Advanced Visualization and Intuitive User Interface Systems for Biomedical Applications

David Quam
Marquette University, david.quam@marquette.edu

ADVANCED VISUALIZATION AND INTUITIVE USER INTERFACE SYSTEMS FOR BIOMEDICAL APPLICATIONS

by

David J. Quam

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

May 2012

ABSTRACT

ADVANCED VISUALIZATION AND INTUITIVE USER INTERFACE SYSTEMS FOR BIOMEDICAL APPLICATIONS

David J. Quam

Marquette University, 2012

Modern scientific research produces data at rates that far outpace our ability to comprehend and analyze it. Such sources include medical imaging data and computer simulations, where technological advancements and spatiotemporal resolution generate increasing amounts of data from each scan or simulation. A bottleneck has developed whereby medical professionals and researchers are unable to fully use the advanced information available to them. By integrating computer science, computer graphics, artistic ability and medical expertise, scientific visualization of medical data has become a new field of study. The objective of this thesis is to develop two visualization systems that use advanced visualization, natural user interface technologies and the large amount of biomedical data available to produce results that are of clinical utility and overcome the data bottleneck that has developed.

Computational Fluid Dynamics (CFD) is a tool used to study the quantities associated with the movement of blood by computer simulation. We developed methods of processing spatiotemporal CFD data and displaying it in stereoscopic 3D with the ability to spatially navigate through the data. We used this method with two sets of display hardware: a full-scale visualization environment and a small-scale desktop system. The advanced display and data navigation abilities provide the user with the means to better understand the relationship between the vessel’s form and function.

Low-cost 3D, depth-sensing cameras capture and process user body motion to recognize motions and gestures. Such devices allow users to use hand motions as an intuitive interface to computer applications. We developed algorithms to process and prepare the biomedical and scientific data for use with a custom control application. The application interprets user gestures as commands to a visualization tool and allows the user to control the visualization of multi-dimensional data. The intuitive interface allows the user to control the visualization of data without manual contact with an interaction device. In developing these methods and software tools we have leveraged recent trends in advanced visualization and intuitive interfaces in order to efficiently visualize biomedical data in such a way that provides meaningful information that can be used to further appreciate it.

ACKNOWLEDGEMENTS

David J. Quam

I wish to express gratitude to my principle advisor, John LaDisa, for his innovative perspective on the utility of immersive visualization in biomedical applications and his ability to link my skill set with this project. It is this forward-focused vision that provided the inspiration for this thesis. Further, I gratefully acknowledge his fervent support of my project through securing unique collaboration and funding opportunities. On a personal note, I wish to thank him for serving as an excellent role model in how to professionally conduct academic research in a way that fosters an entrepreneurial spirit. This project would not have been possible without the vision of Dr. Ronald Woods, Associate Professor of Surgery at the Children's Hospital of Wisconsin. I wish to take this opportunity to sincerely thank him for the support he provided by spearheading this project and providing a vision to which I could apply my creative and engineering skills.

I would be remised to not thank the Marquette University Department of Biomedical Engineering, especially Dr. Kristina Ropella, for providing the financial support that made my graduate studies feasible through teaching assistantships and travel awards. I also express gratitude to my Thesis Committee, consisting of Drs. John LaDisa, Ronald Woods, Laura Ellwein and Lars Olson. I appreciate the time, energy and expertise they have given this project and am humbled by their generosity. I am pleased to acknowledge the generous commitment of time, energy and resources afforded me by the staff at the Discovery World Museum at Pier Wisconsin. I especially wish to thank Paul Hayden for providing excellent support while developing the content for the HIVE. I owe Timothy Gundert a debt of gratitude for establishing the framework for immersive visualization further developed in this thesis. I also wish to thank Dr. Laura Ellwein for developing the method to calculate the orientation of OCT images using the theory of minimum bending energy. I feel the inclusion of her method in this project has strengthened it a great deal.

Finally, I wish to acknowledge the unyielding support of my friends and family, without whom this thesis would not be possible. Most notably I wish to thank my parents David and Kathleen for providing moral support when most needed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY OF TERMS

2D – two dimensional
3D – three dimensional
API – application programming interface
AR – augmented reality
CAVE – Audio-Visual Experience Automatic Virtual Environment
CFD – computational fluid dynamics
CFF – critical flicker frequency
CMOS – complementary metal oxide semiconductor
COM – component object model
COVISE – Collaborative Visualization Environment
CRT – cathode ray tube
DICOM – digital imaging and communications in medicine (medical image storage standard)
FOV – field of view
GPU – graphics processing unit
GUI – graphical user interface
HCI – human-computer interaction
HIVE – Human Interactive Virtual Environment
HMD – head-mounted display
HVS – human visual system
IDE – integrated development environment
IVE – immersive visualization environment
IVS – interventricular septum
LCD – liquid crystal display
LCX – left circumflex coronary artery
LED – light emitting diode
LOS – line of sight
LUT – look-up table
OCT – optical coherence tomography
OSI – oscillatory shear index
PCA – principal component analysis (also called singular value decomposition)
RGB – red green blue (color system)
SDK – software development kit
TAWSS – time-averaged wall shear stress
VB – Visual Basic (computer programming language)
VE – virtual environment
VR – virtual reality
VTK – visualization toolkit
WSS – wall shear stress (interpreted as instantaneous)

Chapter 1: Introduction

1.1 Specific Aims

This thesis aims to describe the utility in using and developing systems for advanced visualization as applied to biomedical research. We demonstrate this through two distinct projects to provide additional benefits in the areas of academic research and medicine.

Aim 1: Develop a Method to Process and Display CFD Simulation Results in an Immersive Visualization Environment

This aim employs visualization in the field of vascular biomechanics research to enhance spatiotemporal understanding of multi-dimensional data sets, such as those commonly generated by computational fluid dynamics (CFD) simulations. These simulations produce large amounts of data that must be compressed or manipulated to be visualized. To reduce the amount of compression or manipulation, we developed a method of rapidly visualizing CFD results in an immersive visualization environment (IVE). The method also establishes a means with which to resolve the data bottleneck that has been developing over the past decade. The applicability of the method is demonstrated with two display topologies displaying CFD results from the carotid and left-circumflex (LCX) coronary arteries.

Aim 2: Flexible Stereoscopic Visualization System for Biomedical and Scientific Datasets with Intuitive Gesture-Based Control Tool

This aim describes a novel visualization system that delivers a hardware and software solution to view medical imaging data with 3D stereoscopic images. We demonstrate results of custom control software that uses gestures recorded by a Microsoft Kinect Camera and processed with open-source algorithms. We also develop an algorithmic framework to create high quality, useful visualizations with open-source visualization software packages. We ultimately assemble a system that integrates the control software and stereoscopic displays configured to perform optimally for use as a pre-surgical planning tool.

1.2 The Data Deluge

Modern biomedical and scientific data is produced in quantities that push the limits of traditional methods of storing, viewing and comprehending it fully [1–4]. The sources of the majority of these data are increasingly monolithic high dimensional datasets, higher resolution images, more sophisticated computer simulations and many other information from computational sources [5,6]. The size of these data is expected to increase commensurately with the increases in computational capacity predicated by Moore's Law [7] thus posing a serious challenge to the way data is handled today [8]. This law roughly predicts a doubling of computational capacity every 18 months by increasing the number of semiconductor transistors on an integrated circuit (IC) chip [9].

In 2011, 1.8 trillion gigabytes of data were created [6]. This surpasses our ability to store such quantities of data [5], let alone completely comprehend it. A typical computer simulation of blood traveling through the human aorta can be comprised of several gigabytes of data, which must be registered to spatial locations along the vessel to be fully appreciated. As an example, consider the hole in ozone. Despite the 1985 British discovery of the hole in the Earth's ozone layer, the data describing the phenomena had been stored in the digital archives of the United States for nearly 10 years, but had not yet been analyzed due to the large amount of data in the processing queue [10]. Collectively, these examples suggest that an alternative to the traditional means of data analysis must be identified if the deluge of scientific data is to be resolved. Scientific visualization may serve this need by providing new ways of displaying large amounts of data at a single point in time.

1.3 Visualization as a Solution

To process, comprehend and make advancements using these massive amounts of data we must develop a means of holistically analyzing data that economizes the time, effort and resources required to process it. Visualization developed as an outgrowth of computer science, and as this thesis contends, suits the needs of the scientific community by allowing researchers to relieve the data analysis bottleneck that has developed in recent years. The term *visualization* has been adopted by many fields of study since its inception, where subtle differences in meaning have developed specific to each. This thesis points to the original definition of *visualization* set forth by DeFanti *et al.* in 1987 [11]:

> “Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization is a tool for […] generating images from complex multi-dimensional data sets.”

In this report the authors describe a method used to achieve results in the form of images. By this definition, visualization becomes limited to software and computing hardware associated with creating the images of data.

In 2006, Schroeder *et al.* published an amended definition of *visualization* broadening the previous meaning to describe a general *process* rather than a specific method to produce visualizations:

> “Visualization is the process of exploring, transforming, and viewing data as images (or other sensory forms) to gain understanding and insight into the data, […] it is naturally interactive, including the human directly in the process of creating, transforming, and viewing data” [12].

Figure 1.1 illustrates how the *process* of visualization described by Schroeder is closely linked with visualization *methods* put forth by DeFanti whereby in the process of conducting visualization, specific methods are called to produce the visual results. In Schroeder’s definition, the data used in visualization can originate from almost any source, including computational methods such as finite-element modeling or CFD, or measured data from sources like CT or MR scanners. Assuming the rest of the visualization process is equipped to handle the data, the process is not concerned with the source.

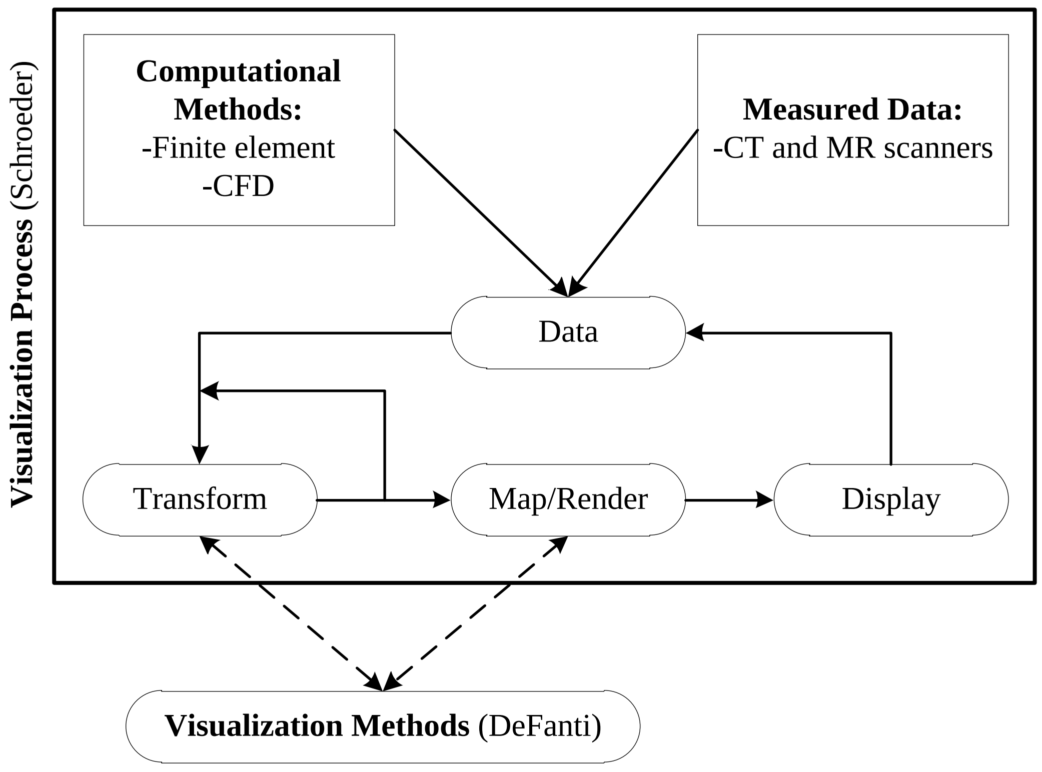


Figure 1.1 - Visualization process and methods
The process of visualizing data is an iterative process whereby data from either computations or measurements are fed to the system and continuously transformed and rendered until the desired features are extracted. Visualization methods provide the tools with which the data is transformed and rendered. This image adapted from [12].

The broadened definition by Schroeder highlights an important consideration in visualization: that it is most useful as a means to gain new insights, rather than a mere post-processing method. This perspective is shared by Fox *et al.* [8], where the authors discuss the need for researchers to be able to discover relationships between and within their data.

In using visualization to achieve this, we rely upon the human visual system (HVS) to perceive the images and relay them to our brain for processing. Nearly one half of the brain's neuronal connections are associated with the HVS [11,13,14]. With such a large share of our intellectual capacity linked to vision, it seems logical to apply this advantage to better understand the results of scientific inquiry. The HVS is well equipped to recognize patterns and detect deviations from expectations to speed up the process of analyzing data. Further, by application of specific visualization methods we can combine sets of seemingly disparate data of high dimensionality and subsequently recognize trends and relationships that might otherwise go unappreciated.

Users are able to more efficiently parse their data by visualizing more than two dimensions at a given point in time. More importantly, however, visualizing multiple dimensions ($\geq$3) simultaneously allows spatiotemporal trends to become visible to the user. Rendering a depth and time dimension, for example, makes possible the appreciation of spatiotemporal varying quantities such as shear stress imparted on the wall of a blood vessel by moving blood throughout the cardiac cycle. Methods for rendering the depth dimension are an area of active research, but we will only discuss the basic concepts. Rendering of the depth dimension can be done via true stereopsis[1] or with depth cues.

The data deluge will continue to grow in the future as the cost of producing data is falling at a rapid pace [4,6,8] in concert with the continued increases in computing power. For this reason, visualization is well poised to serve as a relief to the data analysis bottleneck that will continue to grow. The process of visualizing data leverages the high capacity of HVS to reconcile patterns and trends in seemingly disparate data of high dimensionality.

[1] *Stereopsis* (also known as *binocular parallax*) is the result of the horizontal displacement of the eyes. In ultimately manifests as each eye seeing a different image, which is then integrated by the HVS as the depth dimension.

1.4 Challenges in Visualization

The process of visualization is iterative, as illustrated in Figure 1.1 and often requires continuous refinement of display parameters such as color mapping, opacity mapping or adjusting the size of the field of view. When truly exploring data, *a priori* knowledge of the best parameters is often not known. For this reason, the time expenditure in visualizing data can detract users from using a visualization process to its fullest potential. High performance software packages with large libraries of preconfigured settings can ameliorate this shortcoming; however, the costs of acquiring and maintaining these packages can be prohibitive. It is estimated that 70% of the 1.8 trillion gigabytes produced in 2011 were produced by individuals—not corporations or large organizations [6]. This figure speaks to the fact that the very producers of this large amount of data may not have the resources necessary to use these visualization packages. Lower-cost alternatives are therefore needed to increase access to visualization.

The adoption of the visualization process has been slow due to a variety of factors:

- Costs associated with the procurement of software and hardware/display technologies
- Lack of expertise in the interdisciplinary field of visualization—which involves computer science, computer graphics, display technologies, human-computer interaction and the field of research producing the data of interest
- The relegation of visualization to an “afterthought” status whereby visualization is not practiced until the data has already been produced, rather than in the process of creating the data

Thus, the grand challenge of today's visualization scientists is to address these smaller challenges and find ways of reducing the costs of access, providing resources to overcome the associated knowledge gap and advocating the use of visualization in the data creation process rather than as a post-processing step after the data has been created.

1.5 Current Visualization Methods

1.5.1 Display Technologies

Most displays, whether cathode ray tube (CRT), liquid crystal display (LCD) or projector, are raster devices whereby an image is displayed as a 2D array of picture elements ("pixels") [12] much like the discrete image seen in Figure 1.2. This can result in a loss of data fidelity and resolution [15] if the resolution of the data exceeds that of the display. Much of the scientific data produced today is inherently multi-dimensional and therefore feature selection (also called data abstraction) must be applied to reduce the data to a manageable number of dimensions. Several mathematical and graphical methods have been developed to accomplish this; some are listed below.

- Linear projections via Principle Component Analysis (PCA), also known as Singular Value Decomposition [16] may take the form of projecting the multi-dimensional data onto a 2D plane much like how traditional photography projects the 3D world onto a 2D sensor chip [17], or by resampling the data along a plane to create 2D slices through the data in any orientation.
- Rendering time-varying data as a series of moving images (this may actually provide a secondary advantage in that the HVS is most sensitive to changes in the field of view, rather than static images [18,19]).

- Other methods use depth cues known to produce the visual effect of depth using 2D rendering techniques (see Section 1.5.2).

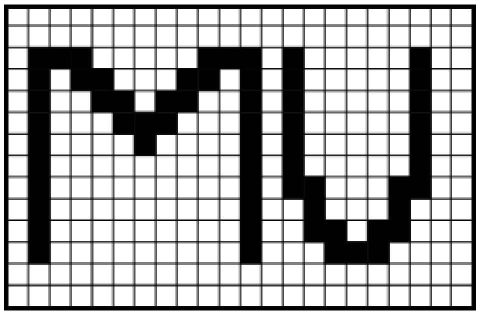


Figure 1.2 – Typical raster display configuration
A raster display is an array of pixels that can images as combinations of pixels. Raster displays can take the form of televisions, computer monitors and projectors.

1.5.2 Computer Graphics and Human Depth Perception

Computers generate images from graphics primitives such as points, lines and polygons [12]. Realistic, and therefore complex, 3D objects are composed of many millions of graphics primitives. At a fundamental level, computers are only capable of rendering combinations of discrete graphics primitives. Software libraries are therefore used to calculate the combinations of millions of graphics primitives that, when aggregated, describe the realistic 3D object.

Humans perceive depth by integrating differing sources called depth cues, whereby we interpret features of an image to communicate information about the relative position of objects in the depth dimension [15,17–19]. As discussed in [15,17] these cues usually have a cumulative effect whereby the greater number of cues visible increases the strength of the perception of depth. In specific combinations, however, depth cues can detract from the perception of depth [17]. These cues can be rendered in a raster display because they can be faithfully recreated in the 2D pixel array. Some examples of depth cues include:

- Occlusion (also called interposition) [17]
- Relative Size
- Relative Brightness

These cues are illustrated in Figure 1.3. There are others; however, the mechanisms by which they communicate depth become quite complex and are beyond the scope of this thesis. In general, the diagrams seen in Figure 1.3 communicate some small feature of the two objects' relative location, such that the viewer is able to draw conclusions on spatial position. Computer-generated images that judiciously combine these and other depth cues create a greater sense of realism than those that do not use depth cues [17,18].

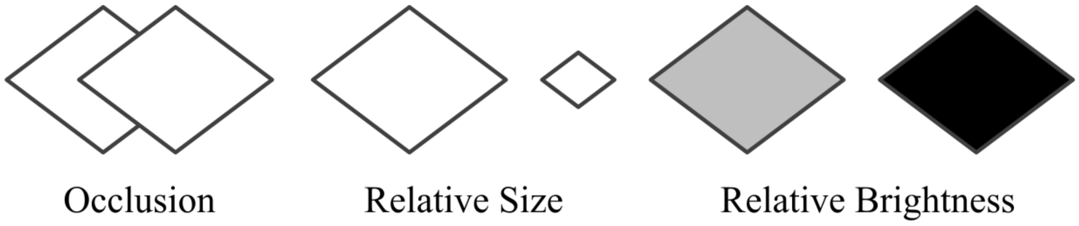
- 

Figure 1.3 – Typical pictorial depth cues
Depth information can be communicated with pictorial depth cues. These cues are interpreted by the human visual system as the depth dimension. This image adapted from [15].

The level of realism that can be achieved by application of depth cues is limited. To properly visualize the depth dimension one must consider the phenomenon of human binocular parallax. The physical location of the human eyes on the head creates an approximate 60 mm horizontal displacement between the images captured by each eye [17,20]. As a result, the HVS produces slightly different images in each eye. This is illustrated in Figure 1.4 in which the same object is viewed from two slightly different positions corresponding to the left and right eyes. This image shows that the left eye captures additional information on the left side of the object, and complementary results are obtained from the right eye. A series of neural integrations is performed by the HVS to register each image and interpret the non-overlapping image data as depth information [17].

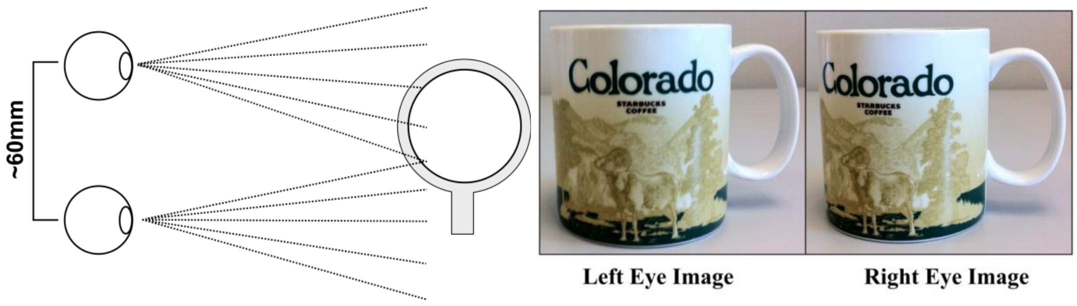


Figure 1.4 - Stereopsis / binocular parallax
The left image is able to observe features on the left side of the mug because of its horizontal displacement relative to the right eye. The right eye is able to observe additional features of the cup not visible to the left. The HVS uses the region of the two images that do not overlap to establish the depth information necessary for depth perception.

Stereoscopic displays—regardless of their mechanism of operation—aim to recreate this binocular parallax to induce perception of the depth dimension by the HVS. The first device to recreate this effect was the stereoscope produced by Charles Wheatstone in 1838 [21,22]. The stereoscopes seen in Figure 1.5 create this effect by presenting a pair of images distinctly to each eye, where the eye's field of view (FOV) is completely encompassed by the respective image. In Figure 1.5A a mirror combines two distinct images and reflects them into the eyes of the observer. In Figure 1.5B a disc contains stereo image pairs on opposite diameters. As the disc is placed into position in the viewer the image pairs are seen separately by each eye and depth is communicated to the user. Finally, Figure 1.5C shows a modern head –mounted display (HMD) that uses small LCD displays for each eye to create the stereopsis effect for a single user. As computer and display technologies have advanced, modern systems that display images with the stereopsis effect do so by means of one of several multiplexing techniques.

