overline{U}$ is the bulk velocity through the outlet and **n** is a unit vector normal to the outlet.

$$\frac{\partial \mathbf{u}}{\partial t} + \bar{U}\mathbf{n} \cdot \nabla \mathbf{u} \qquad 3.13$$

For simulations with a constant volume interface, this simply denotes the difference between the total inlet flux and the total outlet flux from the domain. However, when performing a left ventricle simulation from patient image data, the change of fluid volume that results from left ventricular contraction and relaxation is also necessary to ensure global conservation of mass. The volume of each fluid cell can be summed to obtain an approximation of the left ventricle fluid volume at each time-step. This generates a discrete representation of fluid volume as a function of time. The change in fluid volume, the derivative, is the value needed to maintain global mass balance. Applying a finite difference scheme to this discrete approximation of fluid volume provides a poor estimate of the change in volume. One solution would be to obtain a more accurate fluid volume at each time-step using marching cubes [65]. Unfortunately, this would be prohibitively expensive. Due to the physiologic assumption of smooth interface velocity, an alternative approach is to apply cubic spline interpolation of the fluid volume computed for each target interface. The derivative of this function provides a smooth approximation of the change in fluid volume.

3.7 Verification Case: Elongating Notched Stadium

A test case has been created with a notched oval geometry as shown in Figure 12a. The starting dimensions of the notched stadium is 1.0 unit wide and 9.0 units long with the notch being located in the center. The sequence of target geometries are constructed by stretching the right end while maintaining the left end location. Thus, the right end of the notched stadium will have the largest displacement and interface velocity. The object centerline used for mapping the warp operation is simply the horizontal plane of symmetry as shown Figure 12b for the first frame. The mapping is constructed in the horizontal direction, similar to the stadium of the previous example, Figure 12c. The notched stadium is placed in a volume of fluid and thus providing a case of external fluid flow. These frames are then morphed in the following sequence to provide an oscillating motion for multiple complete cycles: Frame 1,2,3,4,5,5,5,5,4,3,2,1. Target frame 5 is repeated three times to create a longer dwell time at the fully elongated position.

This simulation is performed twice (with and without warping) to examine the usefulness of adding the warp step to the morphing algorithm. The first simulation only uses the blend morphing strategy as illustrated in Figure 8b. The second morphing strategy utilizes the warp operation as well as the blend as illustrated in Figure 8h. Z vorticity contours provide a good visual indication of a physically realistic interface morph. As the stadium stretches to the right, a linear elastic material would produce a linearly increasing intensity of positive vorticity on the top with linearly increasing intensity of negative vorticity on the bottom. The Z vorticity results generated by these simulations are shown in Figure 13. The vorticity of the blend-only morph strategy of Figure 13a is obviously non-physical. All of the horizontal interface motion of this blend

only strategy is concentrated at the right end and near the center. In fact, due to the non-physical nature of the morphing, the vorticity at the notch is occurring in the opposite direction as the intended motion of the interface. Figure 13b shows the expected results with a linearly increasing Z vorticity from left to the right end of the interface. This test case provide clear visual evidence that the morphing strategy of incorporating the advection velocity using a combined warp then blend approach produces a more accurate interface motion and thus the correct boundary condition is applied to the fluid flow solver.

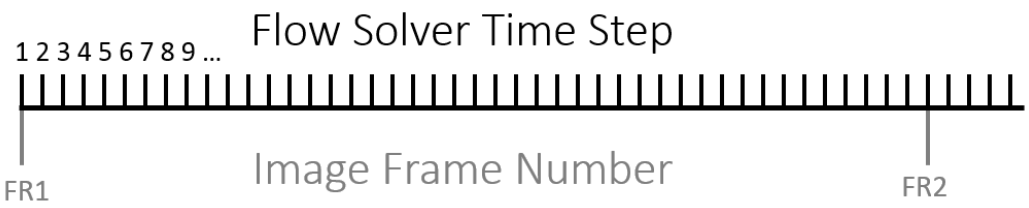


Figure 6: Comparison of flow solver time-step size to image frame duration. The large disparity in time scales necessitates the use of a morphing strategy for intermediate time-steps.

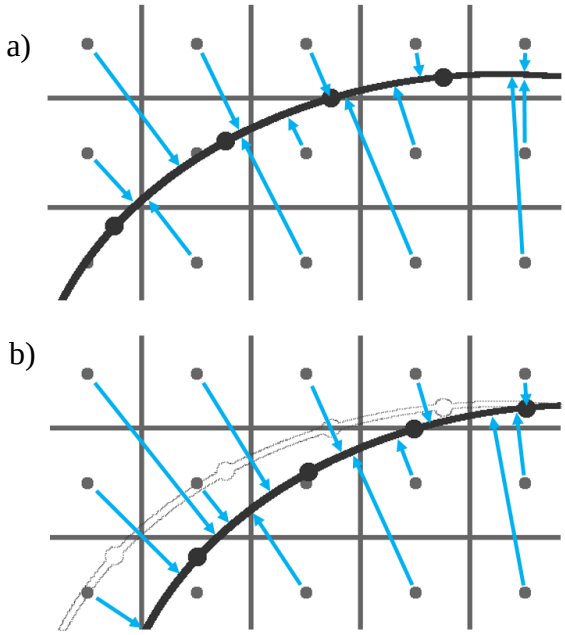


Figure 7: Horizontal interface motion from left to right being tracked across the Cartesian grid using the level set method, a) position 1 (interface left), b) position 2 (interface right).

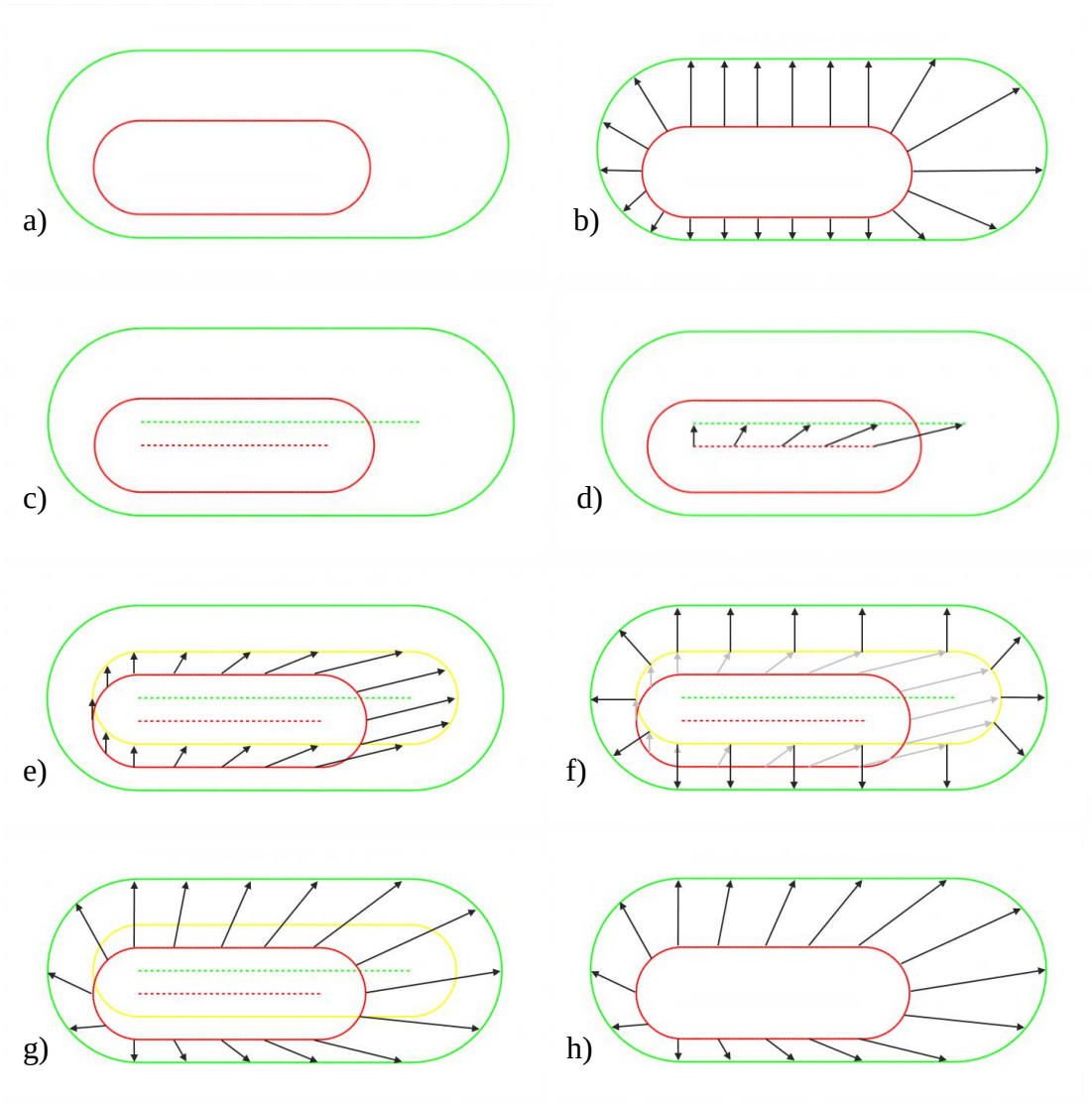


Figure 8: Construction of morph trajectory using a warp then blend approach, a) interface locations, red: starting object location, green: end target location, b) results of neglecting the warp feature by only using a blend operation, note the unrealistic distribution of stretch in the horizontal portions of the interface, c) centerline locations of object and target to be used for warp operation, d) linear mapping of centerline displacement and stretch, e) application of the centerline displacement to the object interface to achieve the intermediate location (yellow) of warp operation, f) construction of the blend displacement from the intermediate location to the target location, g) summation of the warp and blend displacements, h) final warp trajectory.

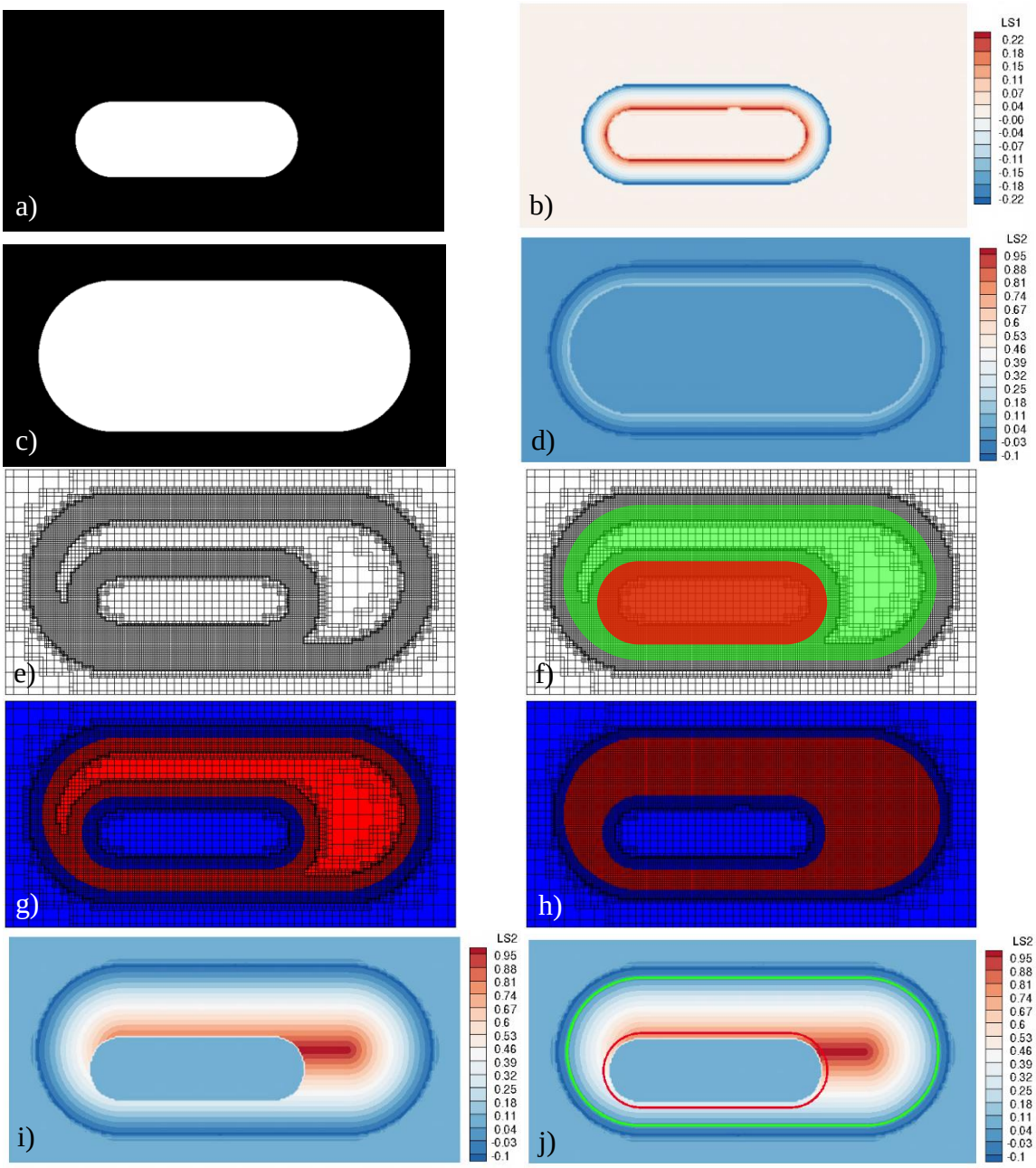


Figure 9: Domain modifications necessary to construct the morph trajectory using a pELAFINT3D, a) image of object initial position, b) narrow band level set field of initial position, c) image of target position, d) narrow band level set field of target position, e) domain mesh refinement for both the interface and target, f) regions inside both the starting interface and target interface found by solving a Laplace equation for each, g) region over which the initial interface must cross to reach the target location (morph zone), h) mesh refinement of all cells located in the morph zone, i) expanded target level set field to include morph zone, j) expanded target level set field with interface locations.

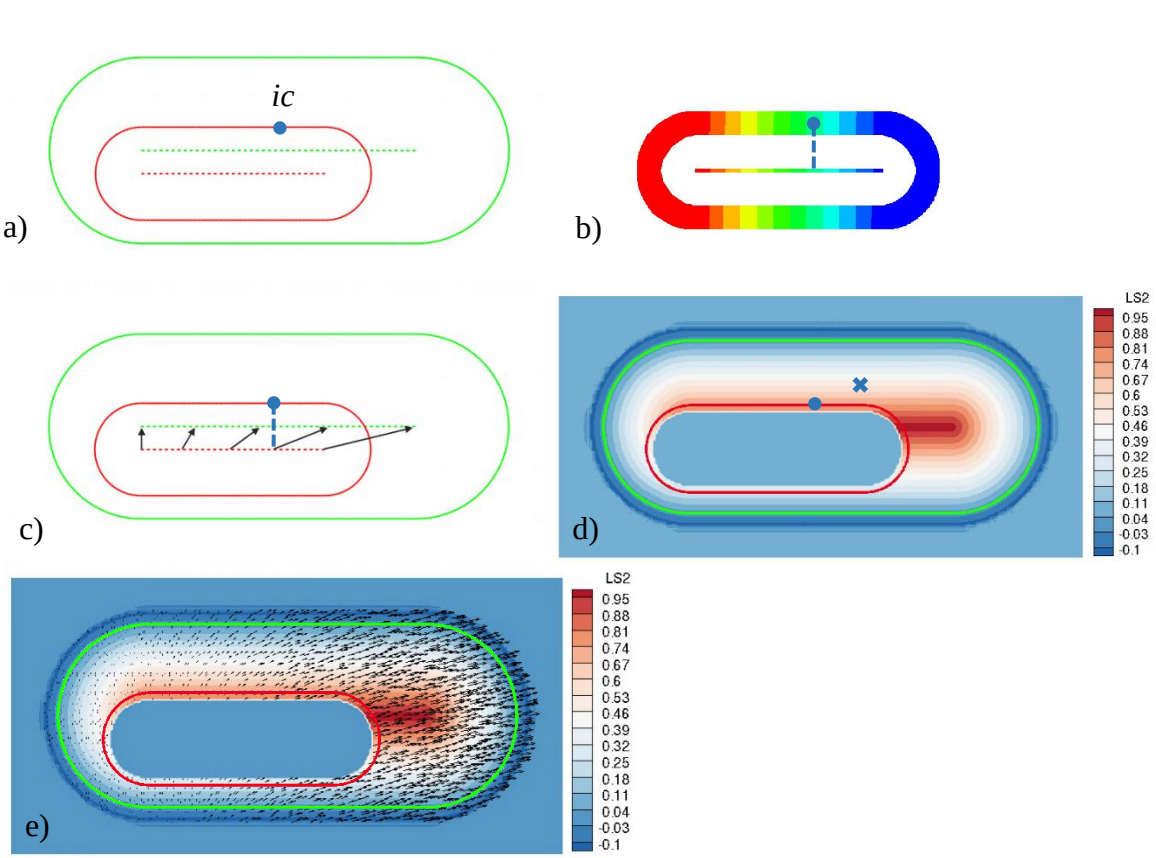


Figure 10: Construction of mean advection velocity for a single morph interval, a) example interface point for which to construct velocity, b) use mapping to identify corresponding location on the centerline representation, c) compute centerline displacement at location from the warp operation, d) centerline displacement is used to offset the initial interface location to compute blend displacement, e) extended combination of blend and warp displacements throughout the thickness of the level set tube.

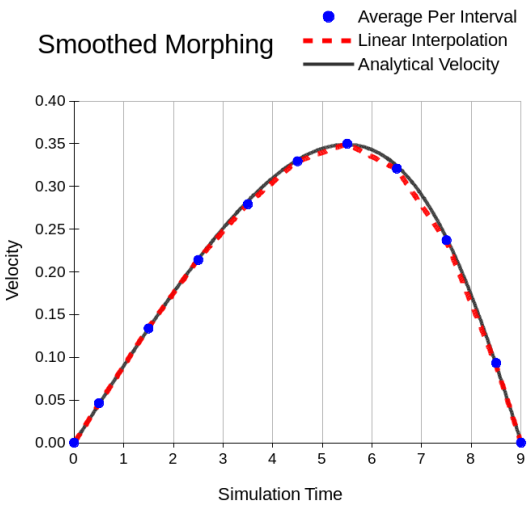


Figure 11: Morph interface velocity computed using interpolation of mean interface velocities with a position correction velocity.

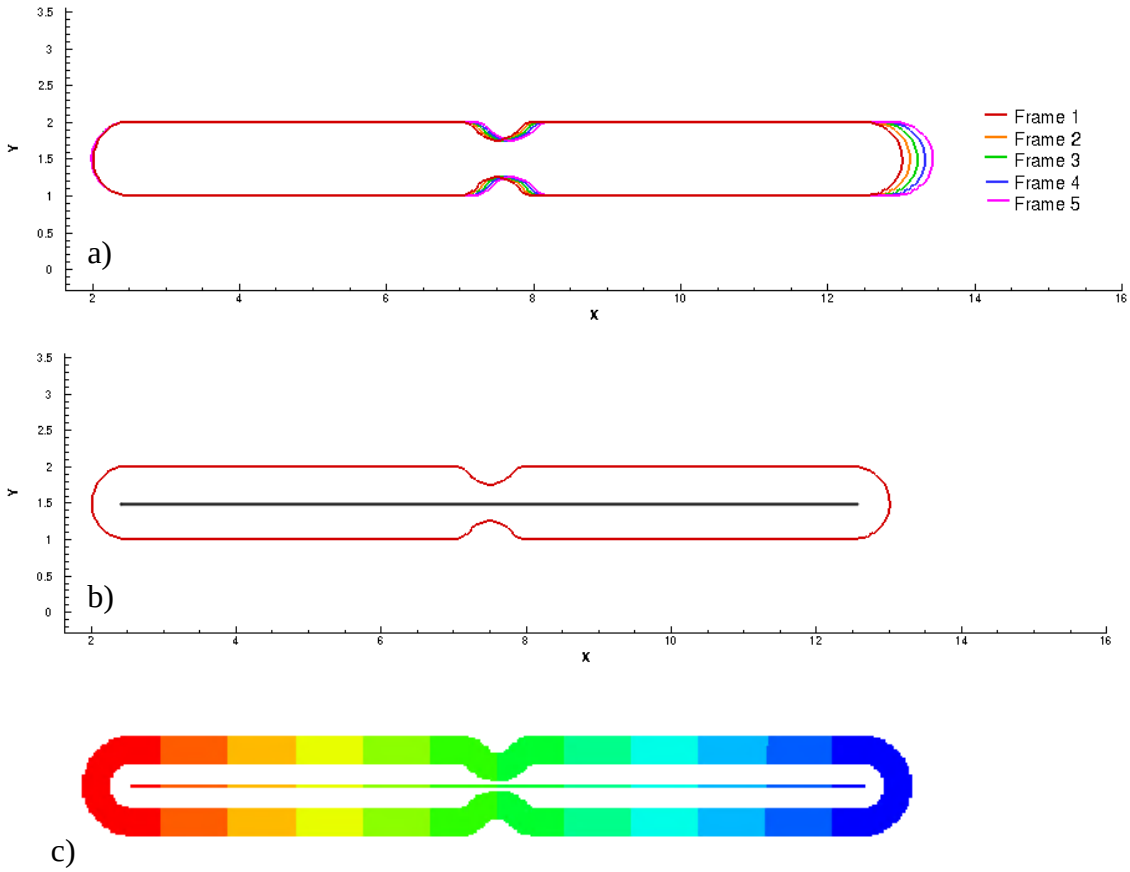


Figure 12: Stretching morph tube immersed in a volume of fluid, a) progression of starting and target interfaces defined for the morph tube verification case, b) frame 1 centerline representation utilized for the warp operation, c) frame 1 mapping of cells in level set tube to the centerline representation.

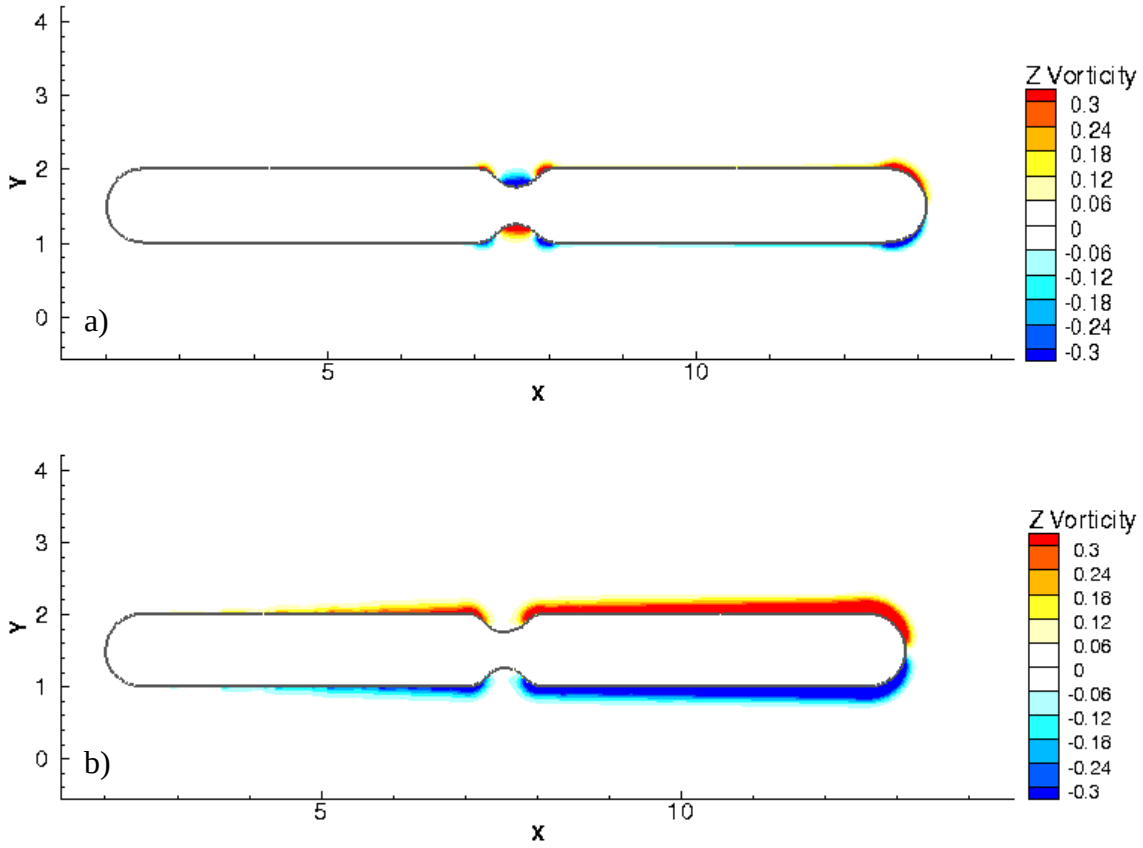


Figure 13: Elongating notched stadium at frame 2 showing the difference in vorticity generated between, a) blend-only implementation of morphing, b) warp-and-blend implementation.

CHAPTER 4 SKELETONIZATION

For simple geometry such as the morph tube described above, defining a low order representation (centerline) of the geometry is obvious. Cardiovascular anatomy is much more complicated and tortuous. A much more robust method must be implemented to automatically compute this representation of the vessels. Due to the tubular shape of cardiovascular vessels, skeletonization has been selected as the means for doing so. The concept of skeletonization was first introduced by Blum 1967 [66]. Skeletonization has been referred to as a compact representation of an object by reducing the dimensionality to a medial axis while preserving the topologic and geometric properties [67]. This can also be thought of as a low order resolution, or the representation of an object which contains less geometric information, while preserving the general shape and structure of an object.

Since there are many methods for constructing a centerline skeleton, the current context must first be considered to constrain the method selection process. Since the goal of this work is to aid the simulation of patient-specific heart valve operation, the solution must be valid for 3D geometry. This eliminates methods developed for 2D space [68] or achieve a skeleton by thresholding a field value, which in 3D space form medial surfaces rather than a singular curve [69]. For example, thresholding the gradient magnitude of the level set field can be used in 2D space to define a curve representing the center of the vessel, but the 3D application results in a medial surface.

Since the level set field describes the interface location with sub-grid resolution, it is desirable to implement a skeletonization method which also allows for a sub-grid

positional accuracy of the skeleton. This eliminates methods which are discretized at the voxel or computational grid size resolution.

The current framework is developed and implemented to be parallel and scalable. This becomes very important in 3D simulations with a large fluid domain such as the left ventricle. As previously mentioned, this parallelization is accomplished via domain decomposition for which each process contains a specific region of the domain with a copy of the boundary layer of cells and their values from adjacent domain partitions. Any skeletonization method under consideration should have a parallelization scheme that is compatible with the currently implemented domain partitioning scheme.

4.1 Selection from Available Methods

Skeletonization is useful for a wide variety of applications including computer vision and image processing [67]. A wide variety of skeletonization methods have been developed with differing properties and advantages. Characteristics which distinguish the usefulness of a specific method include sub-voxel resolution, continuity, topology preservation, and parallelization. To select the appropriate choice, it is helpful to characterize the existing methods into three main categories: Voronoi based, erosion or thinning, and field evolution.

The first category of skeletonization methods utilizes a Voronoi diagram, which is the partitioning of a domain into regions of closest proximity to a cloud of points. This group of methods use the Voronoi-generated features of symmetry to construct the skeleton [70]–[73]. While the work of Naf et al. [71] extended this approach to 3D space, this group of methods is disadvantageous for the proposed application due to the representation of the object interface. The use of an Eulerian level set method to describe

the interface means that there is not direct access to a Lagrangian point cloud needed to construct the Voronoi diagram.

The second set of skeletonization methods to consider is based on morphological erosion or thinning [74]–[77]. These algorithms operate by removing a set of surface voxels in an iterative manner until only a single contiguous string of voxels is selected to represent the skeleton. While this approach has been extended to include 3D space and parallelization [78], [79], it does not offer sub-voxel accuracy. Since the geometry for level set morphing is represented by an Eulerian field at a resolution significantly finer than the coarse, voxel level resolution, this would limit the accuracy of the skeleton and hence would not provide a continuous mean velocity field as computed using Equations 3.7 and 3.9.

The third and final grouping of skeletonization methods consists of those which evolve continuous curves or fields [80]–[83]. One such method of interest from this group is the work of Hassouna et al [84], [85]. This method operates by utilizing a level set representation of the geometry. A wave equation is propagated from a source location with a front speed that is dependent on the level set value, such that the front will travel faster nearer the center of the vessel. A cost minimization algorithm is then used to march a path through the maximum curvature locations of the wave front. The wave is modeled as an Eikonal equation, a PDE, which can easily be solved on the existing computational framework utilizing the same domain partitioning strategy. Finally, the cost minimized path creates a curve in 3D space [85]. While this method was primarily developed for image processing, with some minor modifications (described below), it can be applied to an immersed boundary FSI framework.

4.2 Implementation Details

4.2.1 Skeletonization: Required Computational Domain Modifications

To perform skeletonization using the level set field, the computational domain must be temporarily modified. To illustrate these domain modifications, consider the CT slice displaying the long axis view of the left ventricle shown in Figure 14. The first modification pertains to the automatic Local Mesh Refinement (LMR) [26], [33]. LMR reduces the computational burden of the flow simulation by allowing computational cells of varying sizes. This is advantageous as the regions of high velocity gradients need more cells to accurately resolve the flow, and regions of small velocity gradients can use far fewer cells. Cells near the interface are always refined to the smallest allowed size for accurate level set representation and because these are typically regions of high velocity gradients due to the no-slip boundary condition. The cells in the interior of the fluid region are dynamically refined based on local velocity gradients, as seen in Figure 14b. This mesh refinement scheme is advantageous to the flow solver, but introduces an obstacle when constructing an accurate skeleton, since a higher-resolution skeleton would be achieved if the mesh has maximum refinement at the center of the vessel, where cells representing the skeleton will be obtained. For this reason, the computational refinement is temporarily extended to the entire vessel to construct the skeleton, Figure 14c. Performing the entire simulation at this refinement would completely eliminate the efficiency gains of LMR. Therefore, the domain modifications will be temporary and a skeleton will only be constructed at simulation start, one skeleton for each of the target level sets created from input image frames. A morphing operation must then be performed on the skeleton to determine intermediate locations. After the mesh

refinement is extended throughout the vessel, the level set values must be extended. For computational efficiency the level set values are typically defined only for a narrow band around the interface as shown in Figure 14d. With this configuration, there is no means of determining the distance to the interface for cells near the center of the vessel. To resolve this issue, the values are temporarily extended beyond the narrow band to fill the entire vessel, Figure 14e. Being the farthest from the interfaces, the vessel center will have the largest level set value which then decreases radially towards 0.0 at the vessel wall.

4.2.2 Skeletonization: Implementation

As previously mentioned, the approach implemented here is a variation on the method developed by Hassouna et al. [85]. To illustrate the process, consider a 2D slice through the heart with the left atrium in the lower-left, left ventricle to the right, and the ascending aorta to the upper-left as shown in Figure 15a. The first thing to determine is the wave source location. This will be the point from which all major branches will converge. A consistent source location must be selected as to not impose unphysiological displacement of the skeleton branching as it would inaccurately represent the topological changes of the geometry during the warp operation of the morph process. Heart anatomy offers a good reference to select a source location. Since the fibrous skeleton of the heart is a fairly rigid structure to which the valves are attached, it provides a good reference for which to geometrically construct a consistent wave source position. The location of the fibrous skeleton, through the use of valve annulus identification techniques described in Chapter 2, should already have been determined for the entire cardiac cycle. In Figure 15a, the green circles indicate the annulus locations from which the position of the source

(red circle) is constructed. Once the source location is defined, a low speed wave is propagated from that source location to the entirety of the fluid domain. This is accomplished by solving the following partial differential equation and applying a Neumann boundary condition at all interfaces. As the front propagates away from the interface, the time to reach each cell, T, is recorded as seen in Figure 15b. This information can then be used to determine the shape of the skeleton tree.

$$\frac{\partial T}{\partial t} + |\nabla T| = 0 \qquad 4.1$$

Wave time values assigned to each cell are used to sort the cells into a discrete number of bins as shown in Figure 15c. Data binning is the act of reducing a range of values to a finite set of intervals for which all data points are sorted. The number of bins should be selected such that too few bins will fail to capture the correct tree structure, and too many bins will cause extra computational burden in subsequent steps of the solution process. For this particular case, after evaluating several choices 9 bins were selected to achieve a good balance between computational efficiency and correctly capturing the tree structure. Once every cell is sorted into the appropriate bin, those bins can be subdivided into contiguous regions, or clusters, as shown in Figure 15d. This is accomplished by performing successive flood fill operations in each bin, until all cells of that particular bin belong to a cluster. As indicated by the faint white lines representing the domain partition boundaries, the flood fill operations must be performed using a parallel algorithm which may require a sizable amount of communication. Next, each cluster is searched to identify neighboring clusters in adjacent bins. Endpoint clusters will only have a single neighbor. The number of endpoints will depend on the number of branches

the skeleton has. These endpoint clusters are then one-by-one linked back to the cluster containing the source location. The results of this tree mapping operation are displayed in Figure 15e. Now that the tree structure of the skeleton is known, a field must be generated to guide the specific path of the skeleton. To do this another wave equation must be solved, but this time the speed of the wave front is heavily dependent on the distance from the interface. The speed of the front on cells far from the interface, with large level set values, will be fast while the speed of the front on cells near the interface (i.e. those with small level set values) will be low. This is captured by solving the partial differential equation below with a Neumann boundary condition on the interface. The time for the front to reach the cells is again the output as shown in Figure 15f.

$$\frac{\partial T}{\partial t} + [\phi(\mathbf{x})]^{\beta} |\nabla T| = 0 \qquad 4.2$$

Locations of large iso-contour curvature already give a visual indication of the skeleton path. In fact, the gradient of this field, Figure 15g, is the primary input for minimization. Each endpoint cluster, as previously identified, will be a starting point for the path minimization. Equation 4.3 below is solved using a 2nd order Runge-Kutta implementation. The solution is marched forward in space at a user-defined increment, where the selection of this increment will determine the resolution of the skeleton output. This implementation uses an increment step size of ¼ the minimum cells size providing a sub-grid resolution to the skeleton which matches the sub-grid resolution of the level set field. Termination of the path operation is conducted when the wave source location is reached. This process is repeated for each of the skeleton endpoints creating a skeleton

segment for each of the branches with an endpoint at one end and the source location at the other.

$$\frac{\mathrm{d}\mathbf{C}}{\mathrm{d}t} = \frac{\nabla T}{|\nabla T|} \ , \mathbf{C}(0) = \mathbf{x}_{endpoint} \qquad 4.3$$

Figure 15h shows the final results for this specific frame of the image sequence. This skeletonization procedure is repeated for every frame in the image sequence. Labeling of the three skeleton segments must be consistent between all frames to ensure successful morphing. Segments are matching using a Euclidean distance metric to determine the nearest distance of the segment endpoints to the frame just prior. The results for selected frames are shown in Figure 16. Upon completion of this process the domain is returned to the previous level set tube width and the mesh is coarsened to the previous degree of refinement as shown in Figure 14b and d, respectively.