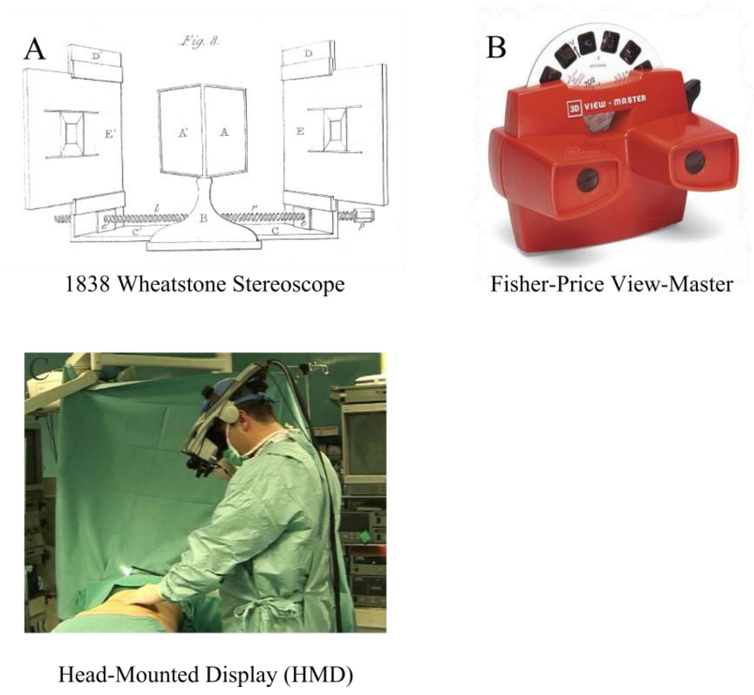


Figure 1.5 - Stereo viewing devices
(A) Original stereoscope device developed by Charles Wheatstone[2]*. (B) Fisher-Price View-Master device. (C) Modern Head-Mounted Display (HMD) being worn by a surgeon*[3]*. All three devices operate by presenting horizontally-offset images to each eye to create the stereopsis effect.*

Image multiplexing allows a single raster display to communicate images for both eyes. A multitude of technologies have been designed to accomplish this, each with its

[2] From [22]
[3] Reprinted, with permission: Urey *et al*., "State of the Art in Stereoscopic and Autostereoscopic Displays," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 540-555, Apr. 2011.

own advantages and disadvantages. Each technology also presents some means of presenting two images either simultaneously or in rapid temporal succession such that the HVS interprets them as a single image. Each method includes some means by which each eye is presented only with the image that recreates its perspective on a real world scene. A brief overview of these technologies follows; however, many technical details have been omitted in the consideration of the scope of this thesis.

1.5.3 Color Multiplexing for Stereoscopy

Color Multiplexing Techniques combine the color spectra of the left and right images to create a single image. The two sub-images can then be separated based on their color content [17,23]. These types of images are known as anaglyphs. The combined color content can be seen along the sacrum in Figure 1.6B. This technology has existed since the 1850s [23,24] but has seen resurgence in recent years with more sophisticated algorithms for color combination emerging [25]. Special filtering glasses are worn to resolve the left and right images. It is convention for left images to be encoded with a red color and right images with a cyan color; which is reflected in the colors of the lenses seen in Figure 1.6A. This technique has the advantage that it can be realized with standard video displays and low-cost eyewear, keeping adoption costs low. Native resolution of the display is also maintained. However, due to the color multiplexing, an inherent loss of color information in each image is to be expected; reds and blues are not visible because they have been filtered out of the image [need to verify/confirm this]. The advanced techniques described in [25] attempt to counter this, but some cross-talk between images remains.

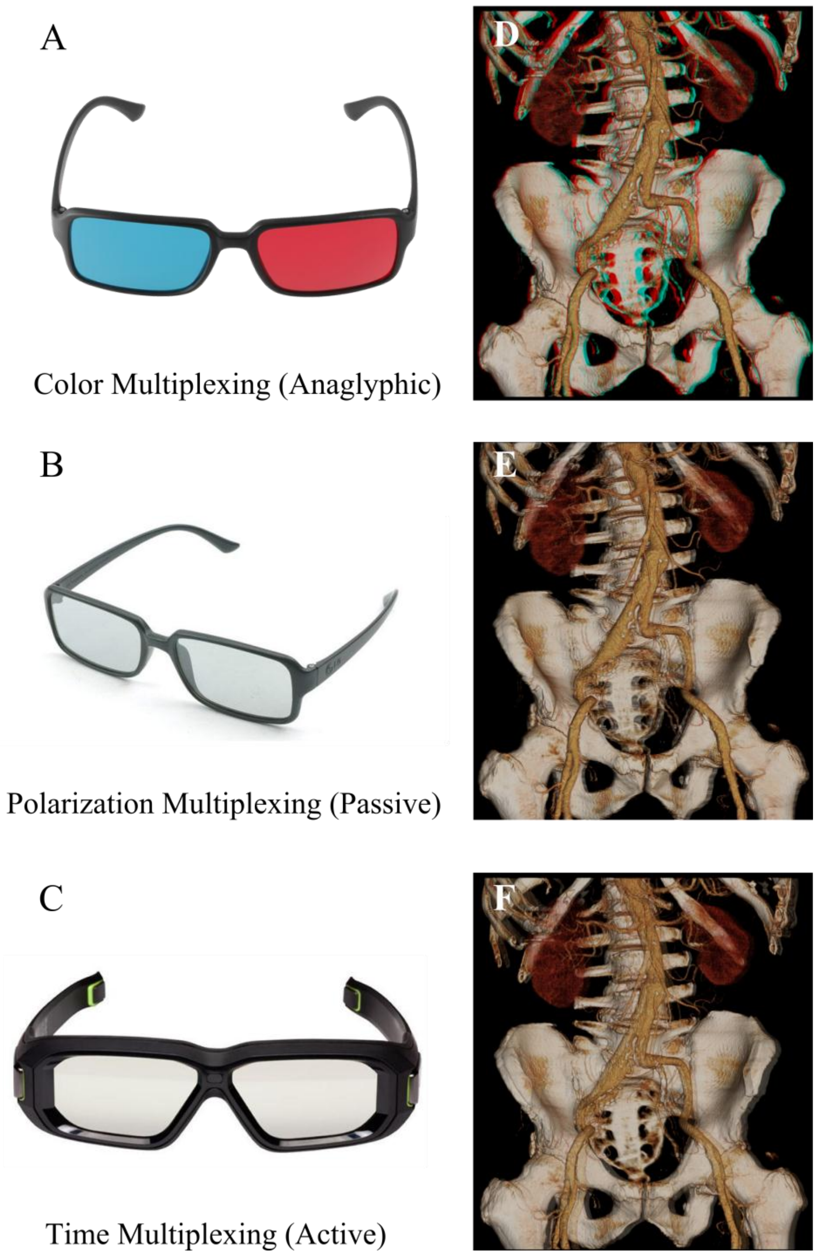

A
D
Color Multiplexing (Anaglyphic)
B
E
Polarization Multiplexing (Passive)
C
F
Time Multiplexing (Active)

Figure 1.6 – Stereo eyewear and multiplexing techniques
The anaglyphic multiplexing technique (A, B) uses color to simultaneously rendering images for the left and right eyes; the accompanying glasses are color-polarized to separate the two images. Similar techniques are used for polarization multiplexing techniques (C, D) where alternating pixel rows are polarized mutually perpendicular and separated by stereo eyewear. In (E, F) time multiplexing techniques flash alternating left and right images that are selectively occluded with the accompanying eyewear.

1.5.4 Polarization Multiplexing for Stereoscopy

Polarization Multiplexing Techniques selectively polarize the light emitted from the display and are able to resolve two separate images creating the stereopsis effect. Alternating horizontal lines (pixel rows) in the display are assigned to each eye. A polarizing filter is applied to each row whereby even rows are polarized in a direction orthogonal to odd rows [26]. These filters can be either linear or circular, where linear opposing linear filters orient sequential rows perpendicular to the next. Circular polarizing filters alternate between clockwise and counter-clockwise orientations. In practice it is more common to polarize with circular filters as the fidelity of depth perception is less affected by the user's head motion [23,26]. The resulting image displayed on the screen will appear striated in the horizontal direction as seen in Figure 1.6D. Images are resolved by glasses whose lenses are polarized to match the polarization of the light emitted from the display as seen in Figure 1.6C. Because the glasses are simple polarizing lenses, the technique has become known as "passive stereo." For each frame shown on the display, 50% of the pixels are dedicated to displaying the left image, while 50% are dedicated to the right. Thus, there is a 50% reduction in horizontal display resolution expected with this stereo technique. Color fidelity and image brightness are maintained, as compared to color and time multiplexing techniques. The glasses are inexpensive, which counters the comparatively higher cost of the displays as compared to those required for color-multiplexing techniques.

1.5.5 Time Multiplexing for Stereoscopy

Time Multiplexing Techniques rely on the fact that the HVS is insensitive to image flickers above a critical flicker frequency (CFF) [27]. This frequency depends on many factors including: image size, brightness, frequency of flicker [28] and properties of the retina [29]. The HVS is not able to perceive the transition between images presented above the CFF. Flickering images presented to the HVS at rates exceeding 60 Hz are generally perceived as continuous [27]. Images displaying the left and right eye vantage points are alternated in rapid succession, though only a single image is displayed at a given moment in time. Battery operated shutter glasses (Figure 1.6E) worn by the user allow one eye (whose vantage point is on display) an unobstructed view, while occluding the other. This state is alternated in synchrony with the displayed images. When viewed with the naked eye, the left and right images will appear to be combined creating a ghost image (see Figure 1.6F); when viewed with the shutter glasses only a stereoscopic 3D image will appear. Active stereo technology has been used extensively in virtual environments due to its relative technological simplicity and its ability to maintain full resolution of displayed content [30,31]. However, because the technology effectively occludes one eye (left eye closed 50% of the time, and vice versa), brightness is noticeably diminished in the stereoscopic images. Newer active stereo systems have attempted to combat this problem by decreasing the percent of the time each eye is occluded [32].

1.5.6 Computer Science and Visualization Software Libraries

Much computer software aimed to make visualization feasible has been developed since 1987. The graphics primitives that computers are capable of rendering must be programmed individually with explicit point-by-point definitions [12]. It is unreasonable to expect non computer scientists to program graphics data on such a level. To resolve this, software tools commonly called middleware have been developed. Middleware bridges the gap between low-level graphics interfaces (colloquially referred to as Application Programming Interfaces, or APIs) that require point-by-point programming by automatically generating the point-by-point data from high-level information provided by the user. These middleware contain functions that translate data commonly visualized, such as CFD results, into the graphics primitives that can be understood and displayed by APIs. Some of the most common graphics APIs are OpenGL, Open Scene Graph and Microsoft DirectX. Ultimately, the graphics API translates the graphics primitives into commands that can be understood and implemented by computer graphics cards and computer displays. One of the most common graphics middleware tools is known as the Visualization Toolkit ("VTK", KitWare Inc., Clifton Park, NY). VTK is well suited for visualizing scientific data. Figure 1.7 shows the hierarchy of visualization tools beginning with the data to visualize visualization software tools, middleware and APIs. Some of the visualization processing takes places in hardware, which are shown by the shaded parts of the pyramid.

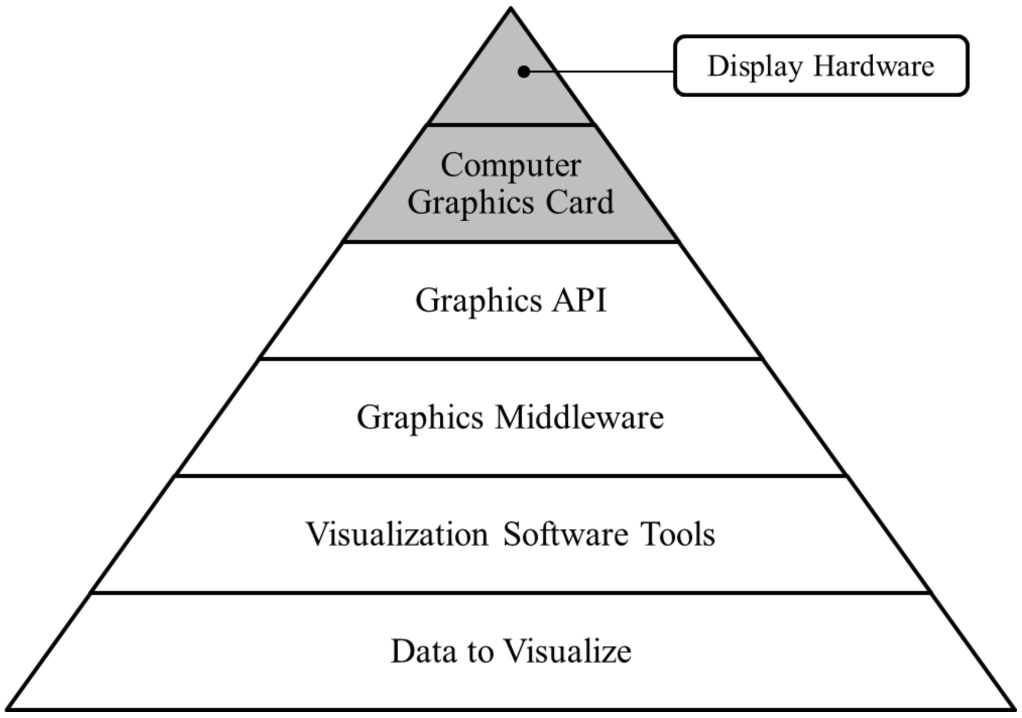


Figure 1.7 - Illustration of software hierarchy used in visualization
The creator of visualization sequences provides data to a visualization software tool, which then transforms the data by means of middleware. The data is then converted to graphics primitives and provided to the computer graphics card and the display. Software housed in specialized hardware has been shaded gray.

Despite VTK’s utility in creating efficient graphics, it still may not be feasible to use discrete VTK functions to visualize the data of interest. One of its practical limitations is that VTK still requires users to explicitly define objects to be rendered, although not to the same level of detail required by graphics APIs. To provide another layer of abstraction between the user’s scientific data and the graphics APIs, there are many software tools available to import the user’s data and transform it into commands used by VTK. These tools are distinct software packages from the “middleware” described above. These software tools are commonly used by scientists and researchers who value an efficient tool to visualize data over a tool such as VTK, which offers greater control, but requires more time and expertise to operate. These software packages can cost upwards of $20,000 to purchase, excluding annual maintenance costs and hardware costs [33]. As mentioned in Section 1.4, costs are a barrier to entry.

1.5.7 Human-Computer Interaction

As stated in its definition [12], the process of visualization entails interacting with the data in order to quickly arrive at new insights and a better understanding of trends and relationships within it. Many methods of interacting with data through visualization have been developed in the past decade. Some of these methods are quite basic—involving a computer mouse and keyboard—while others are more sophisticated and require infrared cameras and joint markers. A second, more mature system is the Ascension Flock of Birds (FOB) tracking system. The FOB tracks user movement by sensing the disruption to magnetic fields caused by motion of magnetic markers worn by the user. The location of the sensors can be used by software to control the visualization. The system works well but can be difficult to interface with software and is expensive when compared to other smaller systems, such as the Microsoft Kinect [34]. The Microsoft Kinect sensor is an emerging device well-suited for Human-Computer Interaction (HCI) applications and will be discussed in further detail in Section 3.1.2.

Other HCI research has been conducted investigating the use of haptic feedback whereby physical sensations are relayed to the user through physical contact with a device. This allows the user's sense of touch to be another dimension in which interaction with the data is possible. These devices require contact with a device and therefore limit mobility and range of motion. HCI devices like the Microsoft Kinect require no physical contact with the user and do not limit his range of motion.

Chapter 2: A Method to Process and Display CFD Simulation Data in an Immersive VR Environment

2.1 Computational Fluid Dynamics

CFD is a tool that can be used to study hemodynamic indices (those associated with the movement of blood) using computer-based vascular representations. This process generally involves creating a vascular model, discretizing the model into a mesh containing millions of elements, specifying rheological properties such as density and viscosity, prescribing the hemodynamic state at the entrance and exit of vessels (known as boundary conditions) and solving applicable governing equations with a powerful computer. Subsequent results such as wall shear stress (WSS; the frictional force experienced by a vessel tangentially due to flowing blood) and strain have been linked to the onset and progression of cardiovascular disease and can therefore be used to augment information obtained in a clinical setting [35–38]. CFD produces time-varying data for the model's millions of elements during an entire cardiac cycle, but the traditional way of viewing this data involves reducing multi-dimensional indices exerted on the walls of an artery to two-dimensions at a single time point and in a predetermined spatial configuration (see Section 1.4). In so doing, relationships between vessel features such as geometry, hemodynamic indices and atherosclerotic plaque morphology can be masked or not fully analyzed by medical professionals.

2.1.1 CFD with Immersive VR

Immersive visualization and virtual reality (VR) with stereoscopic rendering capabilities can mitigate some of these traditional limitations and are routinely used for other disciplines within academic and industry settings, including medical and surgical education [39–42] The emergence and adoption of these technologies in the consumer electronics and entertainment industries points to a progression that may soon extend further into biomedical applications. These advancements facilitate simultaneous viewing of multiple modalities that is particularly exciting for improved disease characterization and longitudinal study when used with CFD. VR was first used with CFD a decade ago to explore datasets at a single time during the cardiac cycle and using idealized representations of grafted carotid arteries [43]. Marked advances in high performance computing, computer graphics and patient-specific CFD modeling have occurred during this time that allow us to study more complex models. For example, a recent investigation demonstrated that the entire patient-specific CFD modeling process including model construction, simulation and quantification of hemodynamics in the carotid arteries is feasible within 49 hours, most of which were dedicated simulation and quantification [44]. With this timeline, it is conceivable for CFD to be used within a clinical setting to augment plaque morphology and flow information obtained from routine imaging modalities. This relatively time-efficient generation of simulation results, if coupled with a means of immersively visualizing indices of interest, could provide more meaningful information to medical professionals monitoring cardiovascular disease progression in at-risk patients.

2.1.2 Objective

The objective of Specific Aim 1 was to develop a method that integrates patient-specific CFD capabilities within an IVE system capable of stereoscopic 3D rendering. Moreover, we sought to simultaneously view spatiotemporal simulation quantities such as WSS, oscillatory shear index (OSI) and velocity along with their relationship to vessel pathologies immersively. To demonstrate how integrating advanced visualization enhances understanding, the investigation begins with a brief review of current patient-specific simulation capabilities followed by presentation of differences in the post-processing operations applied to prepare CFD results for viewing in an immersive environment, as compared to conventional approaches. This process is illustrated with two case studies using patient-specific CFD models generated from the carotid and coronary arteries. Importantly, each example discusses current clinical sequelae and potential sources of long-term morbidity thought to be influenced by adverse hemodynamic alterations, and attempts to highlight additional benefits afforded by immersive 3-D visualization for this purpose.

2.2 Currently Available Software Tools

The practical limitations of visualization middleware (as described in Section 1.5.6 and Figure 1.7) make the case for a higher level software package to make data visualization a realistic possibility. In consideration of the high costs discussed in Section 1.4, these software packages should be relatively low cost. There are commercial and open-source software packages that provide software methods and functions to produce visualizations in a time-efficient manner with higher level software tools. By leveraging such tools it becomes practical to apply these visualization methods to CFD simulation data. Open-source packages have the advantage of low cost and high degree of custom features; however the technical support for these packages is limited and there is usually no guarantee that software development will keep pace with hardware capabilities. Alternatively, commercial packages must provide support for the latest hardware technology in order to be competitive in the market. Two such packages were used during the course of the research and are discussed here.

2.2.1 EON Studio Visualization Software

EON Studio is a commercially available software package that facilitates the creation of 3D virtual environments and data visualizations [45]. The software architecture of EON Studio employs task-specific modules. When specific combinations of these modules are applied to input data, the desired output can be obtained and rendered. The EON package also has a software layer to allow source code to be supplied to automate the assembly and programming of these modules. Finally, the EON product includes a module to support multi-wall immersive rendering (discussed in further detail in Section 2.5.3). EON graphics are generated through the Microsoft DirectX graphics library and can be rendered on a variety of display formats including desktop monitors, Audio-Visual Experience Automatic Virtual Environment (CAVEs) and tiled wall displays. EON can be extended beyond its minimum functionality using company-provided add-on modules for an additional cost. These costs can be high, further illustrating the need for open-source solutions for advanced visualization.

2.2.2 COVISE – Collaborative Visualization Environment

COVISE is a set of three foundational software tools designed to make collaborative visualization across large distances possible [46]. COVISE is available commercially to corporate entities or as an open-source package to academic and other non-profit institutions. COVISE consists of a visual programming interface in which the user selects data and the desired transformations (*e.g.* scaling, isoplanes, volume rendering, slices or PCA) similar to the modules described in EON. It is through this interface that the data is manipulated throughout the course of the visualization. The second component is a display interface that allows for stereoscopic rendering with the Open Scene Graph graphics library.

The display interface is flexible and supports many rendering options such as CAVEs, tiled wall displays, single-desktop monitors and projector-based displays. Multiple formats of stereoscopic rendering including anaglyphic and active stereo are supported. The final component is a master/slave server framework that allows a plurality of computers to be arbitrarily networked to create a rendering cluster to distribute the load of rendering high resolution data sets quickly. Integrated into all the above elements is support for sending and receiving relevant visualization data with another location running COVISE to which vantage points, areas of interest, and audio can be synchronized among multiple parties to provide the same experience.

Collaborative immersive environments have been shown to augment traditional learning by students [47,48]. By sharing the experience the collaborative features of COVISE become apparent. Table 2.1 outlines the major features of each of these two packages.

Table 2.1 – Comparison between EON and COVISE
These software packages further abstract the process of visualization from the use of graphics middleware packages.

	COVISE	***EON***
Primary User Interface	Module-based UI with visual programming	Module-based UI with visual programming
Graphics Middleware	Open Scene Graph	Microsoft DirectX
Open Source	Yes	No
Rendering Formats	Desktop, CAVE, Tiled Wall	Desktop, CAVE, Tiled Wall
Stereoscopic Rendering	Anaglyphic, Active	Active
Extensibility	User created CPP modules	Proprietary modules

2.3 Data Acquisition Methodology

The methods developed to complete this specific aim are divided into four stages as illustrated in Figure 2.1. In general, the data for this aim were obtained through the acquisition of medical imaging data, preparing and conducting CFD simulations and applying a series of post-processing steps to quickly produce a scientific data visualization. Particular emphasis is placed upon the post-processing steps, and as such, will be treated in greater detail in Section 2.4.