4.2.3 Skeletonization: Mapping the Level Set to the Skeleton

Even though the skeleton has been constructed, it still isn't ready for use during the warp step of the morphing process. While the mapping of cells in the level set tube to a location on the skeleton is self-evident for simple shapes, such as the oval previously considered in Figure 10b, this is certainly not the case for many complex and tortuous geometries found in the human cardiovascular system. Thus it is necessary to define a way to map the cells in the level set tube to a reference location on the skeleton.

To illustrate the proposed method, consider the 3D geometry shown in Figure 17a. This is a 3D reconstruction of a left ventricle with the inlet from the left atrium (upper-left) and the outlet to the ascending aorta (upper-right). The result of

skeletonization is also shown with the orange segment representing the left ventricle region, the green segment representing the left atrium, and the blue segment serving as a skeleton for the ascending aorta. To begin mapping interface points to their respective location on the skeleton, first consider computing a simple Euclidean distance metric for every cell in the level set tube to the nearest location on the nearest skeleton segment. The results on the iso-contour of zero level set value is shown in Figure 17b with the contour values representing the arc length percentage of the skeleton onto which the level set cell is being mapped. To inspect the utility of this metric, consider the orange skeleton segment located in the left ventricle. It is apparent from this figure that the cells in the level set tube are not using the entirety of the left ventricle skeleton segment (orange). The usage varies from approximately 60 percent on the right to nearly 90 percent on the left. This underutilization is also apparent when considering the boundary between the left atrium region (green skeleton segment) and the ascending aorta region (blue skeleton segment). Figure 17c shows the major issue with using this approach. Consider two adjacent cells located on the interface marked as green and blue circles. The nearest location on each respective skeleton segment are shown with matching green and blue circles. In both cases the corresponding location on the skeleton segment is approximately 80 percent of the length of the skeleton segment. Recall that when computing the mean advection velocity, as shown in Figure 10c, the mapped skeleton location is used to compute the warp displacement. Consequently, any relative displacement between the two mapped skeleton locations will cause an abrupt change in warp displacement. The end results will be a discontinuous mean advection velocity at the interface.

To resolve this issue a method must be developed as to utilize the entirety of each skeleton segment. The Euclidean distance metric can still be used, but instead it is employed as a means of apportioning all the cells in the level set tube to a specific skeleton segment. An iso-contour surface at the zero level set shows the regions associated with each skeleton segment, as displayed in Figure 17d. All cells in the level set tube are then searched for neighbor cells which are associated with a different skeleton segment. These cells constitute the boundary between mapping regions. To prevent the discontinuity in advection velocity, these cells should be mapped to the source point which is located at 100 percent of the arc length of all three skeleton segments. The mapping variable for these cells is set to 1.0. At the far end of each skeleton segment, there exists a hemispherical shaped region of cells which all map to the single endpoint of each skeleton segment. A value of 0.0 is given to these cells indicating they map to 0 percent of the arc length. These values can be used as boundary conditions to compute a smooth distribution of cells being mapped to the entire length of each skeleton segment. Figure 17e shows the known Dirichlet mapping values on the iso-contour surface at the zero level set. A Laplace equation has been selected due to its elliptical characteristics. For the edge of the tube, a Neumann boundary condition of value 0.0 is applied.

$$\nabla^2 m = 0\,, m_{Bdry} = 1.0, m_{Endpoint} = 0.0, \frac{\mathrm{d}m}{\mathrm{d}t}_{TubeEdge} = 0.0 \qquad 4.4$$

For faster convergence, the field is initialized with the Euclidean distance measurements shown in Figure 17b. The final level set tube mapping results are shown in Figure 17f. The contour plot clearly shows the tube-to-skeleton mapping utilizes the

entire length of all three skeleton segments, thus ensuring that a continuous advection velocity is applied to the interface.

4.3 Skeleton Verification: 3D Y Skeleton

To verify the accuracy of the skeletonization algorithm, a test case can been created in the shape of an idealized bifurcation. The Y Skeleton model was constructed using the 3D CAD modeling software Creo Parametric, version 4.0 (PTC, Needham, MA). Using such a means of construction, provides the exact geometric definition of the centerline which can be compared with the results of the proposed skeletonization method described above. The dimensions used to construct the geometry are shown in Figure 18. This 3D geometry was intentionally constructed to be out-of-plane, to test the implementation with a geometry that occupies more than a single plane. The right leg of the Y Skeleton model is angled out-of-plane at 30 degrees. The diameter of all legs is 1.0.

Skeletonization was performed on the Y skeleton at four different mesh sizes to determine the accuracy and impact of grid resolution. The coarsest mesh size was 0.1 and was cut in half one, two, and three times to achieve grid spacing of 0.05, 0.025, and 0.0125, respectively. Since the diameter of the Y Skeleton legs is 1.0, the approximate number of cells across the diameter was thus 10, 20, 40, and 80.

A visual representation of the results can be seen in Figure 19, computed using the smallest computational cell size. The results qualitatively match very well with the different colors indicating the three different segments or branches detected by the skeletonization algorithm. To quantitatively assess the accuracy of the method, each point of the computed skeleton was compared against the known centerline geometry of

the CAD model. The results are shown in the graph of Figure 20 for each of the computational grid sizes. In all four cases the mean error is shown to be less than half the computational grid size. Inspection of the graph shows a nearly linear relationship between the grid size and the positional error. Figure 21 displays a graph showing the maximum error of skeletonization for each of the four mesh resolutions. Maximum error also appears to have a linear relationship to the computational grid size with the maximum error being approximately equal to the grid size. Location of the maximum error indicates the possibility that minor features of the CAD geometry construction may have contributed to the larger error. The maximum error for all four cases is located on the arcs connecting the three legs. Skeletonization placed the points off the arcs in the medial direction. When creating the CAD model a large blend was added between the two top legs. This may have shifted the symmetry of the model and thus magnified the maximum positional error.

While the mesh resolution clearly impacts the accuracy, the Y Skeleton verification case has demonstrated that the skeletonization method developed and implemented can effectively measure and represent out-of-plane, 3D geometries. The sub-grid accuracy is in line with the sub-grid accuracy of the level set field and thus can achieve a continuous mean morph velocity field needed to advect the interface.

4.4 Verification of Skeleton-Based Morphing: Swimming Eel

Next, the complete skeleton-based level set morphing approach can be verified using a benchmark test case. Tytell et al. investigated the anguilliform fluid dynamics of a swimming American eel. 2D images were captured of the eel swimming while pinned to an experimental tunnel apparatus [86]. 36 video frames were captured for one full

cycle of the tail swimming motion (total time = 0.32s). Select frames from the video sequence are shown in Figure 22. This case presents a good verification opportunity of the full interface morphing strategy developed thus far. Not only is this a biological application in which a large-displacement moving interface is driving the fluid flow, but also because this has been previously simulated using an alternative computational approach [63].

Dillard et al. performed this simulation utilizing a computational framework built around the optical flow concept [63]. Active contours was utilized to independently segment all 36 video frames [63]. While there are numerous disadvantages of the Dillard scheme, it was still successful in generating flow physics which agreed with experimental results. For this reason, the computational results of Dillard et al. will be used for verification of the currently proposed method of level set morphing using skeletons.

Dillard et al. non-dimensionalized the simulation using the eel body length as the characteristic length, L, and placed it in a fluid domain of 5L by 2L. The peak-to-peak amplitude, A, of the tail displacement was approximately 0.7L. The eel length and amplitude are labeled on the first frame of Figure 22. Adaptive mesh refinement was utilized with the minimum grid size being 0.0025. The free stream velocity, U, and the kinematic viscosity were set to achieve a Reynolds number, Re, of 5000. The time scale of the image sequence was thus sets to achieve three different tail beat frequencies, resulting in three different Strouhal numbers, St, of 0.3, 0.5, and 0.7. The Strouhal number, Equation 4.5, is used to characterize the non-dimensional vortex shedding phenomenon of objects in a free stream flow.

$$St = \frac{fA}{U} \qquad 4.5$$

While the Dillard et al. simulation was performed at a lower Reynolds number than the experiment, computational flow results showed good agreement with the particle image velocimetry results of Tytell et al. Vorticity plots showing the structure and frequency of the vortex shedding computed by Dillard are included in Figure 23. As expected, the lower Strouhal number simulations (St = 0.3, St =0.5) produced a 2s-type vortex formation while the higher Strouhal simulation (St = 0.7) starts to show progression towards a 2p-type vortex formation.

In the current work, the same raw video frames were obtained to perform simulations using the proposed method of level set morphing, and compare with previous results. Each image was independently segmented using 3D CAD modeling software Creo. The domain and flow parameters were set to replicate the conditions used by Dillard et al. As necessary for the proposed level set morphing algorithm, skeletonization of each frame was performed on start-up. The computed skeleton results are shown for select frames in Figure 24. The skeletonization algorithm appears to have achieved frame-to-frame consistency and appears to mimic the vertebral structure of the eel. The flow simulation was performed for a full 12 cycles and the computed vorticity plots, shown in Figure 25, were used for verification. For the lower Strouhal numbers of St = 0.3 and St = 0.5, the vorticity results are in excellent agreement with the vorticity shown in Figure 23. While still very similar, the vortical structures of the two solutions deviate slightly at St = 0.7. The results from the currently developed method of level set morphing using skeletons shows what appear to be more-developed 2p-type vortex

structures. It is difficult to know which simulation is more accurate without having experimental results at the Reynolds number of 5000 for which these simulations were performed (recall that the experimental results of Tytell were obtained at Re = 56,000).

One other difference to note is the presence of noise or oscillations in the vortex field near the tip of the tail in the Dillard et al. results. Since these flow field disturbances are not present in the current results, it is likely that the noise arises from one of two sources. The first could be due to an incorrect morph path in Dillard et al., causing a spurious shortening and lengthening of the tail during each morph increment, creating a tip oscillation artifact. The second possible source could be from a discrepancy between the computed interface velocity and the interface position, obtained from and over-constrained interface (described above). Despite these slight discrepancies, this simulation has demonstrated positive verification of the developed approach using skeleton-based level set morphing in a biological application.

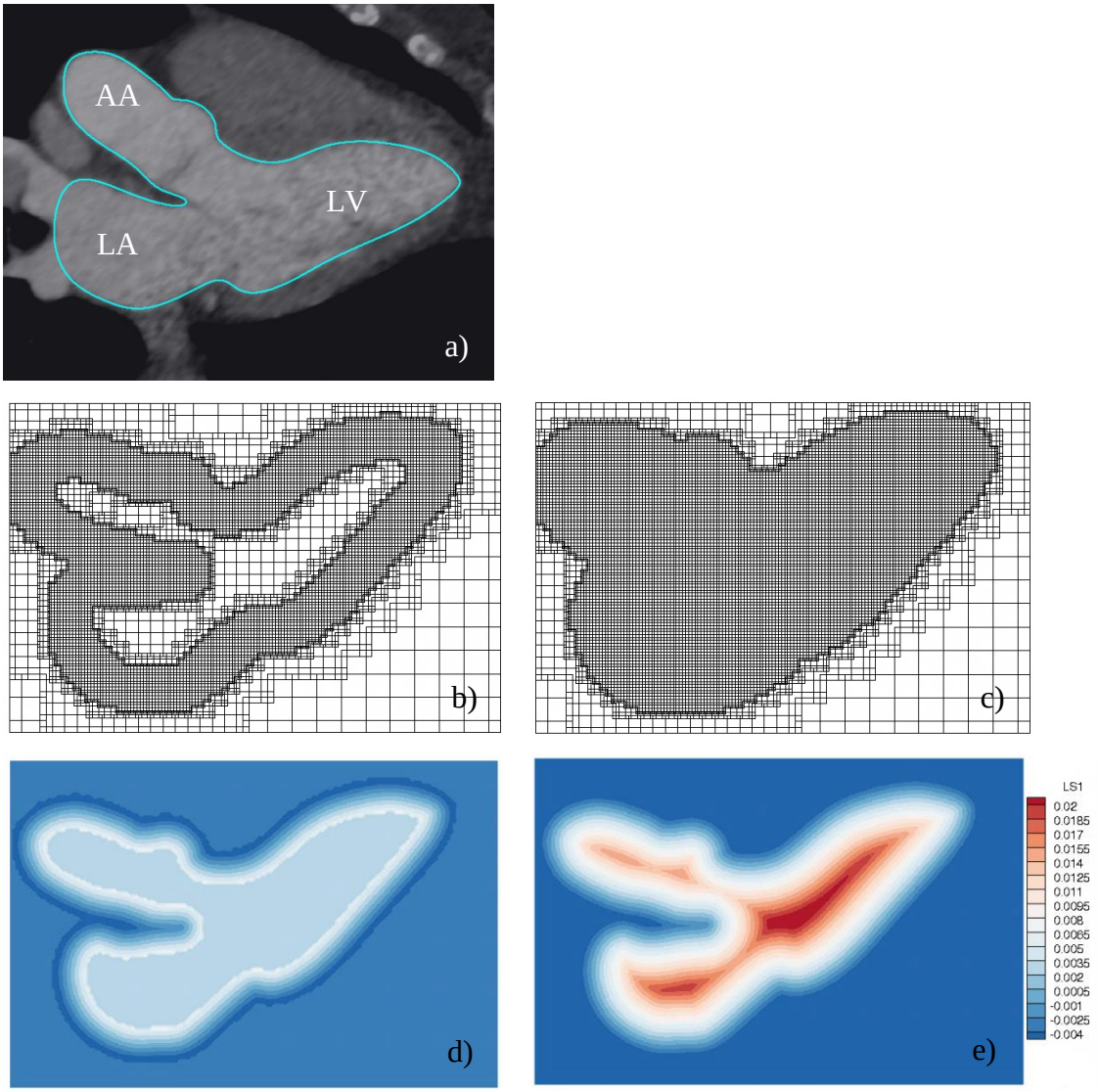


Figure 14: Temporary domain modifications required for skeletonization, a) segmented CT image of left atrium (LA), left ventricle (LV), and ascending aorta (AA), b) default mesh refinement c) entire endocardial region refined for skeletonization, c) typical size of level set narrow band, d) level set values extended to the whole volume for skeletonization.

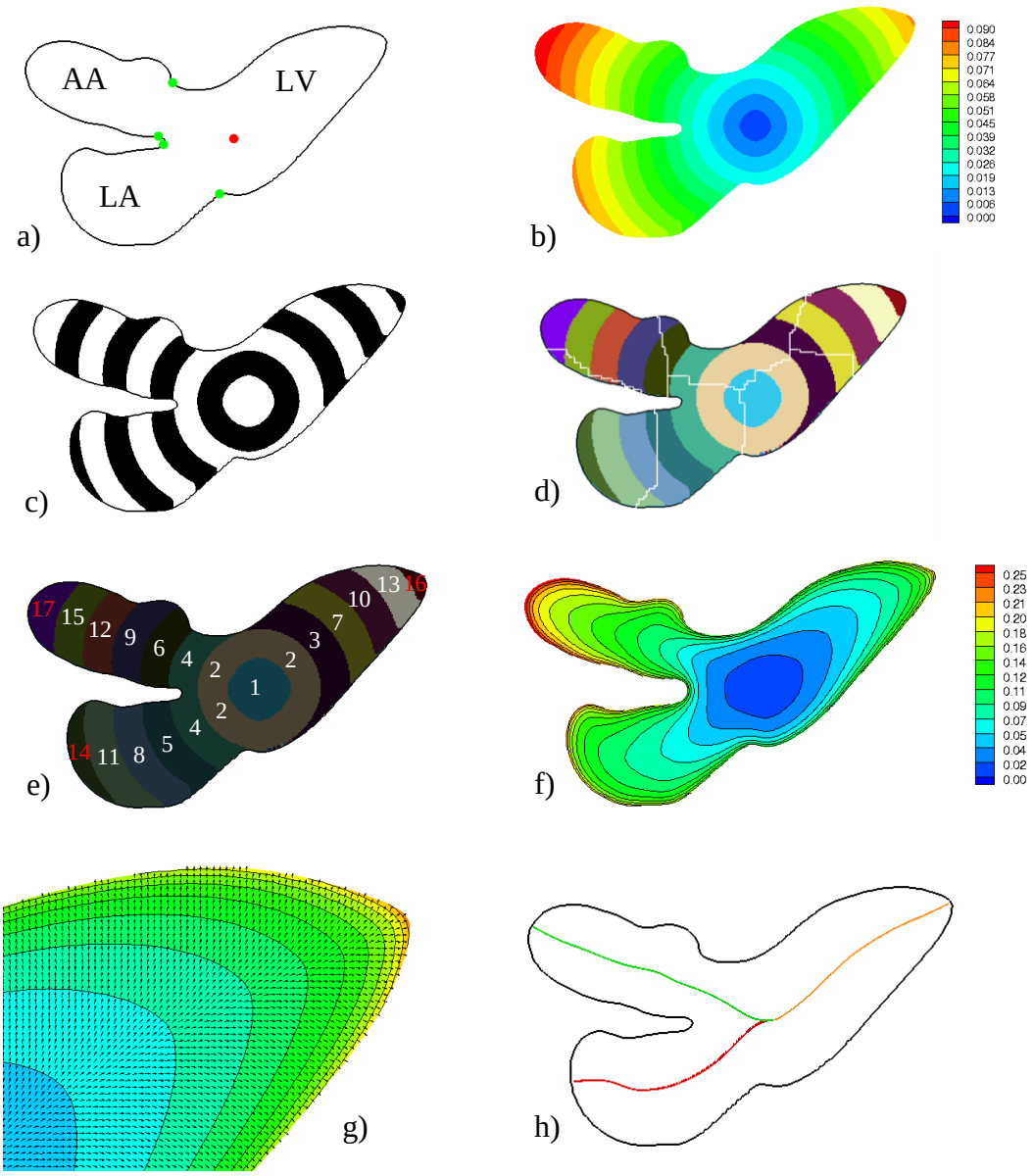


Figure 15: Skeletonization utilizing the level set field, a) wave source (red) constructed from the annulus locations (green), b) results of low speed wave propagation, c) low speed wave results are sorted into a discrete number of bins, d) successive flood fills conducted in each bin to define contiguous clusters, e) skeleton tree map with endpoint clusters in red, f) results of high speed wave propagation, g) gradient of high speed wave results used to minimize the cost path of the centerline, f) skeleton results for a single frame.

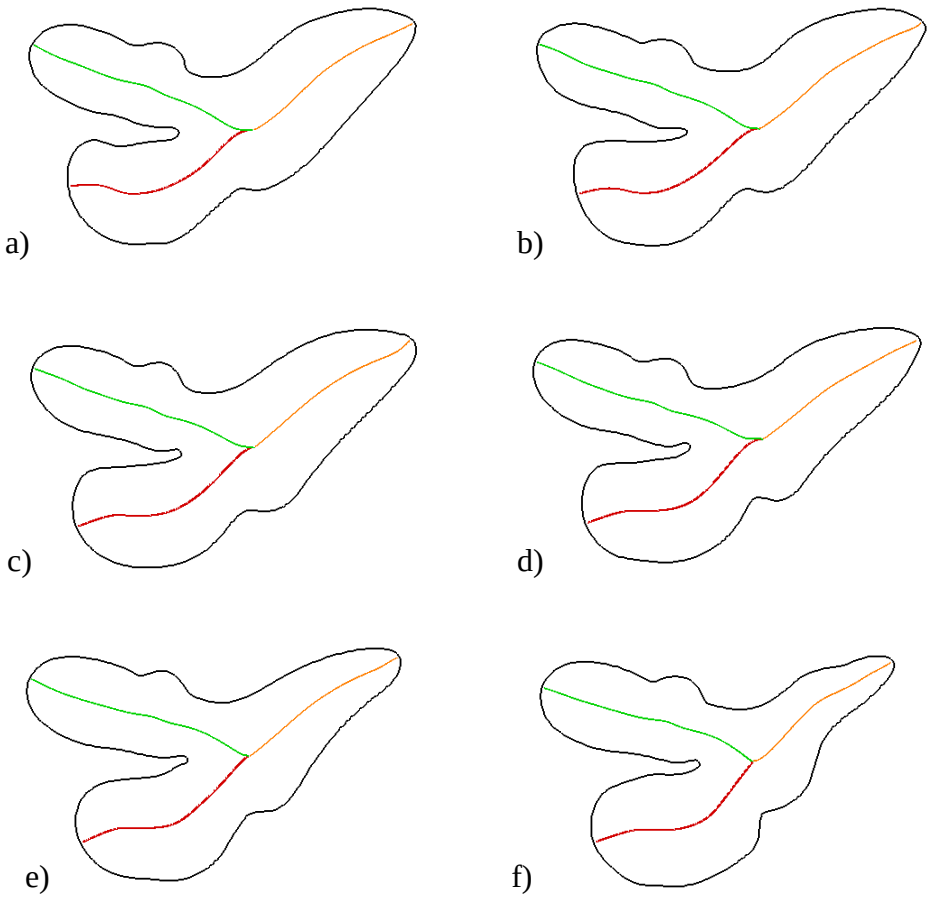


Figure 16: Skeletonization results from select frames of a cardiac cycle imaged by CT with consistent numbering of skeleton segments.

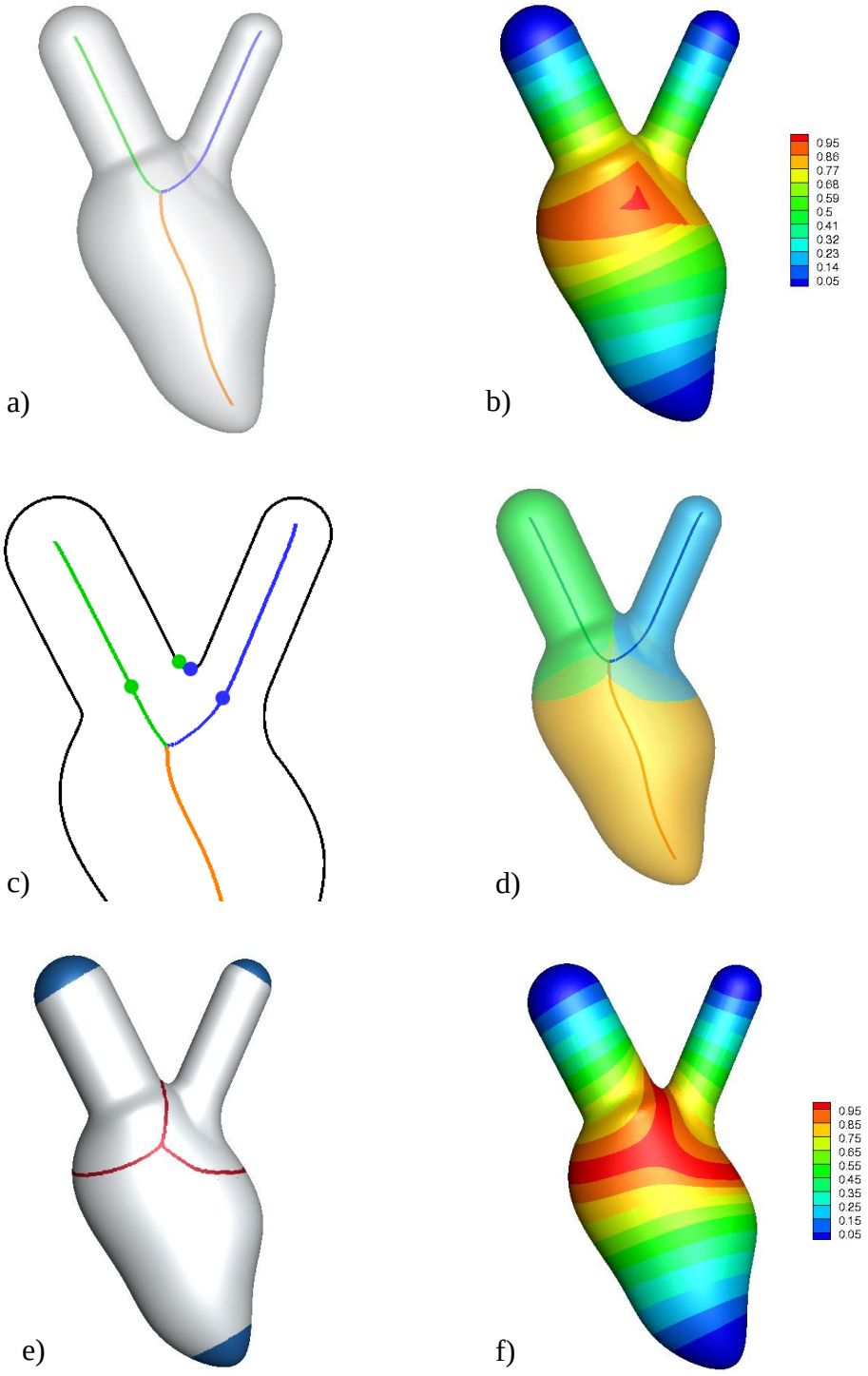


Figure 17: Mapping the level set field to the skeleton, a) skeleton generated for a 3D left ventricle reconstruction, b) Euclidean mapping of the interface to the nearest point on the skeleton, c) Adjacent locations on the interface mapped to distant points on the skeleton, d) Euclidean mapping used to apportion the interface to each skeleton segment, e) identification of interface cells at the boundary of skeleton apportioning or skeleton endpoints, f) results of skeleton mapping achieved by solving a Laplace equation.

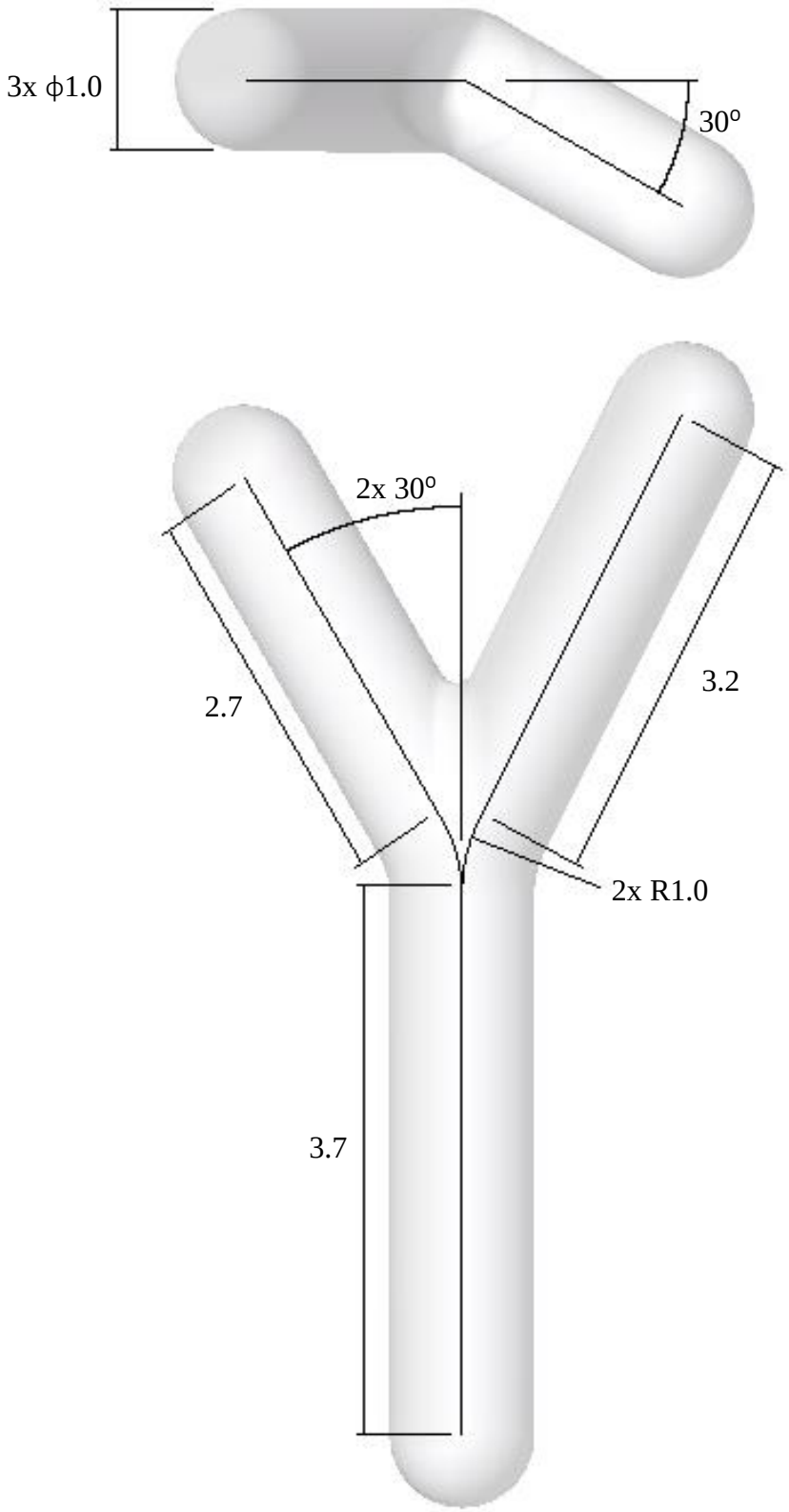


Figure 18: Y Skeleton 3D verification geometry with all pertinent dimensions shown.

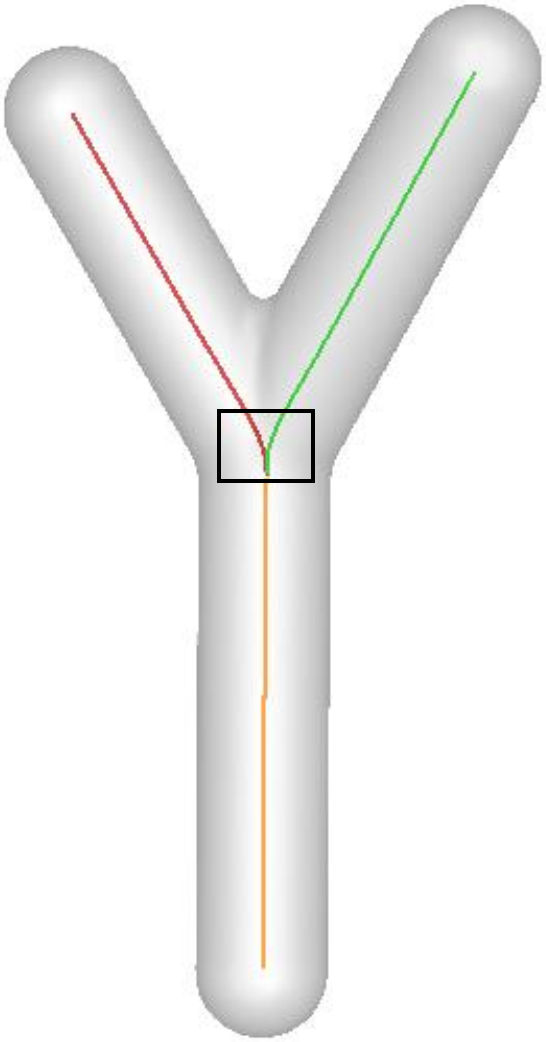


Figure 19: Y Skeleton verification results at computational cell size of 0.0125 using the implemented method for skeletonization with colors indicating the different segments/branches detected. The black box indicates the region of largest positional error.

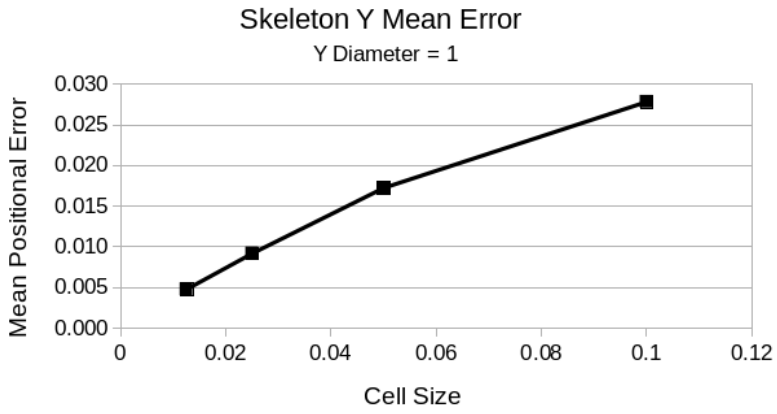


Figure 20: Quantitative verification of Y Skeleton geometry considering the mean positional error at four mesh resolutions indicating a linear relationship between the mean error and the computation grid size.

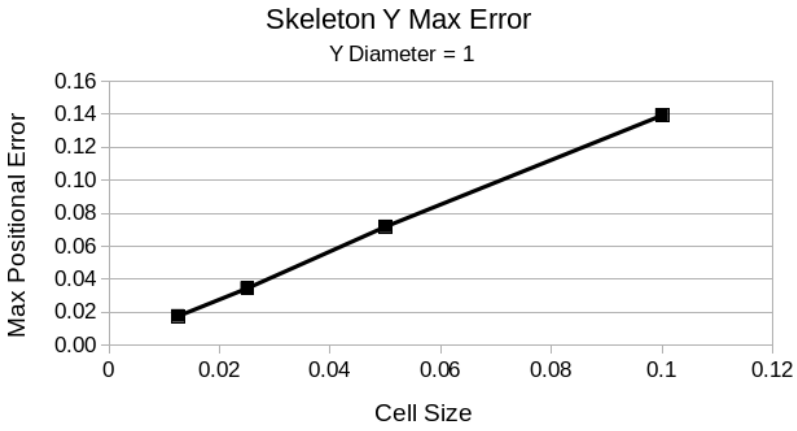


Figure 21: Quantitative verification of Y Skeleton geometry considering the maximum positional error at four mesh resolutions indicating a linear relationship between the maximum error and computational grid size.

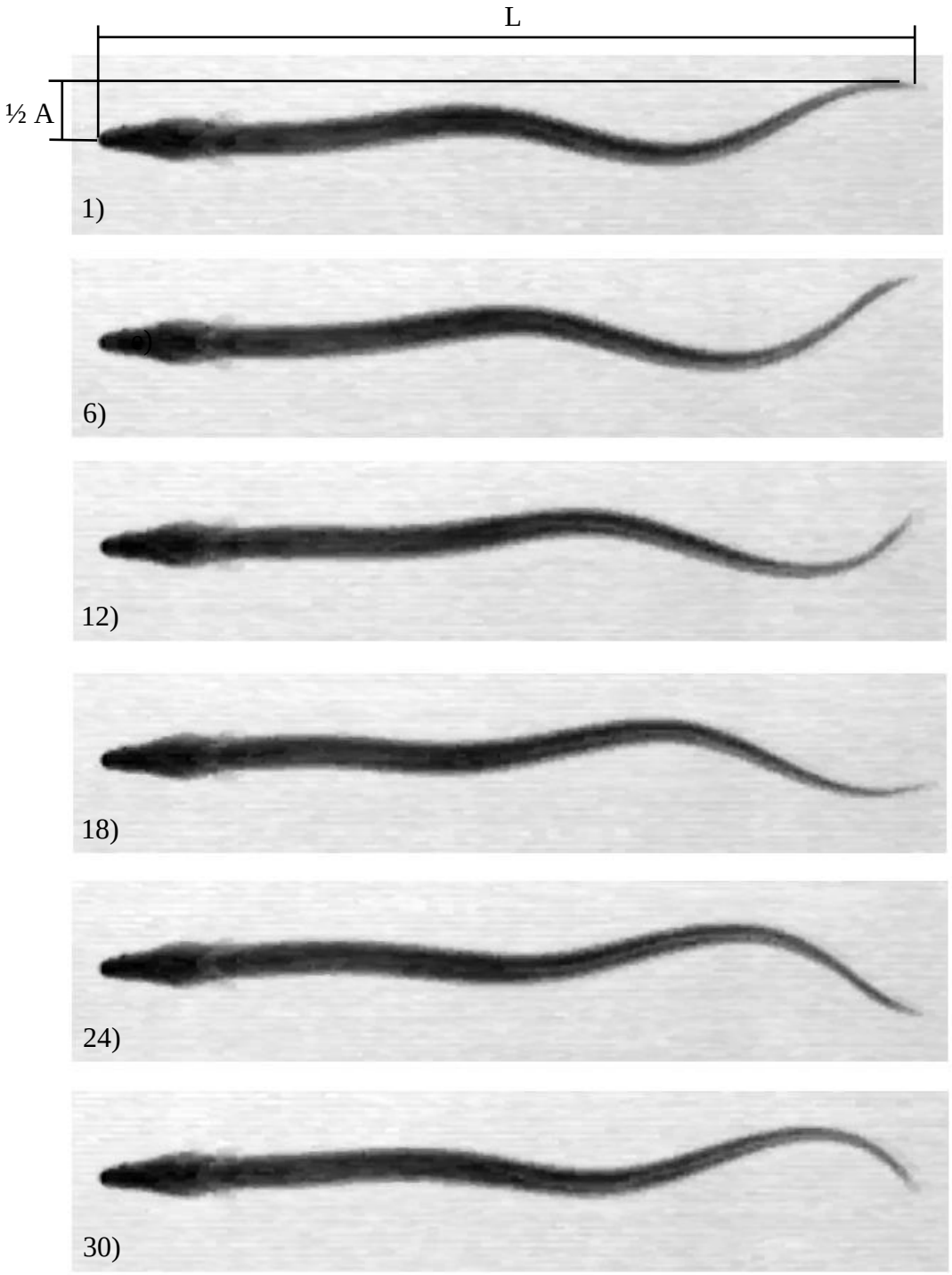


Figure 22: Select frames of the eel input images with the length (L) and stroke amplitude (A) labeled, frames 1,6,12,18,24,30

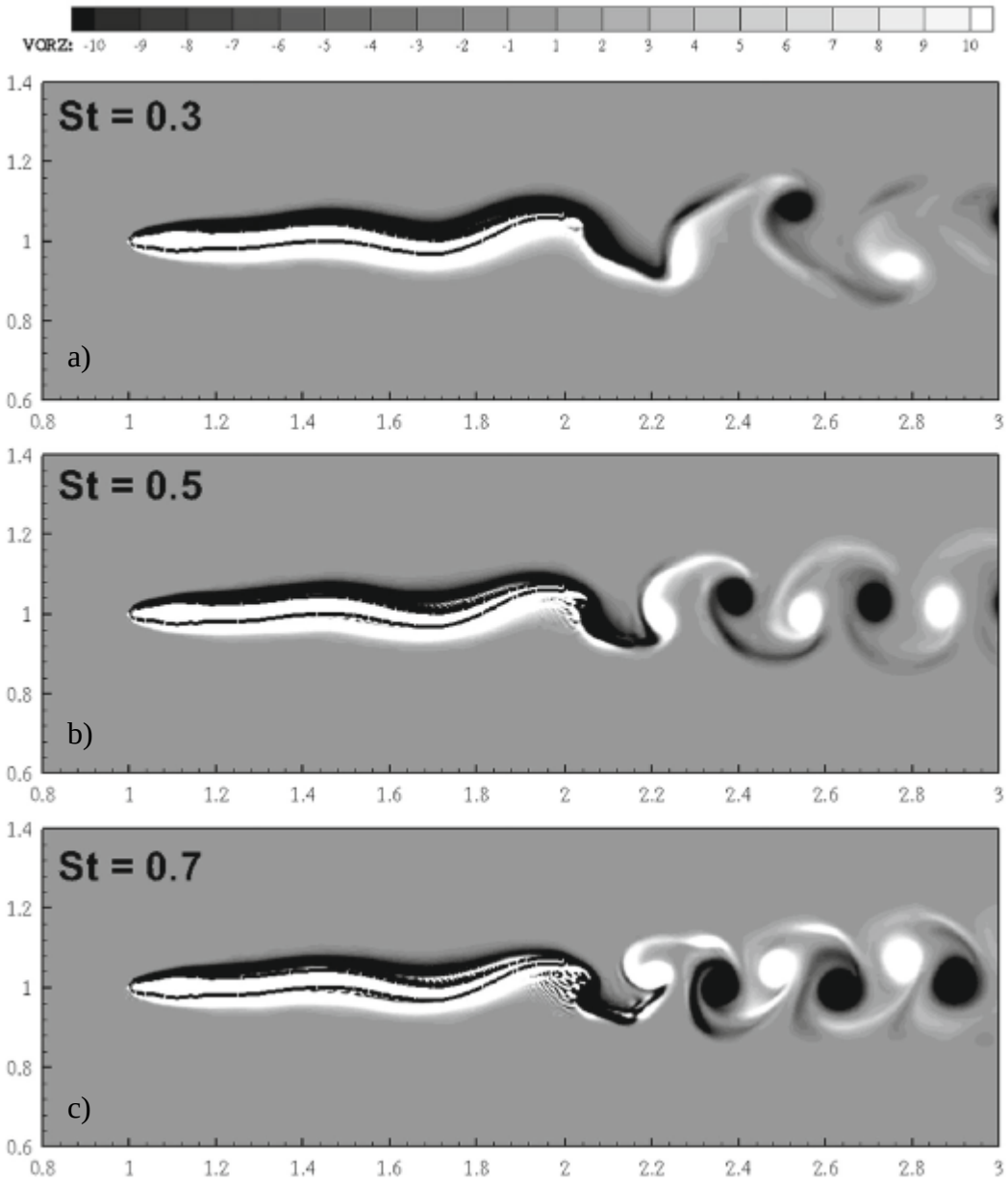


Figure 23: Dillard et al. Vorticity field results of the swimming eel at Re = 5000 and various Strouhal number, a) 0.3, b) 0.5, and c) 0.7.

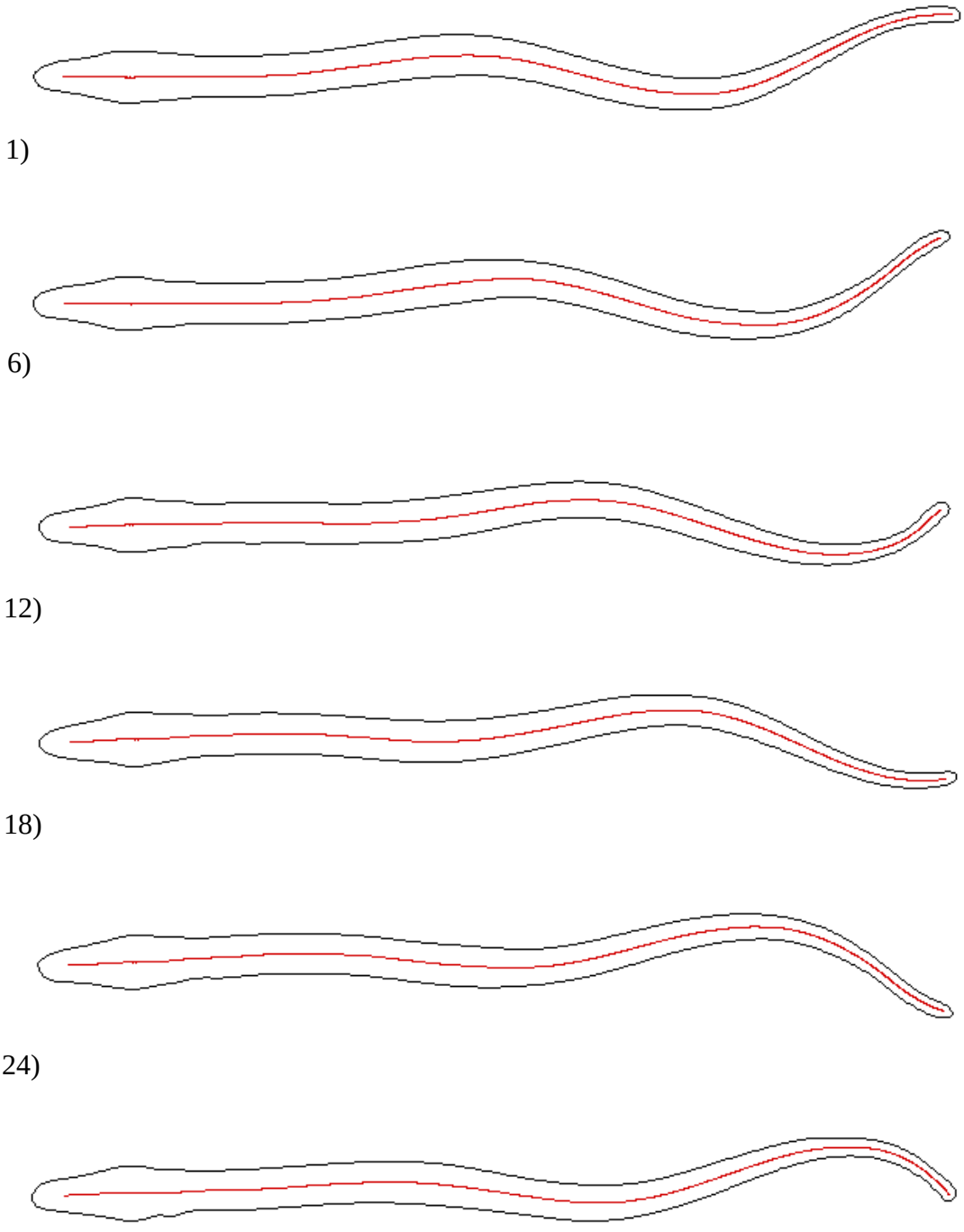


Figure 24: Skeletonization results for select frames of the eel, frames 1,6,12,18,24,30

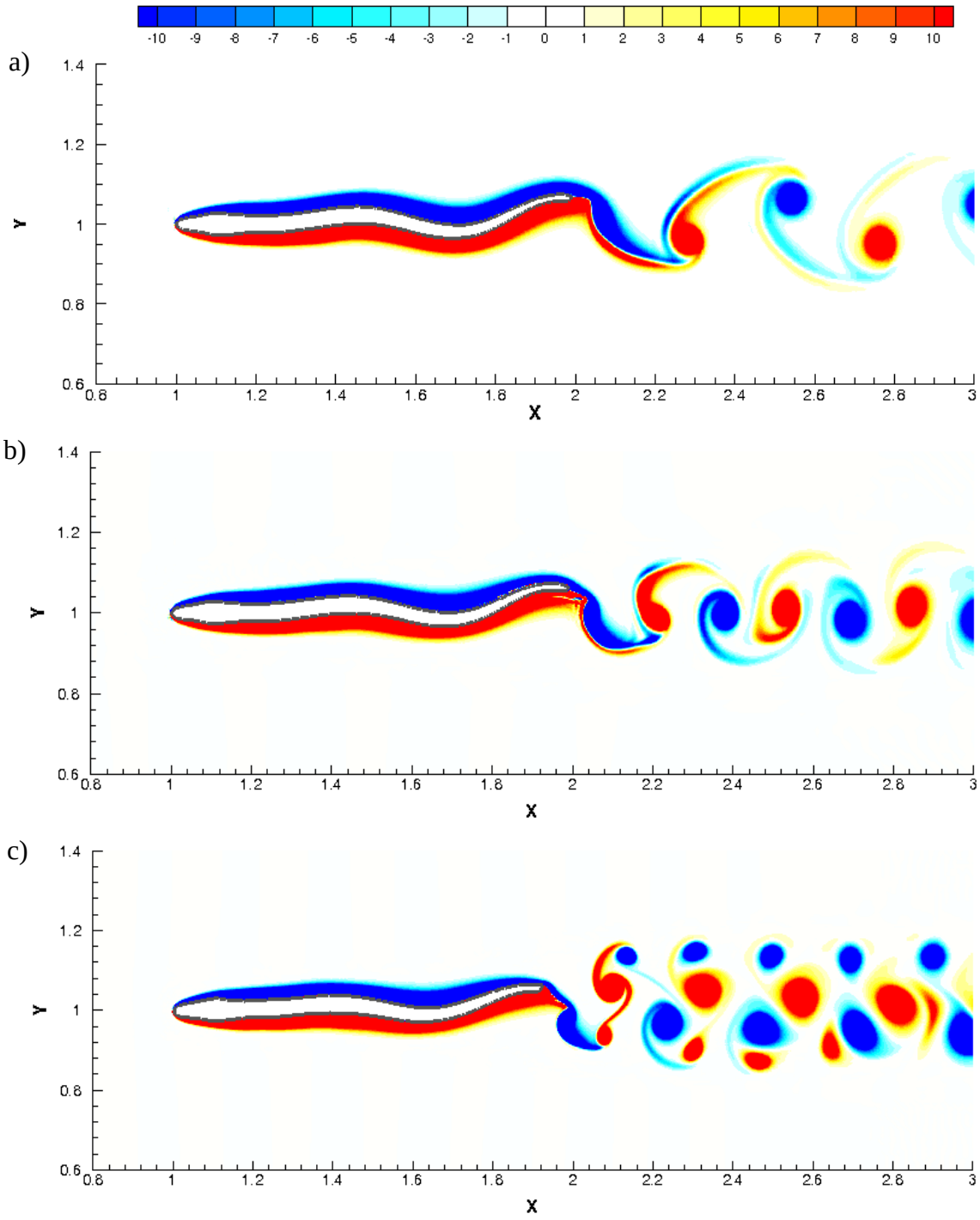


Figure 25: Vorticity field results computed using the skeletonization morphing method to simulate the swimming eel at Re = 5000 and various Strouhal numbers, a) 0.3, b) 0.5, and c) 0.7.

CHAPTER 5 APPLYING BOUNDARY CONDITIONS INSIDE THE DOMAIN

The eel simulation of the previous chapter is an insightful verification case which demonstrates the success of the proposed interface morphing method. While this flow is biological in nature, it doesn't directly apply to the goal of performing left ventricle simulations from patient-specific image data. The most notable difference between the two is the external flow for the eel simulation and internal flow for the left ventricle simulation. For external flow simulations, the inflow/outflow boundary conditions conform well to a cuboidal domain as long as the domain is large enough so that the boundary conditions do not influence the region of interest surrounding the immersed object. This is not the case for internal flow such as the cardiovascular system.

It is impractical to perform patient-specific simulations of the entire cardiovascular system for a variety of reasons. First, the disparate length scales range from centimeters in the aorta to micrometers in the capillaries. Second, medical imaging modalities only capture the region of interest due to physical constraints and concerns about patient safety and comfort. Lastly, the added computational burden would be prohibitive.

Since it is impractical to simulate the entire cardiovascular system, a region of interest must be isolated with the appropriate fluid flow boundary conditions applied where the vessels are clipped. For patient-specific left ventricle simulations, the boundary conditions must be applied upstream of the mitral valve and downstream of the aortic valve. This ensures there is enough fluid domain for both valves to move between open and closed positions. This is illustrated in Figure 26. The task of applying these boundary conditions is challenging since the heart and vessels deform, translate, and

rotate throughout the cardiac cycle. Thus, to maintain a consistent boundary location for the duration of the simulation, the position and cross-sectional shape of the boundary condition must dynamically update with the vessel for every time-step of the flow simulation. This is especially problematic when considering flow solvers operating on a Cartesian grid domain (including the current approach), since those boundary conditions are traditionally applied to the sides of the cuboidal domain.

5.1 Selection of Immersed Boundary Method

The development and implementation of applying inlet/outlet boundary conditions to vessels inside a fixed (typically Cartesian) fluid domain depends upon the immersed boundary method chosen to represent the FSI at the moving interface. The immersed boundary method was first introduced by Peskin to simulate cardiac blood flow [87]. This approach has spawned a full field of methods in which the discretized fluid domain does not deform to match the location of the solid interface. An Eulerian representation of the interface is used to define the position of the interface. The solid is tracked with respect to the fixed fluid domain using *forcing functions* to apply boundary conditions to the fluid governing equations which are shown below for an incompressible Newtonian fluid.

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{\nu}{UL}\nabla^2 \mathbf{u} \tag{5.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{5.2}$$

The forcing functions are used to mathematically impose the moving interface onto the flow field and are typically applied in one of two ways. The first means of doing

so is the continuous forcing approach. This uses the physical properties of the elastic solid boundary to apply a distributed force on the fluid domain cells near the interface [87]–[90]. Using this approach requires that the fluid equations be solved for all computational cells, even ones located in the solid portion of the domain. This method automatically handles the issue of initializing cells which are newly classified as being in the fluid portion of the domain. On the other hand, solving the equation for all the cells in the solid portion of the domain, is computationally burdensome if the geometry doesn't fit well inside of a cuboidal domain. This is the case for objects which are spheroidal in shape, such as the left ventricle. Another disadvantage of this approach concerns the distribution of force over a set of fluid cells near the interface. This acts to smear the interface and causes a loss of accuracy near the wall [91]. This continuing forcing approach also performs poorly when applied to rigid or prescribed moving boundaries. The governing equations are mathematically stiff when operating the elastic boundary in the rigid limit [91].

To mitigate this issue of simulating rigid interfaces, several approaches have been developed such as Goldstein et al. [92] in which a damped oscillator model is applied or Angot et al. [93] which assumes the entire flow occurs in a porous media. These approaches add additional parameters to the immersed boundary interface which impact the accuracy and stability of the solution. Since the moving wall of the left ventricle is a prescribed moving boundary that is generated from segmented patient images, this class of approaches is not ideal.

The second set of immersed boundary approaches is the discrete forcing approach. This approach acts by directly modifying how cells near the interface are

discretized. This category of approaches is implemented in one of two ways: indirect forcing or direct forcing. Indirect, discrete approaches have been successfully applied for a variety of applications [94]–[96], but they still use a distribution function to apply the forcing term and thus suffer from the same diffuse interface issues as the continuous forcing approach described above. Alternatively, using a direct approach rather than applying a force function, the discretization stencil of cells near the interface is modified. This is often referred to as a sharp interface implementation, since this approach maintains a crisp definition of the boundary and accurately models the fluid flow near the wall.

Many approaches have been developed with this direct methodology [27], [28], [97]. One such instantiation of this approach is the Ghost Fluid Method (GFM) [25], [29]. For this method, the finite difference stencil used for the cells near the interface are directly modified so as to extrapolate field values to a set of cells adjacent and external to the fluid interface. These cells are referred to as ghost cells. The value extrapolated to these ghost cells are constructed to conform to the no-slip velocity and Neumann pressure boundary conditions typical of laminar flow. Due to the inherent pressure oscillations occurring at the moving immersed boundary interface, a hybridized least-squares approach can be employed to resolve the issue [98]. Since this GFM approach achieves accurate near-wall results without the computational burden of solving the entire domain, it is clearly the best choice for representing the immersed interface for the stated goal of left ventricle simulation.

5.2 Boundary Cap Method

The conventional approach used to simulate internal flow problems using immersed boundary interface is to apply inflow/outflow boundary conditions to a set of cells laying on the sides of the fixed cuboidal Cartesian domain. Manually constructed flow extensions are typically required to allow the anatomical geometry to extend and cross the sides of the cuboidal domain in a perpendicular manner as needed. Flow extensions are prohibitive for this application due to the burden of construction and the limitation of patient-specific simulations to rigid vessels which would otherwise require a unique flow extension geometry for every flow solver time-step. These flow extensions are also disadvantageous since they have the potential to induce alterations to the flow field that are not present in the cardiovascular circulation.

The ability to apply boundary conditions inside the Cartesian domain is critical for performing left ventricle simulations from patient-specific imaging. The challenge of applying static boundary conditions inside of a 2D Cartesian domain to this type of immersed boundary method was initially investigated by our group in Goddard et al. [99]. Rather than employing cells on the sides of the cuboidal domain, this method identifies a set of cells inside the domain to apply the inflow and outflow boundary conditions, and can be performed even when the orifice is oriented non-orthogonally to the Cartesian grid.

Many modifications must be made to a typical immersed boundary computational framework to accommodate the stair-step configuration of boundary cells, the non-orthogonal direction of the boundary plane, and the full-size cell spacing. Because the previous work was limited to static geometry in 2D space, additional efforts are needed to

extend this method to moving geometry in 3D space in a manner which allows for the simulation of the contracting left ventricle derived from a time series of patient-specific images. Several enhancements are necessary to accomplish this goal.

5.2.1 Boundary Cap: Search Region

To develop the 3D, moving boundary cap approach, it is necessary to determine the region on the grid where the boundary conditions should be applied. For this reason, a suitable search radius must be computed to guide the selection of boundary cap cells. The location of an interior boundary condition (boundary cap) is defined by a center point and a unit normal vector pointing toward the fluid region of the domain. Mathematically, this represents a line in 2D space or plane in 3D space. Since the mathematical definition of a line and plane both extend to infinity, they carry the risk of crossing tortuous geometry in more than one location. A 2D example of this is illustrated in Figure 27 showing the application of a boundary cap to the ascending aorta of an aortic arch. Therefore, it is necessary to define a search radius which is marginally larger than the vessel to which the boundary cap search algorithm is applied. For 2D simulations, this can be accomplished by computing every instance where the boundary line intersects the interface (ie. zero level set value). There will be a finite number of these locations with the closest two being the edges of the vessel containing the cut point [99].

This approach will not directly extend to 3D space since the intersection of a plane and 3D surface yields a 3D curve rather than a discrete number of points. A binning algorithm is one possible solution for the 3D situation in which fluid cells are sorted into a discrete number of bins per a quantifiable metric. This works by identifying all fluid cells adjacent to the boundary cap plane and placing them into a finite number of

bins per a distance metric. This metric is computed by projecting the cell onto the boundary cap plane and measuring the distance from that project to the boundary cap center point. The bins will form contiguous rings of cells extending radially from the boundary cap cut point. The search radius would simply be the distance to the first bin, or ring, which contains zero cells. Determining a suitable number of bins is a challenge for this approach. If too many bins are used, it is possible to get a false positive with a bin located inside the target vessel containing no cells. If too few bins are used, it is possible the gap between vessels will be missed completely. For a simulation of this type, the correct number of bins could depend on a number of factors, including the computational domain size, minimum computational cell size, size of the vessel, and gap distance between vessels. Rather than implementing an ad-hoc relationship as a function of these factors, a more robust solution is desired.

One robust, but more computationally expensive alternative, is to implement a flood-fill algorithm. This algorithm starts at the computational cell containing the initial vessel cut point as the first cell on the propagating front. The neighbor of each cell on the front is checked to see if it is: 1) adjacent to the cut plane, 2) not already included in the flood fill, and 3) located in the fluid portion of the domain. This process is iterated by adding conforming neighbor cells to the front until there are no new cells to add. The final result is a disk of cells collinear to the cut plane in the shape of the vessel cross-section as seen in Figure 28a. The distance to the furthest cell can then be used as a robust measure of the search radius. LMR also uses the search radius to determine which region of cells to refine in preparation for internal boundary caps. Figure 28b shows the successful LMR refinement due to the search radius. Without the search radius

limitation, cells in the descending aorta adjacent to the boundary plane would also be refined. Since the boundary condition must update and follow the moving vessel, the flood fill process must be repeated for every flow solver time-step. Great care should be given to minimize the computational cost. Min, max, and sign, logic are used to optimize the costly neighbor search loop. Another optimization opportunity lies within the parallel communication between domain partitions. FSI solvers utilizing distributed memory, typically divide the domain so that each processor only stores cell information for a specific region, with a copy of cells lying adjacent to the local domain partition but not included. Communication between domain partitions is often the bottle neck for computations. The duplicated layer of copied cells owned by neighboring partitions, the ghost layer, allows for finite difference stencil completion of partition-adjacent cells belonging to the local partition. This flood fill algorithm has been implemented in such a way as to limit the amount of information broadcast between domain partitions. In this scenario, the only information communicated during an iteration is a stopping flag and an array of processor inclusion flags, one for each processor. These flags denote the flood-fill status of each domain partition with one of three options: 1) currently in the flood 2) new addition to the flood, or 3) not yet included in the flood. When the front encounters a cell in a ghost layer of an adjacent partition, the appropriate partition is flagged as being newly added to the flood and the coordinate location of the cell is communicated to the new processor. A new front will propagate from the new seed point on the recently added processor which floods independently of the original front. Ghost layer cells of partitions already being flooded, will not be considered for the front. This strategy requires much smaller packets of information to be transferred to other domain partitions

as compared to other communication approaches, such as exchanging the partition ghost layer for every iteration.

5.2.2 Boundary Cap: Cross-Sectional Area and Mass Flux

For every flow solver time-step, the cross-sectional area of each boundary cap cell subdivide by the boundary cap plane must be correctly calculated in order to accurately determine the mass flux contribution and achieve global mass balance. Marching cubes has been selected as an effective means to quantify the area [65]. The algorithm classifies each cell through which the cut plane passes by identifying to which side of the plane each corner lies. The specific combination of cell corners lying on the fluid side of the plane classify the cell into 1 of 14 possible cut configurations as shown in Figure 29a. A look up table is then used to compute the planer polygon shape bisecting the cell. This polygon is represented by a set of triangles for which the area can simply be calculated. An example of the final triangulated area is shown below in Figure 29b.

5.2.3 Boundary Cap: 3D Neumann Condition

Since the flow solver utilizes a four-step pressure correction method [30]–[32], a Poisson equation is solved to enforce continuity on the provisional velocity field. The equation is discretized using finite difference stencils as shown below where the subscript refers to the neighbor in the cardinal direction with T and B indicating top and bottom directions respectively

$$-\nabla^2\varphi = -\alpha(\nabla \cdot \mathbf{u}^{**}) \tag{5.3}$$

$$-\frac{1}{dx^2}(\varphi_E + \varphi_W - 2\varphi_{IC}) - \frac{1}{dy^2}(\varphi_N + \varphi_S - 2\varphi_{IC}) - \frac{1}{dz^2}(\varphi_T + \varphi_B - 2\varphi_{IC}) = -\alpha(\nabla \cdot \mathbf{u}^{**}) \tag{5.4}$$

To solve this Poisson equation, a Neumann boundary condition is applied to vessel inlets and outlets. Due to the non-orthogonal orientation of the boundary cap, the Neumann stencil is constructed by projecting a probe location normal to the boundary plane onto the next row of cells in the primary direction. An example of the probe construction is shown is Figure 30 for a 2D simulation. In 3D the four cells surrounding the probe location are used to perform an algebraic tri-linear interpolation. The equation below shows the value at the probe location with BC indicating the cell located on the boundary cap and λ denoting the weight from linear interpolation.

$$\frac{\partial \varphi}{\partial n} = \frac{\varphi_P - \varphi_{BC}}{dn} = \frac{\lambda_1\varphi_1 + \lambda_2\varphi_2 + \lambda_3\varphi_3 + \lambda_4\varphi_4 - \varphi_{BC}}{dn} = C \qquad 5.5$$

The algebraic relation is substituted into the Poisson equation for every cell that contains a boundary cap cell in the finite differencing stencil. Since boundary cap cells are selected from existing cells inside the Cartesian domain, non-orthogonal boundary cap planes will generate a pixelated, or stair-step, configuration of cells. Due to this stair-step arrangement of non-orthogonal boundary planes, some boundary adjacent fluid cells may have up to three stencil neighbors with this algebraic substitution for the Neumann condition. Memory pre-allocation of the operator matrix must accommodate this large number of additional values for boundary-adjacent fluid cells.

$$\varphi_{BC} = \lambda_1\varphi_1 + \lambda_2\varphi_2 + \lambda_3\varphi_3 + \lambda_4\varphi_4 - Cdn \qquad 5.6$$

5.2.4 Boundary Cap: Extrapolation of Field Values

During operation of the flow solver, multiple field values, such as provisional velocity, must be extrapolated from interior fluid cells onto the cells of the boundary cap. Two interior probe locations are needed for linear extrapolation. The parallelization strategy of domain decomposition creates a challenge when evaluating these probe locations. The second probe is located far from the boundary cell and may have one or more of the surrounding neighbors needed for interpolation be more than the ghost layer thickness on another domain partition. A solution is needed which is more robust than simple linear interpolation. For this reason, a least-squares approach has been adopted for the 3D implementation. Figure 31 illustrates a 2D example showing the cloud of cells selected for the least-squares evaluation at each probe location.

5.2.5 Boundary Cap: Corner Treatment

The final modification required to enable 3D interior boundary caps is the treatment of numerically unstable corner configurations. One such configuration has been identified as being problematic for the current implementation of GFM, the immersed boundary treatment selected for this current computational framework. As previously mentioned, GFM creates a layer of ghost cells just outside the interface which contain extrapolated values so the stencil of the fluid cells adjacent to the immersed interface can be handled consistently. Each ghost fluid cell mirrors its position across the immersed interface to a probe location in the fluid for which the surrounding cells are used to interpolate a value. This probe value is combined with the boundary condition value at the interface, to extrapolate a value to the ghost cell. The finite difference

stencils can then operate on the boundary adjacent fluid cell in the same way as the rest of the fluid cells.

This approach can become problematic if the cell arrangement at the intersection of the boundary cap plane and the interface occur such that the corner most cell is classified as a fluid cell rather than a boundary cell as shown below in Figure 32a. The ghost cell across the boundary from this fluid cell can generate bad ghost field values. Application of GFM to this ghost cell creates a probe location equidistance to the interface on the opposite side. For the configuration described above, the probe location lies outside the convex hull of the least square cloud as shown in Figure 32b. The likelihood of encountering such a configuration is compounded by the desire to apply boundary caps to moving vessels which generates a very large number of corner configurations during the motion. The treatment must be robust for any configuration of cells at the corner of the interface and the boundary as the boundary cap moves across the Cartesian domain.

A strategy to address this issue is to first identify all cells classified as GFM ghosts that are positioned adjacent to the boundary cap plane. When applying GFM, the fluid cell to across the interface would instead be used to construct the cloud of neighbors for least-squares evaluation. This provides a larger cloud of cells thus reducing the instability. From here, the probe location can be checked to verify that it lies inside the convex hull of the least square cloud. Typically, it is challenging and expensive to determine whether a probe is located inside the convex hull of a point cloud, especially in 3D space. However, in this approach, the fact that the cells lie on a Cartesian grid allows for a simple geometric approximation. The corners of the rectangular prism-shaped point

cloud can be identified and a plane constructed through them as an approximate test to see if the probe lies within the convex hull. If the probe location is found to be outside the convex hull, the least square method is instead evaluated at the center of the point cloud. A 2D example describing the final treatment for a ghost cell with probe location outside the convex hull is shown in Figure 32c.

5.2.6 Boundary Cap: Partitioning

Parallel computing via domain decomposition/partitioning is necessary to perform 3D simulations in a reasonable amount of time. This is true for simulations such as the left ventricle. As previously demonstrated, the non-orthogonal nature of the interior boundary caps requires a larger number of neighbor cells in order to apply the appropriate fluid boundary conditions. This increases the challenge of appropriate and efficient domain partitioning using the boundary cap approach, as compared with a conventional boundary condition applied at the sides of a cuboidal domain. These conventional boundary conditions only need to force a single interior neighbor to be on the same domain partition. This neighbor along with the single layer of partition ghost cells is sufficient to construct all of the appropriate finite difference stencils.