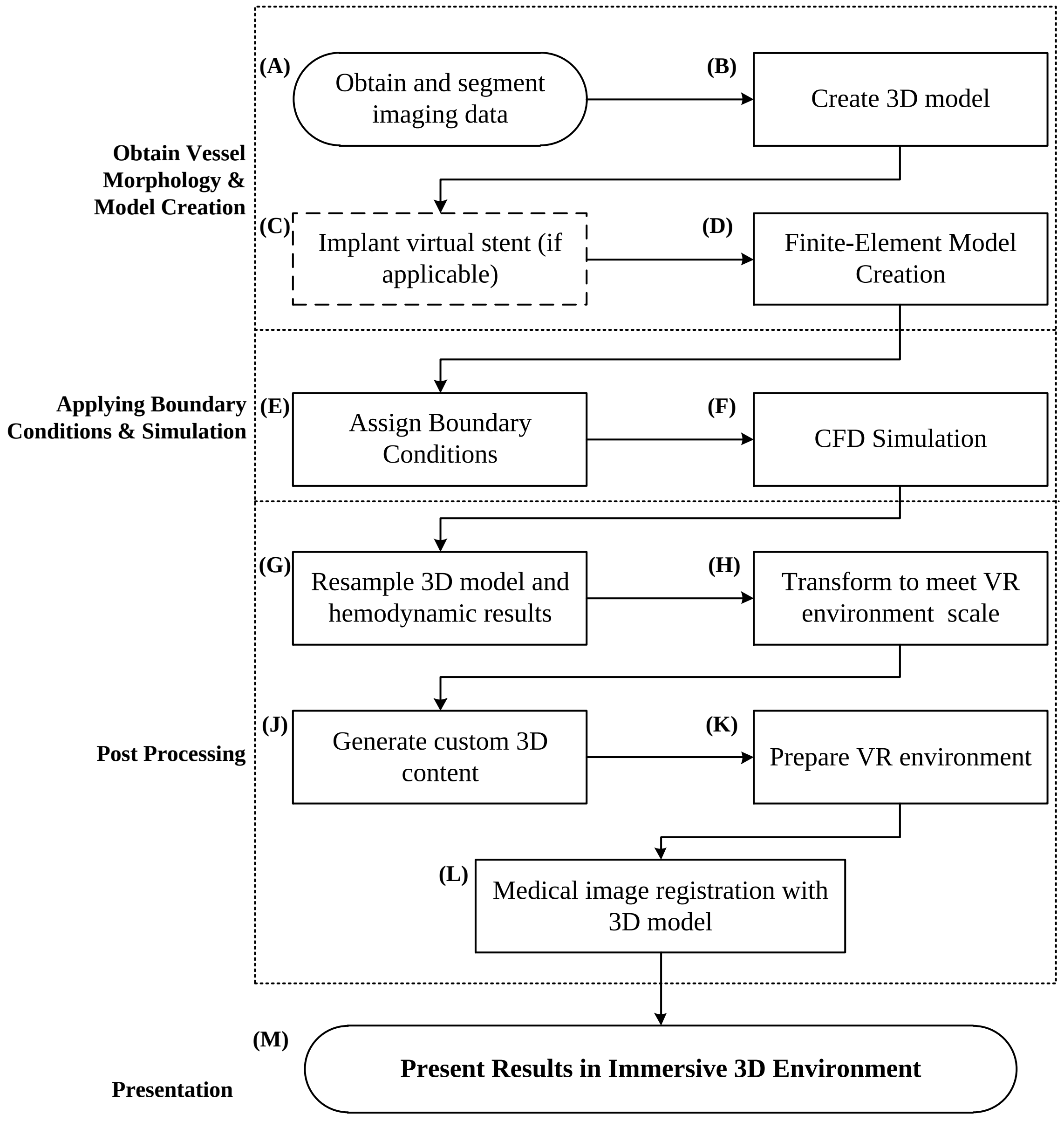

(A)
Obtain and segment imaging data
(B)
Create 3D model
Obtain Vessel Morphology & Model Creation
(C)
Implant virtual stent (if applicable)
(D)
Finite-Element Model Creation
Applying Boundary Conditions & Simulation
(E)
Assign Boundary Conditions
(F)
CFD Simulation
(G)
Resample 3D model and hemodynamic results
(H)
Transform to meet VR environment scale
Post Processing
(J)
Generate custom 3D content
(K)
Prepare VR environment
(L)
Medical image registration with 3D model
(M)
Presentation
Present Results in Immersive 3D Environment

Figure 2.1 – Method for rapidly visualizing CFD results
In (A-D) medical imaging data is obtained, a 3D model created, a virtual stent implanted and the entire mesh discretized prior to simulation. Next, boundary conditions are applied to the discretized model (E) and the CFD simulation conducted (F). Post processing steps are applied to resample the 3D model and hemodynamic data (G) and geometrically transformed for the VR environment (H). Custom 3D content is derived from the hemodynamic and geometric data (J) and sent to the VR environment (K). The final step is to co-register the medical imaging data with the model (L) prior to presentation (M).

The construction of the patient-specific CFD models begins with the acquisition of 3D medical imaging data (Figure 2.1A). The imaging data is then segmented by clinicians to identify the important vessel landmarks including the lumen and wall. We create the patient-specific 3D models of carotid and coronary vasculature (Figure 2.1B) using previously published methods [49–51]. In some models an intravascular stent is virtually implanted (Figure 2.1C) to better emulate patient-specific flow conditions using methods previously described [52]. A meshing algorithm is applied to the 3D model to discretize the volume to a finite-element mesh (Figure 2.1D) as described in [52].

Boundary conditions are applied to the model prior to CFD simulation (Figure 2.1E). Inflow boundary conditions of the vessel can be determined following their direct measurement, or the application of several assumptions from applicable literature. For the examples shown here, inlet boundary conditions were imposed using either normalized waveforms that have been scaled to the patient's body surface area [44,53], or a canine flow waveform that uses values for Reynolds and Womersley numbers that are reflective of human flow [49,54]. Outlet flow boundary conditions are prescribed using a three-element Windkessel model based on the blood pressure measured from the patient. The Windkessel model serves as a surrogate for the downstream impedance to blood flow and total arterial capacitance [55]. CFD simulations are performed using an in-house stabilized finite element solver with a commercial linear solver component LESLIB (Altair Engineering, Troy, MI) to solve the time-dependent Navier-Stokes equations (Figure 2.1F).

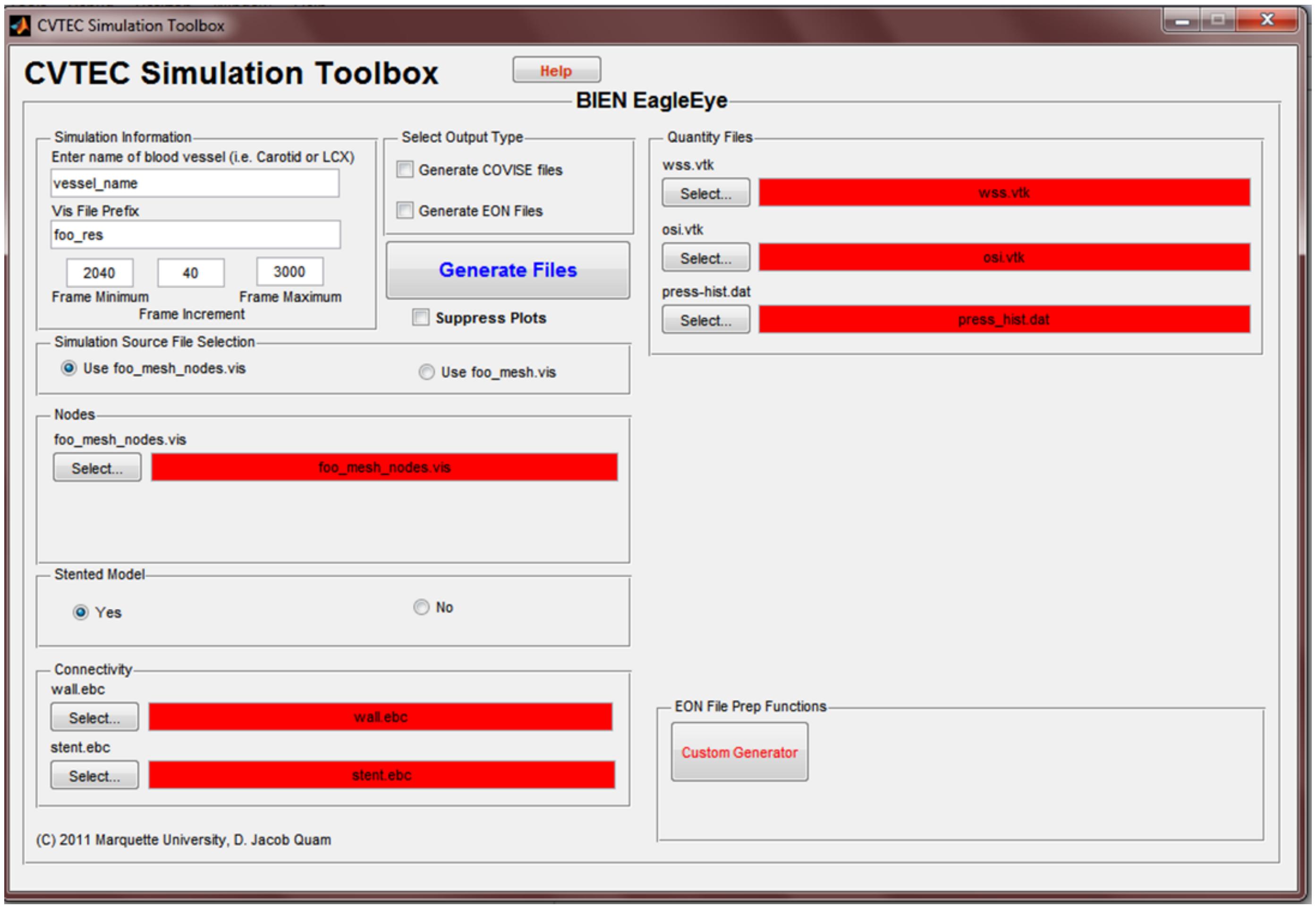
2.4 Post Processing Methods

Once CFD simulation data is obtained the data is further processed using a series of steps designed to prepare the data for use in an IVE. We use MATLAB (The MathWorks, Natick, MA) scripts in a majority of our post-processing methods. This software solution was selected for three primary reasons. First, the Marquette University College of Engineering maintains an active license for MATLAB software, and it therefore represents a tool that is likely to be available to future users of these methods. Further, familiarity with the MATLAB language is wide, and allows for the scripts to be understood by many users. The second reason MATLAB was used is due to the fact that our research group has a large repository of existing post-processing methods written in MATLAB for conducting separate CFD post-processing. Using MATLAB for the post-processing methods described here extends these methods without introducing a new programming language and allows some of the current methods to build from existing ones. The third and final reason for using MATLAB was that it allows easy construction of GUIs using a built-in tool called GUIDE.

In order to import our CFD data directly into the software that runs the IVE (called EON) we use Visual Basic Scripts (*.VB or *.VBS). This choice was dictated by the architecture of the EON software. The software allows scripts to be written in either JavaScript (JScript) or Visual Basic. Visual Basic was selected because it is well documented, runs quickly, and presented a lower learning curve to the developers.

2.4.1 Resample 3D Model and Hemodynamic Results

Post processing of 3D model data is conducted with custom software scripts. All post processing steps involving the use of MATLAB have been combined into a single executable file with the graphical user interface (GUI) seen in Figure 2.2. The GUI clearly specifies the files necessary to carry out the post-processing steps. It also provides an easy to use interface through which the user can locate files that may be in various locations. The GUI also enforces the types of files input to the algorithms. For simulations that conform to the expected use case, pressing the “Generate Files” button will correctly apply each of the post-processing steps described in this section. . Individual steps can be isolated from the rest of the process and executed separately by pressing the “Custom Generator” button at the GUI’s bottom right panel. The order in which these events are applied is important, and the order used by the “Generate Files” button should be observed when applying steps individually. A detailed description of the GUI’s operation can be found in Appendix A on page 119.



CVTEC Simulation Toolbox

Simulation Information
Enter name of blood vessel (i.e. Carotid or LCX): vessel_name
Vis File Prefix: foo_res
Frame Minimum: 2040
Frame Increment: 40
Frame Maximum: 3000

Select Output Type
[ ] Generate COVISE files
[ ] Generate EON Files
[ ] Suppress Plots

Simulation Source File Selection
(●) Use foo_mesh_nodes.vis   ( ) Use foo_mesh.vis

Nodes
foo_mesh_nodes.vis: 🟥 foo_mesh_nodes.vis

Stented Model
[✓ Yes] No

Connectivity
wall.ebc: 🟥 wall.ebc
stent.ebc: 🟥 stent.ebc

Quantity Files
wss.vtk: 🟥 wss.vtk
osi.vtk: 🟥 osi.vtk
press-hist.dat: 🟥 press_hist.dat

(C) 2011 Marquette University, D. Jacob Quam

Figure 2.2 - MATLAB User Interface
A MATLAB graphical user interface was developed with the GUIDE toolbox. All required input files are identified with a red box that turns white when a suitable file has been selected. The entire post processing step can be run with a single click "Generate Files" button, or individual steps can be run selectively by pressing the "Custom Generator" button at the bottom right.

The finite element mesh developed in Figure 2.1D is resampled to remove those points not on the surface of the vessel by comparing their locations to a connectivity matrix supplied by the meshing software (see Figure 2.1G). Duplicate points are removed and the data is resampled and stored in an unstructured grid format in three dimensions as seen in Figure 2.3. VTK functions are used to calculate the unit normal vector for each element of the finite-element mesh. The normal vectors are used to interpolate a smooth vessel surface that improves the viewing experience. The Cartesian coordinates, normal vector and neighbors of each node of the mesh are then stored as a connectivity matrix. This matrix is used to interpolate the data onto a uniform rectilinear grid where the spacing between each node is uniform in all directions. The rectilinear grid is required by the visualization software to reduce memory requirements for displaying the data. A virtual intravascular stent is part of some CFD models. In these cases, the stent model is processed using the same steps described above but as a separate object from the vessel model. The two models are combined in the VR software to allow the visualization properties of each to be controlled independently. The last step is to apply the same resampling procedures to the hemodynamic data obtained from simulation to ensure that each point of data directly maps to only one point on the vessel mesh.

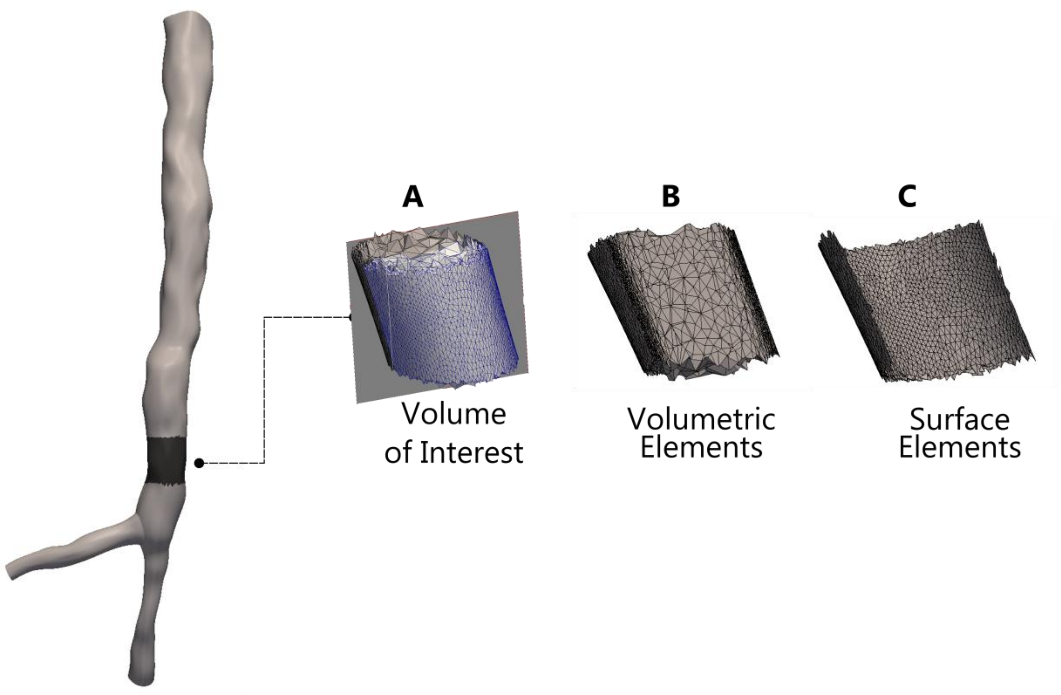


Figure 2.3 - Surface element extraction
A small ROI from a typical aortic model is shown in 3D (A). When cut through the segment's central axis, the volumetric elements are visible (B). Only those elements on the surface of the vessel remain after the surface extraction routine (C).

2.4.2 Transform Data to Meet VR Environment Scale

The geometries of the stent and vessel are rotated such that the direction of blood flow is parallel to the plane of the floor in the VR environment. This is accomplished with a MATLAB program that calculates a standard rotation matrix, Equation [1], about the Y and Z axes where α is the angle of rotation in degrees, and applies it to each node in the mesh. When the rotation operation is complete, the vessel is oriented such that the user's line of sight is along the vessel's central axis in the direction of blood flow. The value of α is not fixed and is unique for each dataset.

[1]
$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

The vessel is scaled by a constant factor to maximize the field of view (FOV) when rendered in the IVE. The factor varies based on the original size and orientation of the vessel in space. Vessels of a shorter length will undergo greater scaling than longer vessels. In addition to scaling, the vessel is translated to fix its geometric center at the origin of 3D space. A general correction factor, Equation [3], is calculated for each point in Cartesian space and applied as seen below. $\eta_{\max}$ and $\eta_{\min}$ represent the spatial boundaries of the dataset, and are used to calculate the translation in Equation [2]. By fixing the point midway between them to the origin, the entire vessel is centered at the desired location.

[2] $$\delta = \frac{\eta_{\max} - \eta_{\min}}{2}$$

[3] $$\eta_{shift} = \eta_{\min} + \delta$$

Manual orientation or location adjustment of the vessel is sometimes required for optimal results; this is easily accomplished with EON's user interface. EON rotations are described in terms of roll (counter-clockwise about the x-axis), pitch (y-axis) and yaw (z-axis). By applying these three transformations to the vessel geometry we ensure that each experience in the VR environment is controlled and represents the data in a consistent manner.

2.4.3 Generate Custom 3D Content

Some visualization content in the IVE is rendered directly from the CFD simulation data, while other content is derived from simulation data (Figure 2.1J). For example, blood velocity is visualized with time-varying vectors (arrows) indicating the direction of blood flow at each point in the cardiac cycle. These arrows are distributed through the vessel’s flow domain by a MATLAB script that accepts user input to determine the density of the arrows and their proximity to each other and the vessel lumen. In additional to the direction of blood flow being indicated by the angle of a particular vector’s arrow, which is calculated for each frame, an arrow’s length and color denote the magnitude of the blood flow velocity. The color is determined through a lookup table whereby the scalar value of the velocity magnitude is linked to a Red/Green/Blue (RGB) color code.

An indication of the temporal position normalized to the cardiac cycle provides a fourth dimension to data analysis in the VR environment. Blood pressure data is extracted from CFD simulation results and assembled in MATLAB to produce a plot of pressure versus time. Most content displayed in the IVE is allowed to move in space as the user navigates about the data; in most cases this is desirable. However, to communicate temporal data, a window with fixed spatial location and scale is needed to properly convey the point in time corresponding to the data being displayed. Therefore, a fixed viewing window is established in the IVE so that as the viewing angle of the data is changed, the pressure plot remains stationary and easy to read. The VR software maintains an internal chronometer that synchronizes events in time. A dot indicator is moved incrementally along the pressure plot at each time point in the cardiac cycle. This movement communicates the passage of time and relates the instantaneous hemodynamic indices (such as WSS) with other values like pressure and blood flow velocity that are displayed simultaneously.

2.4.4 Prepare Data for VR Environment

The VR software (EON Reality: Irvine, CA) manages all data with a hierarchical, modular structure as described in Section 2.2.1. Once the data has been prepared with the methods described above, it is imported to EON using custom Visual Basic (VB) scripts that control where and how the data is stored within the hierarchy (Figure 2.1K). The hierarchy decreases rendering time and allows for a greater level of control in specifying what combinations of VR elements are realized. The file system employed by EON aggregates the necessary data for the simulation into a single resource file, making storage and transport of the VR content simple since only two files must be managed by the user.

The vessel geometry data file is read by a VB script and converted to a 3D structure representing the vessel lumen. Similarly, if present, the stent geometry file is converted to a solid structure and combined with the vessel model. Time-varying hemodynamic data are treated as separate files for each point in time when imported to the IVE. The data contained in each file is then read, processed and rendered for each frame of the simulation. This process renders each file in rapid succession to produce the effect of moving objects in the VR environment.

2.4.5 Medical Image Registration with 3D Simulation Results

Image registration is the final step in the post-processing stage (Figure 2.1L). A separate 3D plane is created in virtual space for each medical image that will be registered with the 3D vessel model. These planes are added at a predetermined interval and do not necessarily represent the proper full size of the corresponding medical images. The planes are transformed using an algorithm specialized for the type of vessel being modeled and the imaging modality used to create a particular CFD model. The algorithm first translates each plane to the origin of 3D space and applies rotations specified by the imaging modality (see Figure 2.4). These rotations can be unique for each, or the same orientation can be applied to all the image data. The final step is to translate the planes back to their original location on the model, where it will have the proper orientation as seen in Figure 2.4. The resulting planes intersect the vessel model at locations and orientations that accurately reflect the relative positions of the anatomy and images at the time they were acquired. Each medical image is then applied to the plane in a way that maintains its aspect ratio. The data is then ready for rendering in 3D within an IVE.

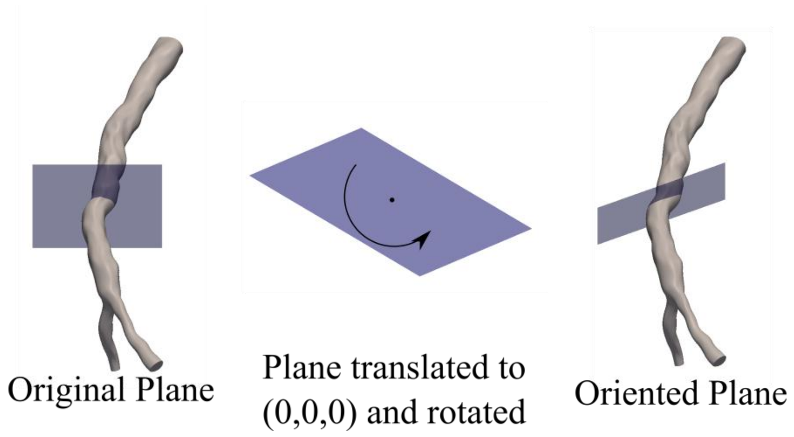


Figure 2.4 - Image Registration Transformations
The image registration process begins with planes that are placed in 3D space at predetermined intervals, than translated to the origin and rotated to match the orientation of the images as they are acquired in the body. The process is complete after the planes are translated back to their location along the vessel.

A detailed description of the MATLAB scripts used in these post-processing methods is found in Appendix A.

2.4.6 Presentation of Results

The Human Interactive Virtual Education (HIVE) environment, located at the Discovery World Museum (Milwaukee, WI), is an advanced visualization center equipped with 4 Christie Digital Mirage S+2K HD projectors capable of stereoscopic projection, 4 HP xw9400 workstations (2.8 GHz dual core processors, 3.5 GB RAM), CrystalEyes 3 active shutter 3D glasses and InterSense IS-900 Head Tracking equipment. The HIVE is designed in the style of the CAVE in which the user is surrounded by content on several screens. The HIVE system employs four large screens (front, left, right, and floor). The hardware layout used at the HIVE is approximated in Figure 2.5.

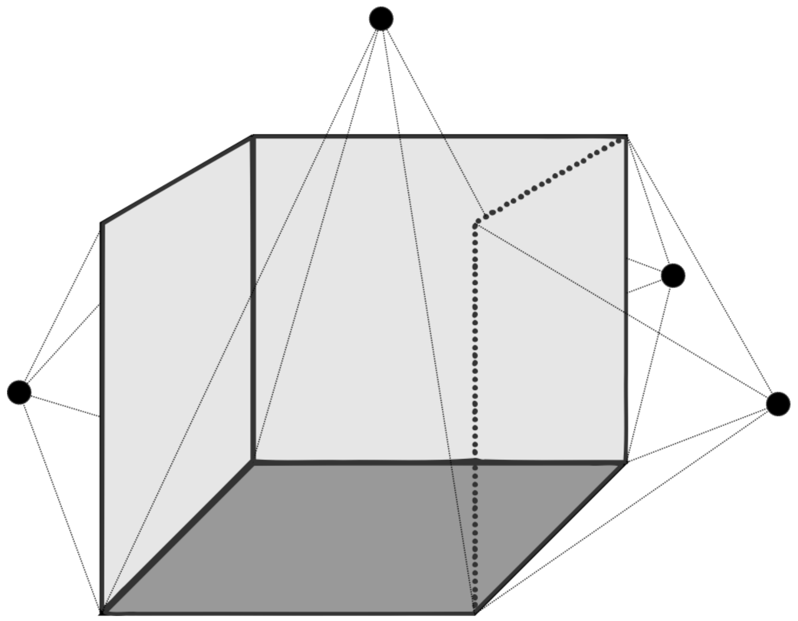


Figure 2.5 - Typical screen topology of IVE
Typical IVEs are rear-projected systems. The block dots in the image represent high-resolution projectors. Each screen is typically 10' on each side.