To develop a suitable partition strategy for the interior boundary caps, it is important to consider all mathematical operations requiring a boundary condition treatment. The solution will, of course, be dependent on the thickness of the partition ghost layer, specific flow solver scheme, and the numerical order of the finite differencing operations. For the specific boundary cap implementation described above, a pencil partition approach will be considered. This approach operates on the set of fluid cells neighboring the boundary cap. The pencil partition is oriented in the primary

Euclidean direction of the boundary cap, i.e. the largest component of the vector perpendicular to the boundary cap plane. The boundary cap cell in the primary Euclidean direction and the corresponding fluid cell on the opposite side are both forced to be on the same partition as the fluid cell. At the smallest possible domain partition, this would consist of a set of three collinear cells beginning with the boundary cap cell and marching in the direction of the fluid. Since a partition ghost layer thickness of one is used for this computational framework, the worst case pencil partition will have 42 ghost neighbors in 3D and 12 in 2D. Figure 33 shows this worst case three cell scenario in 2D. The primary Euclidean direction for this case is in the x (horizontal) direction. Green cells indicate cells owned by the local domain partition with the cell marked with the solid black circle being the boundary cap cell. The green cell to its right with a solid gray circle is the fluid adjacent neighbor and one after that is the fluid interior neighbor. These three cells are surrounded with a single layer of purple shaded ghost cells, 12 in total. This pencil partition strategy must be evaluated with respect to each operation performed on the boundary cap.

First consider the inflow/outflow boundary treatments applied to each orifice. Inlet velocity boundaries are often assigned a Dirichlet boundary condition. In this case the boundary cap cell is given a prescribed value with no other impact to neighboring cells, thus requiring no special treatment for domain partitioning. The outlet boundary condition is often prescribed to be a mass conserving Dirchlet condition via convective outflow boundary condition, which was shown previously in Equation 3.13. To apply the outlet boundary condition a first order derivative must be constructed.

In fact, first order derivatives are constructed on the inlet/outlet boundaries during several steps in the flow solver. This derivative is constructed normal to the boundary, which for the conventional cube-side approach only requires the value of the interior neighbor cell. For the interior boundary conditions a probe location is instead constructed $dx \cdot \mathbf{n}$ away and evaluated using a least-squares approach. Thus, the domain partitioning for the outlet boundary condition of the interior boundary caps must be done in a manner which ensures a sufficient cloud of cells to evaluate the probe location. Figure 33a demonstrates the evaluation of the probe location using least-squares on the smallest possible pencil partition scenario for a 2D simulation.

Upon completion of many flow solver steps, field values must be extrapolated to the boundary cells for the following operations: solving the Helmholtz equation, initializing newly added cells to the flow domain, and the explicit calculations. As previously mentioned, this is accomplished through linear extrapolation of two probe locations positioned at $dx \cdot \mathbf{n}$ and $2dx \cdot \mathbf{n}$ away from the boundary cap cell (where **n** is the normal direction pointing from the boundary into the fluid). Least-squares is used to evaluate fluid values at the two probe locations. The first probe location is evaluated in the same way as the construction of first order derivatives. Due to the distance between the second probe and the boundary cap cell, the risk of an insufficient least-squares cloud is greatly increased if the pencil partition strategy is not used. For example, if the boundary cap cell is the right most cell on the domain partition, the second probe location will be outside the convex hull of the least square cloud extending only one neighbor to the right. As a result, special consideration (i.e. pencil partitioning) should be given for this second probe location. The second probe location can be seen in Figure 33b, again

for the smallest possible domain partition. This figure shows the second probe positioned well inside a sufficient least square convex hull (indicated with the dashed line).

Next consider the explicit terms which are computed from field values obtained from the previous sub-iteration. As before, consider the set of fluid cells that have at least one, but no more than three, boundary cap cells in cardinal directions (E, W, N, S, T, B). For the pressure gradient, convection, and divergence operations, the fluid neighbor cells must have their stencils updated at each sub-iteration to reflect the full cell spacing to accommodate the contribution of (potentially more than one) interior boundary cap cells to each respective stencil. Since the boundary cap cells are adjacent to the fluid cell, no special consideration of the domain partitioning is necessary. This is demonstrated for the smallest possible local domain using pencil partitioning in Figure 33c.

Boundary cap implementation of the Helmholtz equation is carried out in a similar manner as the previous explicit terms. The construction of the operator matrix must be updated for all fluid neighbors to account for the full cell spacing and the possibility of multiple boundary cap cells contributing to the finite differencing stencil of the fluid neighbor. Since the boundary conditions of the Helmholtz equations are Dirichlet at both inlets and outlets, the values are algebraically lifted to the solution array according to the number of fluid neighbors adjacent. As before, since the boundary cap cells are adjacent to the fluid neighbors, a domain partition's ghost layer is sufficient so that no special treatment is required for domain partitioning.

Finally consider the pressure Poisson equation. The boundary condition applied at both inlet and outlet is Neumann. As previously mentioned, the Neumann condition is

applied by substituting the suitable set of fluid neighbors for bilinear interpolation at the probe location positioned at $dx \cdot \mathbf{n}$ away from the boundary cap cell (Figure 30). As previously described through Equation 5.6 for the 3D case, each boundary cap cell will require four fluid or GFM ghost cells to be substituted. Each fluid neighbor cell could have up to three boundary cap cells in its finite difference stencil. These interpolation cells can be quite far from the fluid neighbor, and a single layer of partition ghost cells is likely insufficient to capture all the values needed for bilinear interpolation stencils. Fortunately, the pencil partitioning strategy accommodates the successful Neumann treatment, again for the smallest possible partition in 2D, is shown in Figure 33d.

Upon considering all of the aforementioned constraints, the pencil partitioning scheme appears to be sufficient to address the requirements of the boundary cap when it is located on more than one domain partition. The use of three collinear cells and surrounding ghosts are sufficient to perform all operations previously listed, although an additional treatment is necessary for GFM ghosts which are adjacent to boundary cap cells. As previously mentioned, GFM ghosts at the corner of the boundary cap cut plane and the interface have the potential to construct an insufficient least-squares cloud for evaluation at the GFM probe location. Shifting the selection of the cells used in the least square cloud was shown in the previous section to mitigate the problem. Even when using pencil partitioning, this shift operation could be nullified if the domain partition was located at the interface. To avoid this, in these rare circumstances, the neighbor in the cloud shift direction will also be constrained to the same partition as the GFM ghost cells that are adjacent to a boundary cap cell. This is illustrated in Figure 33e showing the pencil partition to include the corner GFM ghost cell.

5.2.7 Boundary Cap: Motion

When performing FSI simulations on cardiovascular geometry derived from patient-specific images, it is critical that the relative location and orientation of the boundary condition be maintained so as to follow the motion of the moving anatomy. To accomplish this, a landmark or simplified representation must be identified to measure the vessel displacement and orientation. Fortunately, one such representation has already been developed and implemented from the previous chapter, skeletonization. An input parameter of arc-length distance from an anatomical feature, such as the valve annulus, can be used to locate a consistent vessel position for every input image frame, using the corresponding skeleton. For example, one may choose to position the left ventricle outlet boundary condition in the ascending aorta at a specific measurement of two annulus diameters distal of the valve. In this way, regardless of how much the aorta moves and flexes during the cardiac cycle, the outflow boundary condition will be positioned in the same relative anatomical location. Due to the computation framework constraints described in the previous chapter, skeletons are only available for the time-steps corresponding to the input image frames. Cubic spline interpolation can be used to determine the intermediate position and velocity. The boundary cap orientation can also be interpolated by fitting a vector through the adjacent skeleton points to determine a vector which is tangent to the skeleton at that location.

5.2.8 Boundary Cap: Fresh Cell Treatment

Because the interior boundary caps are moving with the vessel throughout the cardiac cycle, new cells may become part of the fluid domain. When the geometry moves in the direction opposite to the boundary cap normal, cells which were previously

considered solid will be reclassified as fluid. Since these cells were previously solid, they will not have fluid field variable values. This presents a problem for the explicit operations performed during the first sub-iteration for the each time-step of the flow solver. To resolve this issue, linear least-squares extrapolation is sufficient to initialize field variable values to all newly added cells.

5.3 Boundary Cap Implementation

Having considered each aspect of the flow solver impacted by a new boundary treatment, it is now important to understand the implementation sequence for the successful use of boundary caps with the proposed level set morphing method on a patient-specific cardiovascular simulation. When the simulation begins, the boundary cap location and orientation is computed for each of the skeletons (one for each input image frame). Cubic spline interpolation is constructed to be evaluated during run-time.

The remaining boundary cap treatments described above are required at each time-step, starting with the update of the boundary cap position, velocity, and orientation. Using those new values, the flood fill operation is performed to determine a suitable search region. The cut plane, as defined by the boundary cap, and the search radius are used to identify fluid neighbors inside the plane and update the mesh refinement so that all the cells near the boundary cap are refined to the smallest level used in the simulation. The boundary cap fluid neighbors are used to apply the pencil strategy for domain partitioning. Next, cells within that search radius are evaluated to see if they are bisected by the boundary cap cut plane. The marching cubes algorithm is performed to determine the contribution of each cell to the cross-sectional which is necessary to compute the total mass flux. Boundary cap cells are then selected from the set of cells bisected by the

boundary cap plane. GFM classifications are updated accordingly [99]. Prior to initiating the flow solver calculations for the given time-step, the operator memory pre-allocation must be modified for boundary-adjacent cells depending on how many boundary cap cells are in the finite difference stencil. This could range from one to three for 3D simulations. The operator construction is then updated to include the extra substitutions to the boundary adjacent fluid cells, as described above. Finally, any cells which are added to the fluid domain as a results of the boundary cap motion (i.e. fresh cells), will be initialized using least-squares linear extrapolation.

Many of the aforementioned boundary cap treatments must be performed for every sub-iteration step of the flow solver, because they are updating field values used by subsequent sub-iterations. First, inflow/outflow velocity boundary conditions are applied, using the mass flux computed in the previous sub-iteration to enforce mass conservation of the convective outflow boundary condition. The normal derivative to the outlet boundary cap is constructed using least-squares to apply the convective outflow boundary condition. The explicit operations that compute the pressure gradient and convection terms are updated as described in Goddard et al. [99]. Values are then extrapolated back to the boundary cap cells using least-squares linear extrapolation normal to the boundary cap plane. Next, the Helmholtz boundary conditions are updated as described in Goddard et al. [99]. The explicit operations that compute the divergence of the provisional velocity, the pressure gradient, and divergence of the second provisional velocity are then updated accordingly followed by boundary conditions for the Poisson equation [99]. Finally, pressure values are extrapolated back to the boundary cap using least-squares linear extrapolation.

5.4 Stationary Boundary Cap Verification

5.4.1 Stationary Boundary Cap Verification: 3D Poiseuille Flow

The implementation of the proposed interior boundary cap approach is first verified using stationary simulations. The first stationary test case is a Poiseuille flow through a 3D pipe, selected because an analytical solution exists and can be used for comparison. The pipe is oriented non-orthogonally to all three of the Cartesian basis vectors, as seen in Figure 34a. This orientation offers the opportunity to test several of the challenging aspects described above. In all three directions, the boundary plane will require cells that are arranged in a stair-step configuration, thus testing the domain partitioning treatment. In various regions of the boundary cap, the adjacent fluid cells will contain one, two, or three boundary cells, which verify the successful memory pre-allocation and operator construction of the Helmholtz and Poisson equations. This non-orthogonal orientation also generates a large variety of corner configurations which tests GFM treatment developed in the prior section.

When applying a non-dimensional inlet plug velocity of 1.0, the analytical solution generates a parabolic flow with the fluid velocity 0.0 at the wall and a maximum value of 2.0 at the centerline. At a Reynolds number of 100, a fully developed paraboloid is observed 6.0 diameters from the inlet. Figure 34b plots the simulation velocity profile against the analytical solution at a cross-section located at 9 diameters from the inlet. The simulation results are in good agreement with the analytical solution.

5.4.2 Stationary Boundary Cap Verification: 3D Aortic Arch

A second, more complicated test case is modeled to further verify the appropriate implementation of 3D stationary interior boundary caps. Here, an idealized aortic arch is

modeled as shown in Figure 36a, with AA denoting the ascending aorta and DA denoting the descending aorta. Uniform steady flow at Re = 100 is applied to the ascending aorta with the other four vessels splitting the outlet flow. The mass flux apportioned to each outlet is shown below in Table 1.

Table 1: Apportionment of outlet mass flux for the aortic arch verification case

Outlet Vessel	**Percent Mass Flux**
Brachiocephalic (BC)	5.5
Left Common Carotid (LC)	4.8
Left Subclavian (LS)	6.9
Descending Aorta (DA)	82.8

Since no analytical solution exists for this geometry, verification is performed through comparison with results using commercial software (ANSYS® Fluent, v17.1). This case was selected due to the large number of orifices, the variety of orifice orientations, and the need for the boundary cap search radius in order to prevent the aortic arch inlet from clipping the descending aorta, similar to Figure 27. Mid-plane velocity magnitude contours are displayed for the boundary cap simulation in Figure 36b and for the commercial software in Figure 36c showing good qualitative agreement. To make a quantitative comparison, data is extracted at three different sections: along the mid-plane positioned in the ascending aorta, immediately after the left common carotid, and in the descending aorta. These locations are displayed in Figure 36. This figure also plots the data at those three sections showing excellent agreement. These two verification studies strongly indicate the successful implementation of stationary 3D interior boundary caps.

5.5 Moving Boundary Cap Verification

As described above, the complexity of the boundary cap treatments is increased by performing a simulation with moving geometry. To verify the successful development and implementation of this capability, a translating 3D Poiseuille case is considered. An angled pipe geometry, similar to the one shown in Figure 34a, begins at rest. After one unit of non-dimensional time, the pipe is translated in the horizontal direction by providing a sinusoidal velocity that ranges from 0.0 to 1.0 units of non-dimensional velocity. The pipe reaches the maximum velocity at eleven units of time. Figure 38 shows the advection velocity magnitude as a function of simulation time with the dashed lines indicating the duration of the acceleration period. Since the level set morphing method isn't used for this simulation, no geometric centerline skeletons are available to update the position and orientation of the boundary cap. Instead, the boundary cap locations are advected with the same velocity as the pipe. This test case not only verifies the dynamic ability to update the location and orientation of the boundary cap for every flow solver time-step, but it also gives good verification of the pencil domain partitioning strategy. Results are shown in Figure 38 for three difference simulation times, $t^* = 2$, $t^* = 10$, and $t^* = 20$ units of non-dimensional time. The front of each figure pane shows the location of the interface inside the cuboidal domain generated by extracting an iso-contour of the level set field at a value of zero. Behind the interface is a mid-plane contour showing the index referring to the processor number that owns that domain partition. 56 processors were used during this simulation, to challenge the robustness of the boundary cap partition capabilities. As the geometry moves from left to right, it is clear by observing the changing partition numbers of the boundary cap regions that

implementation has successfully partitioned and load-balanced so that the single layer of ghost cells is sufficient to perform the flow computations. In the rear of the figure panes, a mid-plane slice shows contours of velocity magnitude. The same analytical velocity profile as the static case is achieved with a maximum centerline velocity of 2.0. This verification case has successfully demonstrated the capability of the interior boundary caps to move during a flow simulation and provide the correct fluid dynamics. It has also clearly demonstrated the success of the pencil strategy for domain partitioning.

The combined success of these stationary and moving verification cases indicates a successful implementation of 3D, moving boundary caps which are now ready to be combined with the proposed level set morphing strategy and applied to left ventricle simulations.

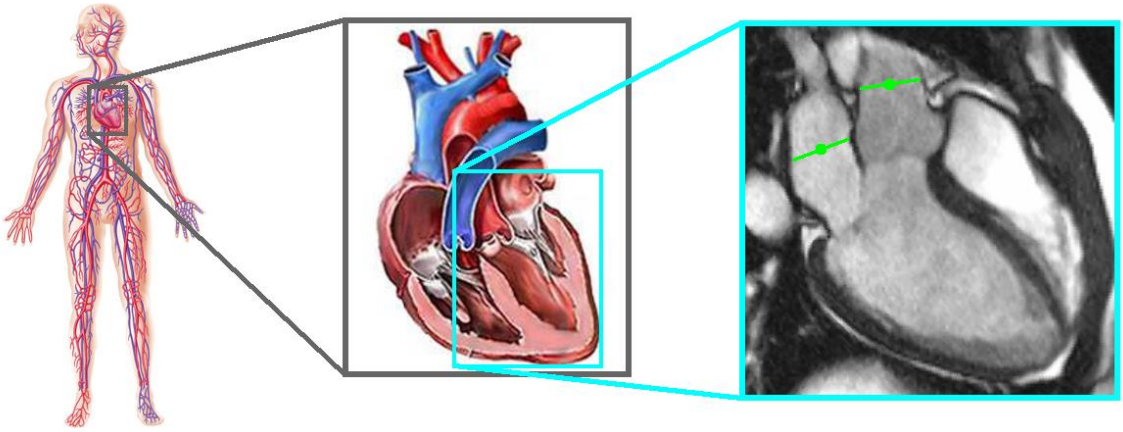


Figure 26: Due to the size, complexity, and disparate length scales of the cardiovascular system, only a portion of the system can be modeled, necessitating geometry clipping and application of physiologic boundary conditions.

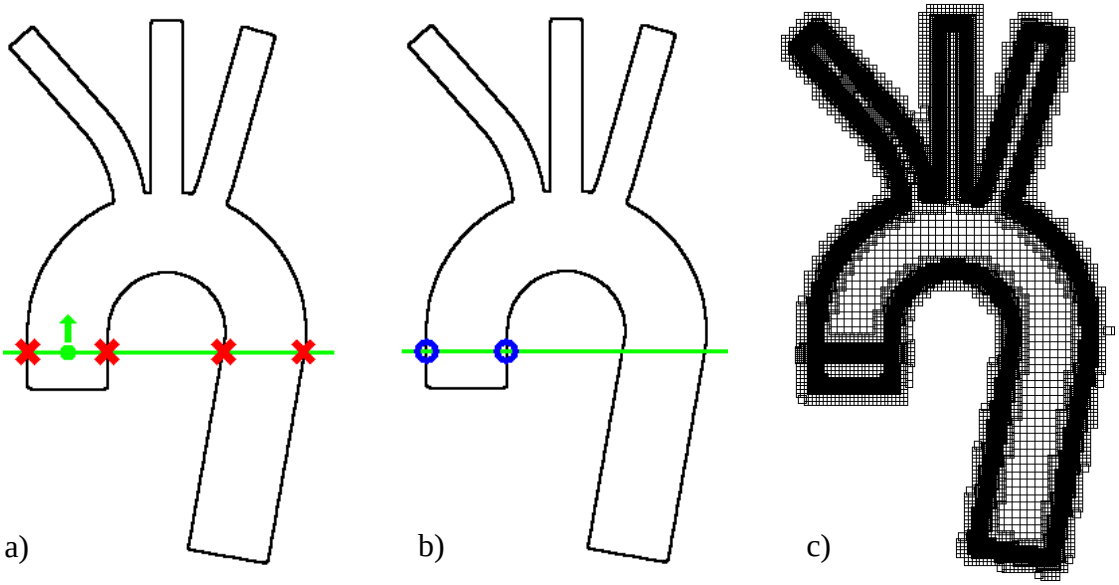


Figure 27: Determining a 2D search radius to constrain the boundary condition to the desired region of the ascending aorta, a) red X's indicate all location of the intersection between the zero level set interface and the boundary cap cut line, b) blue O's indicate the intersection points selected to define the search region, c) mesh refinement correctly indicating that only cells within the ascending aorta are selected and refined.

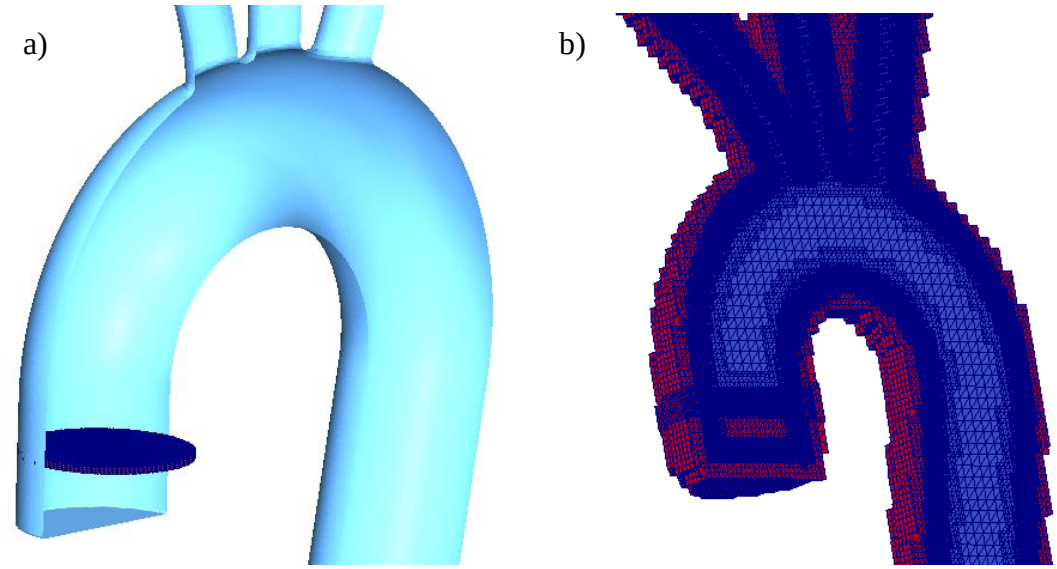


Figure 28: Results of using flood fill approach to define boundary cap search radius, a) disk shaped result of the flood fill algorithm, b) mesh refinement indicating the correct selection of boundary cap cells.

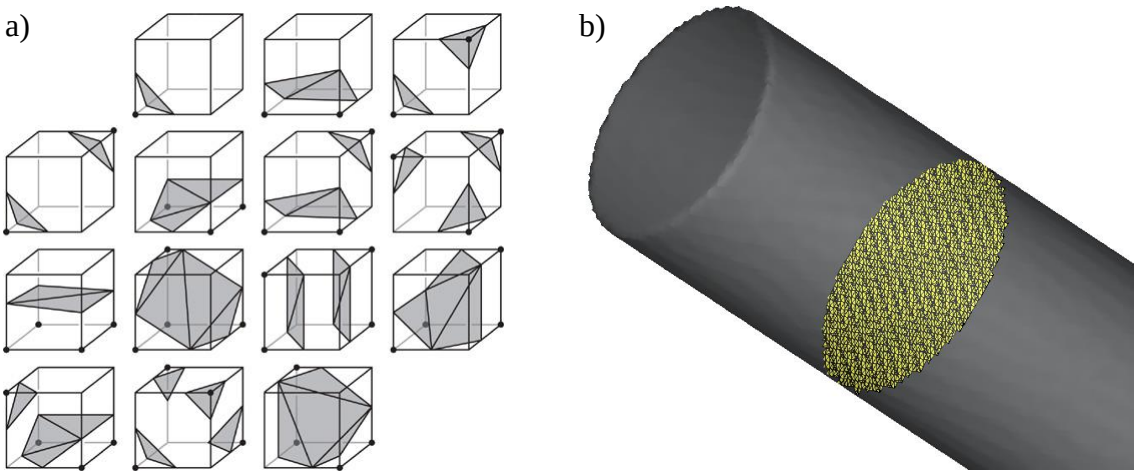


Figure 29: Use of marching cubes to compute vessel cross-sectional area and mass flux of each cell, a) representation of each of the 14 possible cell classifications, b) example result of the triangulated area.

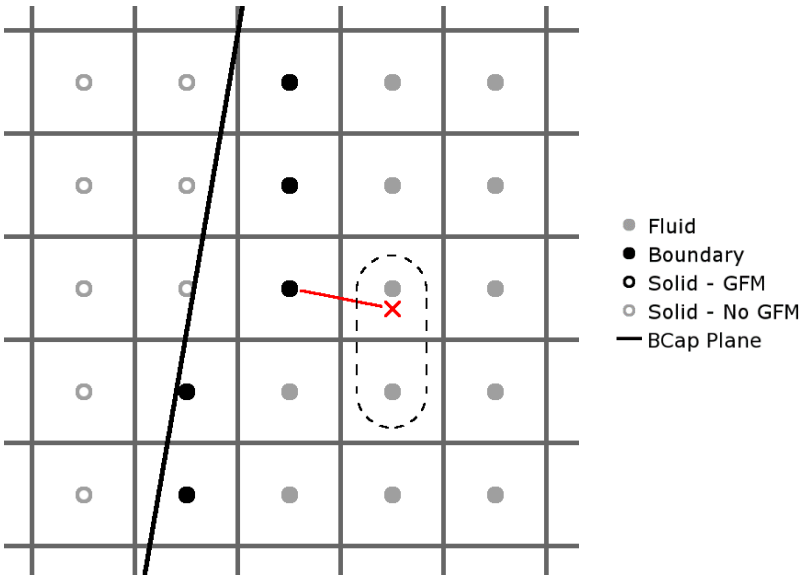


Figure 30: Construction of the Neumann boundary condition to be applied to the Poisson equation showing a 2D example of the cells used for constructing a linear interpolation at a probe location (red X).

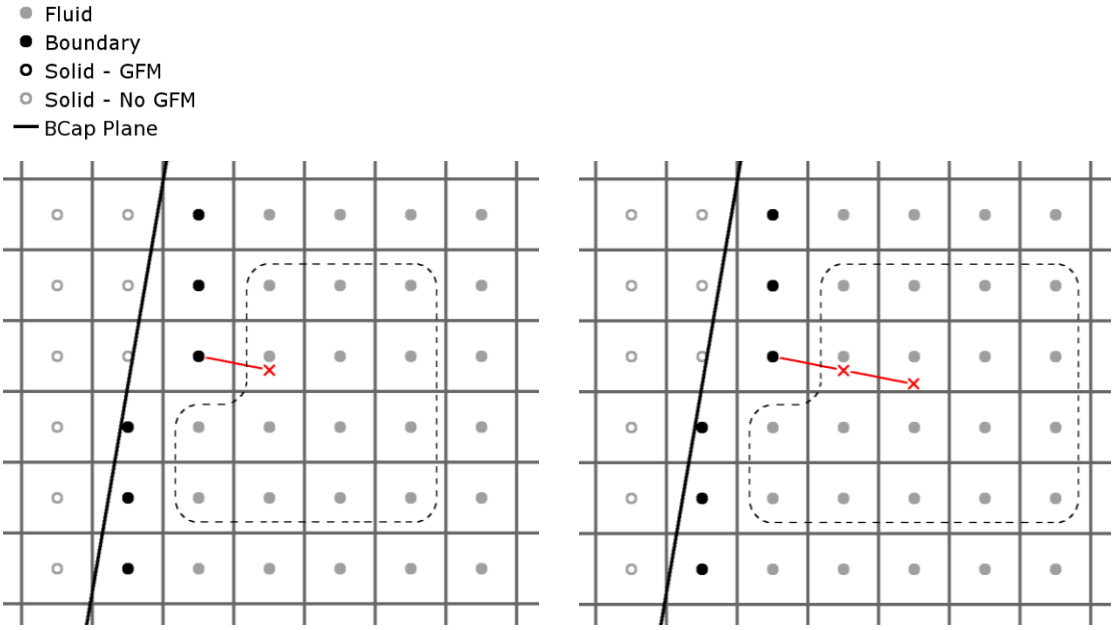


Figure 31: 2D linear extrapolation to boundary using least-squares, a) interpolation cloud for the first probe location, b) interpolation cloud for the second probe location.

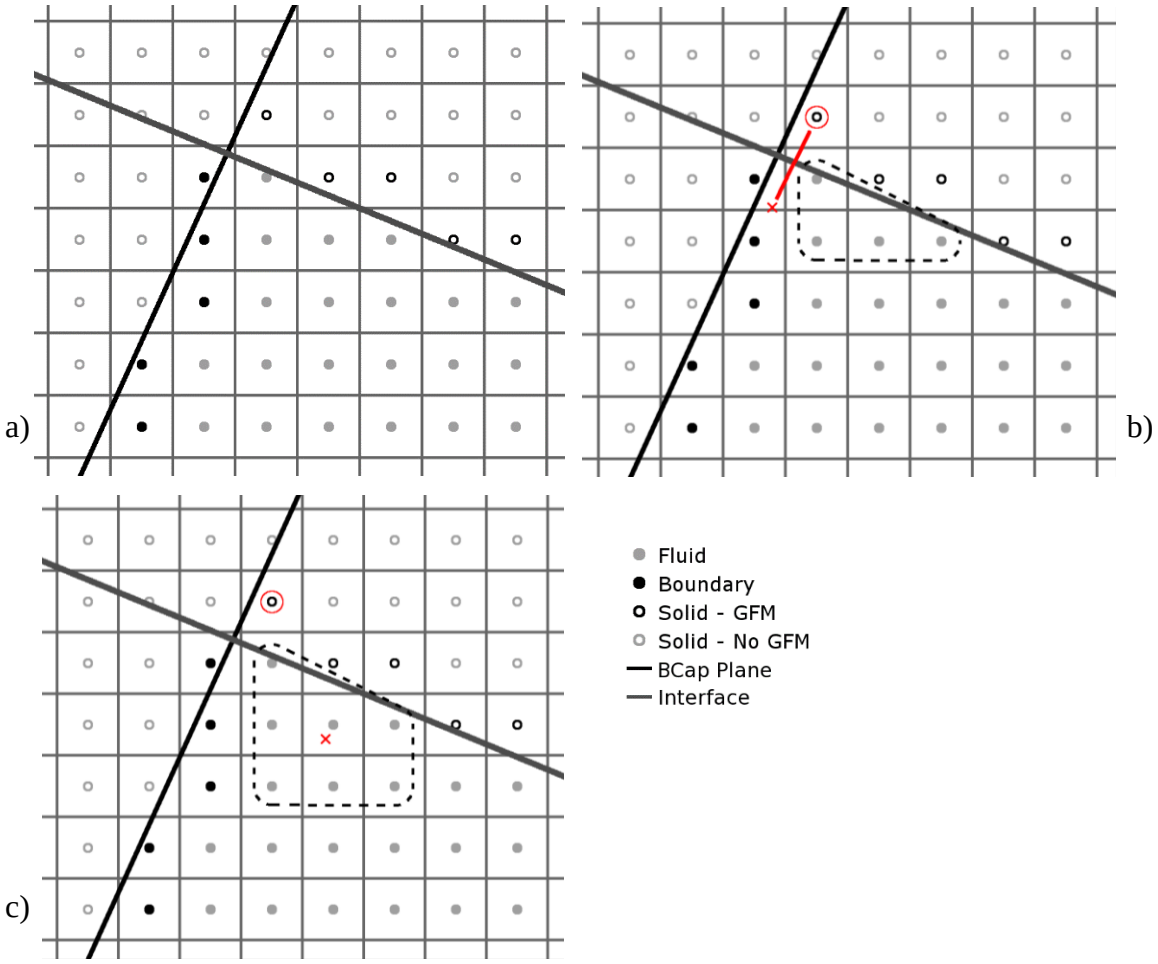


Figure 32: Modified GFM treatment of ghost cells adjacent to the interior boundary plane a) Configuration showing classification of cells, b) Problematic ghost cell (red circle) with projection to probe location (red X) residing outside the convex hull of the least square cloud (dashed line), c) Corner treatment showing expanded least square cloud with probe located at the center.

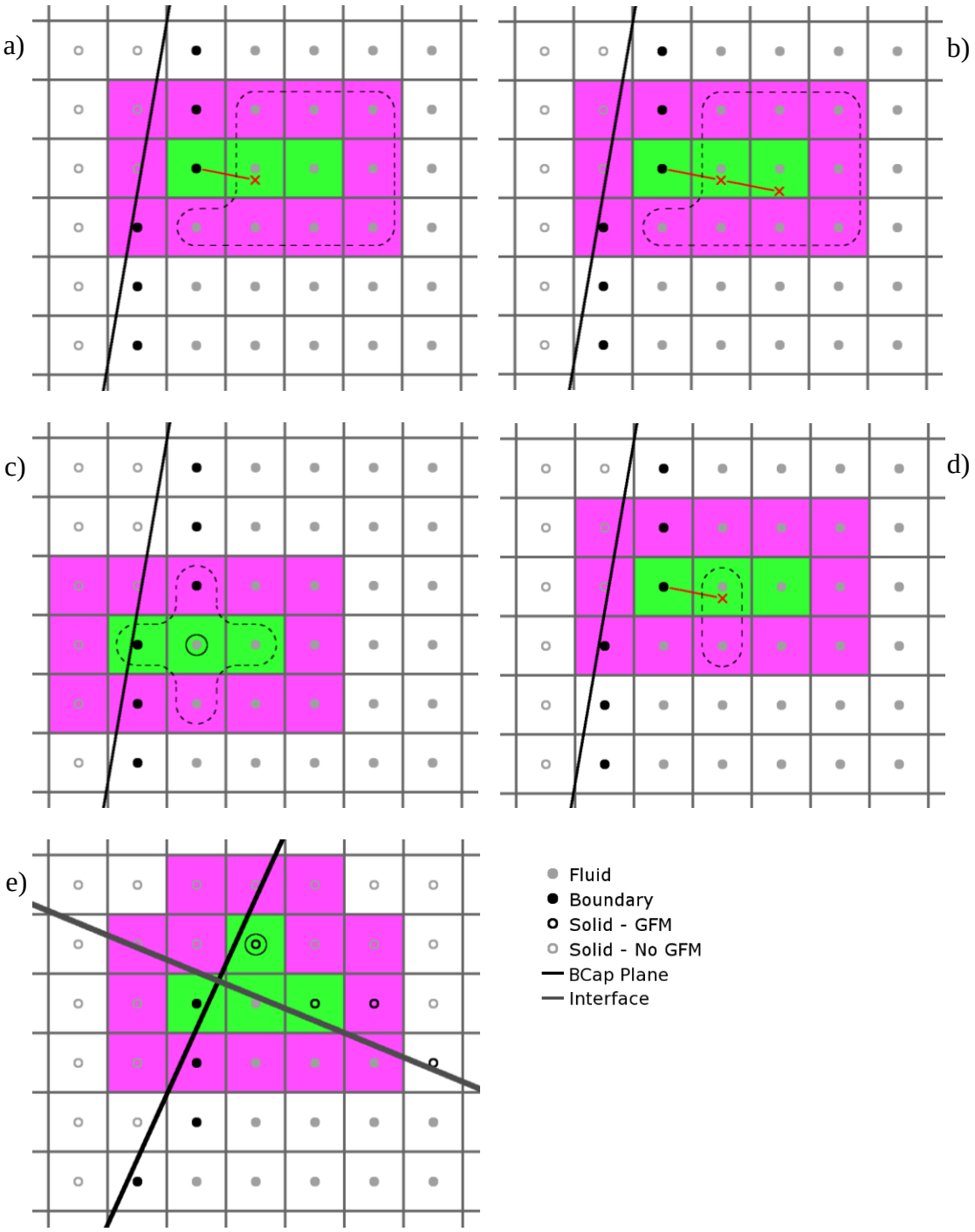


Figure 33: Worst case scenario for boundary cap pencil domain partitioning with green shaded cells located on the current domain partition and the single layer partition ghost cells in purple, a) sufficient least square cloud for first interpolation probe, b) sufficienct least square cloud for second interpolation probe, c) sufficient ghost layer for all fluid adjacent stencils, d) sufficent neighbors for linear interpolation of Neumann boundary condition, and e) modified pencil parition to include corner GFM ghost cell.

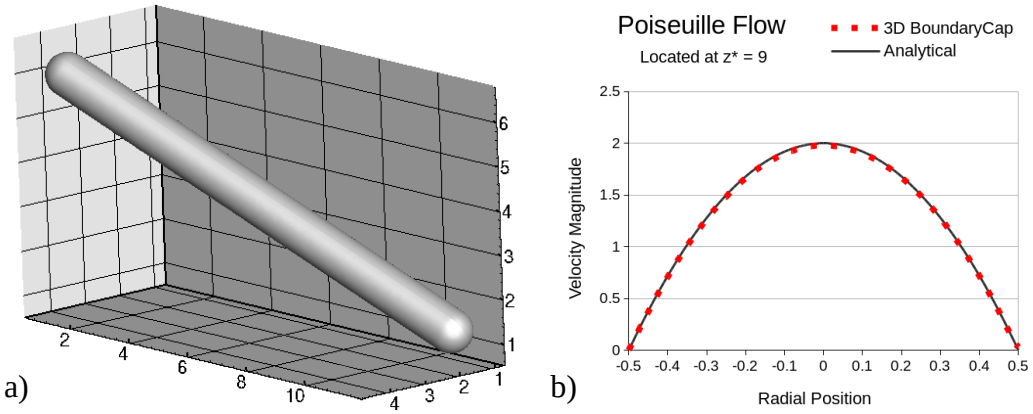


Figure 34: Validation of 3D boundary caps considering Poiseuille flow (Re =100), a) orientation of geometry in domain, b) developed velocity profile 9d from the inlet.

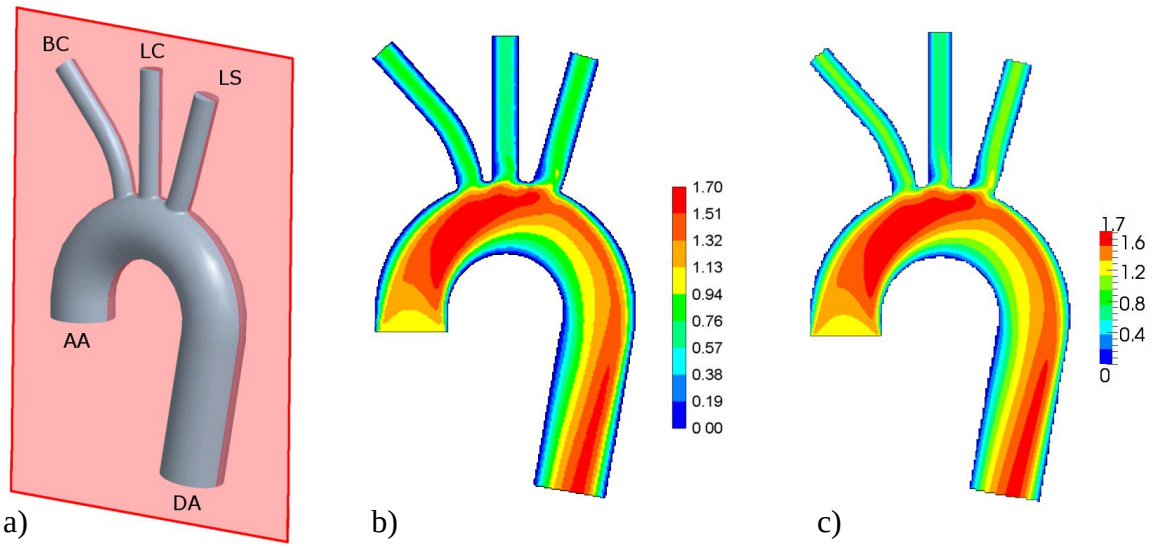


Figure 35: Mid-plane velocity magnitude comparison of 3D boundary caps with commercial software Fluent, a) boundary conditions, b) ANSYS® Fluent, c) pELAFINT3D using boundary caps.

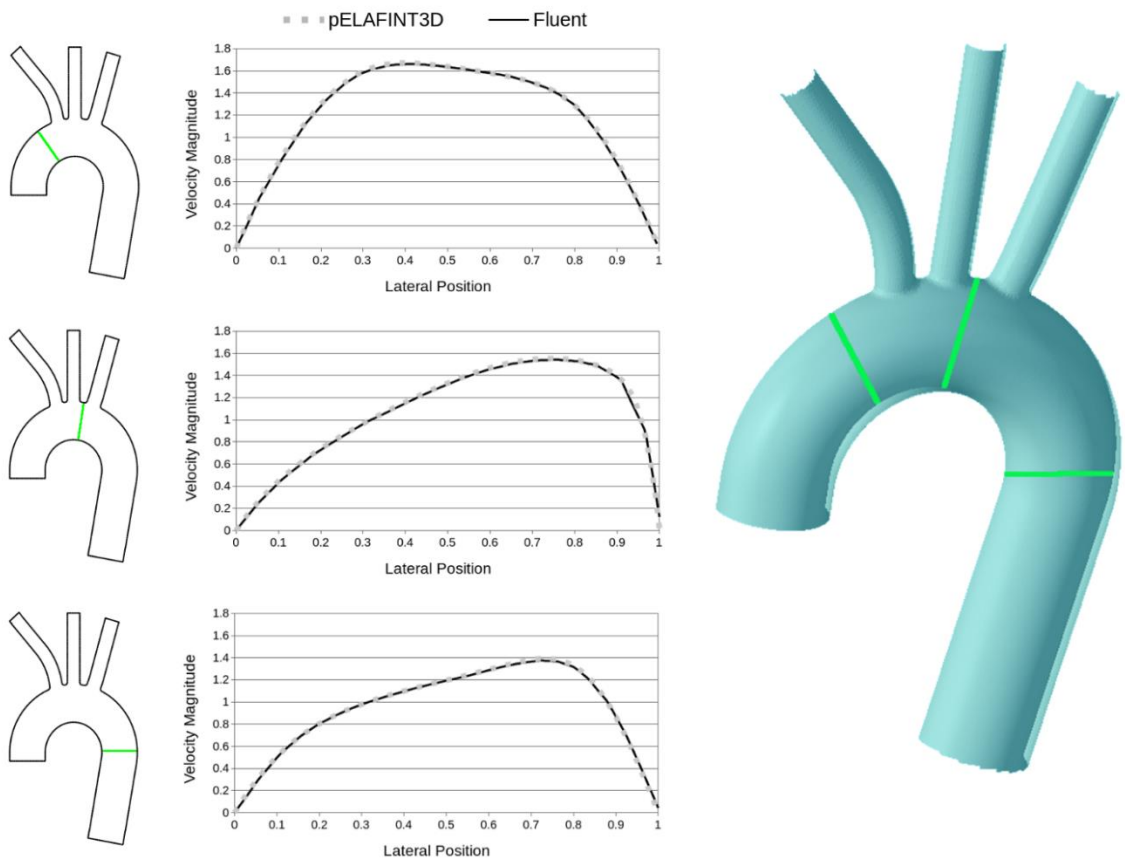


Figure 36: Mid-plane velocity profile comparison of 3D boundary caps with commercial software Fluent.

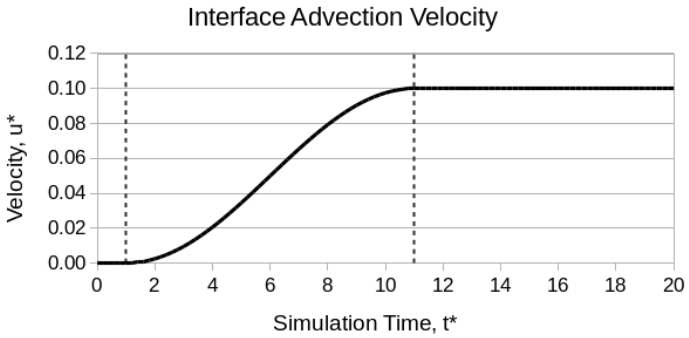


Figure 37: Horizontal advection velocity profile applied to a translating Poiseuille pipe for the verification of moving boundary caps and domain partitioning.

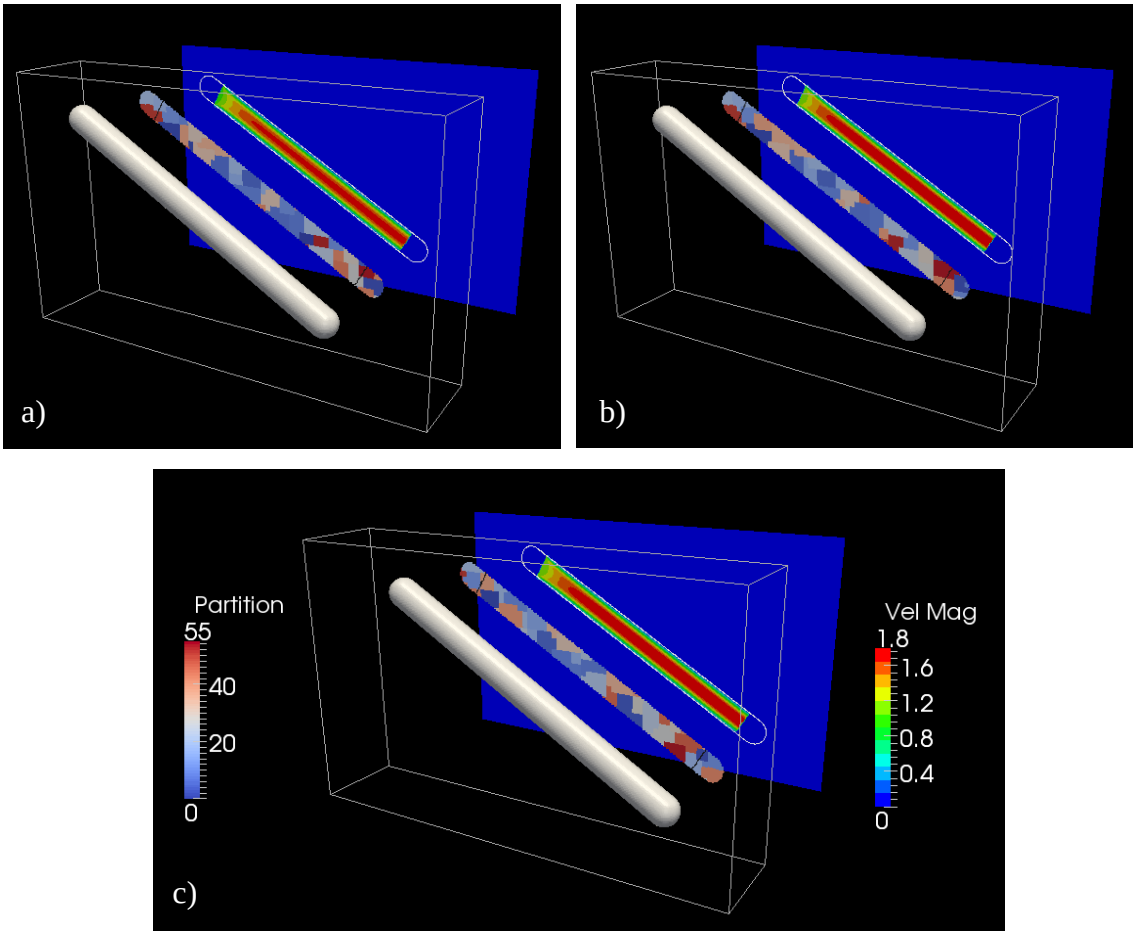


Figure 38: Results of horizontally translating Poiseuille flow (Re =100) for moving boundary cap verification showing the pipe geometry in front, a mid-plane of domain partitions, and a mid-place of velocity magnitude contours in the rear, a) results at t = 2, b) results at t* = 10, c) results at t* = 20.*

CHAPTER 6 RESULTS

As previously stated, the goal of this work is to select and develop the methods necessary to perform patient-specific left ventricle simulations in the clinical environment. Each method has been evaluated against the appropriate selection criteria and verified to show success against suitable benchmark cases. The next step is integrating these methods and applying them to left ventricle simulations. Only then, can the framework be evaluated to show whether this approach accomplishes the stated goals. This is achieved through a set of patient-specific left ventricle simulations which verify the results of the proposed computational framework.

6.1 2D Left Ventricle Simulation from MRI

The first verification case to consider is a 2D left ventricle simulation from a patient-specific image data set. While the fluid flow of the left ventricle has notable 3D characteristics, the 2D simulation is a good first step indicating the successful framework integration and general qualitative flow characteristics. The image data set consists of 32 MRI image frames that represent one cardiac cycle [100]. Select frames from the data set are shown in Figure 39. The left ventricle is positioned in these images such that the apex is located in the upper right. The ascending aorta and left atrium are located in the center left and lower left respectively. Both valves are in the closed position of the starting frame, Figure 39a, indicating the iso-volumetric phase of contraction. Figure 39b and c show the aortic valve in the open position for the systolic ejection phase. The aortic valve closes in Figure 39d, showing the start of diastole. Figure 39e and f show the mitral valve in the open position, indicating the diastolic filling phase of the cardiac cycle.

The endocardial surface must be extracted so it can be used as the moving boundary condition proving forward fluid motion. Segmentation of this surface is performed manually using CAD modeling software PTC Creo Parametric, version 4.0. The geometry is non-dimensionalized using the aortic annulus diameter as the characteristic length. The currently develop method of level set morphing using skeletons is used to interpolate intermediate locations needed for the flow solver. The skeletons are used to position two interior boundary conditions which provide the inflow and outflow boundary conditions.

Since valve leaflet modeling is outside the scope of this work, the leaflets will be excluded from the verification simulations. To achieve similar fluid results to human physiology, the inflow/outflow boundary conditions assigned to the interior boundary caps will provide a reasonable analog to the valve function. The simulation starts at the beginning of the ejection phase. For this portion of the simulation, the interior boundary cap applied to the left atrium is given of a Dirichlet velocity boundary condition of zero, mimicking the closed position of the mitral valve. A second boundary cap with a convective outflow boundary condition is applied to the aorta as would be the case for the open position of the aortic valve during the ejection phase. As the ventricle contracts, global mass conservation, as computed in Chapter 3, provides the mass flux to the outlet boundary condition. At frame 13, where the filling phase begins, the inflow/outflow boundary conditions are changed to mimic the valve positions in the filling phase. For this phase, the boundary cap in the left atrium is assigned a mass conserving inlet plug velocity mimicking the open position of the mitral valve. A zero velocity Dirichlet condition is provided to the boundary cap in the aorta to mimic the closed position of the

aortic valve. For this simulation blood is assumed to be a Newtonian fluid. Two full cardiac cycles are simulated with the results evaluated for the second full cycle.

Flow results for select frames of the cardiac cycle are displayed in Figure 40. While these flow results are limited due to valve exclusion and 2D simulation, they provide good initial indication of the proposed framework success. First, it can be noted the interior boundary conditions are successfully updating position, orientation, and size to follow the motion of the vessels. The interior boundary caps are applying reasonable inflow/outflow boundary conditions resulting in the fluid moving out towards the aorta during systole and fluid moving into the ventricle during diastole. Even with the limitations of this 2D simulation, fluid structures that align with previously reported LV hemodynamics can be identified, such as the fluid rotation during the filling phase as seen in Figure 40d-f [101]. These results are very encouraging and motivate the extension to a 3D simulation of left ventricular contraction.

6.2 3D Left Ventricle Simulation from Patient Data

A 3D simulation must be considered for verification of the proposed cardiovascular modeling framework. The patient-specific data for this test case consists of a time-sequence of points cloud data measured from the endocardial surface of the left ventricle. The specific data from this set is obtained from the Sunnybrook data set, patient 4101 [102]. This patient is described as a healthy, 23 year old male. Consistent anatomical positions were measured for a total of 20 time-steps completing the full cardiac cycle. Seventeen probe locations in total comprise the point cloud for every time-step. The point cloud data for a single frame is displayed in Figure 41a. CAD modeling software PTC Creo Parametric, version 4.0, was used to construct a set of splines through

the point cloud and thus construct a NURBS surface of the internal left ventricle wall. This is shown in Figure 41b for a single frame. Since the point cloud is consistently defined throughout the cardiac cycle, the spline construction can be easily update to achieve a volumetric representation of all input time-steps. Figure 41c shows the generated surface for each of the 20 input time-steps. To perform the desired simulation, suitable flow extensions are constructed to facilitate the inlet flow from the left atrium and outlet flow to the aorta. Both are sized according to typical valve annulus diameters, see Figure 41d. The extension at the mitral valve position (left) has an annulus diameter of 2.88cm while the aortic valve annulus (right) is modeled at 1.92cm. These dimensions are small enough as to consistently update the parametric model constructed of endocardial surface model, but large enough to be within the typical physiological size of a healthy adult. The geometry is then non-dimensionalized using the aortic valve annulus as the characteristic length.

6.2.1 Mesh-Based Benchmark Model

Similar to the 2D left ventricle simulation, this 3D simulation will not include the valve geometry. Without modeling the solid mechanics of the valve leaflets, the flow results will not be physiologic and won't match any patient flow measurements. The best option is to perform the simulation using an alternative method that is already widely accepted to be sufficient. To determine the success of the proposed framework, a benchmark simulation is performed using the traditional approach of polygon mesh morphing. A surface mesh that is consistent in both topology and connectivity will be generated for each of the image frames. This surface mesh can then be immersed into the fluid domain. For each flow solver time-step, the appropriate interface location will be

interpolated and a level set field will be constructed from that surface mesh. Since a level set field is the final step using a mesh morphing approach, the flow solver is not affected. A mesh morphing approach clearly has more computational steps than simply morphing the level set field, but it is the typical approach for left ventricle simulations.

Generation of that consistent (topologic and connectivity) surface mesh for every image frame is not trivial. The surface for the initial frame is constructed with triangular surface elements. Figure 42a shows this mesh for the first frame which represents the iso-volumetric phase of contraction containing a total of 5668 elements and 2837 nodes. A linear elastic material models is applied to all elements and a linear/static FEM simulation is performed using a collapsed node shell element. The specific shell element is formulated using the small displacement assumption. A Mindlin plate element is used to model the bending and shear effects. Reduced integration is applied to the shear stiffness term to minimize the effects of shear locking. A plane stress element is used to model the membrane effects. The first and forth nodes are collapsed to accommodate the triangular shape of the surface mesh. Since the left atrium flow extension is a consistent geometry, a simple distance metric can be used to first determine a set of displacement constraints which fully constrain the rigid body motion of the model. For the remainder of the surface mesh, a load is incrementally applied to each node. The load is proportional to the normal distance to the nearest point on the surface of the next frame. An iterative loop is performed adjusting the load until the mesh has been successfully deformed to match the next frame. This is successively performed for all 19 remaining input data sets. The results for selected frames are shown in Figure 42b-f. These frames are selected to represent the different phases and features of interest throughout the

cardiac cycle. Figure 42b,c represent the systolic ejection phase. Figure 42d,e show the rapid filling portion of diastole with Figure 42e located in the diastasis portion of diastole. Each mesh has a consistent topology, number of elements, and nodal mapping.

Since the mesh is immersed in the computational domain to generate a level set field, only minor changes must be made to the computational framework to accommodate a mesh morphing approach. The computational framework is modified so that upon simulation start, each of these 20 surface meshes is read from file. Since there is correspondence between the node locations on each mesh, cubic spline interpolation can be constructed for all 2837 nodes. During the simulation, each time-step will simply determine where it lies in relation to the frame sequence and interpolate the position accordingly. This cubic spline interpolation is also used to construct the suitable interface velocity which will be applied at the fluid interface for the no-slip boundary condition. 3D boundary caps are applied to this simulation in a similar manner as the previous 2D simulation. Initially, the left atrium boundary cap has an inflow velocity of zero with the aorta having an outflow boundary condition. At frame 8 the boundary conditions are changed with the aorta assigned a velocity of zero and the left atrium a mass conserving plug inlet velocity. Since skeletons aren't constructed for this benchmark simulation, a geometric reference to the rigid extensions is tracked and used to provide the boundary cap position, orientation, and size. Two full cardiac cycles are simulated with all results being extracted from the second cycle. This simulation is performed at a Reynolds number of 1000 which is large enough to be the same order of magnitude observed from measured data, but small enough to reduce the computational

burden of performing the simulation. All results are redimensionalized using a time scale of 70 beats per minute.

When considering the results of this benchmark case, features and metrics should be considered which provide good verification for the proposed framework of level set morphing. For comparison, it is important to consider the interface velocity. This is specifically the case in the long axis (vertical) direction, since obtaining an accurate tangential velocity is challenging, as described in Chapter 3. Figure 43 shows the long axis tangential velocity for the mesh morph simulation at various frames. As expected the apex has a positive vertical velocity during systole, Figure 43b and c, and a negative vertical velocity during diastole Figure 43d and e. The converse is true for the superior region of the left ventricle.

Next, the fluid flow characteristics are considered. Velocity magnitude glyphs are shown for select frames of the cardiac cycle in Figure 44. These results provide a good qualitative and quantitative measures to gauge the accuracy of the proposed level set morph framework. Streamtraces of the mesh morph results, shown in Figure 45, provide insight to the fluid rotation and vortical structures being generated during the cardiac cycle.

6.2.2 Level Set Morphing Framework Results

Now that a benchmark case has been competed by immersing a morphing mesh into the Cartesian domain, a simulation can be performed using the proposed level set morphing framework. As previously described, the level set morphing framework does not use any sort polygonal mesh to represent the surface. Instead it directly deforms the level set field representation of the interface. To provide verification of the level set

morphing method, a simulation is performed on the same geometry, boundary conditions, and flow parameters as the mesh morphing benchmark previously performed. Upon start of the level set morphing simulation, each interface target is analyzed to compute the left ventricle blood volume and a skeleton representation of topology. Between each pair of sequential target interfaces, a steady morphing velocity is computed in two steps. The topological shape change of the skeleton is used to compute a warp velocity and the fine detail changes in the interface are captured with a blend velocity computed from level set values. During the simulation the steady morphing velocities are interpolated between the pairs of sequential frame targets to achieve a continuous interface velocity. Each time the simulation time matches that of an input frame interface, the positional error is measured and used to construct a correction velocity reigning in any unwanted interface deviation. Inflow/outflow boundary conditions are again applied using interior boundary caps, but this time they are positioned using the previously developed skeleton interpolation scheme. This allows them to stay in the same anatomical location as the anatomy moves and deforms. As with the previous benchmark simulation, the exclusions of valve leaflets requires the inflow/outflow boundary conditions to mimic the check valve function normally provided by the heart valves. The simulation is again performed at a Reynolds number of 1000 for two full cardiac cycles.

Before comparing results to the mesh morph simulation, the positional accuracy of the interface can be evaluated against the input geometry. At each frame increment of the simulation, when the supplemental correction velocity is computed, the positional error of the surface is measured and recorded. Figure 46 plots the average and maximum positional error of the interface for each frame of the cardiac cycle. The mean error for

the simulation is redimensionalized to 0.026mm. This error is quite small when comparing to the aortic annulus diameter (characteristic length). The maximum positional error of the simulation occurs on frame 6 with a value of 0.816mm. These results demonstrate the successful ability of the level set morph framework to provide and accurate position of the interface throughout the entire cardiac cycle.

With the interface position verified, the interface velocity should next be considered. Obtaining the correct interface velocity is important to achieve the appropriate solid-to-fluid boundary condition, no-slip, and resulting fluid dynamics. Similar to the benchmark case using a morphing mesh approach, the vertical interface velocity of the level set morphing simulation is plotted on the interface for selected frames as shown in Figure 47. The vertical interface velocity matches quite well and provides good indication that the correct solid-to-fluid boundary condition is being applied at the interface. This suggests that the skeleton is a sufficient low-resolution representation of the left ventricle to apply the correct tangential interface velocity.

Finally, the flow characteristics should be considered. Since the interface position and interface velocities are in good agreement, the fluid dynamics should be expected to also agree. Velocity magnitude glyphs are shown for select frames in Figure 48. These results show good qualitative agreement with the mesh morph results of Figure 44. Streamtraces shown in Figure 49 indicate the fluid rotation and vorticity. Even without the inclusion of valve leaflets, a vortical ring is observed during the filling phase, Figure 49e. This too is consistent with the mesh morph results shown in Figure 45e. Qualitatively, the fluid dynamic results of the proposed level set morph framework match the benchmark results of the mesh morph approach. A quantitative comparison can be

made by measuring the near outlet velocity profile during the systolic phase. This is done for the systolic ejection frames 3 and 5, at a position of half an aortic annulus diameter upstream from the location of the aorta boundary cap. These positions are illustrated for frames 3 and 5 in Figure 50a and b respectively. Figure 50c plots the redimensionalized velocity profiles for frame 3 showing good agreement between the results. While the magnitude of the level set morph velocity is slightly higher, the shape of the velocity profile is very close. Figure 50c plots the velocity profiles for frame 5. The magnitude and shape of the velocity profiles are very close.

These benchmark cases have provided qualitative and quantitative measures to assess the capability of the proposed level set morph framework to perform left ventricle simulations derived from patient-specific data. While there are some very slight discrepancies between the level set morph results and the benchmark mesh morph results, it is difficult to know if these are significant without the inclusion of the valve leaflet models. Overall, the results are very positive and the level set morph results match the benchmark mesh morph results well.

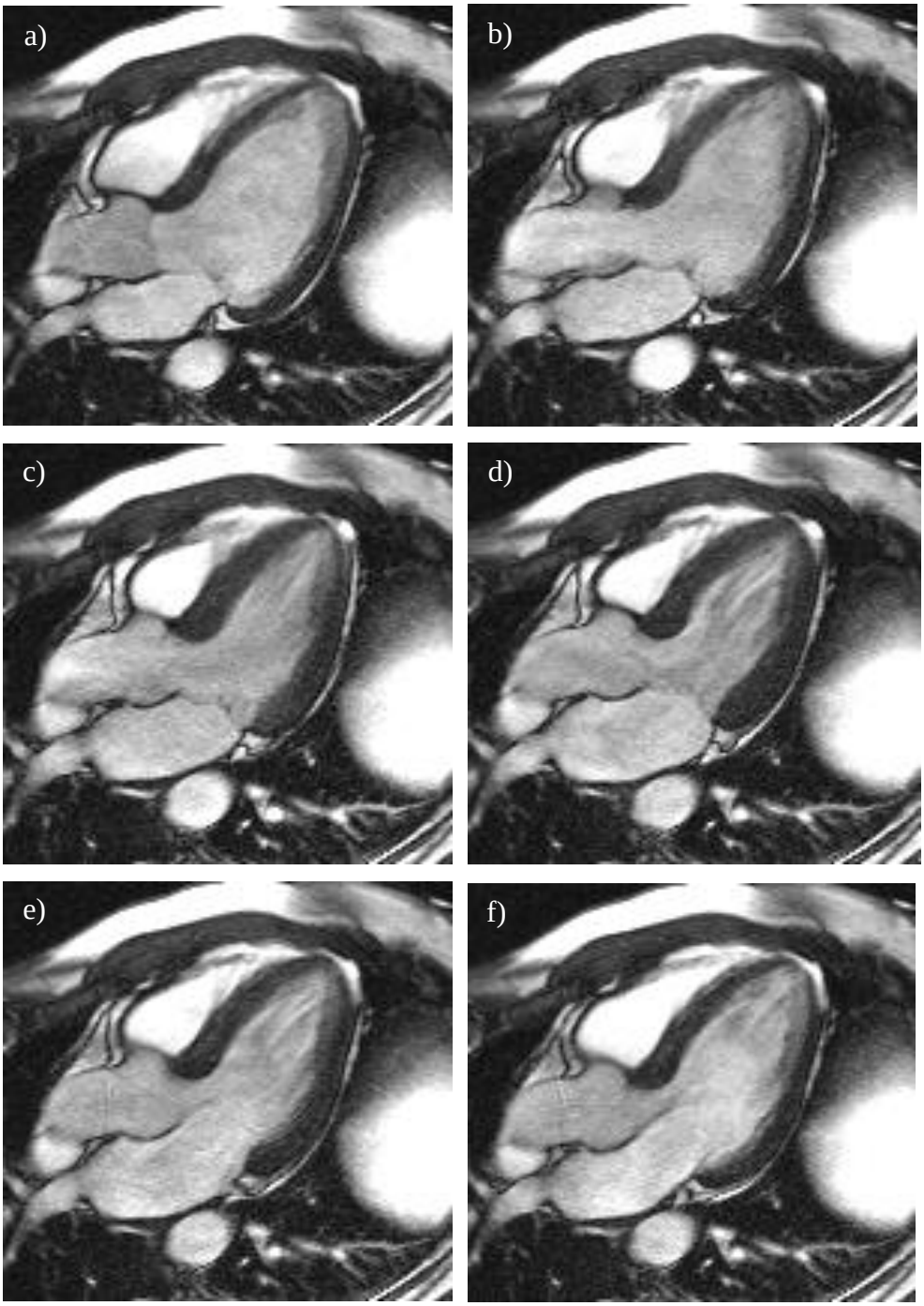


Figure 39: Long axis left ventricle MRI image data used for 2D flow simulation at select frames showing the left atrium (bottom-left), left ventricle (upper-right), and ascending aorta (center-left), (1,5,9,14,16,18).

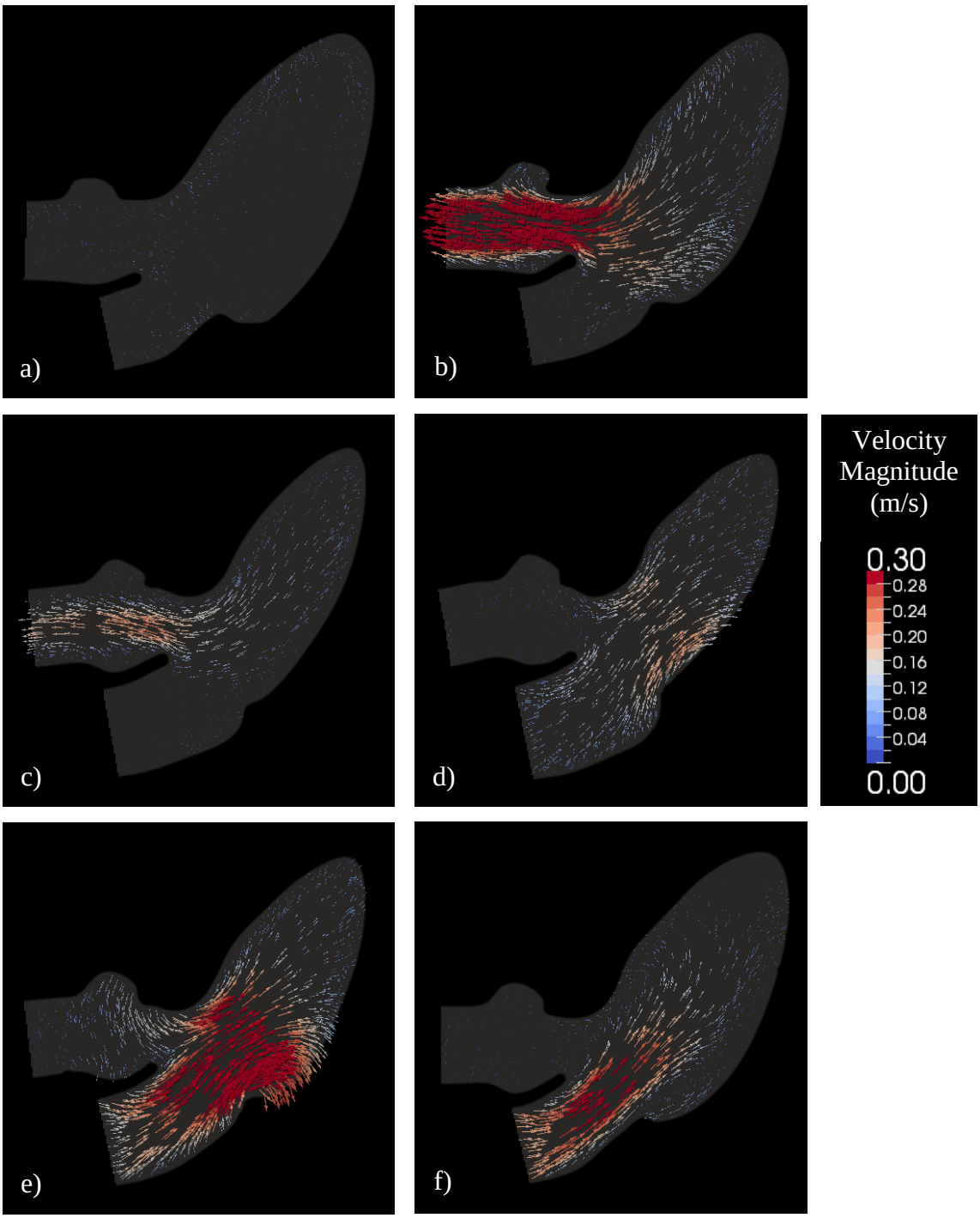


Figure 40: Velocity magnitude results of 2D left ventricle simulation at various frames (1,5,9,14,16,18).