The VR simulations and analyses are displayed in 3D with active stereoscopy (Figure 2.1M). One frame of 3D simulation content is created from two projections of a single 3D object using unique points for the left and right eyes. The EON visualization file must be configured for projection in the IVE by programming the size and relative location of the screens in the HIVE. EON software manages the synchronization of the active shutter glasses and the alternating frames for the left and right eyes.

2.4.7 Simplified Methods for Expedited Results

In some cases, a less sophisticated method of viewing results in stereoscopy is an acceptable alternative to the more advanced methods described in Section 2.4. In these cases, COVISE software can be used in place of EON. When the user wishes to examine a single hemodynamic index—such as WSS or OSI—with less emphasis on data exploration, COVISE is a logical means of conducting visualization. COVISE offers similar results in less time, where the tradeoff is a lack in the ability to explore data immersively and that COVISE does not offer the same ability to view medical imaging data co-located with the CFD results. If the user is willing to forego these features, COVISE may be an acceptable solution.

Once the data has been read by COVISE, a representation of scalar value (such as TAWSS, OSI, or pressure) can be rendered onto the surface of the vessel. As compared to the process required for EON to render CFD results, the only required post-processing step is the resampling and surface extraction step described in Section 2.4.2; the other steps are not necessary. By eliminating these steps the time required to process the data is reduced. As described in Section 2.2.2, COVISE is capable of stereoscopic rendering in a variety of display formats, including desktop monitors, CAVE environments and tiled wall displays. For this aim, COVISE rendering software was used to drive a tiled-wall display with commodity monitors and PCs to create a rendering cluster capable of displaying CFD simulation results in stereoscopic 3D. These methods have been previously described in [56].

2.5 Results

The method developed and described above that integrates patient-specific CFD capabilities with an immersive visualization system capable of 3D stereoscopic rendering has been applied to two regions of the vasculature to highlight its flexibility in application. The carotid and left-circumflex (LCX) coronary arteries were modeled, simulated and rendered in an IVE.

2.5.1 Transformation and Registration Process

In preparing the vessel geometries for use in the IVE (Figure 2.1H) we rotate them such that the principal axis of the vessel is parallel to the floor of the IVE. This ensures that the initial viewing angle in the VR environment has a sufficiently wide FOV to observe all parts of the vessel. The carotid artery is rotated 235° about its Z axis, whereas the coronary artery is rotated 190° to achieve the same result. Both arteries are rotated -90° about the Y axis. This difference originates from the original orientation of the finite-element models, which are based on the orientations of the vessel within the patient's anatomy. We scale the coronary vessel model by a factor of 2.853 in all three directions, whereas the scale of the carotid model remains at 1. These results are summarized in Table 2.2 (below).

Table 2.2- Summary of transformation parameters
The parameters used during application of the visualization methods described here are specific to the region of the vasculature to which they are applied.

Transformation	*Carotid Study*	*LCX Study*
Z Rotation (degrees)	235	190
Y Rotation (degrees)	-90	-90
Scale (unitless)	1.0	2.853
Velocity Vector Spacing Constants: intra-vector/wall distance	1.5/0.5	3.0/1.0

The magnitude of the velocity vectors vary in magnitude relative to the maximum velocity in the cardiac cycle. The user is able to visualize the speed and direction of the blood at each point of the cardiac cycle using these vectors. However, they must be spaced throughout the volume of the vessel in such a way that their value is not lost due to too many vectors being present. A function was written to place seed points for these vectors at parameterized intervals throughout the lumen. The LCX coronary artery is a long, slender vessel with an average diameter of 2.85 mm [57] whereas the average diameter of the distal common carotid artery ranges from 7.8 to 8.8 mm for women and men respectively [58]. The carotid case spaces vectors with an intra-vector distance of 1.5 mm, and the minimum distance between a vector and the wall is 0.5 mm. Placing the vectors near the wall allows us to illustrate the no-slip conditions at the wall and also makes possible the visualization of areas of flow reversal because we are able to see the direction of the velocity arrows reverse, indicating retrograde flow. An intra-vector spacing constant of 3.0 and minimum wall constant of 1.0 were used in the coronary case. These settings decrease the density of the vectors in the vessel, but also decrease the computational expense associated with rendering a large number of elements.

Figure 2.6 demonstrates the image registration techniques developed and applied to carotid artery imaged with Magnetic Resonance (MR) imaging and the LCX coronary artery as obtained with Optical Coherence Tomography (OCT) imaging (Figure 2.7). MR images are acquired in the anatomical transverse plane, such that they are aligned orthogonal to the vessel's central axis. For this reason, registration of MR images only requires the images to be spaced evenly to correspond to the MR slice thickness (2 mm). In contrast, OCT imaging does not necessarily produce images orthogonal to the vessel's central axis, as is the case in MR. We calculate the most probable path taken by the imaging wire [49] and use it to calculate the rotations necessary to duplicate the image orientations as they were obtained *in vivo*. Equations [4] and [5] describe the angles calculated at the lower left corner ordered pair (x, y) of each image coordinate in the virtual space, where z is the depth dimension.

[4]
$$\alpha = \arctan\left(\frac{x}{z}\right)$$

[5]
$$\beta = \arctan\left(\frac{y}{x\sin(\alpha)+z\cos(\alpha)}\right)$$

Equations [6] and [7] show the rotation matrices used to first rotate the image plane about the Y axis then the X axis. The combination produces the final orientations seen in Figure 2.7.

[6]
$$R_y = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

[7]
$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix}$$

2.5.2 Visualization Results

Figure 2.6 and Figure 2.7 were generated using the visualization method described here. These images highlight the differences in image acquisition discussed in Section 2.4.5. Figure 2.6 shows the axial MR images with regular, 2 mm spacing that requires little processing effort to register them with the 3D vascular model. The OCT images demonstrate a less regular orientation than MR images due to the more sophisticated acquisition and registration methods (see Figure 2.7). The MR images have been processed to show plaque components and general morphology. TAWSS for the carotid and coronary arteries has been co-registered with imaging data in Figure 2.8.

Velocity information from the vessel is seen in Figure 2.9, where images obtained from three points in the cardiac cycle have been presented. Notice that the longest, darkest color vectors are observed immediately before peak systole. Velocity magnitude decreases as the cycle progresses. Notice also the near-wall velocities are markedly less than the velocities near the vessel center. OSI and TAWSS have also been rendered using the same method in Figure 2.9D and E. Figure 2.7 shows the LCX solid model with select OCT images displayed. These are the OCT images used in the construction of the CFD model. The orientation of the images reflects that of the imaging wire during the endoscopic study. The OCT images do not contain the same information on plaque morphology as in MR, but do provide high axial resolution data of the vessel lumen. Figure 2.7L shows a region in which a thrombus has formed as well as the turquoise tracing of the recently deployed stent.

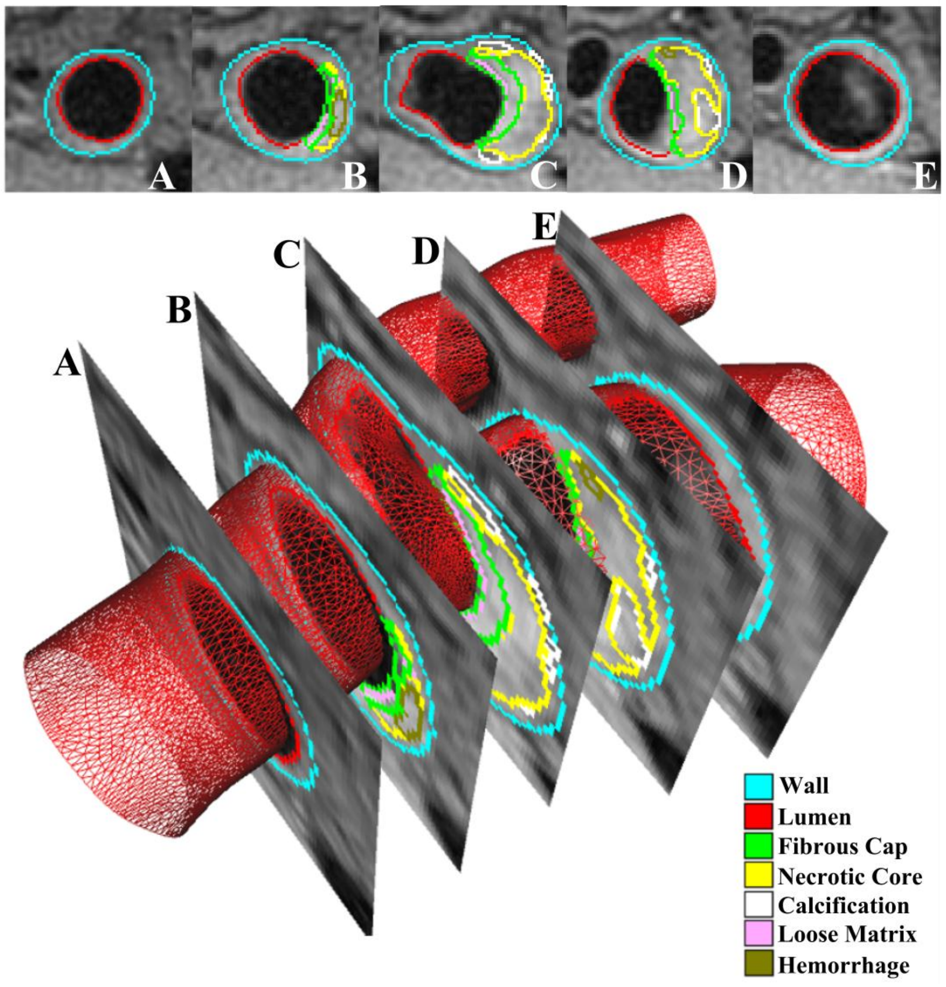


Figure 2.6 - MR image registration on 3D solid model of carotid artery
Slices A-E have been segmented to differentiate the plaque morphology according to the legend at the bottom left. In each slice the inner-most layer, the lumen, is used to create the vessel wall. The MR acquisition sequence collects transverse plane images which are thus in the plane of the vessel.

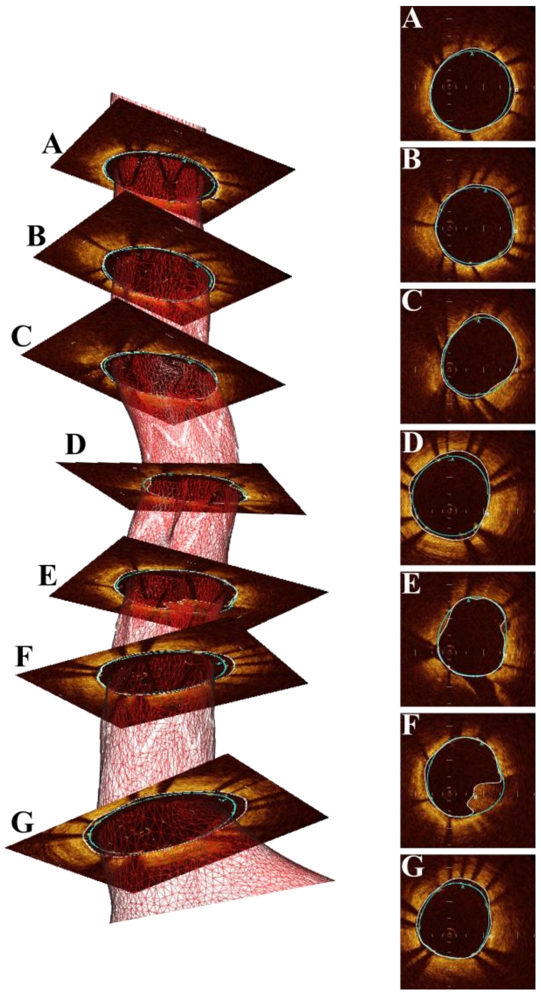


Figure 2.7 – OCT image registration on 3D model of LCX coronary artery
OCT images are acquired orthogonal to the plane of the imaging wire as it is retracted through the vessel; this produces images that are not necessarily orthogonal to the vessel's central axis. This can be especially appreciated in slices E and F. The thrombus seen in slice F is well resolved with OCT's high axial resolution. The outline of the stent is also well resolved as seen in the turquoise traces.

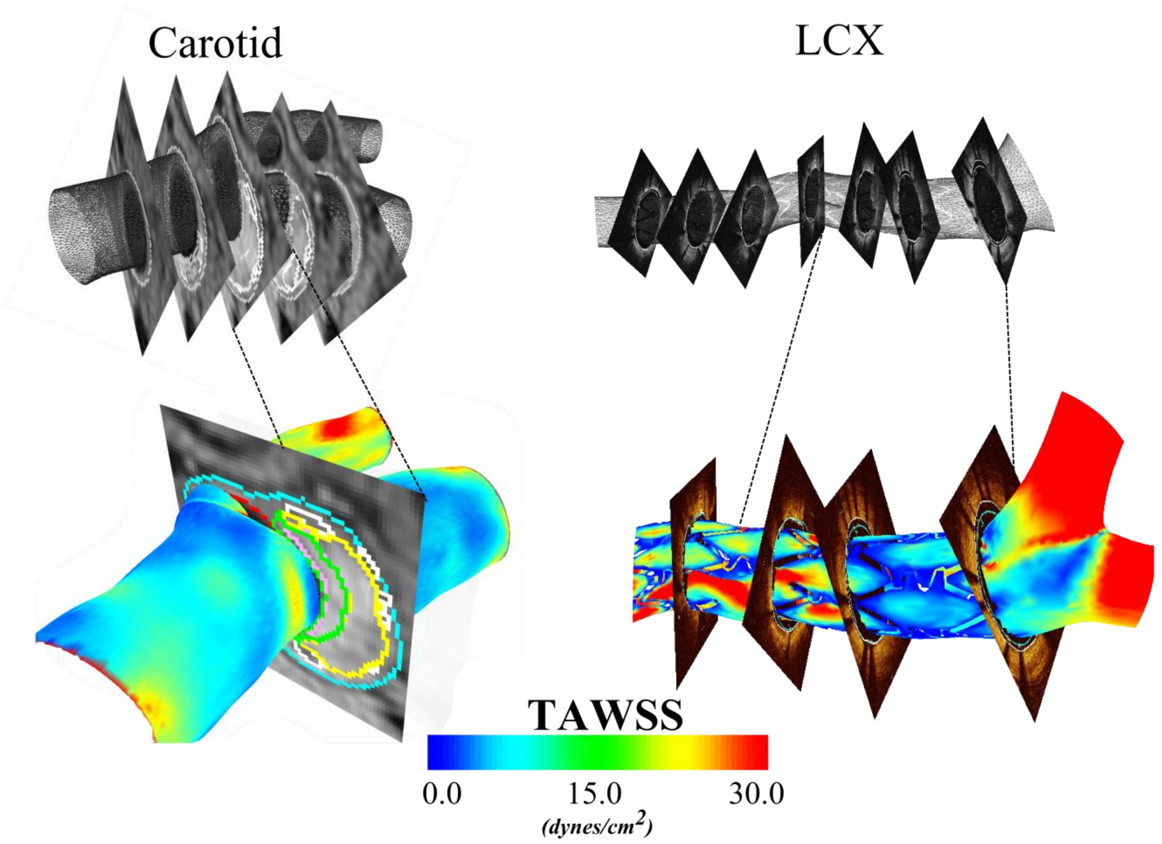


Figure 2.8 – WSS as Visualized with the IVEs
The IVE is capable of rendering many hemodynamic indices. Shown here is TAWSS co-located with the medical imaging data. At left the user can appreciate the co-location of the region of low TAWSS with increased wall thickness. At right, OCT images elucidate the relationship between stent location and regions of altered TAWSS.

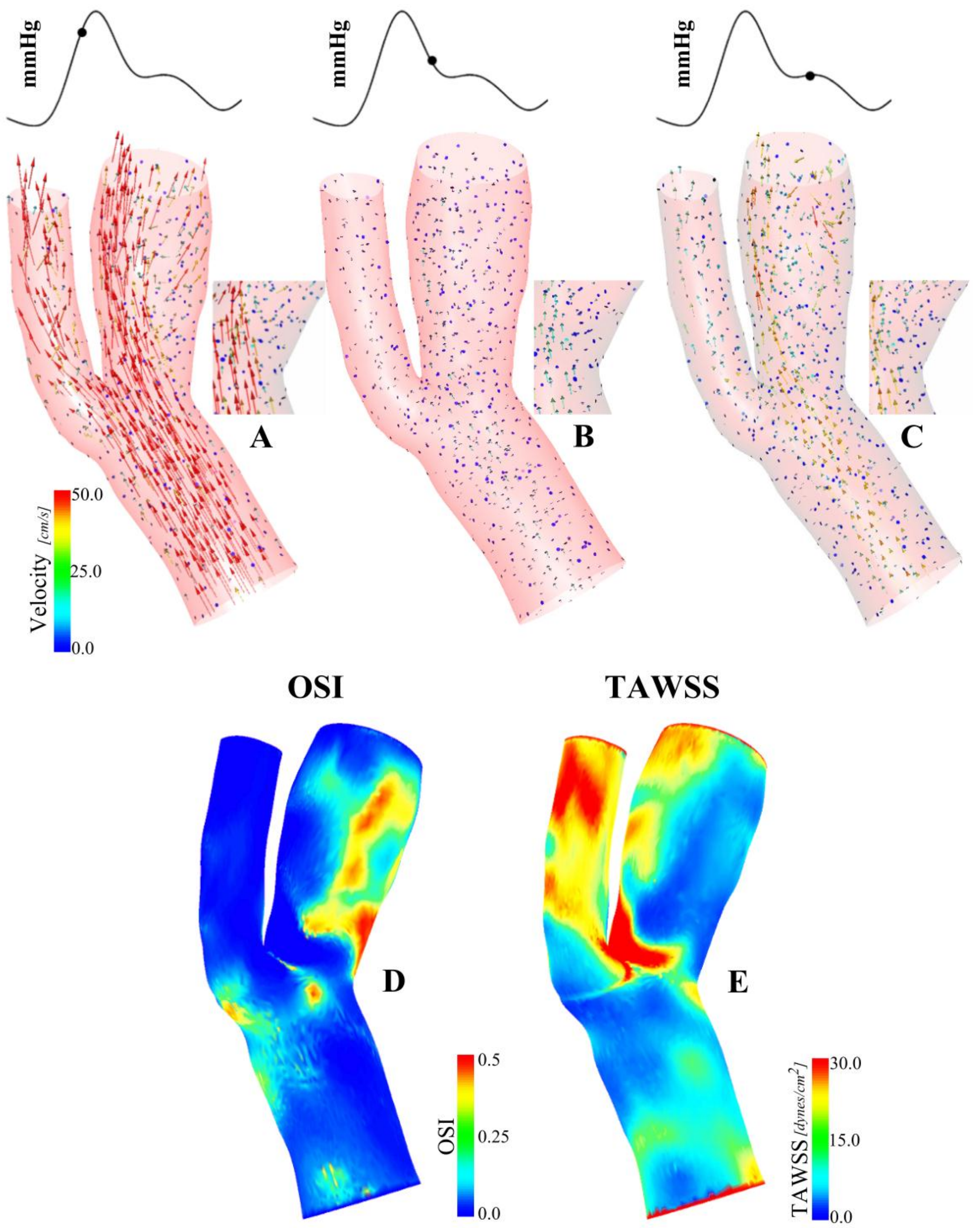


Figure 2.9 – Temporal progression of velocity at predetermined spatial locations
A, B and C show the velocity vectors at specific temporal locations as indicated with the pressure waveforms. The velocity is communicated by the vector magnitude and color; after reaching maximum velocity immediately prior to peak systole, the velocity decreases with time. (At top right) The inset shows a region of flow reversal which is consistent with the high values of OSI seen in the bottom left panel of the figure.

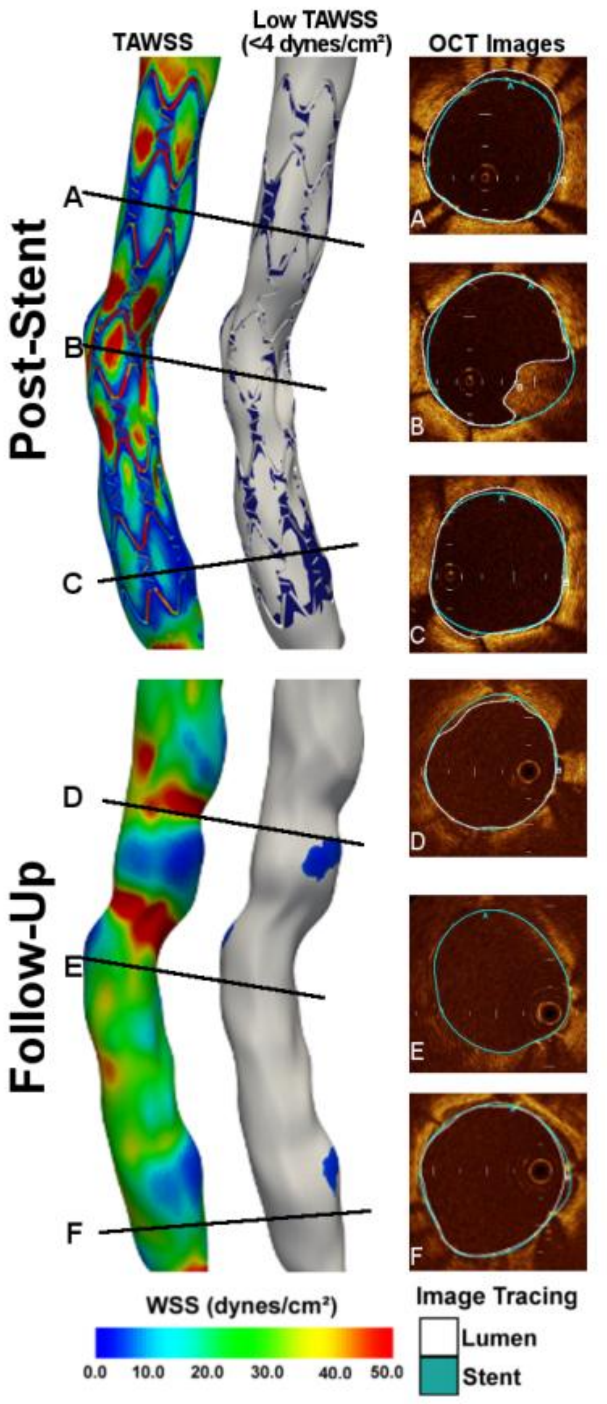


Figure 2.10 – Renderings of LCX coronary artery with quantitative results
These images show the CFD data at two time points- immediately following the stent implantation and at a 6-month follow-up appointment. OCT was used to acquire the slices seen at the right. The transverse lines in each model correspond to the OCT slices. We can anecdotally correlate the region of thrombus with high TAWSS seen in slice B with a region of relatively diminished TAWSS at the follow-up time point.

2.5.3 Human Interactive Virtual Environment (HIVE)

The results of applying this method to the carotid and LCX coronary arteries were realized in the HIVE at the Discovery World Museum. The immersive environment provides large-scale images that approach 10' in each dimension with a resolution of 2000x2000 pixels. Figure 2.11 shows a male of average height standing next to the projected wall of the HIVE. The user is also wearing the CrystalEyes glasses that facilitate active stereoscopic 3D content. The results generated by application of the method provide sufficiently high resolution data to provide a compelling experience. The velocity vectors described in Section 2.4.3 are seen pointing into the plane of the screen, indicating that blood is flowing in this direction. The fixed-location window used to indicate temporal location within the cardiac cycle is also seen in the upper left corner of the screen.

When displaying CFD data, the HIVE shows the flow of blood parallel to the floor, thus maximizing the user's FOV. Navigation within the IVE is carried out using a series of predetermined vantage points. The vantage points describe the viewer's position, orientation and FOV relative to the vessel in the immersive space. When the user selects a new vantage point using the computer terminal the visualization software interpolates a smooth transition to the new vantage point following a spline path. This increases user comfort while visualizing data in the IVE by reducing unnatural movements in space.