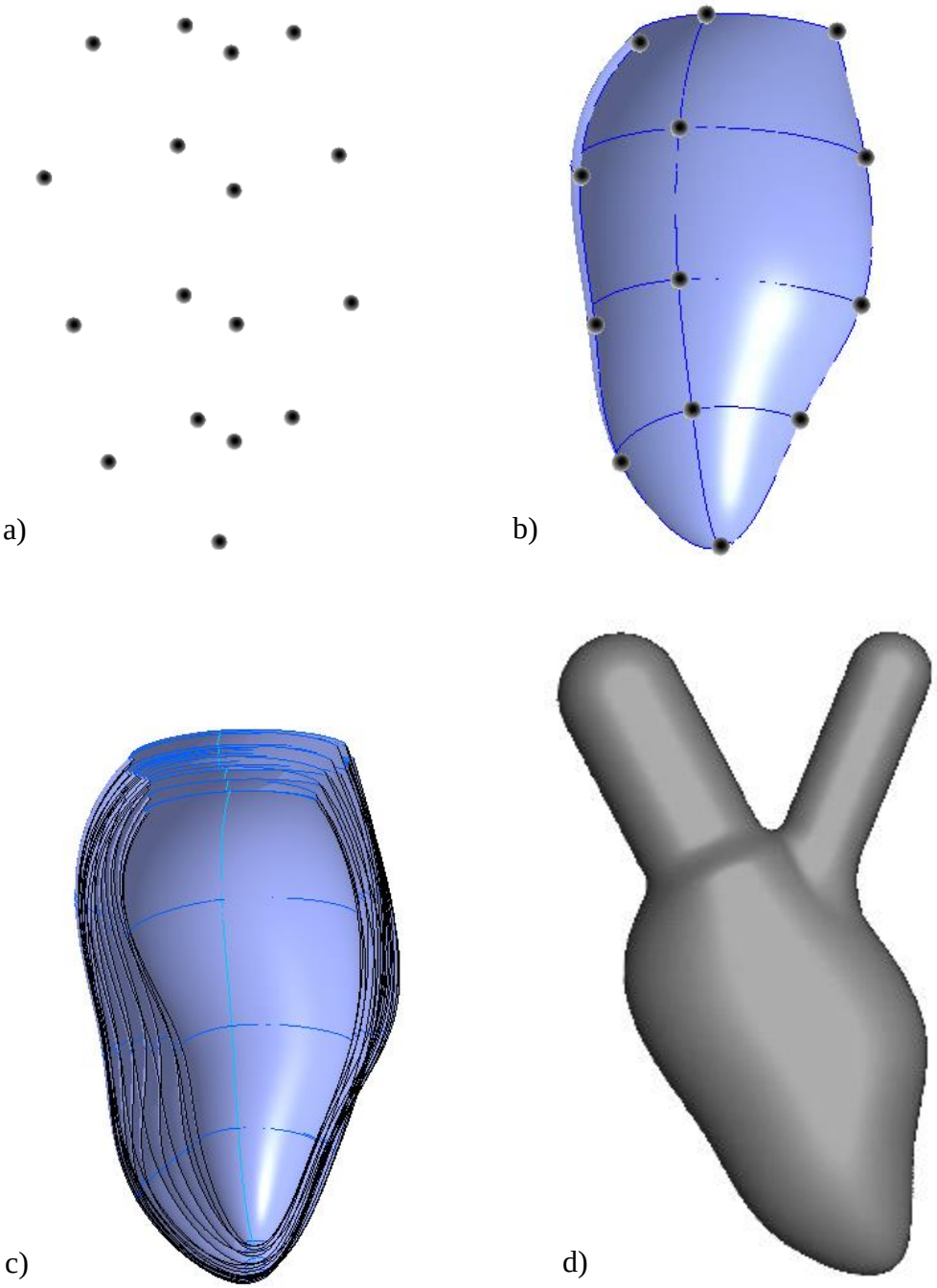


Figure 41: 3D reconstruction of left ventricle anatomy, a) Sunnybrook point cloud data (patient 4101), b) endocardium reconstruction using Creo Parametric, c) reconstruction of all frames, d) final geometry with suitable extensions for left atrium (left) and ascending aorta (right).

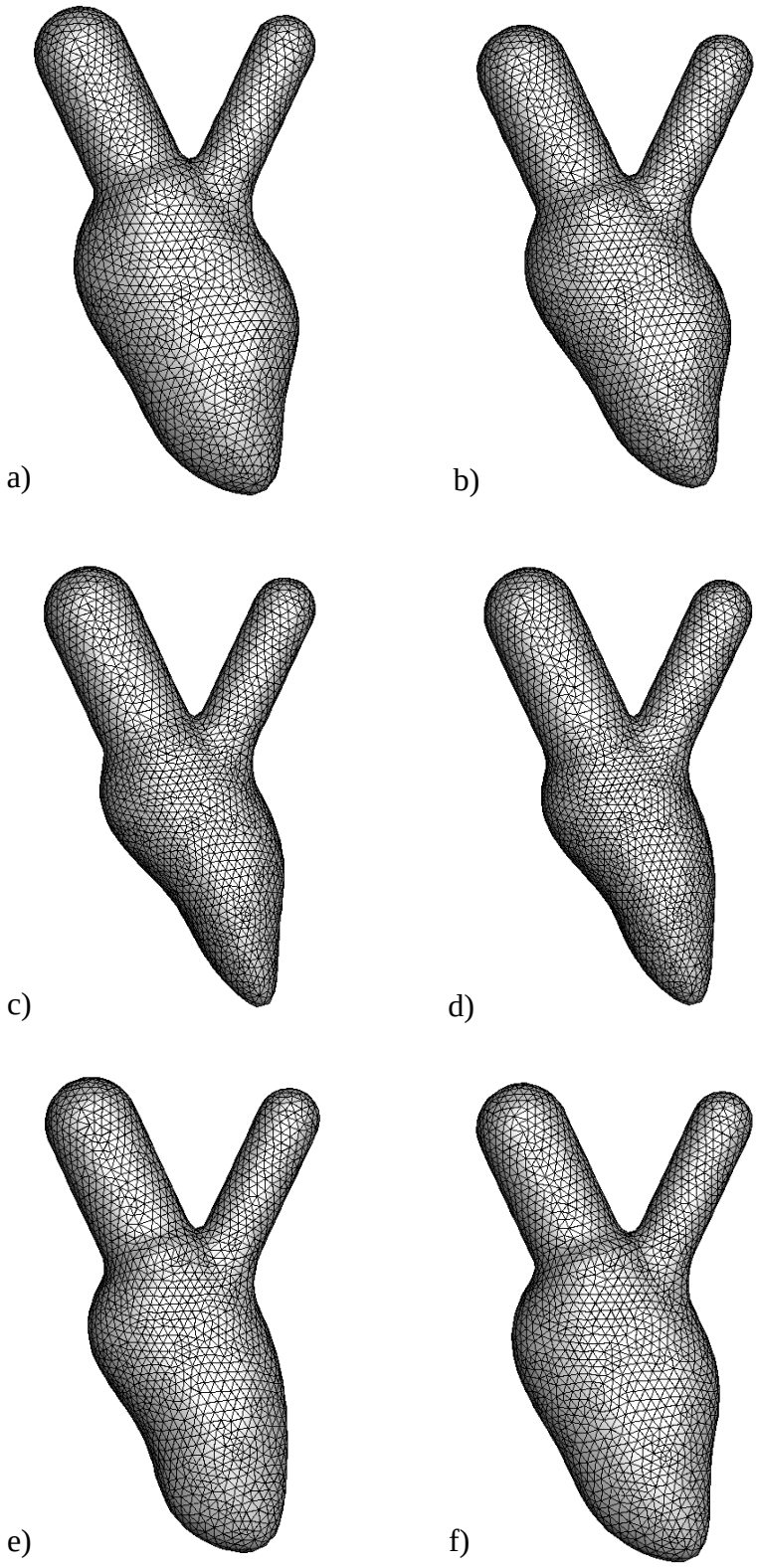


Figure 42: Input triangular surface meshes for 3D left ventricle benchmark simulation using the mesh morphing framework for select frames of the cardiac cycle (1,3,5,7,10,16).

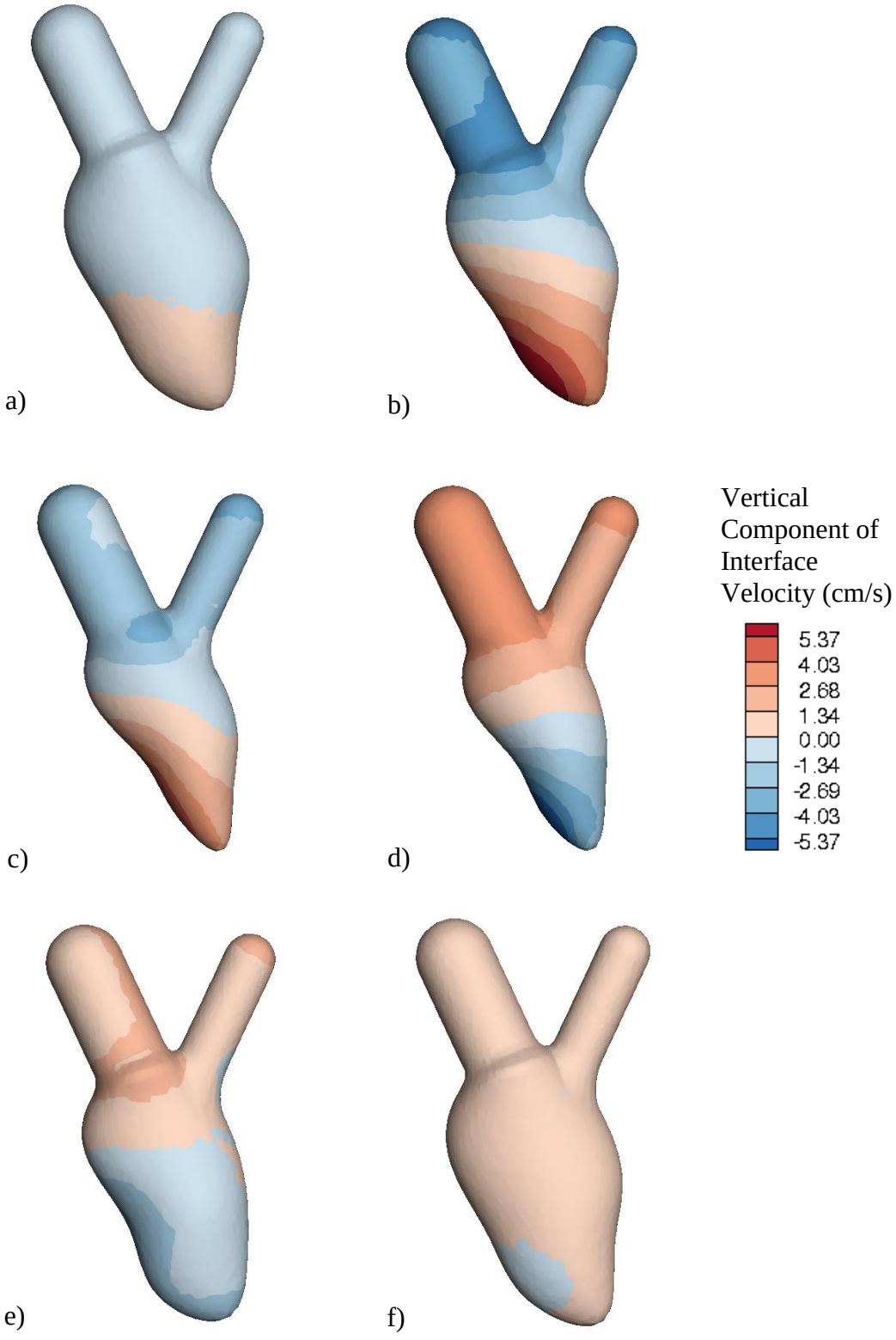


Figure 43: Vertical interface velocity results of 3D left ventricle benchmark simulation using the mesh morphing framework for select frames of the cardiac cycle (1,3,5,7,10,16).

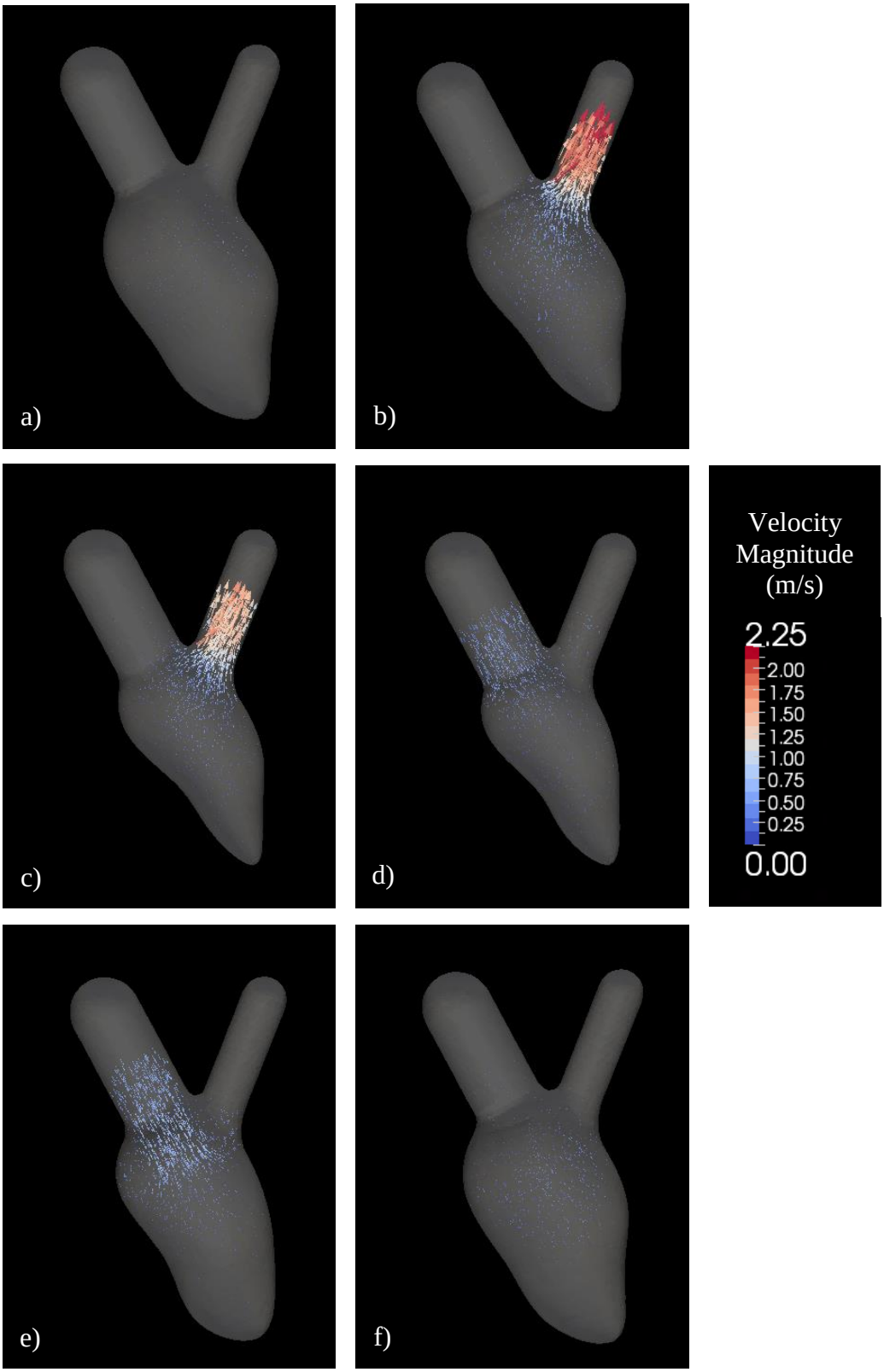


Figure 44: Velocity magnitude results of 3D left ventricle benchmark simulation using the mesh morphing framework for select frames of the cardiac cycle (1,3,5,7,10,16).

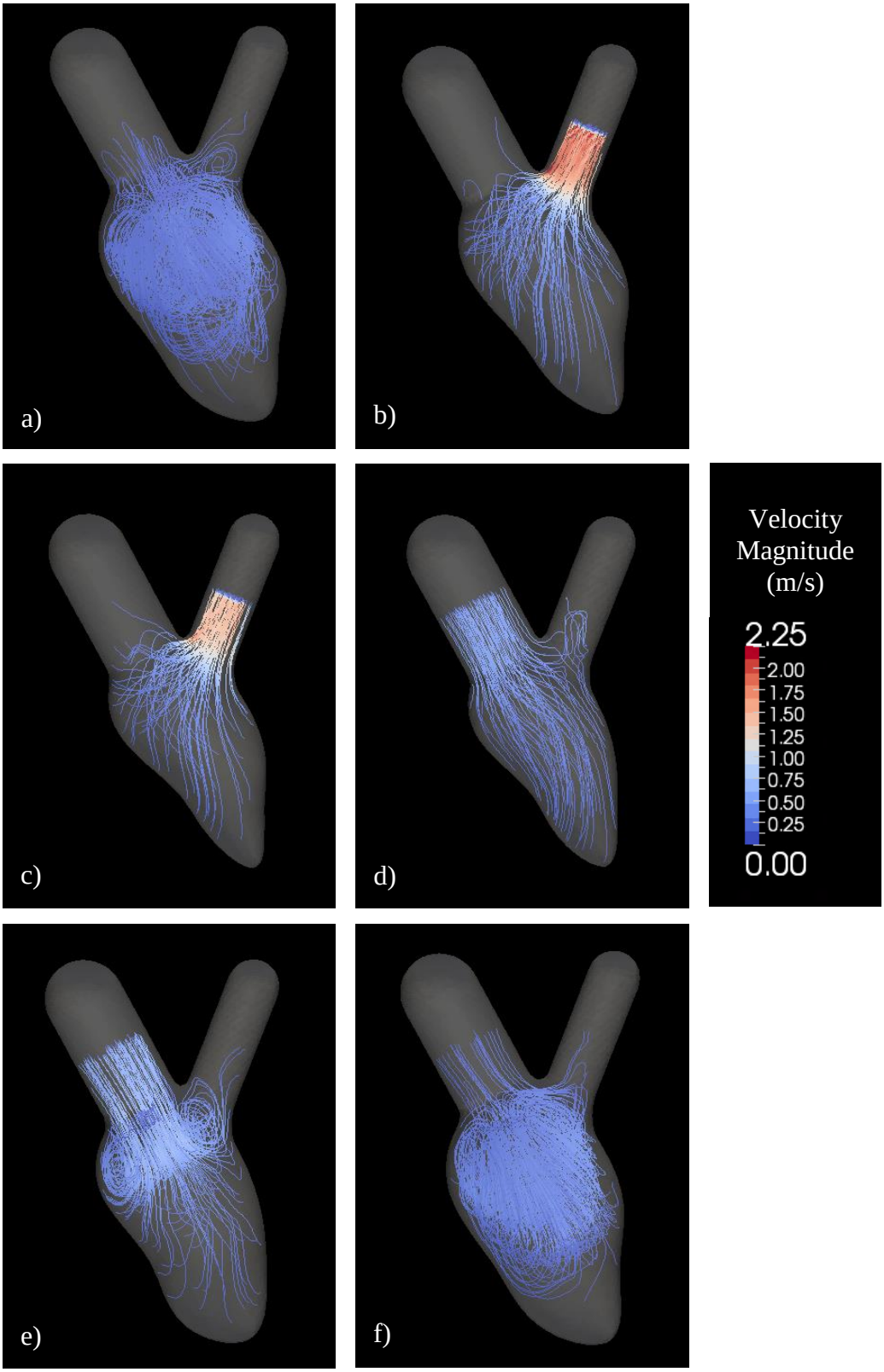


Figure 45: Stream trace results of 3D left ventricle benchmark simulation using the mesh morphing framework indicating fluid rotation and vortical structures for select frames (1,3,5,7,10,16).

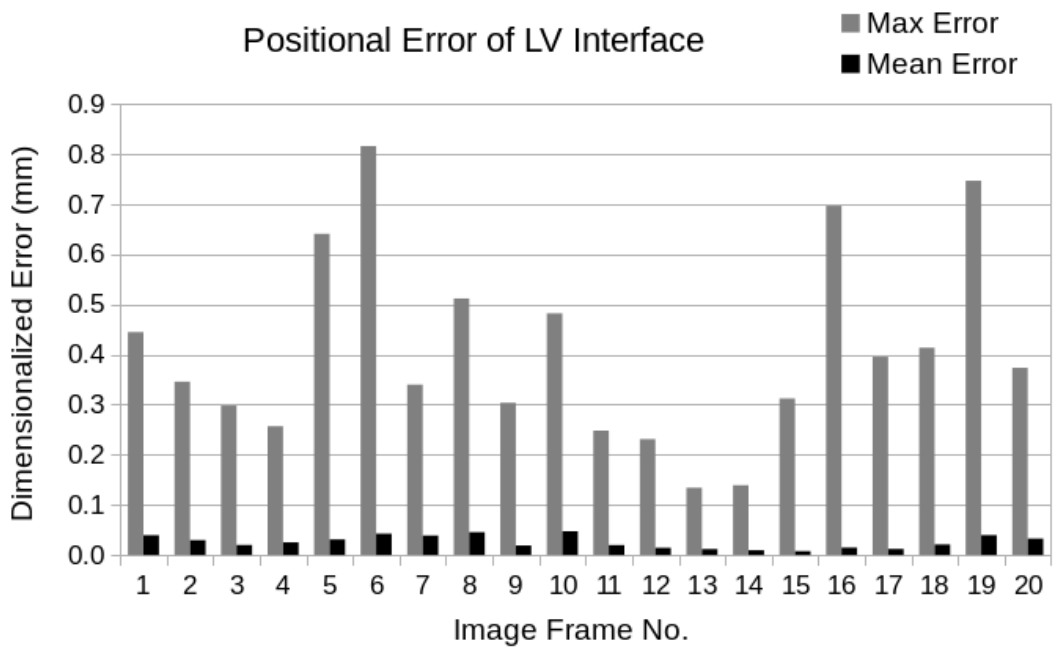


Figure 46: Plot showing the mean and median positional error of the interface using the level set morphing framework.

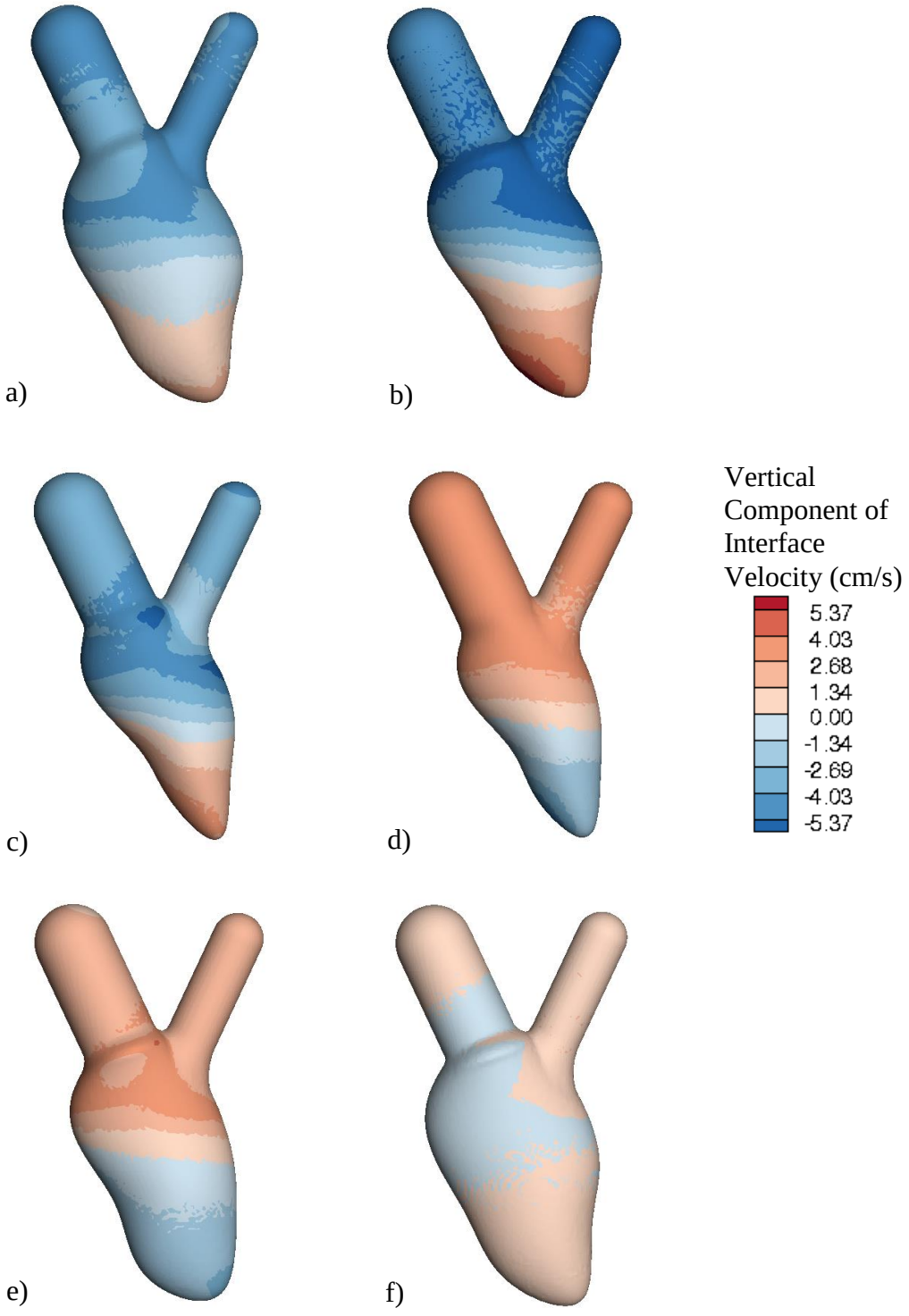


Figure 47: Vertical interface velocity results of 3D left ventricle benchmark simulation using the proposed level set morph framework for select frames of the cardiac cycle (1,3,5,7,10,16).

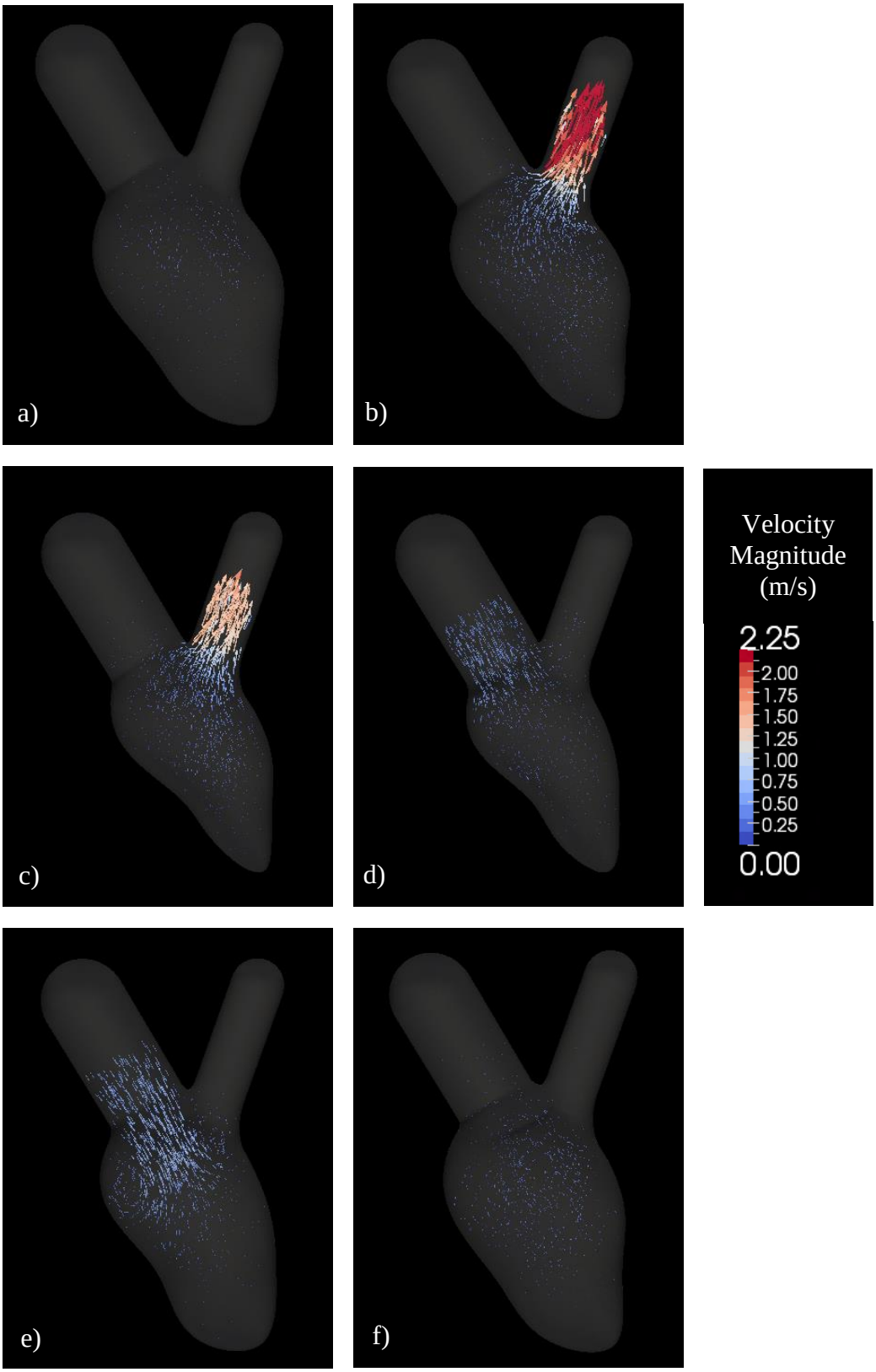


Figure 48: Velocity magnitude results of 3D left ventricle simulation using the proposed level set morph framework for select frames of the cardiac cycle (1,3,5,7,10,16).

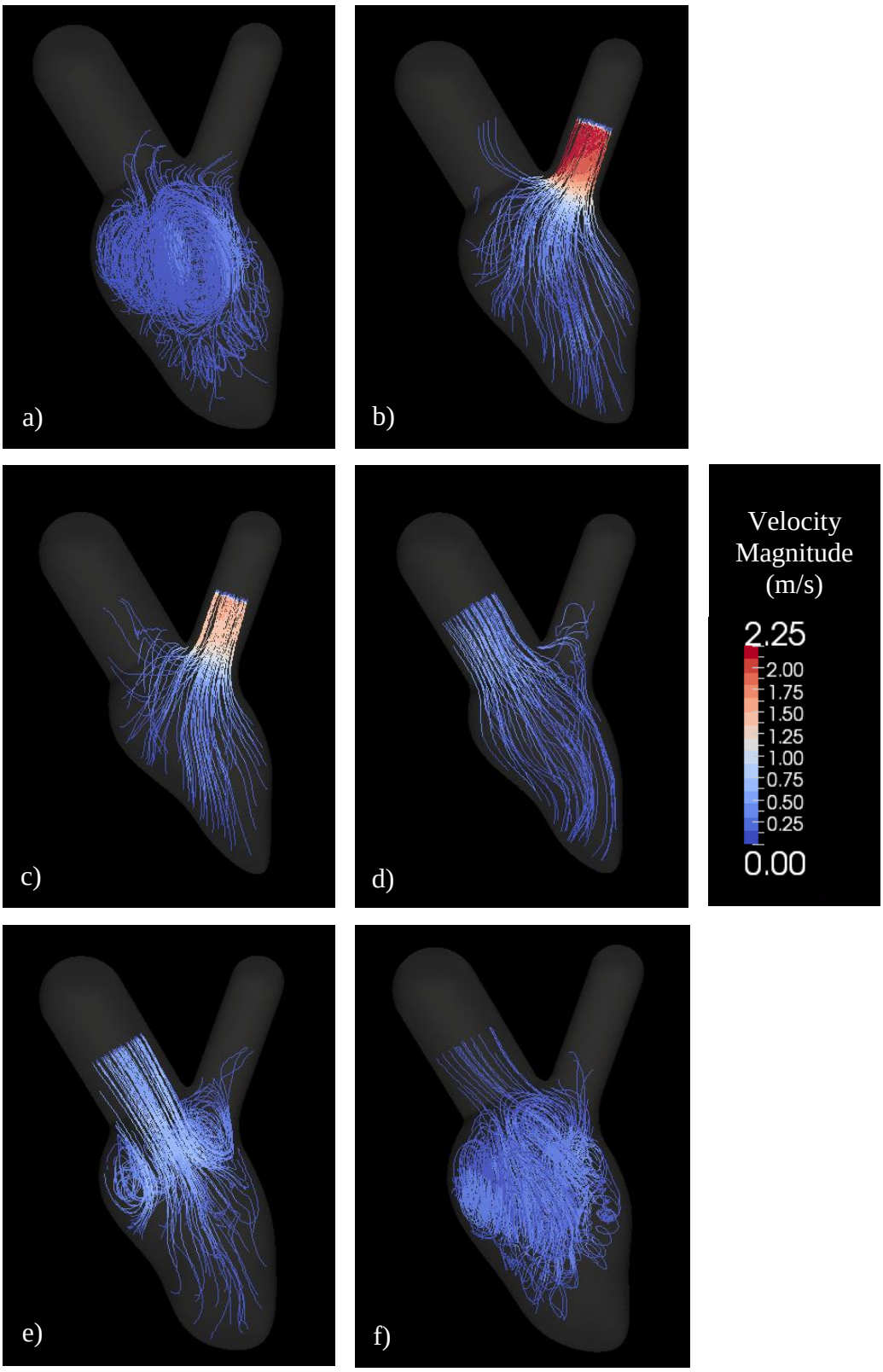
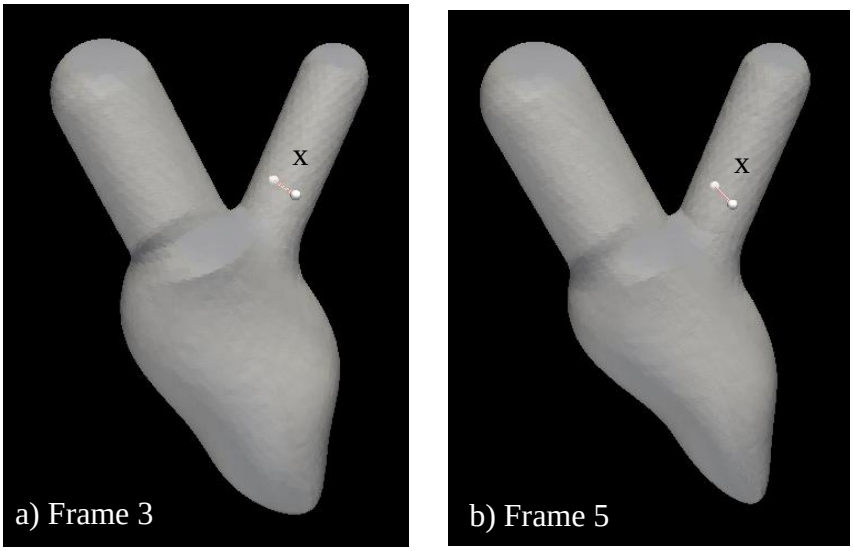
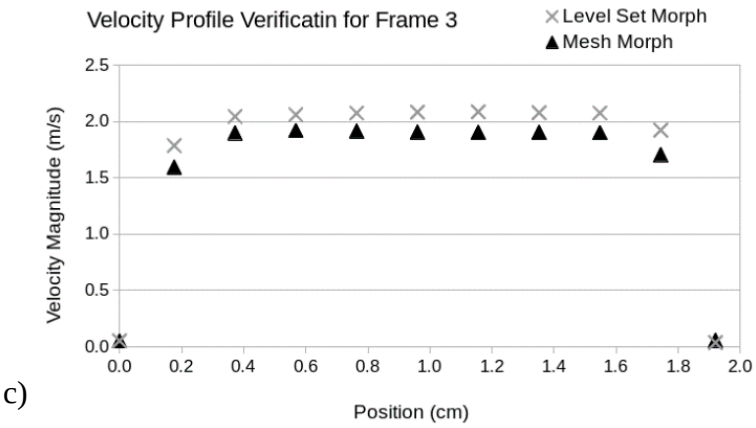
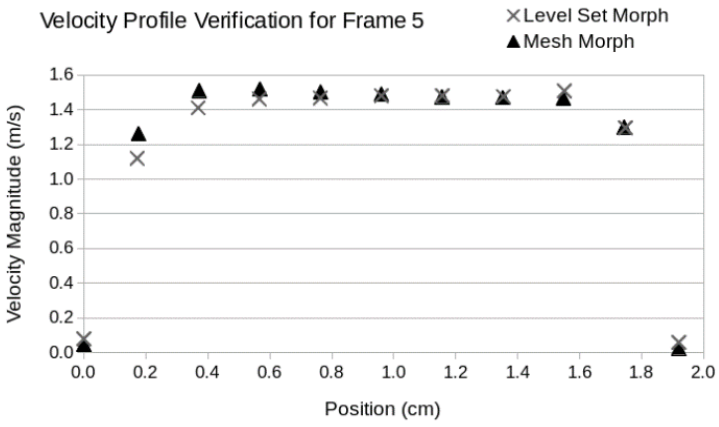


Figure 49: Stream trace results of 3D left ventricle simulation using the proposed level set morph framework indicating fluid rotation and vortical structures for select frames (1,3,5,7,10,16).





c)



d)

Figure 50: Velocity profile comparison between a benchmark simulation using a mesh morph and the proposed framework using a level set morph, a) location of profile 0.5 annulus diameter upstream from the outlet boundary condition (X) for frame 3, b) location of profile 0.5 annulus diameter upstream from the outlet boundary condition (X) for frame 5, c) velocity profile comparison plot for frame 3, d) velocity profile comparison plot for frame 5.