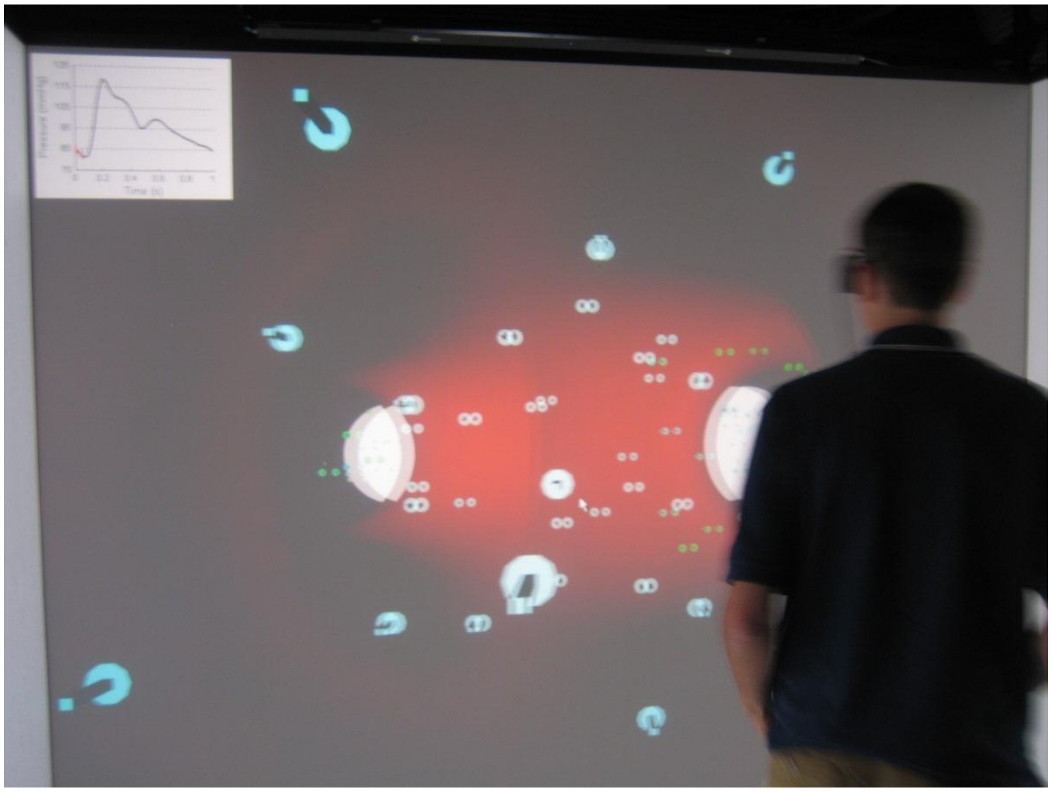


Figure 2.11 - Scale of the IVE
Seen in the figure is an average height male against the 10x10' square, back-projected screen. The ghosting effect inherent to active stereoscopic rendering can be observed along the right edges of the velocity vector arrows.

2.5.4 Mini-CAVE Hardware Prototype

Figure 2.12 shows the multiple configurations achieved with a set of four NEC MultiSync LCD 1880SX monitors arranged in a tiled-wall display. Panels A-C of Figure 2.12 demonstrate TAWSS on the LCX coronary artery, while Panel D shows TAWSS mapped to the surface of the carotid artery. The computer monitors used in Figure 2.12 are commercially available displays and can be procured at a modest cost. Because they are not specifically designed for rendering stereoscopic 3D images, stereo format is limited to anaglyphic stereo. Panel D shows a miniaturized CAVE topology that mimics that full-scale version seen in Figure 2.11. Similar immersive effects can be realized using this technique. The added benefit of using this software and hardware combination is that the expedited methods discussed in Section 2.4.7 can be applied. These results have been previously reported in [56].

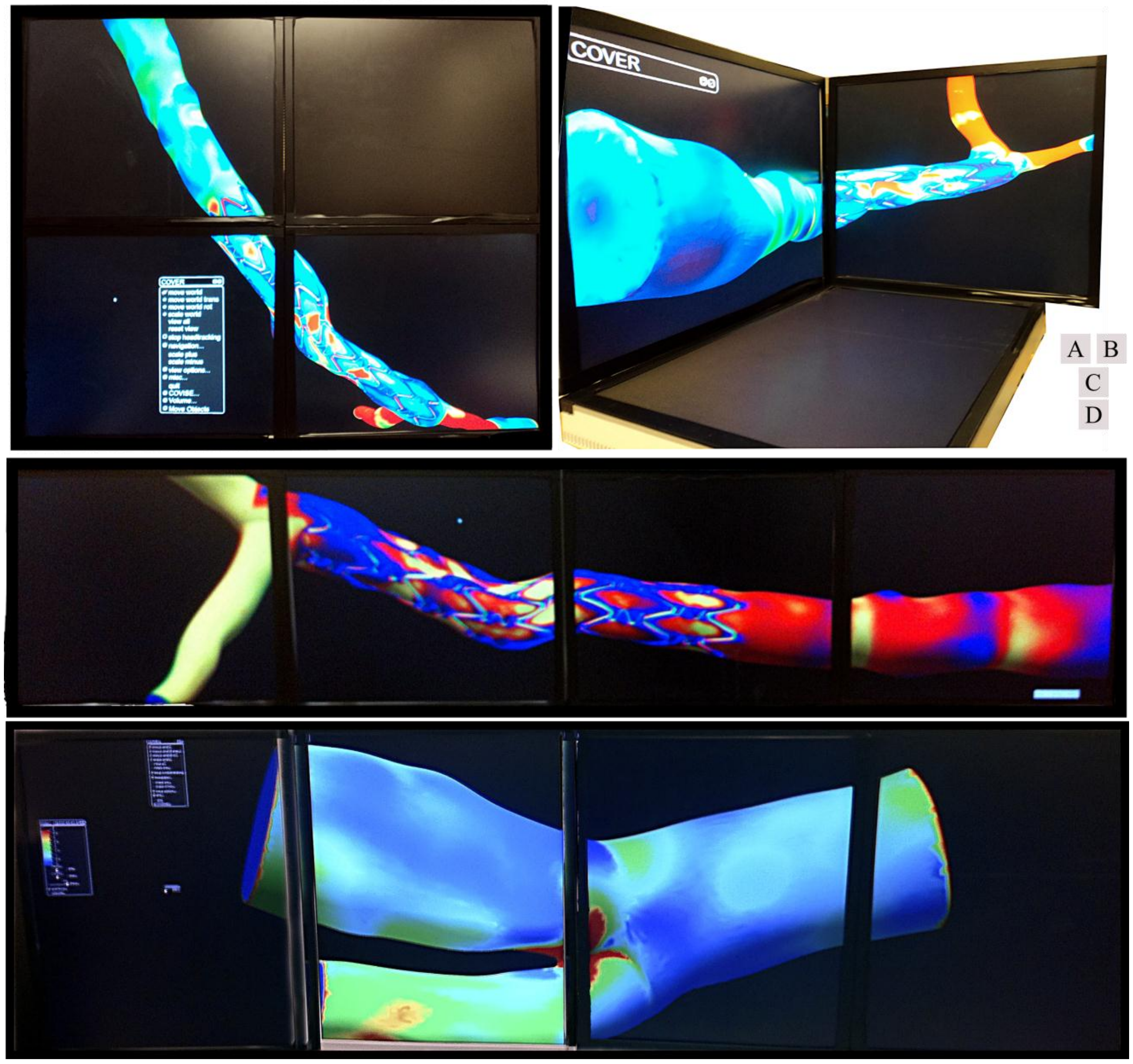


Figure 2.12 – MiniCAVE display topologies
Panel A is a 2x2 monitor tiled-wall display optimized for viewing models that have a low height to width ratio. Panel B exhibits a desktop-size CAVE simulator in which the walls and floor render images to create the immersive environment. Panel C shows TAWSS mapped with a non-standard color map along the LCX coronary artery. The 1x4 topology seen in C is well suited for vessel with a high height to width ratio. D shows TAWSS of the carotid artery mapped in a 1x4 tiled display with the long axis of the monitors emphasized.

2.6 Discussion

The objective of this aim is to develop a method to rapidly visualize patient-specific CFD simulation results for an IVE. The steps defined here describe a modest amount of post-processing conducted after the execution of the CFD simulations. In many cases, these steps have been shown to take less than 60 minutes, with much of that time dedicated to computation and not direct user interaction. Previous work done to integrate CFD with immersive visualization has used idealized vascular representations, rather than the patient-specific models shown here [2,43]. Whereas the work conducted in [43] emphasizes developments made in data interaction and user interfaces, our method attempts to highlight the utility of using IVEs to gain to scientific insights and less on the data interaction. We do so by simultaneously viewing spatiotemporal data and discussing clinical sequelae that can be appreciated with the current method.

2.6.1 Immersive Visualization for Patient-Specific CFD Data

The renderings seen in Figure 2.6, Figure 2.7, Figure 2.9 and Figure 2.11 show results rendered in the HIVE. The environment allows the user to be fully surrounded by the data, such that he observes the physiologic processes on a scale not possible with a standard desktop monitor. As described in [2], standard 2D and 3D desktop monitors are capable of rendering scenes as though observed through a window. IVEs allow users the ability to actively learn—viewing objects from within, looking underneath and handling objects in the virtual environment. It has been shown that when viewed from multiple angles and orientations, individuals better understand and retain spatial information and understand complex phenomena [59,60]. Previous use of IVEs for CFD results have focused on the interaction, and less on the meaning of the data. Our work builds upon this and uses current hardware capabilities to bring the focus of IVE use onto the data and how to gain new insights into biomedical research.

In addition to the large-scale images expected in an immersive environment, stereoscopic rendering adds an additional level of realism and information not possible with standard displays. As discussed in Section 1.5.1, any 2D display must apply some kind of feature extraction (such as PCA or simply sampling a 2D slice) to display multi-dimensional data on a raster display. In so doing, varying amounts of data are either eliminated or aggregated before being displayed in a decreased number of dimensions. These limitations are mitigated by the multiple screens of an immersive environment. Even if a fully immersive environment is not available, modern computer monitors are capable of rendering stereoscopic images, serving as an acceptable surrogate to the full immersive experience offered by true IVEs. Using a 3D computer monitor would limit the out-of-plane visualization possible with an immersive system, but the 3D shapes and details of the vessel and hemodynamic indices would be maintained.

2.6.2 Extension to Other Imaging Modalities

The increased prevalence of clinical imaging for noninvasive diagnostics has led to a great diversity in the modalities available clinically. We have used two modalities in the present study, MR and OCT, but contend that the method is extensible to other imaging modalities. A valuable feature of the results presented here is the differentiation of plaque morphology in the Carotid artery. Multiple MR imaging sequences (T_1, T_2, TOF and proton density weighted) were acquired and analyzed in order to acquire these results [44]. Recent advancements in dual energy CT (DECT) now offer similar tissue differentiation capabilities as multi-sequence MR [61]. Results have been presented demonstrating successful plaque-component differentiation using DECT scanners [62]. DECT may offer increased spatial resolution over MR images, although CT does introduce the use of ionizing radiation. Despite these concerns, this progression indicates that CT images could be adapted for use with this and maintain the ability to relate plaque structure with variations in hemodynamic quantities.

2.6.3 Case Study 1: Carotid Artery

In the first case study we present results obtained from a carotid artery investigation. In Figure 2.6 the reader is directed to observe the increased vessel wall thickness (where thickness is assumed to be the difference between the wall and lumen) visible in slices C and D along the external carotid artery, compared to the thickness in A. When viewed in an IVE, the wall thickness can be seen relative to the vessel length and diameter. This information can help the user build a 3D mental image of the carotid vasculature as it exists within the body. Such knowledge may be useful in clinical settings when planning procedures or analyzing results of medical imaging protocols.

The inset of the top right image illustrates how the visualization system displays a region of flow reversal, which is to be expected when one considers the high values of OSI seen at the same vessel location. OSI is an index of directional changes in WSS, where low OSI indicates the WSS is oriented predominantly in the direction of blood flow, while a value of 0.5 is indicative of bidirectional WSS with a time-average value of zero throughout the cardiac cycle [63]. The regions of high TAWSS near the flow divider of the bifurcation are consistent with previous studies [64,65]. The velocity vectors make it possible to correlate regions of low or no flow with the vessel’s structural features. As an example, the reader is directed to the top right panel of Figure 2.9 where the inset image clearly shows flow reversal with the backwards pointing velocity vector.

No single radiological viewing plane is currently capable of producing images in an IVE such as those seen in Figure 2.6; our method provides it. The IVE renders this composite image and allows the user to appreciate the direct spatial relationship between the flow domain (indicated with the wireframe 3D model), the medical imaging data and the vessel and plaque morphologies. These features relate the structure of the carotid artery, but we can also integrate vessel function by rendering WSS and OSI on the vessel surface and comparing these values to the structural information obtained previously. When examined in the context of vessel geometry and the cardiac cycle the user can begin to establish the spatiotemporal link between the three features. Figure 2.6 illustrates that regions of low TAWSS have been shown to be preferential locations for the development of atherosclerotic regions as seen by the co-location of the fibrous cap/necrotic core with the low TAWSS at slice C. The user can appreciate this by mapping the instantaneous or time-averaged WSS values to the vessel surface and rendering the segmented image slices as seen in the bottom right. By developing this new method of visualizing the CFD and imaging data simultaneously we have established a new way to use and understand the data not previously done.

2.6.4 Case Study 2: LCX Coronary Artery

The second case study highlights results from the LCX coronary artery. Rendering the OCT slices in an IVE allows spatial appreciation for the tortuous path through which the endoscopic imaging wire traveled during the OCT procedure. This is a surrogate for understanding the spatial dimensions and directions of the actual LCX coronary artery. By examining the model with the OCT slices shown, users can learn about the coronary circulation. Figure 2.7 shows the high axial resolution for which OCT has established a standing in clinical practice as a viable alternative to intravascular ultrasound (IVUS) [66].

It has been shown that low WSS creates an environment favorable to the development of atherosclerosis, and that these areas of low WSS can be characterized in vivo using the same techniques described here [67]. We can apply this method to in vivo patient data from at least two points in time to establish points of comparison needed to track disease progression. Figure 2.10 shows the same region of the vasculature with data captured at the time of stent placement and at a 6-month follow-up appointment. Representing data in this fashion allows conclusions to be drawn about parameters that change as a function of time. We have shown two temporal locations, but this practice could be extended to include many points representing snapshots of vascular performance throughout the course of a longitudinal clinical study or to track disease progression. With such data available it would be possible to visualize changes in hemodynamic quantities in both time and space.

2.6.5 Limitations

The current method is robust and flexible in its application when applied in the scope of specific considerations. The method is designed to accept CFD simulation results produced by the software used in our research group. The MATLAB functions written to process these results expects a specific format in order to perform as expected. This format generally conforms to the standardized VTK file format, which is a well-documented standard. Many other types of files can easily be converted to VTK format with open source software tools. As such, simulation results not originally in the accepted format can likely be converted by using one of several software tools commonly available.

The data interaction ascribed to visualization by [12] calls for natural manipulation of data by users. The current method indeed allows users to interact with data, but in a prescribed fashion using the predetermined vantage points described in Section 2.5.2. The current method could be enhanced by integrating the head-tracking hardware into the visualization code, allowing the user to re-orient the data using his head's position in space.

The type of clinically relevant information depends on the imaging modality used to acquire the anatomical data. MR imaging can provide specific information on plaque morphology and its spatial relationships with vessel structure. Alternatively, OCT provides high axial resolution of a vessel's lumen and does not pose MR compatibility concerns immediately following stent deployment. OCT does not currently provide the same degree of plaque differentiation abilities offered by MR, but does offer a demonstrated ability to differentiate layers of the vessel wall (*e.g.* internal and external elastic lamina) [68]. This method does not amplify the information captured by the selected imaging, but instead allows users to see the present information in more useful and compelling ways.

2.7 Conclusions

The use of an IVE presents many opportunities for enhanced learning through direct interaction with data and allows for a greater level of spatiotemporal knowledge of simulation results. The method described in this section has been developed for use with cardiovascular CFD results and a full-scale IVE. However, we have also shown flexibility in implementing these methods wherein a variety of small-scale displays can be used to realize these results with varying degrees of immersion. We have described the utility in applying this method to two specific regions of the vasculature, the carotid and LCX coronary arteries, with either MR or OCT imaging data. IVEs provide users with tangible data that can be explored in both time and space, thus providing enhanced learning and familiarization with the data of interest. This method may have clinical applications to those fields of practice that rely on knowledge of multi-dimensional data.

Chapter 3: Stereoscopic Visualization System with Intuitive Gesture-Based Control for Biomedical and Scientific Datasets

3.1 Introduction and Motivations

The ability to interact with data in any visualization setting is important for gaining the most information and insight from the data [2,11,69]. For this reason, extensive work has been conducted to advance the field of human-computer interaction (HCI). The goal of HCI is to create interface devices, methods and algorithms for allowing natural, intuitive interaction between humans and computers. This field naturally extends to visualization as computers are the engines that render and process the data used in visualization. 2D input/control devices such as keyboards and mice offer a familiar HCI, and have good spatiotemporal resolution overall. These same devices, however, can become an ergonomic obstacle requiring the user to step out of an IVE to interact with data. One possible solution is to use 3D input devices such as motion-tracking cameras that track color markers placed on the body, or instrumented gloves that transform body movements to commands. These devices suffer from high noise interference and often operate at a less desirable spatiotemporal resolution than their 2D alternatives [2]. These shortcomings allude to a need for a more robust and simple solution for data interaction that does not require body markers and extra equipment worn by the user.

3.1.1 Pre-Surgical Planning Tools

The inter-individual heterogeneity of human anatomy requires cardiovascular surgeons to use diagnostic imaging to familiarize themselves with the specific conditions present in patients prior to surgery [70]. General knowledge of cardiac anatomy may be insufficient because many patients present with cardiovascular architecture that has been distorted by disease. This is compounded by the fact that the myocardium is strikingly anisotropic and complex compared to other parts of the body where the anatomy is more homogeneous. Providing these clinicians with a means to manipulate the patient-specific imaging data allows them to gain a stronger spatial knowledge and greater confidence when performing the procedure.

3.1.2 Microsoft Kinect Camera

The Microsoft Kinect camera is a commercial product originally developed for the Xbox computer entertainment system. It was released to the public in late 2010 and was well received by the scientific and consumer entertainment communities [71]. The Kinect device is a small, 3D, depth sensing camera with a small array of microphones capable of active noise cancellation [72]. Figure 3.1 shows the camera with major features indicated.

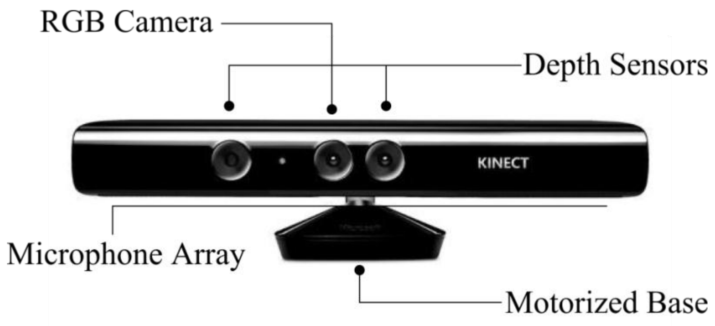


Figure 3.1 - Kinect Camera components
The Kinect camera system contains a standard web-cam type RGB camera. It captures depth information in the scene by emitting infrared light from an aperture, and detecting the returned infrared light through a second aperture. An array of microphones is positioned along the base to capture direction-sensitive audio information.

The Kinect camera is equipped with an RGB camera to capture video at frame rates of approximately 30 frames per second (Figure 3.2A). To capture depth information, the camera software continuously tracks the distance between the camera plane and objects within the FOV. The camera emits a beam of near-infrared light through the left depth sensing aperture (see Figure 3.1) which is then reflected off objects in the FOV and received by the other depth sensor [73,74]. Algorithms built into the device convert the signal intensity data into distance in units of millimeters by comparing the incoming data to a control image. The distance between the camera and the nearest object at the particular (x, y) coordinate of the FOV is the depth information returned to the computer [75]. This strategy of collecting depth information makes the camera vulnerable to occlusion errors whereby objects are lost in the depth dimension if they are behind an object nearer the camera. The depth data is a scalar map that contains no actual video or picture information about the captured scene. Many Kinect applications apply a look-up table (LUT) to the depth data to visualize the scalar data as an image. The Kinect performs image segmentation on the depth data to differentiate users from the background. The Kinect camera employs firmware to perform frame-by-frame skeletal joint tracking in real time. 20 skeletal joints (*e.g.* knees, hips, elbows, wrists) are tracked in the (x, y) plane and depth direction of each frame. These algorithms provide a single spatial position for each joint (*i.e.* the front and back sides of the hand are indistinguishable to the camera). Figure 3.2B shows upper skeletal joints mapped atop a standard video stream image. Figure 3.2C shows depth data that has been segmented to isolate the user. The background of Figure 3.2C shows pixels of varying grayscale values whose intensity is determined by the distance of the object represented by the pixel.

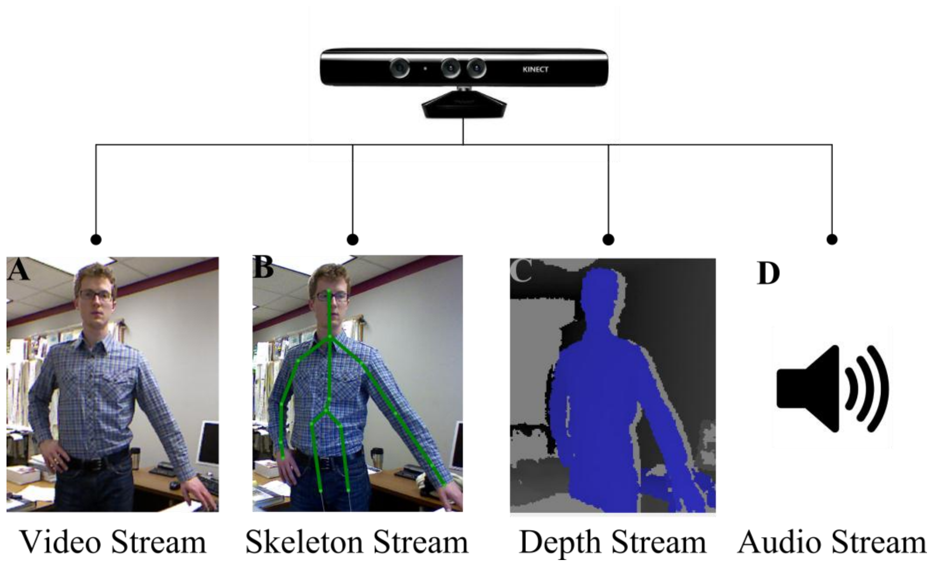


Figure 3.2 – Video, skeleton and depth images from Kinect camera
A standard RGB camera captures video (A). Algorithms track 20 skeletal joints and update the spatial locations of each with every frame (B). The Kinect camera calculates distance from the sensor in units of millimeters for each pixel in the FOV (C). An array of microphones captures noise-cancelled audio (D).