CHAPTER 7 LEFT VENTRICLE TWIST

As previously stated, the purpose of performing left ventricle simulation from patient-specific images is to correctly model the dynamics of valve leaflet motion. To do so, complete and accurate motion of the left ventricle wall must applied to the fluid flow simulation. Regardless of the left ventricle morphing strategy selected, the limited information contained in those medical images presents challenges in capturing and measuring certain aspects of motion. A medical image captures discrete intensity values on a discrete Eulerian domain consisting of a finite number of voxels (3D) or pixels (2D). The intensity value can offer information about different tissue types and their properties, thus allowing clinicians to discern specific organs and tissues, where regions of high intensity gradients are intuitively assumed to be the boundary or interface between differing tissue types. Through the use of common segmentation methods such as active contours [54]–[56], the large intensity gradients can automatically be extracted to create the geometric boundary needed for engineering analysis. Often a sequence of medical images is taken at discrete (order of milliseconds) time intervals allowing the clinician to infer motion. This allows for the extraction of the geometric boundary at many discrete points in time. Since the series of time-sequenced patient geometries are generated from an Eulerian reference frame, the Lagrangian information can only be measured for specific landmark locations. Information pertaining to the exact interface displacement or interface velocity at any specific location cannot be directly measured. This Lagrangian information is needed to apply the no-slip boundary condition when simulating a solid moving through a fluid domain, which is the case for a left ventricle simulation regardless of the method chosen for interface morphing. To illustrate this

challenge inherent to medical images, consider the case of a rotating circle as shown in Figure 51. Since there are no landmarks on the circle it is impossible to know from the image sequence whether the circle is rotating or not. In contrast, consider an image sequence which contains sufficient landmarks allowing one to infer the tangential motion and thus the rotational speed of the circle, as depicted in Figure 52.

This example is very similar to the case of left ventricle simulations. It is known that the muscle fibers of the myocardium are arranged in a geodesic manner causing the left ventricle to twist as it contracts [103]. Unless additional information is supplied, the boundary condition obtained from the series of images, will not include the twist component of the interface velocity. Since the no-slip boundary condition is used, the missing component of interface velocity (twist component) will also be missing from the fluid velocity near the interface and thus effecting the remainder of the flow field. This limitation exists regardless of the interface morphing method selected, since the twist information is not available from the series of medical images. The magnitude of the twist velocity has been measured to be on the order of cm/s, [60] while the outlet jet velocity through the aorta is on the order of m/s [61]. A comparison of these two fluid velocities shows approximately two orders of magnitude difference, suggesting the twist velocity of the ventricle may not have a noticeable impact on leaflet dynamics. On the other hand, there is evidence to suggest the vortices induced in the left ventricle influence the filling dynamics [62]. While the twist velocity has been included in one left ventricle simulation [18] and not in others [19], [19], [104], the author knows of no direct comparison for which the same left ventricle simulation was performed with and without

the inclusion of the tangential twist velocity, which is necessary to determine the role that this plays on the flow and the resulting impact on the valve leaflet dynamics.

7.1 Application of a Supplemental Tangential Interface Velocity

To run a comparison study, a means of applying a supplemental tangential surface velocity must be added to the computational framework. Two obvious options are to 1) add the supplemental tangential interface velocity to the level set advection velocity or 2) apply it to the immersed boundary treatment of the moving interface. To avoid any unintended interface distortions during the level set morph, the latter choice in which the tangential velocity is supplied to the immersed boundary treatment is chosen for implementation. As previously mentioned, the immersed interface treatment utilized for the selected framework is a least-squares implementation of the Ghost Fluid Method, solved on a four-step pressure correction scheme [30]–[32]. In this case the supplemental tangential velocity is added to the interface velocity of the level set field when applying the Dirichlet (no-slip) boundary condition to the Helmholtz equation being solved for the provisional velocity. This modification is shown in the equation below for a Dirichlet boundary condition where the ghost cell velocity is extrapolated from the values of the interface velocity and a probe positioned inside the fluid domain. The supplemental tangential velocity is simply added to the interface velocity to provide the correct interface boundary condition.

$$\mathbf{u}_{ghost} = 2\left(\mathbf{u}_{int} + \mathbf{u}_{\sup}\right) - \mathbf{u}_{probe} \qquad 7.1$$

To verify successful implementation of the supplemental tangential surface velocity, an analytical test case similar to the rotating circle described in Figure 51 is considered. Taylor-Couette flow is a classic fluid mechanics problem consisting of fluid between two long concentric cylinders with the inner cylinder rotating at an angular velocity, Ω [105]. A diagram of this problem is shown in Figure 53. As the angular velocity is increased, the flow becomes increasingly unstable. The Taylor number is a non-dimensional value which indicates the point in which the flow will reach the first instability. The Taylor number, Ta, is defined in the equation below.

$$Ta = \frac{r_i(r_o - r_i)^3\Omega_i}{\nu^2} \tag{7.2}$$

As long as the flow is below a Ta ≈ 1700, the flow should be stable and match the analytical solution shown below [106].

$$u_\theta = \Omega_i r_i \frac{{r_o}/{r} - {r}/{r_o}}{{r_o}/{r_i} - {r_i}/{r_o}} \tag{7.3}$$

In this test case, the only interface velocity applied to the fluid is that of the supplemental tangent surface velocity. A 3D simulation of this test case is performed at Ta = 1000. Flow results are visualized in Figure 54. As predicted by the analytical solution, the fluid velocity is largest near the center rotating cylinder and quickly drops to become zero at the wall of the outer cylinder. A radial polyline of data is extracted from the flow results to compare the simulation to the analytical. As can be seen in Figure 55,

the results match very well, thus giving good indication of successful implementation of the supplemental tangential velocity.

7.2 Mapping the Twist Velocity to the Left Ventricle Interface

It is clear from studies measuring the myocardial twist velocities that the magnitude and direction of twist varies depending on position within the left ventricle [103], [107]. The values are measured and reported as a function of position along the long axis of the left ventricle. Thus a means of mapping the axial position to the interface is needed to apply a suitable representation of the supplemental twist velocity. Conveniently, the level set morph framework already uses one such mapping generated from skeletonization. Thus the comparative study will apply a time-varying function extracted from Jung et al. [107], to the portion of the interface mapped to the skeleton segment which descends toward the apex of the left ventricle, displayed as the orange region of Figure 17d. The time varying twist function is plotted in Figure 56. Since the skeleton mapping is unique for each morph target, the supplemental twist velocity must be computed for each flow solver time-step using much of the same information as the level set interface morphing routines. First, the non-dimensional time location in the cardiac cycle, which is the same time value used for computing the global mass flux, is applied to cubic spline interpolation of the twist velocity function. Next, the skeleton mapping value must be interpolated similar as was done in Chapter 3. Note that α denotes the skeleton mapping percentage and t denotes the simulation time. This is computed for each cell in the level set tube that is associated to the skeleton segment that descends into the left ventricle.

$$\alpha_{ic} = \left(1.0 - MOD\left(t, t_{fdur}\right)\right)\alpha_{sk1} + MOD\left(t, t_{fdur}\right)\alpha_{sk2} \quad 7.4$$

Since the supplemental tangent velocity is only being applied to a region of the interface, a tapered weighting function must be mapped for the surface. An example of the weighting distribution of the velocity function is shown in Figure 57. This is accomplished by inverting and thresholding the interpolated skeleton mapping value. As can be seen from this figure, interpolated skeleton mapping values of less than 0.8 are weighted at a value of 1.0, while the skeleton mapping values of 1.0-0.8 are tapered from 0.0-1.0. This specific implementation is achieved through the equation below.

$$w_{ic} = 5.0 \cdot MIN(1.0 - \alpha_{ic}, 0.2) \quad 7.5$$

The weighting value combined with the interpolated twist velocity provides the supplemental twist velocity magnitude, but a direction vector is still needed. This can be defined in a variety of ways, but the author selected to make a local evaluation of the vector based on the skeleton mappings. For this approach, the direction vector will be established through the use of the skeletons. Using the existing tube-to-skeleton mapping created for level set morphing, a location can be identified on the skeleton at each end of the current frame interval. The neighboring skeleton nodes are used to determine a vector tangent to the skeleton at that location for each of the two frame interval skeletons. These two direction vectors are interpolated to achieve a skeleton direction for the current time-step as shown in the equation below.

$$\mathbf{n}_{skel} = \left(1.0 - MOD\left(t, t_{fdur}\right)\right)\mathbf{n}_{sk1} + MOD\left(t, t_{fdur}\right)\mathbf{n}_{sk2} \quad 7.6$$

Cross product of this skeleton direction vector and the interface normal, which is provided by the morphing level set field, produces the final direction vector for which the supplemental twist velocity is to be applied. This is shown in the equation below where $\mathbf{n}_{ic}$ denotes the interface normal at the current cell.

$$\mathbf{n}_{sup} = \mathbf{n}_{ic} \times \mathbf{n}_{skel} \tag{7.7}$$

The final supplemental twist velocity is computed using the following equation with $u_{twist}(t)$ representing the twist velocity function of Figure 56.

$$\mathbf{u}_{sup} = w_{ic} \cdot u_{twist}(t) \cdot \mathbf{n}_{sup} \tag{7.8}$$

7.3 Left Ventricle Simulation with Supplemental Twist Velocity

A comparable level set morph simulation to that of the previous chapter is performed, but also includes the supplemental twist velocity as defined in the previous section. The flow results of this simulation are shown below in Figure 58 for the same frames as previously plotted for the simulation that neglected twisting velocity. Stream trace results are also plotted in Figure 59. The general characteristics of the flow field seem fairly unchanged indicating that the twist velocity has a relatively minor effect on the fluid dynamics. A closer inspection of the region near the aortic valve will indicate likelihood of the supplemental twist velocity having an impact on the simulation of valve leaflet dynamics. To inspect the flow at the valve location, cross-sectional fluid velocity data is extracted from a position half the annulus diameter upstream from the location of the outlet boundary condition. This data is extracted for frames 3 and 5 corresponding to the ejection phase of systole. Figure 58 plots the comparison of velocity profiles between

the level set morph with and without the supplemental twist velocity. The difference between the results is noticeable, but very small. While this may be evidence that the twist velocity for left ventricle simulations can in fact be neglected, it is the author's opinion that simulations which include heart valves must be performed to make any such conclusion. This is especially important when considering how the vortical structures (created during the filling phase) may affect the closing dynamics of the mitral valve. While these results are not definitive, this has shown itself to be a simple and effective means of applying a supplemental tangential surface velocity to a level set morphing simulation.

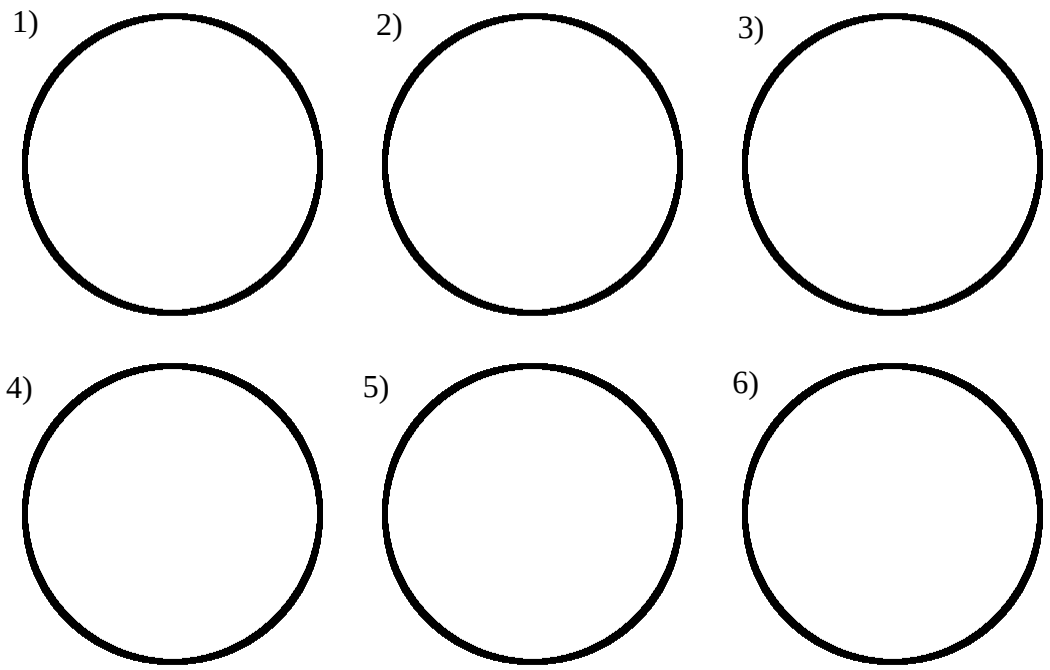


Figure 51: Image sequence of a rotating circle demonstrating that without landmarks, it is not possible to detect the motion.

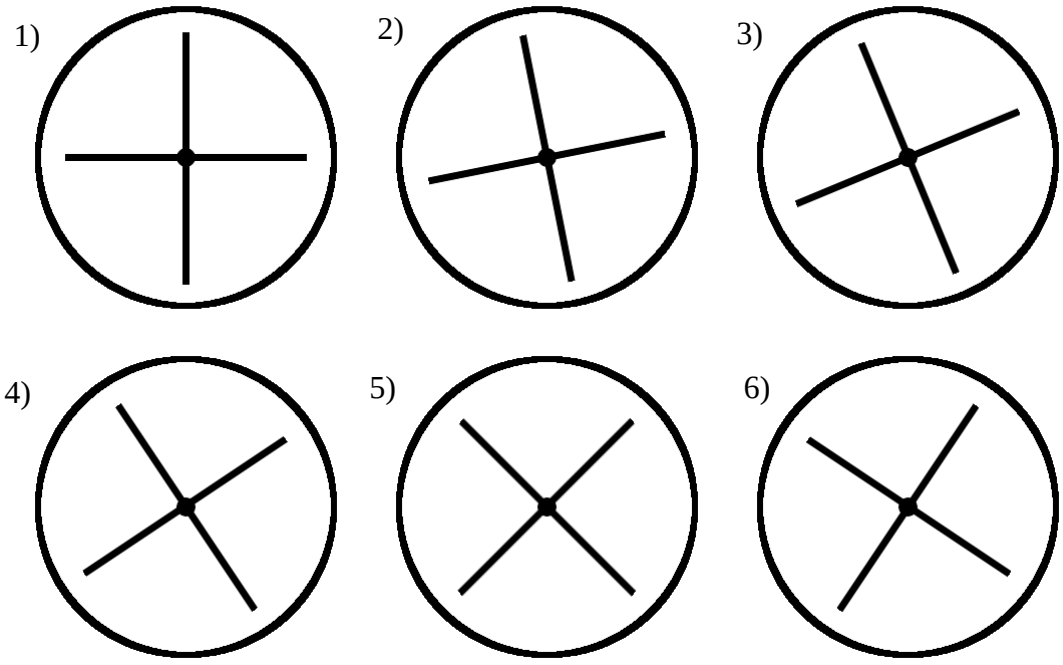


Figure 52: Image sequence of a rotating circle showing how an effective landmark can infer object motion.

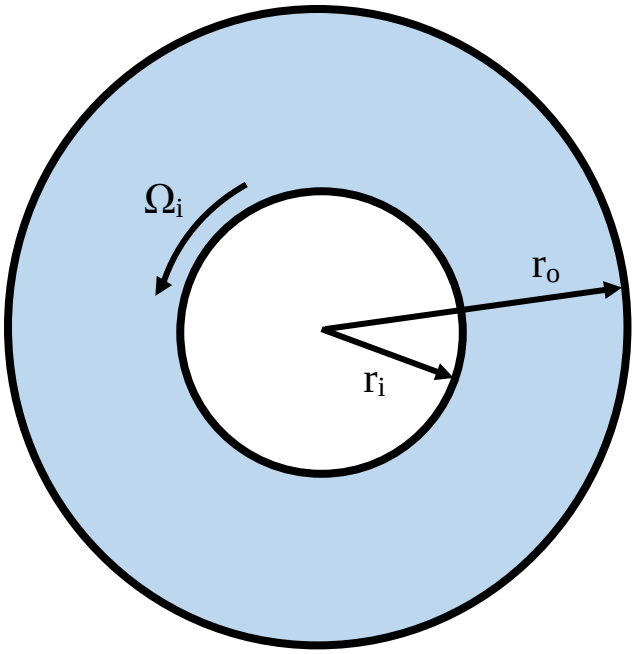


Figure 53: 2D diagram describing the configuration of Taylor-Couette flow between two cylinders with the inner one rotating.

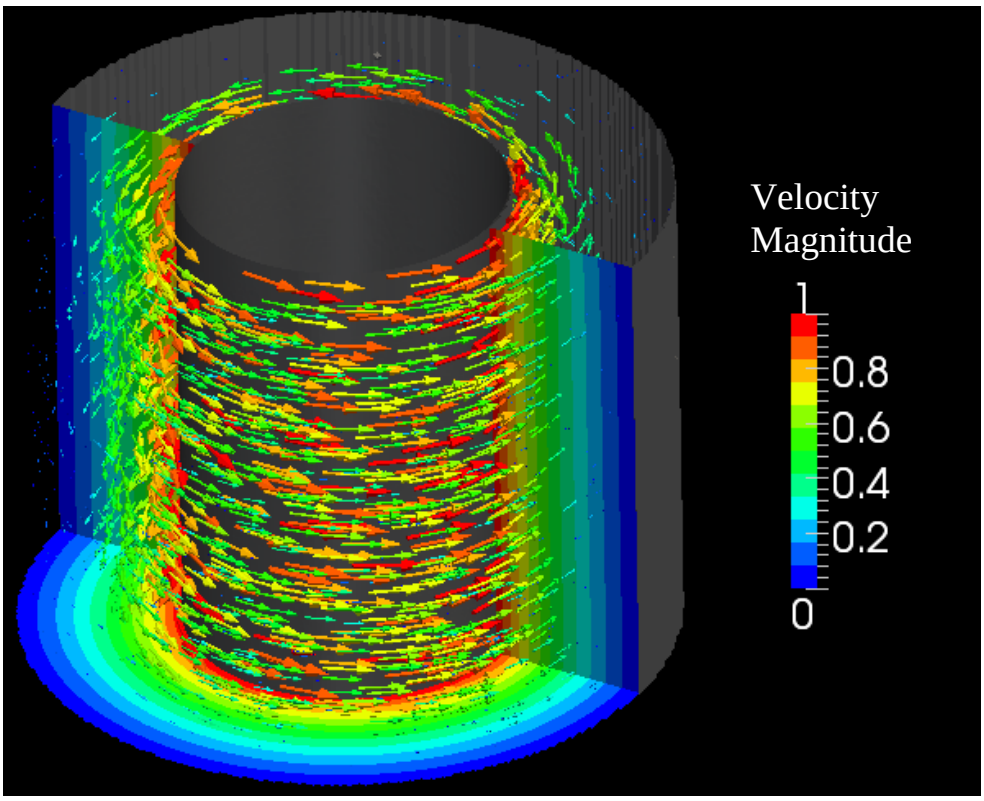


Figure 54: Velocity magnitude results of Taylor-Couette flow at a Taylor number of 1000 with the outer cylinder stationary and the inner cylinder rotating

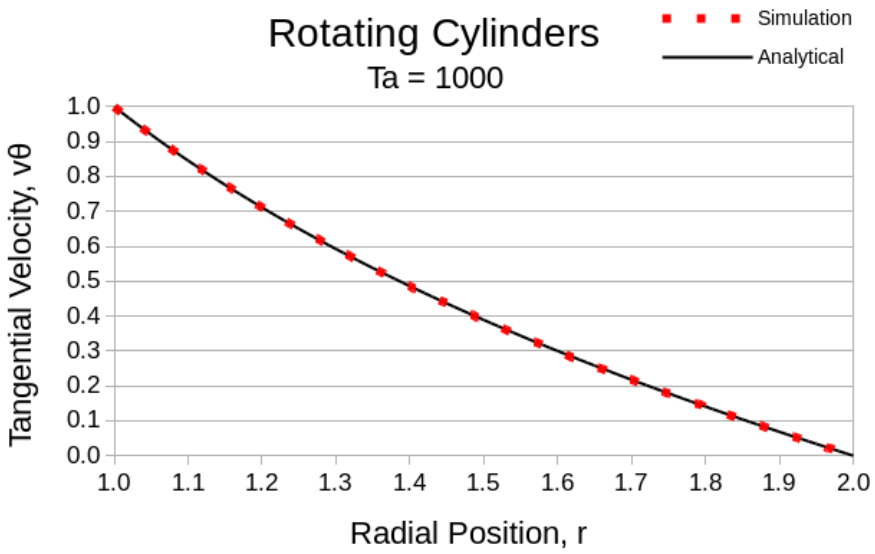


Figure 55: Plot comparing analytical solution of Tayor-Couette to a simulation where the flow is induced through the supplemental interface velocity.

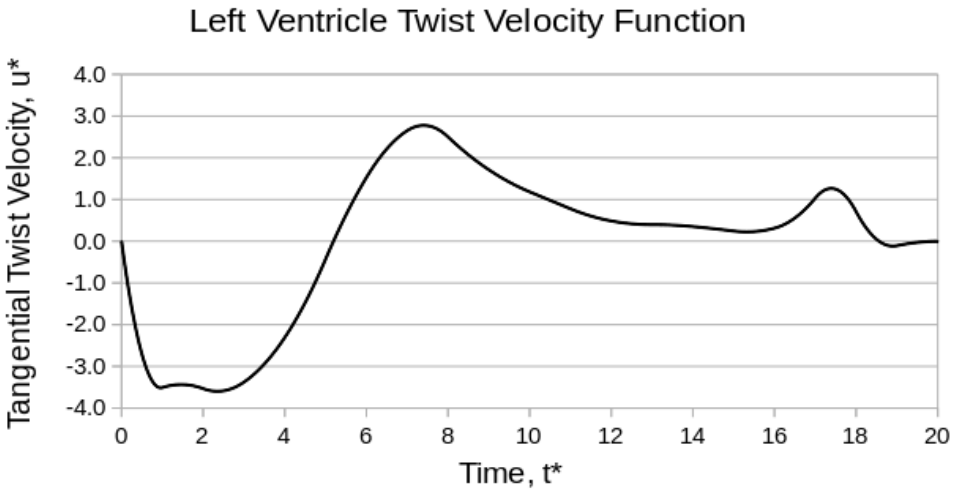
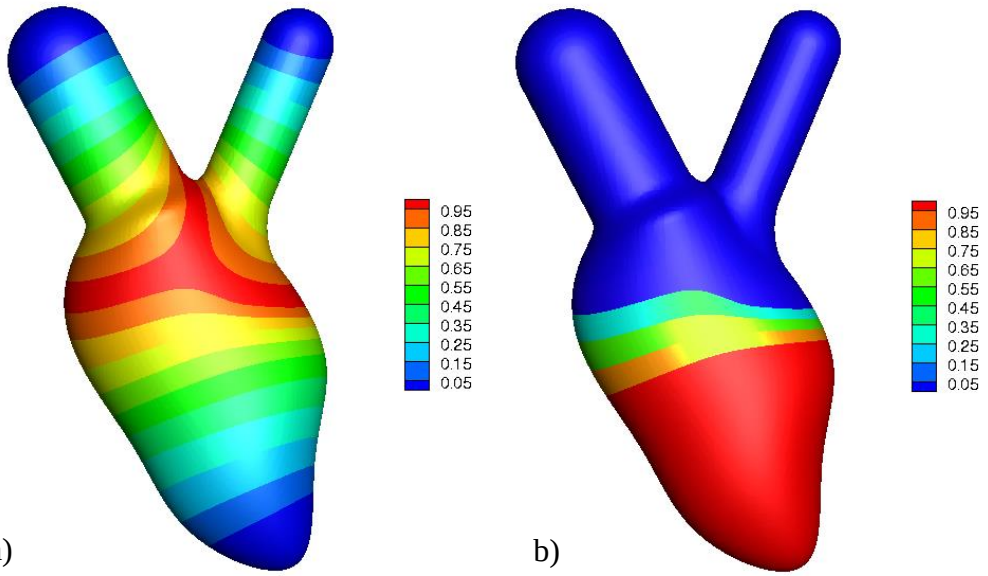


Figure 56: Plot showing the non-dimensional supplemental twist velocity applied to the left ventricle simulation.



a) b)

Figure 57: Level set tube mapping for applying the supplemental twist velocity, a) the existing skeleton mapping is used to construct the twist velocity weighting, b) supplemental twist weighting used to apply the twist function to the lower portion of the ventricle and taper to zero near the skeleton branch location.

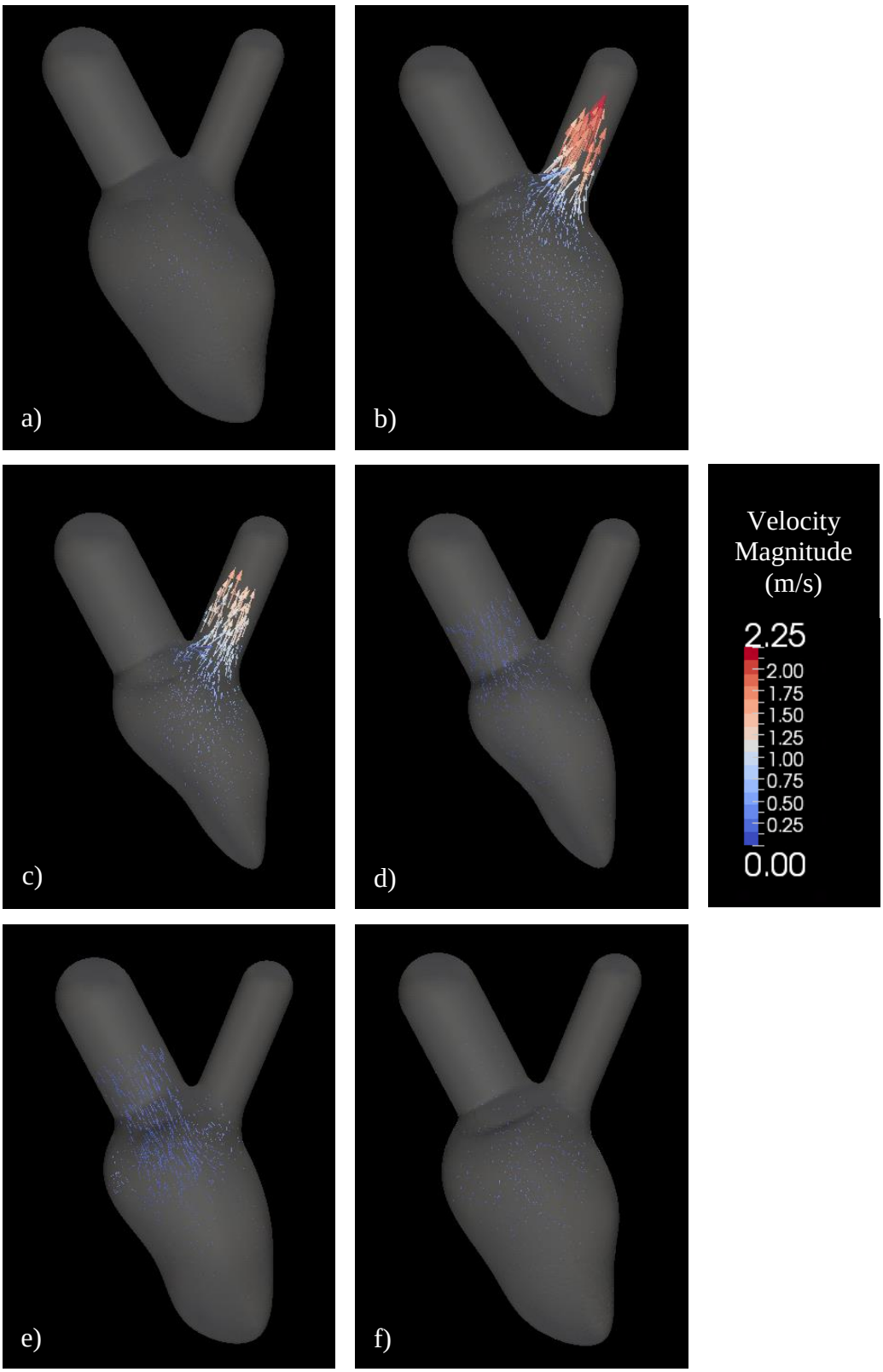


Figure 58: Velocity magnitude results of 3D left ventricle simulation using the proposed level set morph framework with supplemental twist velocity for select frames of the cardiac cycle (1,3,5,7,10,16)