3.1.3 Objective

Considering the need for enhancing a user's ability to interact with data in a manner that is not intrusive on range of motion or cost, the Microsoft Kinect camera is well positioned to meet the demand for a low-cost, contact-free data visualization interaction device. The non-contact feature is especially attractive for use in a sterile environment in which cardiovascular surgeons frequently work. This aim will produce a control application that uses hand gestures recorded by the Microsoft Kinect camera and processed by open-source recognition algorithms. The application will control visualization content produced by a custom algorithmic framework designed to produce meaningful visualizations of medical imaging data tailored for use as a pre-surgical planning tool. The entire system shall be used with a stereoscopic display that allows users to gain the benefits afforded by stereoscopic rendering.

3.2 Methods

We have created a series of 4 methods to meet the objectives outlined in Section 3.1.3. The relationship between these methods is illustrated in Figure 3.3, and further detail is shown in Figure 3.4. The series begins with Interaction Data, where the Microsoft Kinect data is sent to open-source drivers (OpenNI Environment and NITE Algorithms) and processed to recognize gestures in the user's motions. Computer events are sent to the Software Solution, which is composed of three primary pieces: the Real-Time GUI, State Machine and a Configuration Dialog. These three pieces work in concert to receive the Interaction Data and translate it into commands for the Stereoscopic Player, which plays customized Stereoscopic Video content. The Stereoscopic Player displays 3D, stereoscopic video output at high resolution via the

Hardware Solution, consisting of specialized video cards, an NVIDIA 3D Driver system and a specialized computer monitor. A User Guide on using the visualization system is provided in Appendix C on page 137.

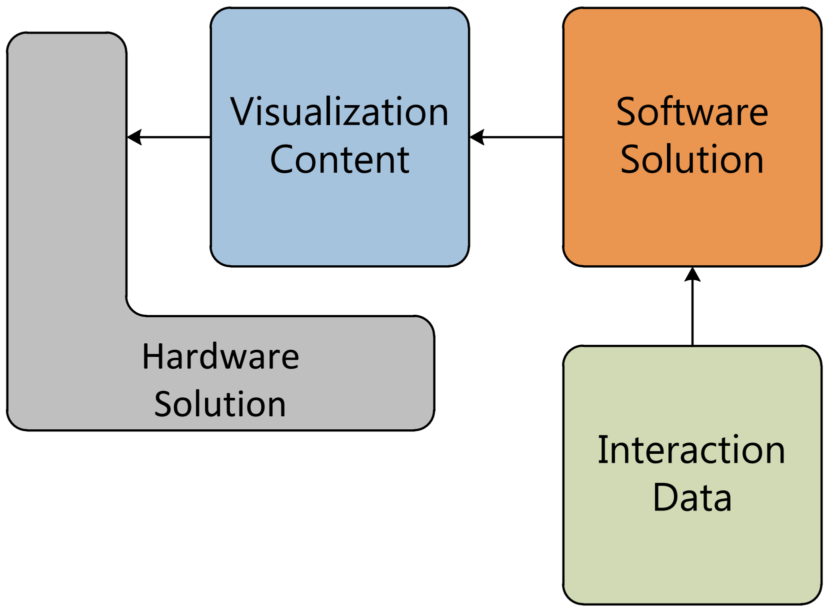


Figure 3.3 – Solution components
We have created a series of 4 methods to meet the stated objectives. The Interaction Data is supplied by the Kinect camera and sent to the Software Solution. Visualization Content created specifically for this aim is created and replayed with the Software Solution. The Hardware Solution is used to render the Visualization Content in stereoscopic 3D.

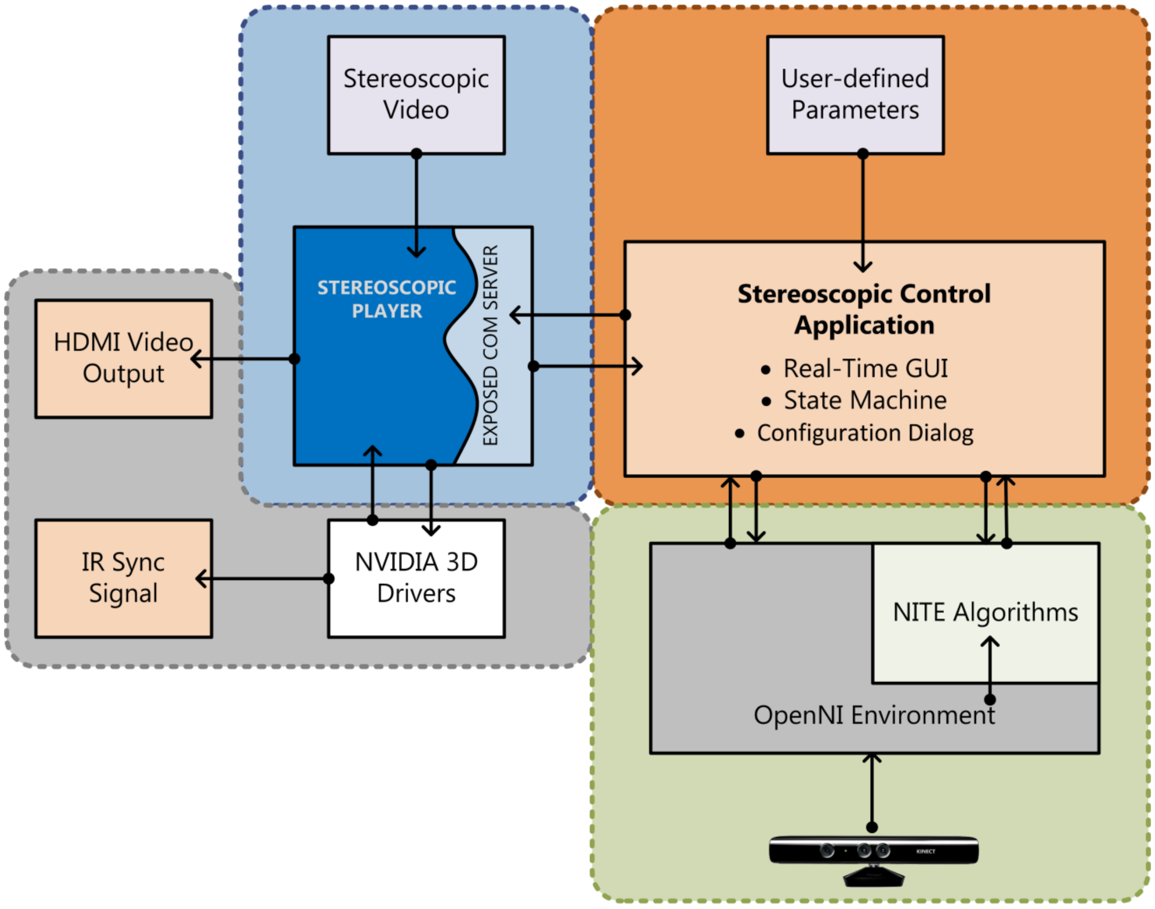


Figure 3.4 – Detailed view of system components
The stereoscopic control application expects three inputs to instantiate the OpenNI and NITE drivers and algorithms. The software creates a COM server object with the Stereoscopic Player application to establish two-way communication. When gesture-driven commands begin playback, the Stereoscopic Player outputs an HD video signal and synchronization signal in conjunction with the NVIDIA 3D Vision drivers.

3.3 Interaction Data

3.3.1 Gesture Libraries and User Interfaces

The 3D tracking of skeletal joints provides the Interaction Data to operate algorithms that detect gestures and postures performed by the user. It is possible to link preconfigured software events to these gestures and postures if the application supports an event handing paradigm. The software community has produced a small number of libraries that contain the necessary tools to detect gestures and postures directly from Kinect camera data. This eliminates the need for application developers to write gesture and posture recognition algorithms from scratch. The OpenNI software standard [76] establishes an extensible interface that integrates Kinect camera data with task-specific middleware. Kinect camera middleware is distinct from the graphics middleware discussed in Section 1.5.6. The OpenNI standard dictates how recognition middleware is written to be universally useful (Figure 3.5). The OpenNI library is a set of functions and drivers that allow computers to run recognition middleware that uses Kinect data. NITE Algorithms are an example of OpenNI middleware written in accordance with the OpenNI standard and designed to recognize hand gesture. The general term for a gesture-based control system is natural user interface, or NUI. Much like a GUI where the user interacts with graphics, NUIs allow users to interact with computers in a way that is natural. NUIs are not limited to gesture-based control; voice recognition is another common form of NUIs supported by the Kinect camera system.

3.3.2 Processing of Kinect Data

Interaction Data originates at the Kinect camera, which is then processed in the OpenNI environment. The OpenNI environment contains a set of drivers that process the Kinect data and broadcasts it to the NITE Algorithms, as seen in Figure 3.5. The NITE Algorithms calculate the position of the user’s hands and evaluate the displacement, velocity and relative position. When the displacement and velocity data meet the criteria established by the User-Defined Parameters dialog (Figure 3.4), a computer event is sent to the Software Solution which handles the event.

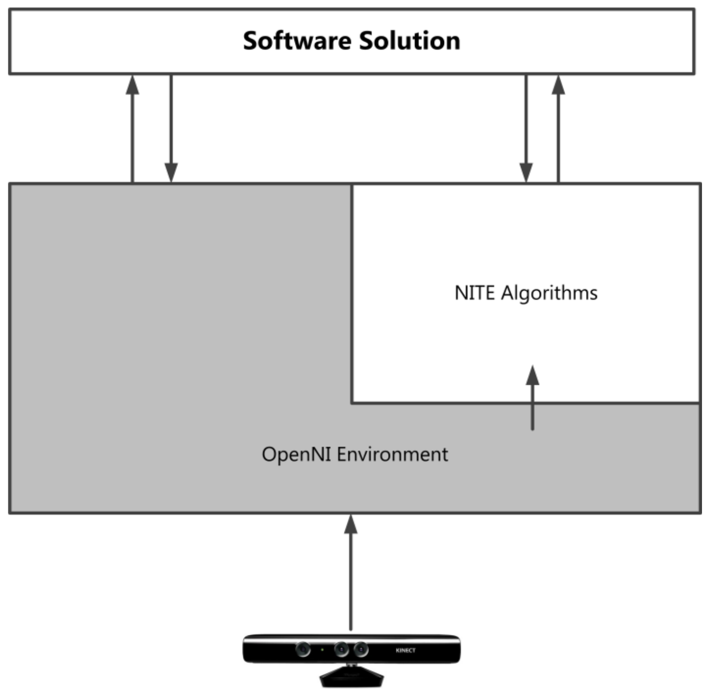


Figure 3.5 - OpenNI extensible interface
The OpenNI interface provides the drivers necessary to use the Kinect camera and makes the Interaction Data available to the NITE Algorithms.

3.4 Software Solution

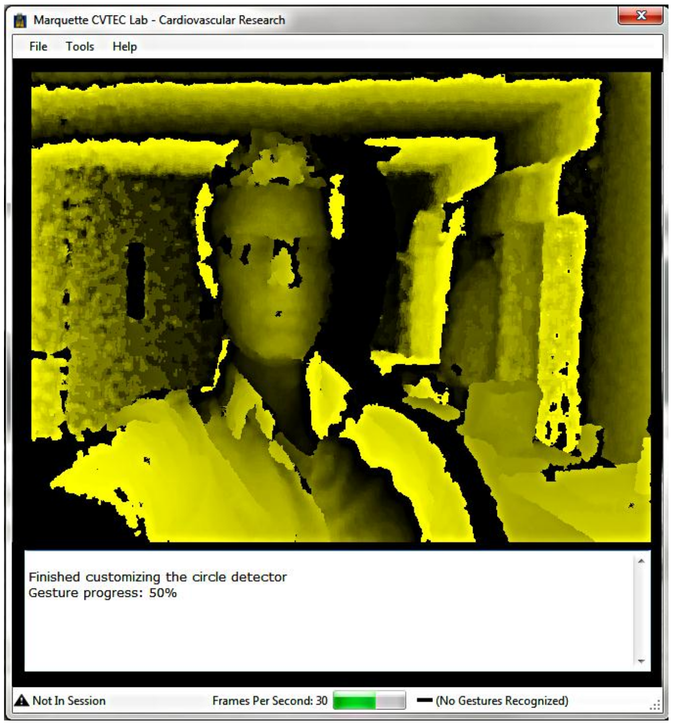
It is common for software developers to reuse small executable files in order to save time and effort in the development process. These executable files provide access to many low-level functions within the operating system and larger applications. Consider a custom application used in the course of academic research that includes a function to send information to a desktop printer. The developer may choose to write a large subroutine to temporarily store, convert and transmit the data to the printer, or he may instead access an existing executable file designed to handle all the requisite steps. These small executable files are collectively known as component object model (COM) objects. COM objects contain a collection of computer functions and the necessary resource files such as print preview function and the data needed to carry out the function. Software developers regularly extend the functionality of their applications using COM objects from other applications [77].

COM objects must first be created by software developers in order for other developers to use them. A COM object that delivers executable methods to other applications is called a COM server. COM servers can be created using any computer language, but the details are beyond the scope of this thesis. COM servers are accessed with an IDE by reading the specially compiled library files that are provided with the server. These libraries provide information about the expected inputs, outputs and syntax of the server functions.

A stereoscopic video player [78] with an exposed COM server is used to playback stereoscopic video files. The major features of the player can be controlled with the COM interface and Software Solution (see Figure 3.3). The Software Solution is written in Visual C# with Microsoft Visual Studio because it supports strongly typed functions associated with the stereoscopic player COM server. The Software Solution integrates the Interaction Data from the Kinect camera and the Stereoscopic Video Player in order to control its functionality through gestures detected by the NITE Algorithms. The system accepts input in the form of stereoscopic video files, User-Defined Parameters for the NITE Algorithms and Interaction Data. The system outputs a high-definition 3D stereoscopic video signal and a synchronization signal required for active shutter glasses. A detailed description of the software's architecture is provided in Appendix B.

3.4.1 Software Solution: Real-Time GUI

The user interface seen in Figure 3.6 was designed as a multi-threaded Windows Forms application. Once the Kinect camera is connected the Interaction Data is available and a second thread is created to run the gesture control algorithms with minimal impact on the performance of the rest of the application. When these algorithms detect a gesture, an event is sent to the primary thread, which contains event handling subroutines. These subroutines call the low-level functions that ultimately carry out the desired command on the Stereoscopic Video Player. Several GUI functions were written to allow the user to open files in a variety of 3D formats using the menu at the top of Figure 3.6. These functions provide the necessary stereoscopic video input required by Figure 3.4. The NITE Algorithms can be tuned to improve performance and accuracy in difficult environments using a GUI seen in Figure 3.8. This feature is discussed further in Section 3.4.3. The implementation details for the Real-Time GUI and the Software Solution are discussed in detail in Appendix B.



Finished customizing the circle detector
Gesture progress: 50%

Not In Session
Frames Per Second: 30
(No Gestures Recognized)

Figure 3.6 – Software Solution: Real-Time GUI
The Real-Time GUI is created as a Windows Forms Application with a viewing panel and toolbars. The depth image and textual feedback regions can be selectively hidden or shown depending on the level of information required by the user. Status icons in the window footer indicate the session state, frames per second and recently recognized gestures. This information aids in the training and familiarization process of using the software.

Table 3.1 – Gestures and associated commands

Flow Mode	*Gesture*	*Command*	*User-defined parameters*
Hand	Swipe Right	Play/pause	Velocity, duration, steady duration
Hand	Swipe Left	Full screen toggle	Velocity, duration, steady duration
Hand	Swipe Up	Zoom in / decrease FOV	Velocity, duration, steady duration
Hand	Swipe Down	Zoom out / increase FOV	Velocity, duration, steady duration
Hand	Push	Enter slider mode	None
Slider	Circle	Quit	Radius, sensitivity
Slider	Push	Enter selection	None
Slider	Slider Bar	Select playback position in normalized time	Bar height, width

3.4.2 Software Solution: State Machine

The swipe gestures used in a majority of the playback controls require the user to extend his hand beyond the normal bounds of the trunk. The natural follow-up movement after completing a swipe gesture will be to retract the hand back to its resting position.

Without correction, the NITE Algorithms will interpret this follow-up movement as a second swipe in the opposite direction. The Real-Time GUI employs a state machine to prevent this. The state machine is designed to internally redirect the flow of camera data to specific algorithms (Figure 3.7F). When the NITE Algorithms receive camera data they process it and trigger control events; when no data is received the algorithms enter an idle state and wait for the next packet of data to arrive. Figure 3.7 illustrates the state machine in which the Flow Router (Figure 3.7B) acts as the switch, directing the Interaction Data to the proper broadcaster (Figure 3.7C and D) depending on the current state.

The system operates in one of three states: Hand, Slider or Steady Mode. When in Hand Mode, Interaction Data is directed to the Primary Broadcaster (Figure 3.7C) by the Flow Router and on to the gesture recognition algorithms connected to it. If the system is in Slider Mode, all Interaction Data is sent to the Auxiliary Broadcaster (Figure 3.7D) and none to the Primary Broadcaster. After a gesture is recognized the system is put into Steady Mode. When in Steady Mode all Interaction Data is sent to the Steady Detector gesture recognition algorithm. The Steady Detector is a type of gesture recognition algorithm that identifies when the hand is maintaining a relatively stationary position. The degree to which the hand must be held steady can be adjusted. When the hand is determined to have been steady for the specified period of time (see Table 3.2) the data flow is returned to the Flow Router and the nodes are again able to detect gestures.

The Auxiliary Broadcaster (Figure 3.7D) was implemented to allow the same gesture to have many different commands depending on the state of the system. When in Slider Mode, the Slider gesture recognition algorithm calculates the horizontal position of the hand as a normalized distance from the origin. The normalized distance is then used to advance to the specified position in the video sequence. For example, if the user selects 0.75 using his hand and performs the push gesture, the Real-Time GUI will send a command to the stereoscopic player which begins playback at the time point 75% of the total time. The user can enter or leave Slider Mode with the Push gesture. When the system is in Hand Mode, the standard gestures listed in Table 3.1 are detected and the associated commands are sent to the Real-Time GUI.

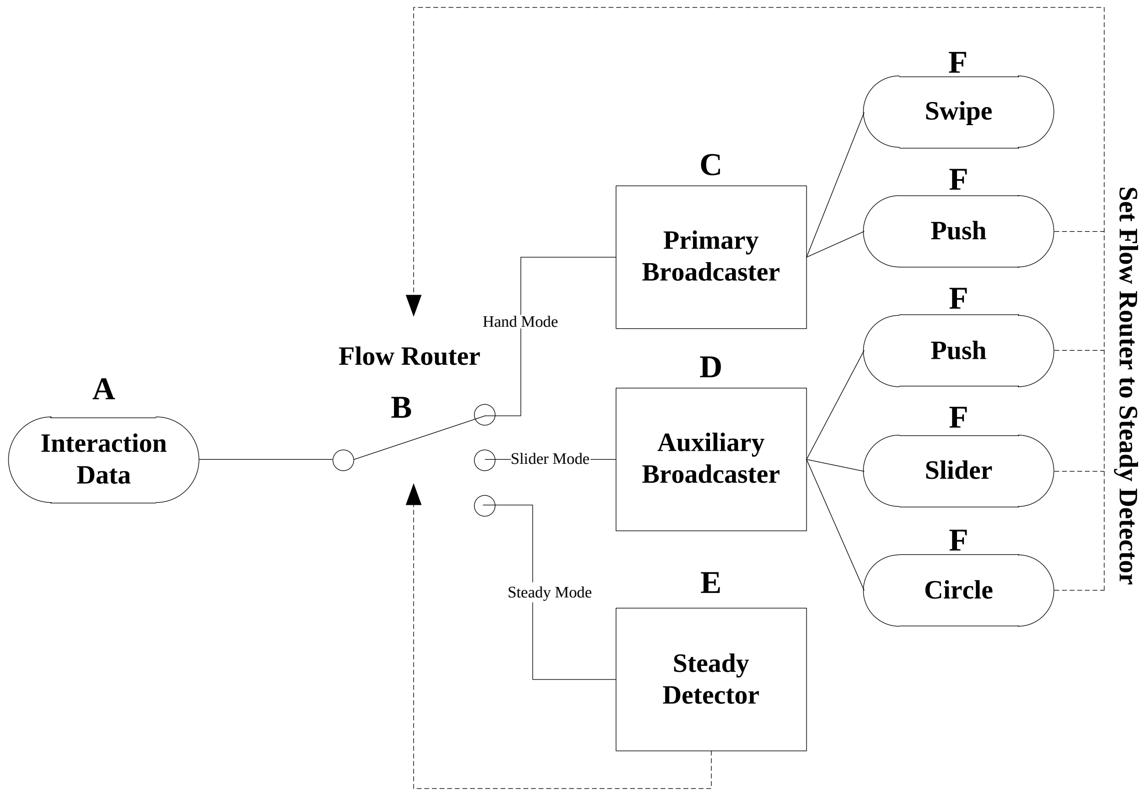
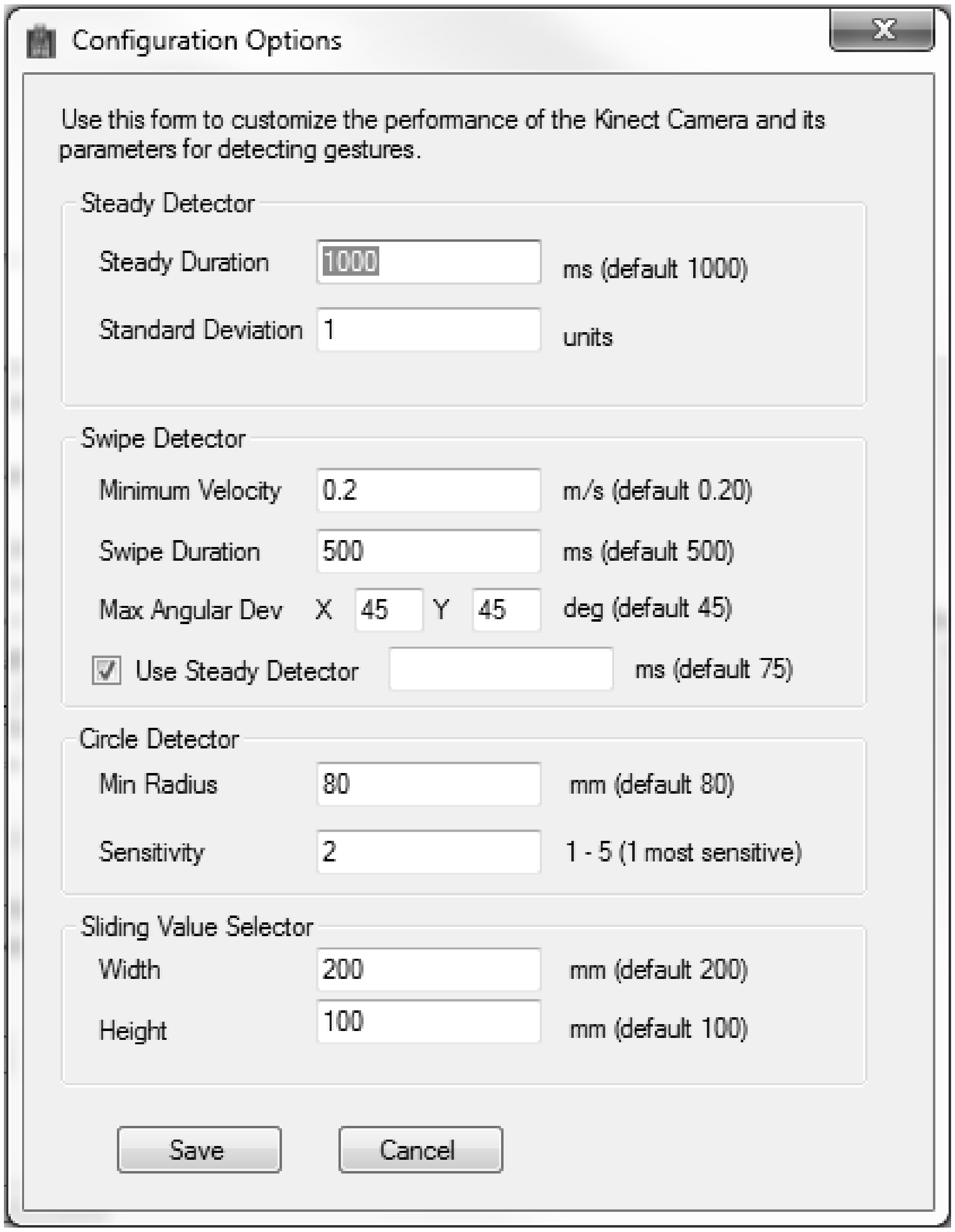


Figure 3.7 – State Machine
Hand position data arrives in each frame of Kinect camera data. When the system is in Hand mode, the position data is sent to the Primary Broadcaster (C), which then sends the unchanged data to the push and swipe detectors (E). When a gesture is detected in either of nodes, all data is rerouted to the Steady Detector (B), which keeps all data from the other detectors until the hand has been steady for the specified period of time. The user can alternate between Hand and Slider modes with the push gesture (F, G).