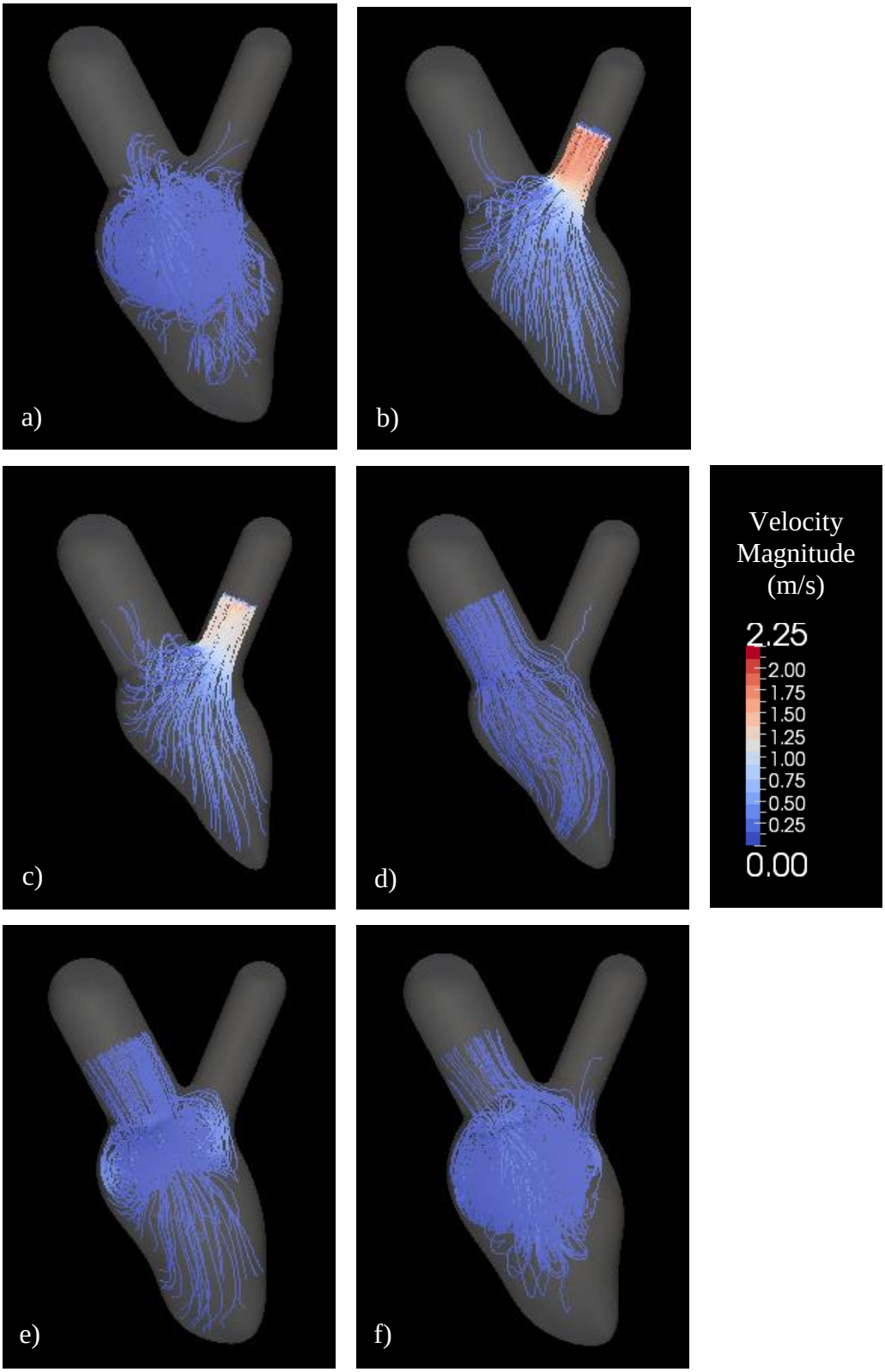
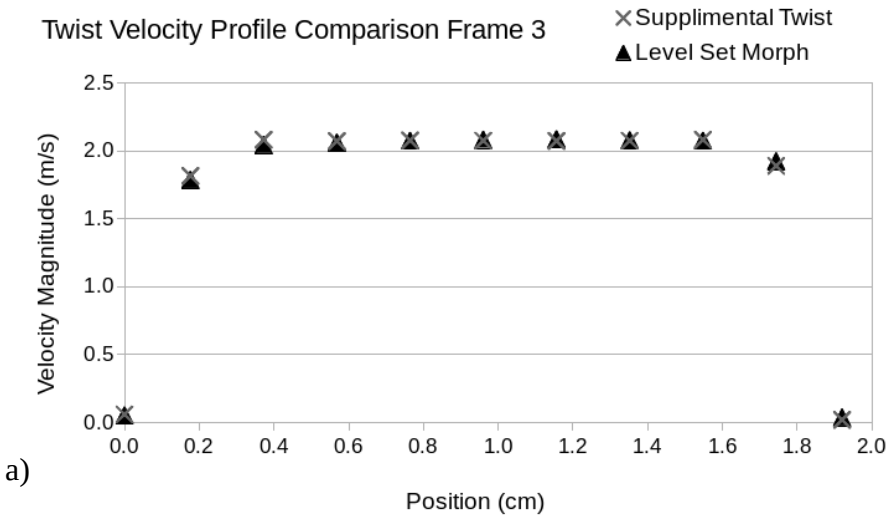
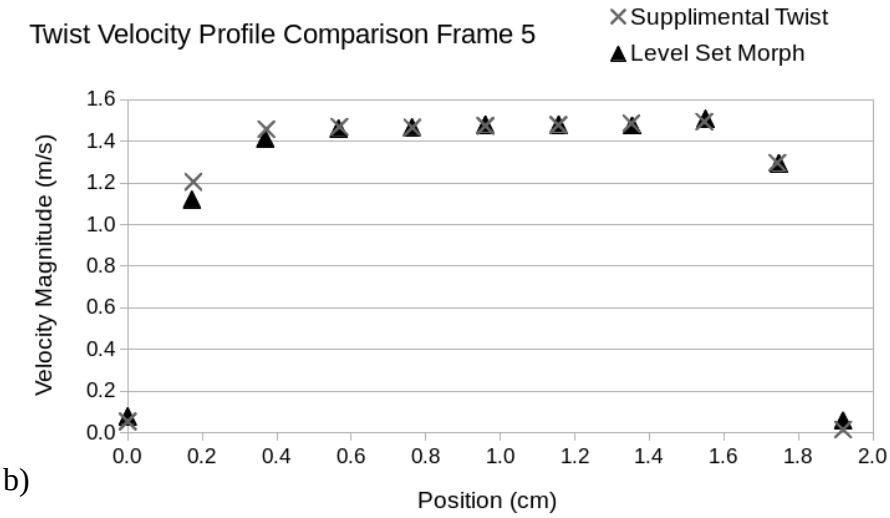


Figure 59: Stream trace results of 3D left ventricle simulation using the proposed level set morph framework with supplemental twist velocity indicating fluid rotation and vortical structures for select frames (1,3,5,7,10,16)



a)



b)

Figure 60: Velocity profile comparison using the level set morph framework with and without the supplemental twist velocity, a) velocity profile comparison plot for frame 3, b) velocity profile comparison plot for frame 5

CHAPTER 8 CONCLUSIONS

8.1 Conclusions and Limitations

The framework developed here accomplishes the goal of providing the left ventricle moving boundary condition, derived from patient-specific images, to a fluid-structure interaction simulation of the heart in a manner that is conducive to the clinical environment. The framework accomplishes this goal through use of a novel skeleton-based level set morphing approach. The skeletonization method has been evaluated to show sub-grid accuracy that depends primarily on the grid resolution. It has also been shown to be successful for applications involving tortuous geometry in the cardiovascular system. The level set morphing algorithm first utilizes the skeleton to perform a warp operation for the large scale topological changes and then a blend operation to adjust the small scale surface features. A Laplace solution is used to map the skeleton to the interface for the warp operation. This level set morphing approach using skeletons has been successfully verified against an alternative interface morphing method for the case of a swimming eel. The last piece of the framework is the capability to apply boundary conditions inside the computational domain. Through multiple verification cases, this method has been shown to work in 3D space and accommodate moving geometries. All of these pieces are successfully integrated to perform FSI simulations on patient-specific data. A 3D verification case has shown the proposed framework generates similar results as the traditional approach of morphing a polygonal mesh.

The method described here of applying boundary conditions inside the computational domain is specifically developed for a discrete forcing approach using a sharp interface method. Since diffuse interface approaches require that the fluid

equations are solved for the entire domain, the requirements and implementation would be significantly different.

Results from this new level set morphing approach show excellent agreement with the traditional surface mesh-based morphing, but the verification is somewhat limited. While the capabilities were demonstrated on an idealized, 3D geometry and 2D patient-specific image data, verification has not yet been performed on 3D geometry segmented from patient image data. This is an important step, since the segmented geometry would not be as smooth as the surfaces reconstructed from the patient point cloud data. Computed tomography (CT) would likely be the source of the image-derived patient data.

Caution should be taken before employing this framework as a general solution to image-based modeling with moving boundaries. The applicability of the level set morphing method developed here largely depends on how well the skeleton represents the topological deformation. The skeleton is integral to constructing physically realistic tangential interface velocities. For the swimming eel, a skeleton is clearly a good low-resolution representation. The eel has a long slender body with a vertebral column running down the centerline. This is confirmed through the impressive accuracy of the results. A skeleton is also a reasonable low-resolution representation of cardiovascular geometry. Most cardiovascular vessels are tubular in shape, with the slight exception being the chamber of the left ventricle. This shape is typically described as prolate spheroidal. But as demonstrated, using a wave source location inside the chamber, the skeleton representation generates a sufficient characterization of the bulk deformation.

This produces a warp velocity such that the interface motion and fluid dynamics are consistent with the more traditional approach of morphing a polygonal surface mesh.

8.2 Future Directions

Future directions of this work can be primarily divided into three categories: development of the supporting workflow, technical improvements and validations of the level set morphing method, and computational framework development/optimization.

Other aspects of the workflow required to perform patient-specific heart valve modeling can be identified by reexamining

Figure *5*. The activities adjacent to the *interface morphing* step should be considered first as to allow higher throughput of patient-specific data. Segmentation of the left ventricle is the most integral need. While capabilities exist elsewhere to perform this type of segmentation [54]–[56], this step requires dedicated expertise in image segmentation and a toolset, such as ITK [108], Mimics [109], or Seg-3D [110], to generate a consistent geometry for every frame of the cardiac cycle. Skilled technicians typically spend a great deal of time, not only developing the image processing pipeline, but also in executing the pipeline on all 3D image data sets of the full image sequence [10]. This approach, however doesn't concur with the previously stated goal of automating the image-to-simulation work flow for use in the clinical environment. It is the opinion of the author that machine learning must be utilized to automate the image segmentation, which is an ongoing research effort in medical imaging groups [111]. The initial challenge to this approach is developing a sufficiently large set of properly segmented images for training.

To support the addition of the heart valves to the FSI simulation, three additional image processing capabilities must be refined and automated. The first is the segmentation of the heart valves. While this task has proven itself to be challenging, it is the opinion of the author that machine learning will accomplish this goal if enough training data is made available. Optimism is found though similar challenges such as the work of Gerard et al. [112] which uses a Convolutional Neural Network to successfully segment lung fissures from medical images of which the features are barely visible to the human eye. The second and third image processing needs consist of automated tools for feature identification of valve annuli and papillary muscle attachment locations. Existing algorithms have been developed for these tasks [12], [13], but these must be further investigated to determine whether they provide sufficiently accurate and consistent results from a common input image data set. Their robustness should also be considered as they must also be integrated in an automated manner.

Progress in developing the supporting image processing capabilities will be crucial for certain aspects of the technical development of the proposed level set morphing strategy. Currently, the proposed strategy has only been verified against a reconstructed left ventricle geometry from patient point cloud data. While the results of Chapter 6 are very exciting and encouraging, verification should be conducted against segmented patient data. Ideally, this should be performed against a sizable number of patients to demonstrate the robustness of this process and verify that is can offer an automated solution to fill the stated goals.

The primary technical aspect currently limiting the proposed level set morphing strategy concerns the interface connection between the morphing left ventricle and the

valve geometry. It is the expectation of the author that the Finite Element Method will be used to represent and simulate the solid mechanics of the moving heart valves and chordae tendineae. The previously mentioned features of valve annuli and papillary muscle attachments are key. A means of influencing the level set morph near those features must be developed to ensure the zero level set iso-contour is coincident such to provide a fluid-tight connection to the heart valve finite element mesh.

The third area of future work concerns framework development. Framework development and scalability are important for completing this type of large simulation in a reliable and timely manner. The primary area of concern affecting the scalability, is the level set portion of the code. In the current approach, during simulation start-up, each frame of the medical image sequence is converted to a level set field on the computational fluid domain. This includes arrays for the *level set normal*, *zero iso-contour cells, level set-to-fluid index mapping, skeleton segment association, skeleton mapping*, and *average morph target array*. Not only is the memory footprint very large, but each of these arrays must be managed and communicated during the automated mesh refinement and dynamic domain repartitioning operations of the framework, thus greatly hindering the scalability of the method. It may be possible to only carry two target frames worth of level set data as only two are needed for interpolation at any point in time. The challenge with this approach concerns some of the initial operations performed during initial start-up as previously described. The first concerns the global mass conservation applied to the outlet boundary condition. As previously described, the fluid volume of each morph target is measured so cubic spline interpolation can fit a function to be evaluation during the simulation to achieve the correct mass balance. Secondly, the

morph zone is computed on initial start-up by solving a Laplace equation on each of the morph targets to determine the region of the fluid domain over which the interface will cross during the simulation. The final challenge concerns skeletonization and skeleton mapping. As previously mentioned, the skeletonization process requires temporary refinement of the entire fluid region of the domain to the smallest grid size allowed. This is temporarily performed during start-up and then the mesh coarsened before the first time-step of the flow solver is executed. Any such strategy to accomplish this framework optimization must account for all of these mentioned start-up operations.

Despite some of the limitations to the proposed framework, a large potential awaits the future extension of this work. Additional effort is likely to bring patient-specific cardiovascular modeling into the clinical environment and improve the quality of patient care.

REFERENCES

[1] J. Xu, S. Murphy, K. Kochanek, and B. Bastian, "Deaths: Final Data for 2013," *Natl. Vital Stat. Rep.*, vol. 64, no. 2, Feb. 2016.

[2] O. O. Al-Radi, "Understanding the pathophysiology of mitral regurgitation: the first step in management," *Geriatr Aging*, vol. 6, pp. 42–45, 2003.

[3] D. J. LaPar *et al.*, "Mitral valve repair rates correlate with surgeon and institutional experience," *J. Thorac. Cardiovasc. Surg.*, vol. 148, no. 3, pp. 995–1004, Sep. 2014.

[4] M. L. Neal and R. Kerckhoffs, "Current progress in patient-specific modeling," *Brief. Bioinform.*, vol. 11, no. 1, pp. 111–126, Jan. 2010.

[5] J.-C. Bernhard *et al.*, "Personalized 3D printed model of kidney and tumor anatomy: a useful tool for patient education," *World J. Urol.*, vol. 34, no. 3, pp. 337–345, Mar. 2016.

[6] S. Kulp *et al.*, "Using High Resolution Cardiac CT Data to Model and Visualize Patient-Specific Interactions between Trabeculae and Blood Flow," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2011*, 2011, pp. 468–475.

[7] 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 *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2010, pp. 504–507.

[8] V. Mihalef, D. Metaxas, M. Sussman, V. Hurmusiadis, and L. Axel, "Atrioventricular Blood Flow Simulation Based on Patient-Specific Data," in *Functional Imaging and Modeling of the Heart*, 2009, pp. 386–395.

[9] K. B. Chandran, S. E. Rittgers, A. P. Yoganathan, S. E. Rittgers, and A. P. Yoganathan, *Biofluid Mechanics : The Human Circulation*. CRC Press, 2006.

[10] R. Mittal *et al.*, "Computational modeling of cardiac hemodynamics: Current status and future outlook," *J. Comput. Phys.*, vol. 305, pp. 1065–1082, Jan. 2016.

[11] P. Gade, "Development of a surgical simulation toolkit for mitral valve repair surgeries," ProQuest Dissertations Publishing, 2014.

[12] R. J. Schneider, D. P. Perrin, N. V. Vasilyev, G. R. Marx, P. J. del Nido, and R. D. Howe, "Mitral Annulus Segmentation From 3D Ultrasound Using Graph Cuts," *IEEE Trans. Med. Imaging*, vol. 29, no. 9, pp. 1676–1687, Sep. 2010.

[13] Y. Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu, "Four-Chamber Heart Modeling and Automatic Segmentation for 3-D Cardiac CT Volumes Using Marginal Space Learning and Steerable Features," *IEEE Trans. Med. Imaging*, vol. 27, no. 11, pp. 1668–1681, Nov. 2008.

[14] R. J. Schneider, D. P. Perrin, N. V. Vasilyev, G. R. Marx, P. J. del Nido, and R. D. Howe, "Mitral annulus segmentation from four-dimensional ultrasound using a valve state predictor and constrained optical flow," *Med. Image Anal.*, vol. 16, no. 2, pp. 497–504, Feb. 2012.

[15] J. R. Cebral *et al.*, "Clinical application of image-based CFD for cerebral aneurysms," *Int. J. Numer. Methods Biomed. Eng.*, vol. 27, pp. 977–992, 2010.

[16] C. A. Taylor and D. A. Steinman, "Image-Based Modeling of Blood Flow and Vessel Wall Dynamics: Applications, Methods and Future Directions," *Ann. Biomed. Eng.*, vol. 38, no. 3, pp. 1188–1203, Mar. 2010.

[17] A. Caballero *et al.*, "Modeling Left Ventricular Blood Flow Using Smoothed Particle Hydrodynamics," *Cardiovasc. Eng. Technol.*, vol. 8, no. 4, pp. 465–479, Dec. 2017.

[18] T. B. Le and F. Sotiropoulos, "Fluid–structure interaction of an aortic heart valve prosthesis driven by an animated anatomic left ventricle," *J. Comput. Phys.*, vol. 244, pp. 41–62, Jul. 2013.

[19] 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 *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2011, pp. 1692–1697.

[20] E. Votta *et al.*, "Toward patient-specific simulations of cardiac valves: State-of-the-art and future directions," *J. Biomech.*, vol. 46, no. 2, pp. 217–228, Jan. 2013.

[21] F. Sotiropoulos, T. B. Le, and A. Gilmanov, "Fluid Mechanics of Heart Valves and Their Replacements," *Annu. Rev. Fluid Mech.*, vol. 48, no. 1, pp. 259–283, Jan. 2016.

[22] I. Borazjani, L. Ge, and F. Sotiropoulos, "Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies," *J. Comput. Phys.*, vol. 227, no. 16, pp. 7587–7620, Aug. 2008.

[23] L. Ge and F. Sotiropoulos, "A numerical method for solving the 3D unsteady incompressible Navier–Stokes equations in curvilinear domains with complex immersed boundaries," *J. Comput. Phys.*, vol. 225, no. 2, pp. 1782–1809, Aug. 2007.

[24] J. H. Seo *et al.*, "Effect of the mitral valve on diastolic flow patterns," *Phys. Fluids*, vol. 26, no. 12, p. 121901, Dec. 2014.

[25] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, and A. von Loebbecke, "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries," *J. Comput. Phys.*, vol. 227, no. 10, pp. 4825–4852, May 2008.

[26] J. Mousel, "A massively parallel adaptive sharp interface solver with application to mechanical heart valve simulations," *Theses Diss.*, Jan. 2012.

[27] S. Marella, S. Krishnan, H. Liu, and H. S. Udaykumar, "Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations," *J. Comput. Phys.*, vol. 210, no. 1, pp. 1–31, Nov. 2005.

[28] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, "A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)," *J. Comput. Phys.*, vol. 152, no. 2, pp. 457–492, Jul. 1999.

[29] Y.-H. Tseng and J. H. Ferziger, "A ghost-cell immersed boundary method for flow in complex geometry," *J. Comput. Phys.*, vol. 192, no. 2, pp. 593–623, Dec. 2003.

[30] H. Choi and P. Moin, "Effects of the Computational Time Step on Numerical Solutions of Turbulent Flow," *J. Comput. Phys.*, vol. 113, no. 1, pp. 1–4, Jul. 1994.

[31] J. Yang and F. Stern, "Sharp interface immersed-boundary/level-set method for wave–body interactions," *J. Comput. Phys.*, vol. 228, no. 17, pp. 6590–6616, Sep. 2009.

[32] J. Yang and F. Stern, "A simple and efficient direct forcing immersed boundary framework for fluid–structure interactions," *J. Comput. Phys.*, vol. 231, no. 15, pp. 5029–5061, Jun. 2012.

[33] T. Tu, D. R. O'Hallaron, and O. Ghattas, "Scalable Parallel Octree Meshing for TeraScale Applications," in *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, 2005, pp. 4–4.

[34] S. I. Dillard, J. A. Mousel, L. Shrestha, M. L. Raghavan, and S. C. Vigmostad, "From medical images to flow computations without user-generated meshes," *Int. J. Numer. Methods Biomed. Eng.*, vol. 30, no. 10, pp. 1057–1083, Oct. 2014.

[35] V. author Govindarajan, "Three dimensional fluid structural interaction of tissue valves /," University of Iowa, Iowa City, Iowa, 2013.

[36] F. Lazarus and A. Verroust, "Three-dimensional metamorphosis: a survey," *Vis. Comput.*, vol. 14, no. 8, pp. 373–389, Dec. 1998.

[37] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "MAPS: Multiresolution Adaptive Parameterization of Surfaces," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1998, pp. 95–104.

[38] T. Beier and S. Neely, “Feature-based Image Metamorphosis,” in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1992, pp. 35–42.

[39] S.-Y. Lee, K.-Y. Chwa, J. Hahn, and S. Y. Shin, “Image Morphing Using Deformation Techniques,” *J. Vis. Comput. Animat.*, vol. 7, no. 1, pp. 3–23, Jan. 1996.

[40] L. G. Brown, “A Survey of Image Registration Techniques,” *ACM Comput Surv*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

[41] A. M. Tekalp, *Digital Video Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2015.

[42] J. R. Kent, W. E. Carlson, and R. E. Parent, “Shape Transformation for Polyhedral Objects,” in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1992, pp. 47–54.

[43] R. E. Parent, “Shape transformation by boundary representation interpolation: A recursive approach to establishing face correspondences,” *J. Vis. Comput. Animat.*, vol. 3, no. 4, pp. 219–239, Oct. 1992.

[44] Y. M. Sun, W. Wang, and F. Y. L. Chin, “Interpolating Polyhedral Models Using Intrinsic Shape Parameters,” *J. Vis. Comput. Animat.*, vol. 8, no. 2, pp. 81–96, Mar. 1997.

[45] T. Kanai, H. Suzuki, and F. Kimura, “3D geometric metamorphosis based on harmonic map,” in *Proceedings The Fifth Pacific Conference on Computer Graphics and Applications*, 1997, pp. 97–104.

[46] T. M. Hong, N. Magnenat-Thalmann, and D. Thalmann, “A general algorithm for 3-D shape interpolation in a facet-based representation,” vol. 88, no. 30, pp. 229–235.

[47] M. M. Profir, “Mesh morphing techniques in CFD.,” presented at the International Student Conference on Pure and Applied Mathematics, 2011.

[48] M. Alexa, “Recent Advances in Mesh Morphing,” *Comput. Graph. Forum*, vol. 21, no. 2, pp. 173–198, Jun. 2002.

[49] D. E. Breen and R. T. Whitaker, “A level-set approach for the metamorphosis of solid models,” *Ieee Trans. Vis. Comput. Graph.*, vol. 7, pp. 173–192, Apr. 2001.

[50] R. T. Whitaker, “A level-set approach to image blending,” *IEEE Trans. Image Process.*, vol. 9, pp. 1849–1861, Nov. 2000.

[51] R. T. Whitaker, “Algorithms for implicit deformable models,” in *Proceedings of IEEE International Conference on Computer Vision*, 1995, pp. 822–827.

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

[53] J. A. Sethian and P. Smereka, "Level Set Methods for Fluid Interfaces," *Annu. Rev. Fluid Mech.*, vol. 35, no. 1, pp. 341–372, 2003.

[54] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, pp. 266–277, Feb. 2001.

[55] M. Niethammer, A. Tannenbaum, and S. Angenent, "Dynamic active contours for visual tracking," *Ieee Trans. Autom. Control*, vol. 51, pp. 562–579, Apr. 2006.

[56] H. Y. Lee, N. C. F. Codella, M. D. Cham, J. W. Weinsaft, and Y. Wang, "Automatic Left Ventricle Segmentation Using Iterative Thresholding and an Active Contour Model With Adaptation on Short-Axis Cardiac MRI," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 4, pp. 905–913, Apr. 2010.

[57] C. Petitjean and J.-N. Dacher, "A review of segmentation methods in short axis cardiac MR images," *Med. Image Anal.*, vol. 15, no. 2, pp. 169–184, Apr. 2011.

[58] P. Peng, K. Lekadir, A. Gooya, L. Shao, S. E. Petersen, and A. F. Frangi, "A review of heart chamber segmentation for structural and functional analysis using cardiac magnetic resonance imaging," *Magn. Reson. Mater. Phys. Biol. Med.*, vol. 29, no. 2, pp. 155–195, Apr. 2016.

[59] M. Lynch, O. Ghita, and P. F. Whelan, "Segmentation of the Left Ventricle of the Heart in 3-D+t MRI Data Using an Optimized Nonrigid Temporal Model," *IEEE Trans. Med. Imaging*, vol. 27, no. 2, pp. 195–203, Feb. 2008.

[60] J. Hennig, B. Schneider, S. Peschl, M. Markl, and T. K. J. Laubenberger, "Analysis of myocardial motion based on velocity measurements with a black blood prepared segmented gradient-echo sequence: Methodology and applications to normal volunteers and patients," *J. Magn. Reson. Imaging*, vol. 8, no. 4, pp. 868–877, Jul. 1998.

[61] C. M. Otto, "Valvular Aortic Stenosis," *J. Am. Coll. Cardiol.*, vol. 47, no. 11, pp. 2141–2151, Jun. 2006.

[62] B. Baccani, F. Domenichini, and G. Pedrizzetti, "Vortex dynamics in a model left ventricle during filling," *Eur. J. Mech. - BFluids*, vol. 21, no. 5, pp. 527–543, Sep. 2002.

[63] S. I. Dillard, J. H. J. Buchholz, and H. S. Udaykumar, "From video to computation of biological fluid–structure interaction problems," *Theor. Comput. Fluid Dyn.*, vol. 1–2, no. 30, pp. 41–66, 2016.

[64] B. Horn and B. Schunck, "Determining Optical Flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1981.

[65] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1987, pp. 163–169.

[66] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Form*, W. Dunn, Ed. MIT Press, 1967, pp. 362–380.

[67] P. K. Saha, G. Borgefors, and G. Sanniti di Baja, "A survey on skeletonization algorithms and their applications," *Pattern Recognit. Lett.*, vol. 76, pp. 3–12, Jun. 2016.

[68] D. Attali and A. Montanvert, "Computing and Simplifying 2D and 3D Continuous Skeletons," *Comput. Vis. Image Underst.*, vol. 67, no. 3, pp. 261–273, Sep. 1997.

[69] F. Rolland, J.-M. Chassery, and A. Montanvert, "3D Medial Surfaces and 3D Skeletons," in *Visual Form: Analysis and Recognition*, C. Arcelli, L. P. Cordella, and G. S. di Baja, Eds. Boston, MA: Springer US, 1992, pp. 443–450.

[70] J. W. Brandt and V. R. Algazi, "Continuous skeleton computation by Voronoi diagram," *CVGIP Image Underst.*, vol. 55, no. 3, pp. 329–338, May 1992.

[71] M. Näf, G. Székely, R. Kikinis, M. E. Shenton, and O. Kübler, "3D Voronoi Skeletons and Their Usage for the Characterization and Recognition of 3D Organ Shape," *Comput. Vis. Image Underst.*, vol. 66, no. 2, pp. 147–161, May 1997.

[72] R. Ogniewicz and M. Ilg, "Voronoi skeletons: theory and applications," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992, pp. 63–69.

[73] R. L. Ogniewicz and O. Kübler, "Hierarchic Voronoi skeletons," *Pattern Recognit.*, vol. 28, no. 3, pp. 343–359, Mar. 1995.

[74] C. Pudney, "Distance-Ordered Homotopic Thinning: A Skeletonization Algorithm for 3D Digital Images," *Comput. Vis. Image Underst.*, vol. 72, no. 3, pp. 404–413, Dec. 1998.

[75] C. Arcelli, G. S. di Baja, and L. Serino, "Distance-Driven Skeletonization in Voxel Images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 4, pp. 709–720, Apr. 2011.

[76] G. Németh, P. Kardos, and K. Palágyi, "Thinning combined with iteration-by-iteration smoothing for 3D binary images," *Graph. Models*, vol. 73, no. 6, pp. 335–345, Nov. 2011.

[77] M. Sonka, M. D. Winniford, X. Zhang, and S. M. Collins, “Lumen centerline detection in complex coronary angiograms,” *IEEE Trans. Biomed. Eng.*, vol. 41, no. 6, pp. 520–528, Jun. 1994.

[78] C. M. Ma and M. Sonka, “A Fully Parallel 3D Thinning Algorithm and Its Applications,” *Comput. Vis. Image Underst.*, vol. 64, no. 3, pp. 420–433, Nov. 1996.

[79] Y. F. Tsao and K. S. Fu, “A parallel thinning algorithm for 3-D pictures,” *Comput. Graph. Image Process.*, vol. 17, no. 4, pp. 315–331, Dec. 1981.

[80] N. Ahuja and J.-H. Chuang, “Shape representation using a generalized potential field model,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 169–176, Feb. 1997.

[81] R. Kimmel, D. Shaked, N. Kiryati, and A. M. Bruckstein, “Skeletonization via distance maps and level sets,” in *Vision Geometry III*, 1995, vol. 2356, pp. 137–149.

[82] F. Leymarie and M. D. Levine, “Simulating the grassfire transform using an active contour model,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 1, pp. 56–75, Jan. 1992.

[83] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker, “Hamilton-Jacobi Skeletons,” *Int. J. Comput. Vis.*, vol. 48, no. 3, pp. 215–231, Jul. 2002.

[84] M. S. Hassouna and A. A. Farag, “Robust centerline extraction framework using level sets,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, 2005, vol. 1, pp. 458–465 vol. 1.

[85] M. S. Hassouna and A. A. Farag, “Variational Curve Skeletons Using Gradient Vector Flow,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2257–2274, Dec. 2009.

[86] E. D. Tytell and G. V. Lauder, “The hydrodynamics of eel swimming: I. Wake structure,” *J. Exp. Biol.*, 2004.

[87] C. S. Peskin, “Flow patterns around heart valves: A numerical method,” *J. Comput. Phys.*, vol. 10, no. 2, pp. 252–271, Oct. 1972.

[88] E. M. Saiki and S. Biringen, “Numerical Simulation of a Cylinder in Uniform Flow: Application of a Virtual Boundary Method,” *J. Comput. Phys.*, vol. 123, no. 2, pp. 450–465, Feb. 1996.

[89] R. P. Beyer and R. J. LeVeque, “Analysis of a One-Dimensional Model for the Immersed Boundary Method,” *SIAM J. Numer. Anal. Phila.*, vol. 29, no. 2, p. 33, Apr. 1992.

[90] M.-C. Lai and C. S. Peskin, “An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity,” *J. Comput. Phys.*, vol. 160, no. 2, pp. 705–719, May 2000.

[91] R. Mittal and G. Iaccarino, “Immersed Boundary Methods,” *Annu. Rev. Fluid Mech.*, vol. 37, no. 1, pp. 239–261, 2005.

[92] D. Goldstein, R. Handler, and L. Sirovich, “Modeling a No-Slip Flow Boundary with an External Force Field,” *J. Comput. Phys.*, vol. 105, no. 2, pp. 354–366, Apr. 1993.

[93] P. Angot, C.-H. Bruneau, and P. Fabrie, “A penalization method to take into account obstacles in incompressible viscous flows,” *Numer. Math.*, vol. 81, no. 4, pp. 497–520, Feb. 1999.

[94] Roberto Verzicco, Jamaludin Mohd-Yusof, Paolo Orlandi, and Daniel Haworth, “Large Eddy Simulation in Complex Geometric Configurations Using Boundary Body Forces,” *AIAA J.*, vol. 38, no. 3, pp. 427–433, 2000.

[95] R. Verzicco, M. Fatica, G. Iaccarino, P. Moin, and B. Khalighi, “Large Eddy Simulation of a Road Vehicle with Drag-Reduction Devices,” *AIAA J.*, vol. 40, no. 12, pp. 2447–2455, 2002.

[96] E. Balaras, “Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations,” *Comput. Fluids*, vol. 33, no. 3, pp. 375–404, Mar. 2004.

[97] F. Gibou and R. Fedkiw, “A Fast Hybrid k-Means Level Set Algorithm for Segmentation,” presented at the 4th Annual Hawaii International Conference on Statistics, Mathematics, and Related Fields, 2005.

[98] C. Lu, “Multi-scale modeling of shock interactions with a cloud of particles using an artificial neural network for model representation,” *Procedia IUTAM*, no. In press, 2012.

[99] A. Goddard, “Applying vessel inlet/outlet conditions to patient-specific models embedded in Cartesian grids,” University of Iowa, 2015.

[100] J. Schulz-Menger *et al.*, “Standardized image interpretation and post processing in cardiovascular magnetic resonance: Society for Cardiovascular Magnetic Resonance (SCMR) Board of Trustees Task Force on Standardized Post Processing,” *J. Cardiovasc. Magn. Reson.*, vol. 15, no. 1, p. 35, May 2013.

[101] G.-R. Hong *et al.*, “Characterization and Quantification of Vortex Flow in the Human Left Ventricle by Contrast Echocardiography Using Vector Particle Image Velocimetry,” *JACC Cardiovasc. Imaging*, vol. 1, no. 6, pp. 705–717, Nov. 2008.

[102] P. Radau, Y. Lu, K. Connelly, G. Paul, A. J. Dick, and G. A. Wright, "Evaluation Framework for Algorithms Segmenting Short Axis Cardiac MRI.," *MIDAS J.*, p. 658, Jul. 2009.

[103] P. P. Sengupta, A. J. Tajik, K. Chandrasekaran, and B. K. Khandheria, "Twist Mechanics of the Left Ventricle: Principles and Application," *JACC Cardiovasc. Imaging*, vol. 1, no. 3, pp. 366–376, May 2008.

[104] H. Gao, C. Berry, and X. Luo, "Image-Derived Human Left Ventricular Modelling with Fluid-Structure Interaction," in *SpringerLink*, 2015, pp. 321–329.

[105] G. I. Taylor, "VIII. Stability of a viscous liquid contained between two rotating cylinders," *Phil Trans R Soc Lond A*, vol. 223, no. 605–615, pp. 289–343, Jan. 1923.

[106] E. L. Koschmieder, "Turbulent Taylor vortex flow," *J. Fluid Mech.*, vol. 93, no. 3, pp. 515–527, Aug. 1979.

[107] B. Jung, M. Markl, D. Föll, and J. Hennig, "Investigating myocardial motion by MRI using tissue phase mapping," *Eur. J. Cardiothorac. Surg.*, vol. 29, no. Supplement_1, pp. S150–S157, Apr. 2006.

[108] "ITK - Segmentation & Registration Toolkit." [Online]. Available: https://itk.org/ITK/project/license.html. [Accessed: 26-Oct-2018].

[109] "3D Medical Image Processing Software | Materialise Mimics." [Online]. Available: https://www.materialise.com/en/medical/software/mimics. [Accessed: 26-Oct-2018].

[110] "Seg3D." [Online]. Available: http://www.sci.utah.edu/cibc-software/seg3d.html. [Accessed: 26-Oct-2018].

[111] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.

[112] S. E. Gerard, T. J. Patton, G. E. Christensen, J. E. Bayouth, and J. M. Reinhardt, "FissureNet: A Deep Learning Approach For Pulmonary Fissure Detection in CT Images," *IEEE Trans. Med. Imaging*, pp. 1–1, 2018.