3.4.3 Software Solution: Configuration Dialog

The ability to tune the gesture recognition algorithms was added to maximize the number of environments in which the system is effective. The algorithms used to detect the gestures listed in Table 3.1 depend on the ability to detect motion of the hand. Due to the effect of camera/hand distance, hand movement near the camera will appear to create a larger displacement than the same motion at the far end of the camera’s FOV.

Considering that the effective range for the depth-sensing camera is 3.5 meters [75] this effect may become quite pronounced at far distances. By tuning the parameters of the algorithms the user can correct for this to allow more natural gestures to be used no matter the environment. Swipe velocity and duration are the two parameters most likely to be adjusted, as they are involved in correcting for the problem described previously. These parameters are described in greater detail in Table 3.2. Users may also adjust the sensitivity of the steady detector to best match their work habits. Suggested values for tuning the detection parameters can be found in Appendix B on page 135.



Configuration Options

Use this form to customize the performance of the Kinect Camera and its parameters for detecting gestures.

Steady Detector

Steady Duration	1000	ms (default 1000)
Standard Deviation	1	units

Swipe Detector

Minimum Velocity	0.2	m/s (default 0.20)
Swipe Duration	500	ms (default 500)
Max Angular Dev	X 45 Y 45	deg (default 45)
✓ Use Steady Detector		ms (default 75)

Circle Detector

Min Radius	80	mm (default 80)
Sensitivity	2	1 - 5 (1 most sensitive)

Sliding Value Selector

Width	200	mm (default 200)
Height	100	mm (default 100)

Figure 3.8 - Customization dialog
The NITE Algorithms allow the user to customize the detection parameters for a specific application. The Real-Time GUI exposes these parameters to the user and offers a mechanism by which they can be reconfigured at run-time.

Table 3.2 - User-defined parameters
The table enumerates the various categories of user-defined parameters, their description and their range of recommended values. These parameters can be used to adjust the performance of the NITE Algorithms. Unless specified, the application will use the default value.

	Parameter	*Description* ***[units]***	*Default Value*	*Min Value*	*Max Value*
Steady Detector	Steady Duration	Minimum period of time the hand must be relatively stationary before the session manager to set back to the flow router. **[ms]**	1000	200	10000
	Standard Deviation	The standard deviation of hand movements that is considered to be steady. Increasing this number makes the system less sensitive to unintentional hand movements. **[m/s]**	1	0.1	6.0
Swipe Detector	Minimum Velocity	The swipe velocity below which the NITE Algorithms will not throw gesture event. **[m/s]**	0.2	0.1	1000
	Swipe Duration	The minimum temporal duration of hand movement to trigger swipe event. **[ms]**	500	350	1000
	Max Angular Deviation (X, Y)	Maximum deviation	45, 45	0	90
	Use Steady Detector	Boolean variable that turns the swipe detector's internal steady detector on or off. If on, the swipe detector steady duration is added to that of the entire system. **[bool]**	True	False	True
Circle Detector	Min Radius	Radius of smallest diameter circle detected by NITE Algorithms in display's coordinate system. **[mm]**	80	40	1200
	Sensitivity	Maximum allowed deviations from perfect circular path of the hand. **[count]**	2	1	5
Value Selector	Width	The sliding value selector's width measured in units of the display's coordinate system. **[mm]**	200	0	Screen width
	Height	The sliding value selector's height measured in units of the display's coordinate system. **[mm]**	100	0	Screen height

3.5 Visualization Content

A method was developed to produce meaningful visualizations tailored for use by a cardiothoracic surgeon using open-source visualization tools. This framework is outlined in Figure 3.9. These algorithms automate most of the intermediate steps required to produce valuable visualization content. Minimal user interaction is required to run these algorithms. They are written as Python scripts to control ParaView [79], an open-source parallel visualization tool. The algorithms can be applied in any order so that a final visualization meets the needs of the user fully. Once rendered with a combination of these algorithms, the visualization content can be manipulated using the stereoscopic control application. It is not feasible to control all features of ParaView with hand gestures; for this reason, we sample the data to establish pre-recorded vantage points that are of the highest clinical value. These pre-recorded vantage points were selected in consultation with an expert in the field.

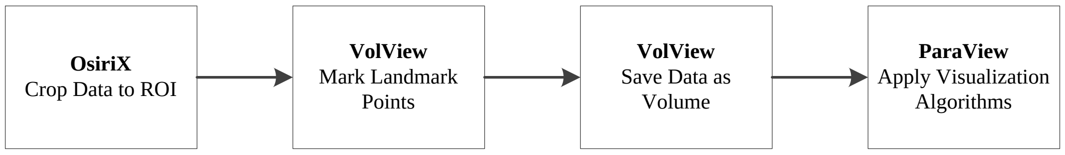


Figure 3.9 - Visualization Framework
The framework allows users to create meaningful visualization content using open-source software tools.

3.5.1 Data Selection

The goal of producing visualization content that is tailored for use in cardiovascular surgery informs the selection of datasets to test our methods. Eight datasets have been selected to represent a wide cross section of possible pathologies commonly encountered by the surgeon, which are summarized in Table 3.3.

Table 3.3 - Summary of datasets processed

Name	Modality	No. Images	Voxel Size [mm]	Description
AGECANONIX	CT	355	0.47 x 0.47 x 0.50	16 slice cardiac and coronary study
ARTIFIX	CT	64	0.39 x 0.39 x 2.5	Cardiac study of patient with a dilated aorta
CETAUTAMATIX	MR	88	0.93 x 0.93 x 1.40	Normal cardiac MRI/MRA study
FEROVIX	CT	53	0.43 x 0.43 x 0.5	Cardiac study of patient with pulmonary stent
FIVIX	PET/CT	513	0.35 x 0.40 x 0.40	64 slice CT angiogram with PET study
MAGIX	CT	76	0.40 x 0.40 x 2.0	64 slice CT angiogram
RATIB	CT	264	0.43 x 0.43 x 0.50	Normal
TOUTATIX	CT	309	0.33 x 0.33 x 0.50	CT cardiac scan of patient with main coronary branch emanating from pulmonary artery

3.5.2 Image Processing Methods

Each dataset must first be cropped to the region of interest, illustrated in Figure 3.10. DICOM files are loaded into OsiriX, open-source DICOM software package [80], and cropped using a 3D sculpting tool. In typical cardiac images, this involves manually removing the pulmonary vasculature, ribs and excess tissue information superior and inferior to the myocardium. However, not all structures will be visible with all modalities. For example, bone is often not visible with MR images. The system should be set to parallel projection to avoid perspective errors. The complexity and time required for completion of this step is directly proportional to the tortuosity and complexity of the anatomy. The edited DICOM files are exported as a new set of discrete DICOM slices, retaining the original metadata. The cropped slices are then aggregated and converted to a volumetric dataset using the open-source application VolView [81]. The volumetric data is saved in *.vti* file format, a uniform, rectilinear grid format that allows other software packages to process the data more efficiently.

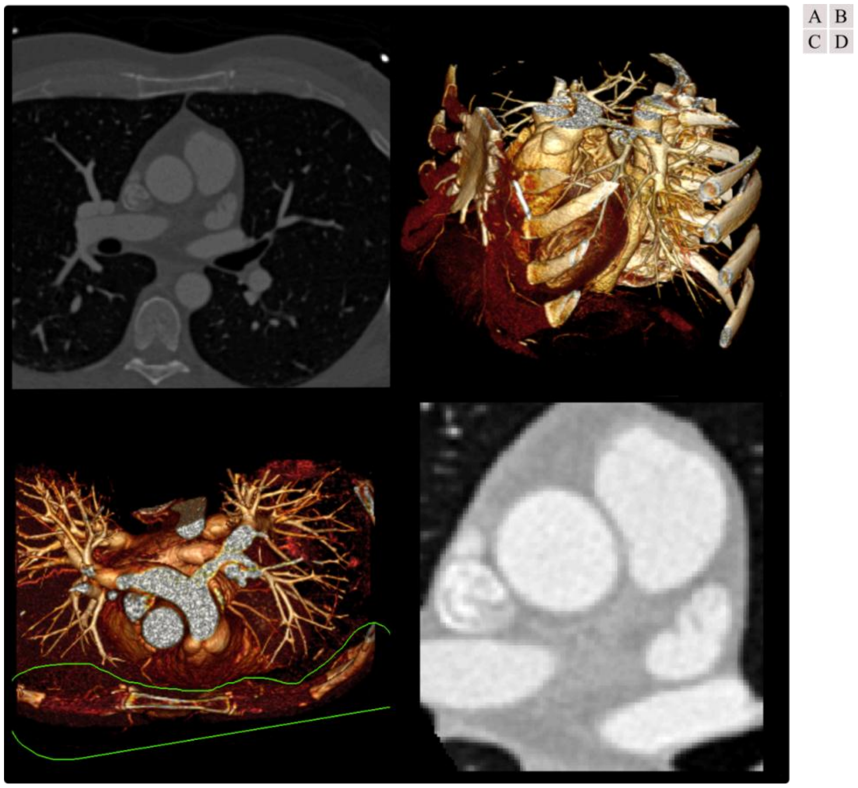


Figure 3.10 – Image processing steps
Panel A shows a standard thoracic CT image. B is the volume rendering of A with thresholding applied. C shows the volumetric data as cropping is being applied. The green outline shows how the user is able to selectively remove portions of the data using a standard computer mouse. The algorithms automatically apply the cropping data to all relevant slices resulting in D.

It is necessary to mark landmark points on the images in order to establish the viewpoints vectors in the 3D volumes after the images have been processed. This is accomplished with VolView, where points are marked on a 2D image, and the software determines depth location based on slice number. Two specific procedures for establishing 3D content are demonstrated. The first procedure establishes a vector along the line-of-sight (LOS) of the surgeon as if he were standing on the patient's right side. The LOS vector enters the myocardium on the anterolateral side of the right atrium and exits on the lateral wall of the left ventricle as seen in Figure 3.11. Resampling the viewpoint vector along this vector allows the surgeon to preview the heart's anatomy prior to a procedure. The second procedure orients the DICOM images along two vectors. The first runs through the interventricular septum to the center of the heart; the second bisects the atria through the interatrial septum (see Figure 3.12).

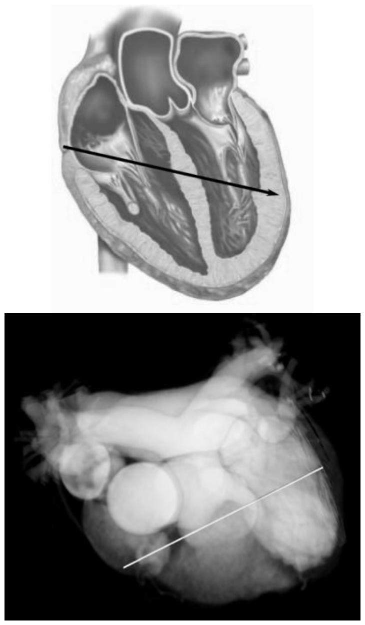


Figure 3.11 – Approximation of LOS vector
The white LOS vector extends from the anterolateral wall of the Right Atrium through the heart to the lateral wall of the left ventricle.

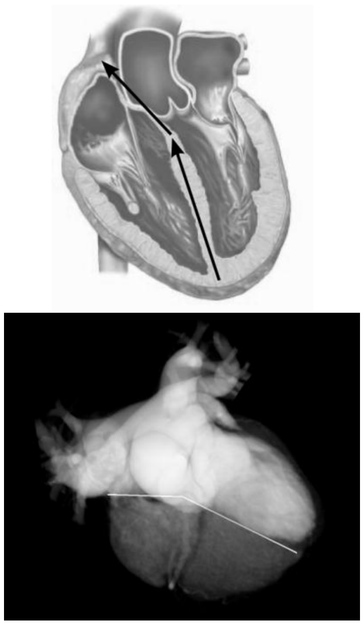


Figure 3.12 – Approximation of interventricular septal (IVS) vector
The two white vectors bisect the septa of the heart beginning at the apex, into the center and through the base of the heart.

3.5.3 Visual Content Generation

Renderings are created after the data has been cropped and the landmark points recorded at a resolution not less than 1920x1280 pixels. An algorithm was written to process the landmark points and calculate the length, direction and orientation of the vectors defined previously (see Figure 3.13). This data is used as input to the ParaView *Slice* filter that is used to resample the DICOM images. The slice filter locates a 2D sampling of the 3D data at a specific location and orientation. A grayscale color map was designed to view major structures within the heart while maintaining transparency. Since

ParaView renders the slices and the volumetric data in the same space it is important to maintain the ability to see through the volume to the slices. This transparency is seen in Figure 3.11 and Figure 3.12 where the white vector is visible. An opacity map developed for this application emphasizes the densest material and makes less dense material transparent or translucent.

Once the slices have been calculated and rendered in ParaView, visualizations are created with additional algorithms. Any combination of rotations, movements and transformations are possible, but results are optimized for use as a pre-surgical planning tool. These involve a minimal amount of rotation to allow the user to maintain spatial orientation relative to the heart. To provide detailed anatomic detail, the active slice is slowly translated along the vectors seen in Figure 3.11 and Figure 3.12. Several macro functions were written to automate the process of creating the visualization sequences. These functions were designed to be applied to an existing ParaView dataset; this allows maximum flexibility in how the visualizations are created with minimal alterations to the macro functions. By applying varying combinations of these macro functions, the user is able to create a visualization that is tailored to his specific need. Two stereoscopic files are produced. These files are input to the Stereoscopic Video Player described in Figure 3.4.

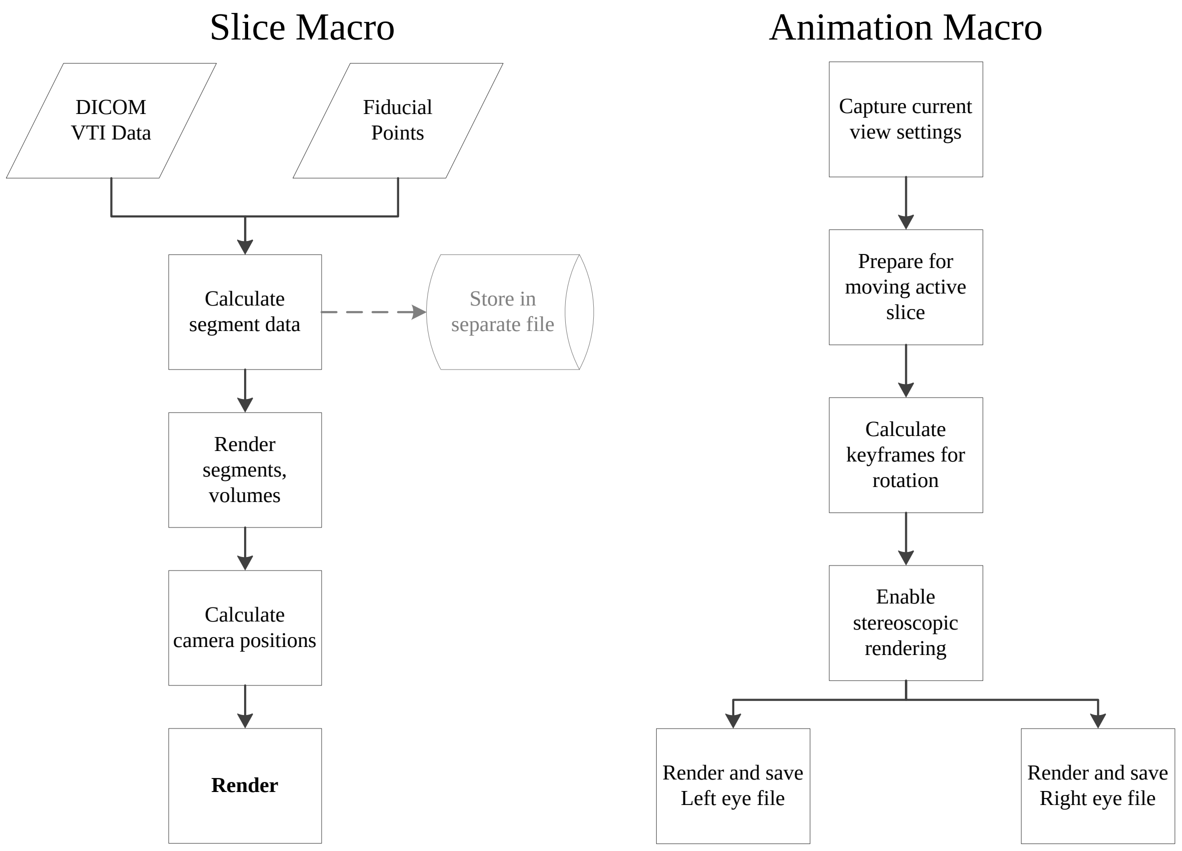

Slice Macro
DICOM VTI Data
Fiducial Points
Calculate segment data
Store in separate file
Render segments, volumes
Calculate camera positions
Render
Animation Macro
Capture current view settings
Prepare for moving active slice
Calculate keyframes for rotation
Enable stereoscopic rendering
Render and save Left eye file
Render and save Right eye file

Figure 3.13 – Flow charts describing ParaView Macros
The slice macro (left) prepares and renders a DICOM slice using two sets of data. The Animation Macro (right) first captures the current view settings allowing it to be applied to any existing ParaView rendering. Two stereoscopic (left and right eye) files are produced by this macro.

3.6 Hardware Solution

To deliver the system that integrates these software tools with a stereoscopic display device we use commercially available products. A specialized GPU is used to achieve time multiplexing compatible with a commercially available stereoscopic hardware package distributed by the NVIDIA Corporation. Figure 3.14 illustrates the primary connections between the various pieces of hardware used in the project.

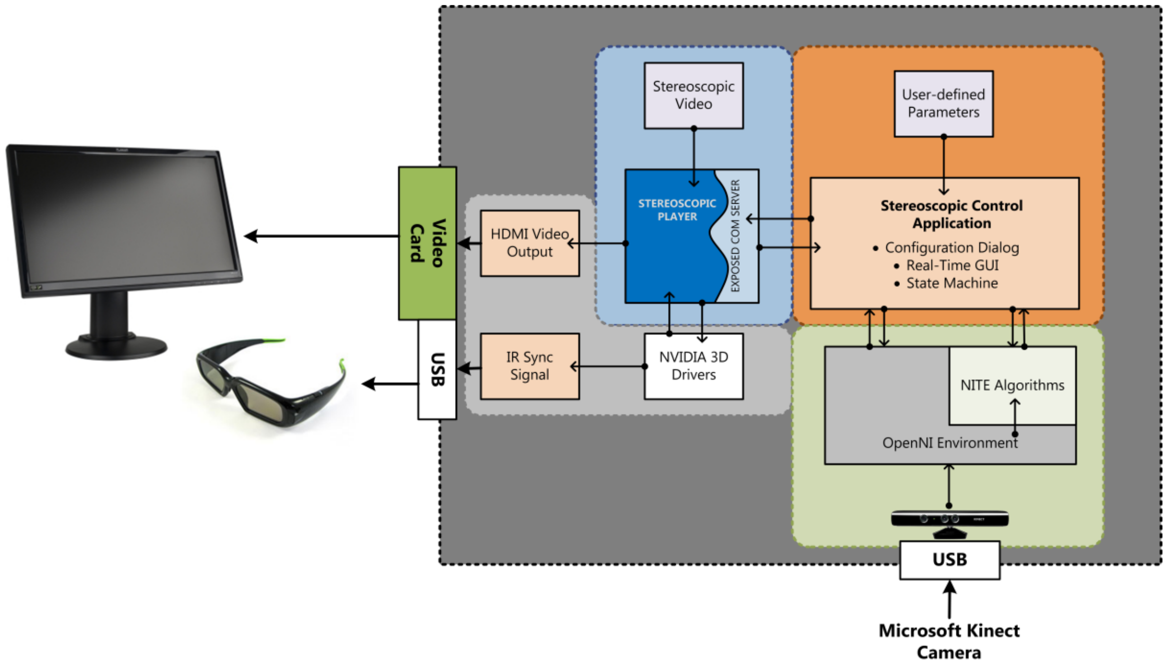


Figure 3.14 - Hardware Topology
The hardware used in this system involves assorted pieces of commercially available products. The stereoscopic effect is achieved by the NVIDIA graphics card, IR emitter and glasses maintaining a synchronized shutter to the high performance display. The stereoscopic control application integrates data from the Kinect camera to produce commands that control the video player.

3.6.1 Commercial Stereoscopic Solutions

We have elected to use a turn-key solution for creating time-multiplexed (active) stereo images to keep the costs manageable. The NVIDIA Corporation distributes a hardware/software product including graphics drivers and 3D shutter glasses (see) to allow users to quickly create stereoscopic renderings. Active shutter glasses and an infrared emitter maintain synchrony with the quad-buffered display (driven by a high-end graphics card, see Table 3.4). Because the images must be alternated at a rate of 120Hz it is not practical to render each video frame immediately prior to display; rather, the system maintains a separate memory buffer in which the next frame is rendered while the previous frame is being displayed.

Table 3.4 - Bill of Materials

Description	*Quantity*	*Est. Cost*
Planar SA2311W 3D Vision Monitor	1	$300
BenQ MS612ST Stereoscopic Projector	1	$499
Da-Lite 57.75"x77" Rear-projection holoscreen	1	$2,905
NVIDIA GeForce GT430 graphics card	1	$67.00
NVIDIA 3DVision2 system	1	$150.00
Microsoft XBOX Kinect Camera	1	$150.00

The NVIDIA system requires the display device to be certified for performance and reliability. A small number of commercially available televisions, computer monitors and LCD projectors have been certified for use with the NVIDIA 3D Vision system. The monitor and projector listed in Table 3.4 are certified by NVIDIA to function according to expectations.

3.6.2 Display Devices

Our visualization system can be realized with one of several display devices. The NVIDIA stereoscopic system can be used with any approved display device, and the stereoscopic control application is display independent. We present results using the Planar SA2311W monitor, which has a large screen size, high contrast ratio, low ghosting (between time sequential frames) and high brightness.

For larger displays a projector is a viable alternative to a desktop monitor. Projectors offer larger display areas than a single monitor and the screen can be retracted or otherwise hidden when not in use; this is an advantage over the fixed-size of desktop monitors and televisions in which the display's required storage space is constant. Time-multiplexed stereoscopic techniques are less sensitive to viewing angle than polarization multiplexing techniques [26]. Because of this, time-multiplexed stereoscopic images can be viewed from extreme viewing angles with limited effect on the perceived image [23,26].

3.7 Results

3.7.1 Hardware and Software Performance

The stereoscopic control application performed as expected. The Real-Time GUI (Figure 3.6) can be opened as a standalone executable file on any computer with the applicable camera and video drivers installed. Once the Real-Tim GUI loads successfully, the user opens files of interest with second easy to use GUI. The files can be located locally or on a remote computer. Opening remote files may increase latency and generally decrease performance. The gesture detection criteria for the NITE Algorithms can be adjusted at runtime as the User-Defined Parameters from Figure 3.4 to maximize performance under a variety of environmental conditions and user preferences. Figure 3.8 shows the dialog box used to adjust the User-Defined Parameters. When adequately adjusted for the current environment, the software successfully detects gestures and correctly executes the associated player command within a short period of time. Latency between gesture recognition and command execution is small and estimated to be less than 500 ms under normal conditions.

Inaccurate gesture detection can occur if the User-Defined Parameters are not well suited for the environment. Appendix B lists suggested settings for the User-Defined Parameters for two common user environments and provides a brief discussion on how to intuitively adjust them to best suit other environments. The NITE Algorithms interpret hand motion most accurately when the user ensures the angle created between his and the camera’s lines of sight is less than 35º. When this threshold is exceeded detection errors can occur. Appendix B contains a detailed discussion on the impact of orientation on NITE Algorithm performance.

The 3D stereoscopic effect is available when the player is used in Full Screen mode. This is the result of the stereoscopic player being a child process of the stereoscopic control application. This is because the computer's operating system treats the player differently when it is seen as a child of a parent process. Once in full screen mode, the player is able to adjust the parallax, vertical and horizontal offsets of the 3D content in order to maximize the benefits afforded by advanced visualization. The active shutter glasses must be used within 15 feet of the emitter, and with a direct line of sight, to achieve the 3D effect. The glasses must maintain a line of sight with the emitter since it operates on the IR wavelength. Satisfactory stereoscopic depth perception was achieved within the range of $\pm 60°$ of the screen's perpendicular axis. This provides 120° across which the 3D effect is both perceptible and effective in conveying depth. Beyond this range, visual disparities caused by scene geometry render the depth information unreliable [17].

3.7.2 Volumetric Medical Data Processing Results

The visualization methods described in Sections 3.5.2 and 3.5.3 were successfully applied to all data described in Table 3.3. The image processing and annotation results can be seen in Figure 3.15. These algorithms produced visual content commensurate with the acquisition resolution; those datasets acquired at higher spatial resolutions produced results of a higher fidelity compared to those acquired at lower resolutions. Higher resolution is desirable as it affords greater detailed to be realized in the final visualization sequences, thus providing more information to the user. The anatomic variability seen in Figure 3.15 supports our claim that the framework is flexible and can be applied to data of heterogeneous origins.

Rendering of the Visualization Content was conducted on an HP Tower computer with an Intel Core2 Duo 3.17 GHz Processor, 4GB DDR3 RAM equipped with 32-Bit Windows 7 SP1. An NVIDIA GeForce 430 1 GB GPU was added to the PCI-e slot for additional rendering capabilities. The most complex visualization sequence derived from the Agecanonix dataset required approximately 1:41 to render 500 frames for both Left and Right eye stereo images for a total of 1,000 images in 101 minutes, or about 10 frames per second. The simplest sequence applied to the Agecanonix dataset required 3 minutes to render the same number of frames.

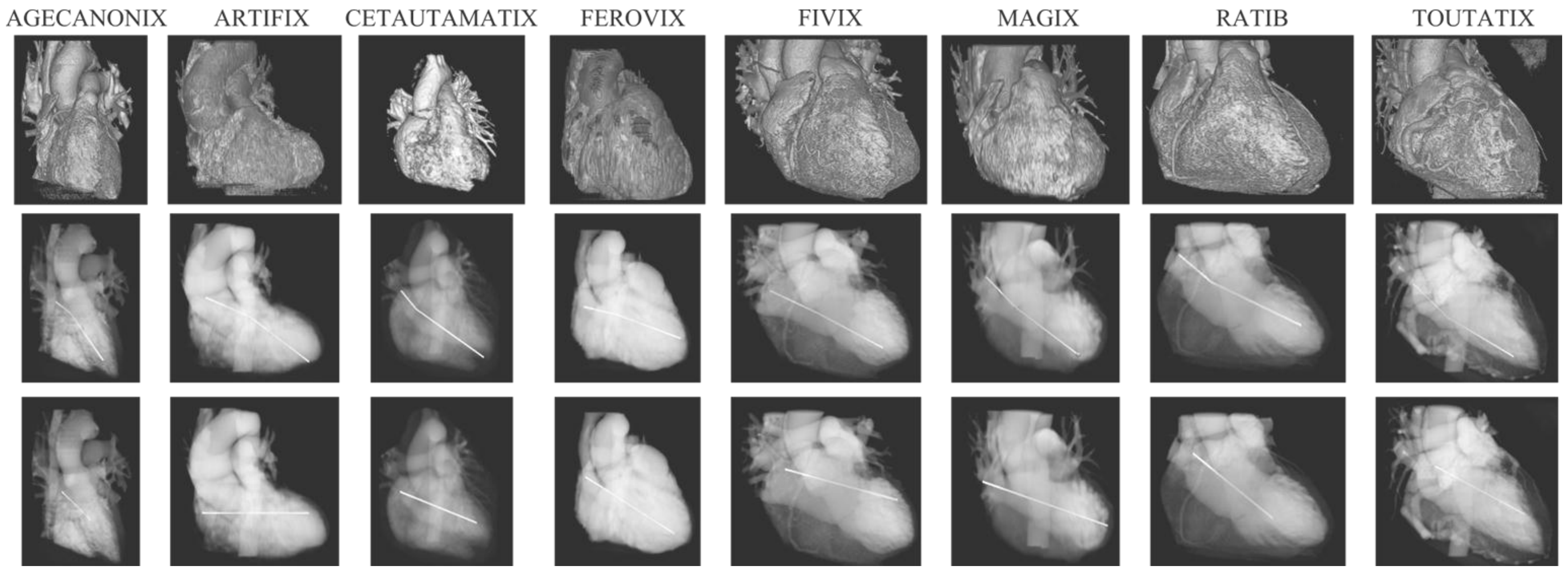


Figure 3.15 - Image processing results
The result of cropping, converting to volumetric data and annotating each dataset is shown above. Some datasets exhibit higher spatial resolution, namely the FIVIX and TOUTATIX sets, as compared to others. These high resolution volumes produce higher quality visualization results because more detailed data can be displayed to the user. The top row shows surface morphology of the myocardium. The second row demonstrates the interventricular and interatrial septal vectors. The third row shows the surgical LOS vector.

3.8 Discussion

The objective of the current work was to develop a visualization system that provides a method for viewing stereoscopic medical imaging data with a gesture-based control system. A further goal of the work was to develop an algorithmic framework to create high quality, useful visualizations with open-source visualization software packages. The application of advanced visualization techniques to medical imaging data is not itself unique; however, when coupled with a gesture-based system for controlling the visualization we extend the current techniques for reviewing medical imaging data. To date, we have used high-end, commercial display hardware to stereoscopically render high-resolution medical imaging data that is viewing angle independent to improve upon previous techniques. We have developed custom software that integrates data from a 3D, depth-sensing camera with open-source gesture recognition algorithms to control a sequence of visualizations using hand gestures. Finally, we have written a series of algorithms to produce high-quality visualizations from a multiple imaging modalities designed to produce content tailored for use a pre-surgical planning tool by a cardiothoracic surgeon.

3.8.1 System Performance

The stereoscopic control application performed within the defined parameters. The application is easy to use, as it is based on the Windows Forms Application template. Using this template provides the necessary graphics and interface elements to make the control application appear like many other commercial applications written for the Microsoft Windows operating system. The user interface is natural, and follows the conventions used by most other applications. This provides a familiarity to the user so that the time required to become familiar with the software is minimal. The elements of the user interface are named logically to facilitate ease of navigation for novice users. The software loads quickly because it is contained in a small amount of code and leverages shared libraries on the computer, reducing the amount of code that must be compiled with the stereoscopic control application.

The software link between the stereoscopic player, stereoscopic control application and the gesture recognition algorithms performs at a level that does not impede normal use. The latency between gesture detection and command execution is small and therefore does not pose a performance concern. Running the application on a multi-threaded computer is recommended to allow the various pieces of the software to execute on separate computational threads. Ideally, the gesture recognition algorithms are processed on a separate thread from the rest of the application.

3.8.2 Display Techniques and Hardware Considerations

The application of stereoscopic rendering to biomedical data has been previously reported [82]. Such systems use older, CRT monitor and polarizing filter technologies to display results and require the user to wear depolarizing glasses. Our system uses a time-multiplexing technique to achieve the stereoscopic effect, which is not affected by viewing angle. Polarization-multiplexing techniques are sensitive to the user's viewing angle relative to the screen, and off-axis viewing can completely obscure the 3D effect by introducing ghosting effects [26]. The wide viewing angle of our system (~120°) provides sufficient flexibility in viewing position relative to the screen, allowing multiple users to view the display simultaneously and observe the stereoscopic images with equal depth effects.

Our system employs a commercial, high resolution computer monitor which offers greater resolution than the CRT monitor described previously [82]. Because the stereo data is presented as time-multiplexed information (not row-interlaced as used by polarization multiplexing techniques), the left and right eyes each receive images at the monitor's full resolution. With the increased display resolution we are better able to display the detailed anatomic and functional data that is produced by modern medical imaging equipment. The high contrast ratio of the Planar monitor (1000:1) [83] used in the current system offers good differentiation for details in image data. The refresh rate of the monitor (120 Hz) is sufficient to avoid flicker effects and the brightness is not greatly diminished. Related to the refresh rate of the monitor is the perceived flicker of the stereo images through the active shutter glasses. The NVIDIA active shutter glasses provided good stereo performance with a minimal amount of perceived flicker. The flicker that was perceived was likely due to the interference from the overhead fluorescent lights and the other active computer monitors. The amount of perceived flicker decreased when all other light sources (including windows, overhead lamps and other computers) were eliminated.

3.8.3 Gesture-Based Controls

To extend the current techniques for integrating stereoscopic visualization of medical imaging data we have developed software to allow a user's hand motions to control the visualization content. Previous systems have used head tracking as a means of providing interactivity with the data [82], but out system allows the user to have a greater level of control. To the best of our knowledge, no other system uses a 3D, depth-sensing camera to accomplish this. Gesture recognition is a technology that is gaining momentum in the mainstream technology industries, including television and computerized entertainment markets [84,85]. Using a gesture recognition device eliminates the need for more costly equipment because gesture can be recognized with as little as one image capture device. A typical Kinect camera is available from an online retailer for $134 [86] compared to a basic VICON-brand motion capture system, which captures body motion using the same technology as the Kinect and starts at over $10,000 [87].

Controlling a computer system with gestures also eliminates the need for physical contact with an interface device. This feature is especially attractive for medical applications in which a sterile field may need to be maintained. A surgeon who wishes to review the medical imaging data—with or without the stereoscopic display—may have to touch controls wrapped in a shroud to maintain sterility. With gesture based controls, the surgeon simply makes hand gestures in view of the camera in order to control the medical image display. This better maintains the sterile field, while also eliminating the need for disposable covers for the manual controls.

Gestures can be described mathematically as the trajectory of a point in space; the recognition algorithms employed in this system can be trained to recognize custom gestures by performing the custom gesture in view of the camera. This affords great flexibility to the users of this system, who can map customized gestures to specific features of the stereoscopic player to expand its gesture-based functionality. Using custom gestures in addition to or instead of the standard gestures recognized by the algorithms provides a more intuitive user experience. These gestures can be used in place of or in addition to traditional user interface devices such as a mouse and keyboard.

3.8.4 General Applicability

Previous work done to combine medical science with advanced visualization has often focused on the design of a system to accomplish a specific task—such as training and education [42,88], virtual procedures [89,90] and even autopsy [91]. Our system is designed to be flexible in application to make it useful to as large a part of medical and scientific communities as possible. The graphics hardware is capable of displaying almost any form of medical or scientific data—such as CT, MR, OCT and simulation datasets from sources such as CFD. The visualization results produced by the open-source algorithms in Section 3.5 are intended to be one of many types of data that can be visualized with the display and control hardware. Because the foundational format of the data visualized in this system in based on the VTK file format standard, the list of possible data compatible with the current system is extensive.

3.8.5 Limitations

The application relies upon many of the video and display drivers common to the Microsoft Windows operating system. For this reason, the application will only function on Microsoft Windows. The software must be compiled on the host computer in order to locate and link to the shared libraries required for operation. Using the Microsoft Visual Studio IDE automates this process and greatly reduces the amount of time required to locate the required libraries and compile the application.

The current embodiment of this visualization system requires the user to wear active shutter 3D glasses. The glasses require electricity and must be charged with a USB power cord. When not performing the shutter action, the glasses typically diminish the brightness of the images observed through them, making it impractical to wear them unless viewing 3D visualizations. This challenge could be overcome with the use of autostereoscopic displays that do not require the user wear any glasses. Such displays will be discussed in further detail in Section 4.1.2.

The use of gestures affords many advantages over traditional user interface devices. However, further development of the detection algorithms may be needed before gesture-based controls can be used for safety-critical applications. The accuracy of the gesture detection algorithms is sufficient for the applications described here, but additional fidelity in detection is required before such hardware/software systems could be safely used for applications directly impacting a patient’s health and safety, such as robotic surgery.

3.9 Conclusion

The objectives of this aim were to produce a control application that uses hand gestures recorded by the Microsoft Kinect camera and processed by open-source recognition algorithms. We have delivered an application that recognizes hand gestures and uses them to control the 3D stereoscopic visualization content. This content was produced with a custom algorithmic framework designed to produce meaningful visualizations of medical imaging data tailored for use as a pre-surgical planning tool. The system is completed with a specialized display that allows users to gain the benefits afforded by stereoscopic rendering. These accomplishments meet the stated objectives, and deliver a system that is well-suited for use by a cardiovascular surgeon.

Chapter 4: Conclusions and Future Directions

4.1 Future Directions

The practice of integrating advanced visualization technologies with gesture recognition and 3D depth-sensing cameras has a bright future. This thesis describes the first steps in realizing such future possibilities. Future developments will likely focus on extending the functionality of gesture-based controls and their ability to directly manipulate medical and scientific data. New display technologies suggest that the use of specialized eyewear may soon be obsoleted in favor of new stereoscopic techniques that do not require glasses. New areas of applications for these technologies will also be discussed.

4.1.1 Application Development

The current system relies on pre-configured visualization sequences to deliver the data. Future versions of this system might employ the use of a Virtual Reality Peripheral Network (VRPN) in which joint position data from the Kinect camera is delivered to the visualization software (such as ParaView) through a network connection [92]. Using such a system would allow data to be manipulated in real-time. A software solution designed to manipulate data in real-time requires two components. First, it must provide the means to rotate and scale the object of interest within the field of view. In ParaView this is achieved by changing the position and focal point of the camera. Second, the software must allow the user to apply transformation filters to reveal new and interesting features of the data. Such filters provide the means to transform the data and gain new insights which is an important part of the visualization process.

The ParaView client can be used to visualize data with an easy to use GUI. The capabilities of ParaView's visualization features can be extended through the use of plugins. Plugins are small applications written in C++ that add specific features to the ParaView client. Plugins can only be executed by ParaView because their architecture is designed to be run from within the ParaView Client. For example, a plugin has been designed by a third party to read and visualize a specialized file type '*HD5Part*,' which describes particle simulation data. Since this file type is not normally supported by ParaView, a software developer wrote a C++ plugin to extend ParaView's capabilities to read and visualize it.

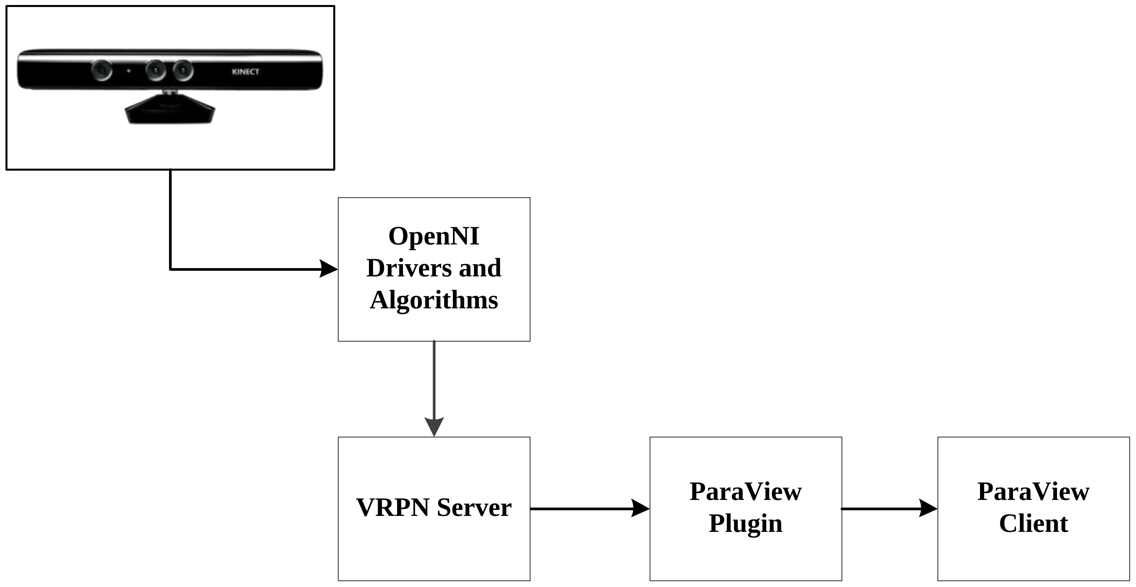


Figure 4.1 - Data pathway between the Kinect and ParaView
Data travels from the Kinect camera to the OpenNI Drivers which send hand position data to the VRPN Server. The VRPN server broadcasts the position data to a custom ParaView plugin which processes the data and creates commands for ParaView Client.

Camera Position and Orientation: ParaView makes it possible to manipulate the position and orientation of the camera and its focal point. The *(x,y,z)* coordinates describing these camera properties are easily accessible through the plugin interface. The Kinect camera can supply the plugin with position information, which the plugin can then use to change the camera's position and orientation. At normal performance levels, the Kinect system supplies Interaction Data at approximately 30 Hz. It may be necessary to down sample the camera data to a rate that is consistent with ParaView's ability to receive and carry out instructions on changing the position of the camera. This down sampling can be performed in either the VRPN server or in the plugin itself. It may be favorable to perform this step within the plugin, as it will have access to the state of the ParaView client, which would allow for adjustable down sampling depending of the current abilities of the client.

The plugin must have a means to map the *(x,y,z)* position data of the hand into viable position coordinates of the camera. For example, the hand's *x*-position could map to the camera's elevation, the *y*-position to the azimuth and the *z*-position the roll. Elevation, azimuth and roll are illustrated below. A 360^{o} sweep of Azimuth produces a full orbit around the camera's focal point, which is held steady throughout the sweep (Figure 4.2). It is ParaView's convention to place the camera's focal point at the geometric center of the object/volume of interest. This proposed mapping of hand data to camera position is one of many possible solutions and may not lead to optimal results.

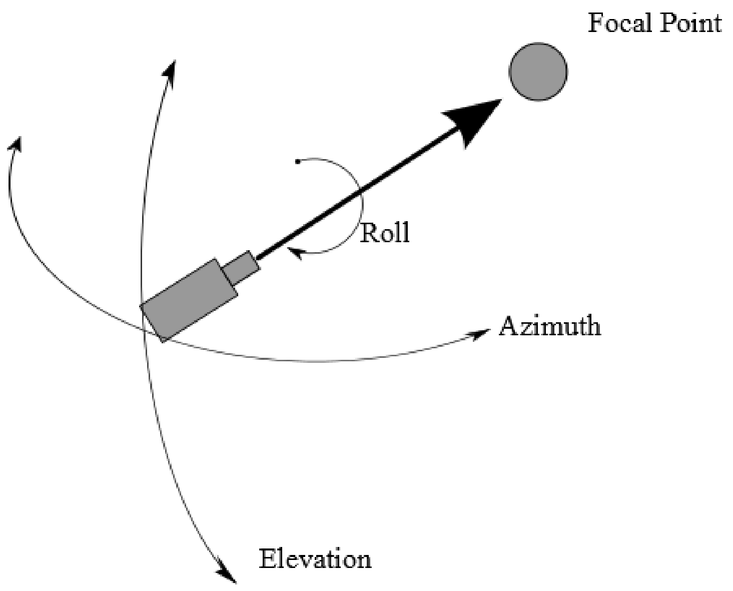


Figure 4.2 - Elevation, Azimuth and Roll camera operations
By linking specific hand position data coordinates to these operations the developer would be able to use the Kinect camera to control the camera position and orientation in real time. Graphic adapted from [12].

Potential Roadblocks and Solutions: The process of adjusting the camera's position and orientation requires ParaView to render the volumetric data for each incremental change in any position field. This describes a state in which ParaView will attempt to fully render the volume for each increment is elevation, azimuth, position or orientation. By attempting to do this, performance will be slowed and memory use will grow rapidly. In order to maintain real-time performance (defined as *latency <125 ms* for hand/eye coordinated events) [2], it is likely that using a Level of Detail (LOD) filter will be required. ParaView has a built-in LOD filter that down samples the object of interest while commands are received. When the object is sufficiently down sampled, the graphics can be rendered in synchrony with the camera repositioning commands, thus maintaining the real-time data interaction. When new commands cease to arrive for a specific period of time, a full volume rendering can be performed to produce graphics at a more realistic resolution. The settings for the LOD filter will likely need to be adaptive to the size and complexity of the volume being rendered, meaning that a separate function will be needed to set up and control the adaptive LOD filter.

Application of Filters: The application of ParaView filters extends the range of what can be visualized and appreciated using the ParaView client. For example, using the Slice filter allows users to view 2D image slices sampled from within a complex multi-dimensional data set such as CT medical imaging data. The information required by the ParaView Slice filter can be supplied by the Cartesian coordinates of the hand, which is tracked by the OpenNI Algorithms and broadcast by the VRPN Server (see Figure 4.1 and Figure 4.3). These coordinates are then received by the custom plugin and mapped to a specific property of a ParaView filter.

It is possible to design a custom plugin such that the *(x,y,z)* position of the hand is mapped to *(x,y,z)* specific filter properties. It would also be possible to use gestures to select the current property whose value is set by the hand position. A state machine similar to (but more complex than) the state machine developed in Aim 2 could be used to control the flow of data between setting the filter property values and selecting the active filter property. An example describing the use of the Kinect camera to set the Slice filter is described below.

The Slice filter requires these properties be supplied with data before the filter is applied:

- Input (of type *vtkDataSet*)
- Slice Type (choose from Plane, Box or Sphere)
- Input Bounds (x,y,z coordinates describing slice’s location in space)

- Slice Offset Values

As seen in Figure 4.1 and Figure 4.3 Interaction Data is obtained by the Kinect camera and processed by the OpenNI Drivers and Algorithms. These isolate the position of the dominant hand and also scan the data for any particular gestures that are performed by the user. A continuous stream of hand position data is sent along a VRPN server to the custom ParaView plugin which receives the data and routes its numeric value to the property of its active filter. When the user is satisfied with the current value assigned to the filter property, he performs a gesture, which is recognized by the OpenNI Algorithms and an event is triggered in the ParaView plugin. The plugin sets the “Next Property” (see Figure 4.3) to the “Active Property” and the hand position data is once again used to set the value of the property. Specific gestures may allow the user to navigate between the many properties of a filter and adjust them as he deems necessary. When all filter properties have been assigned to the satisfaction of the user a set of instructions is sent to the ParaView client. The client receives these instructions and applies the filter to the data.

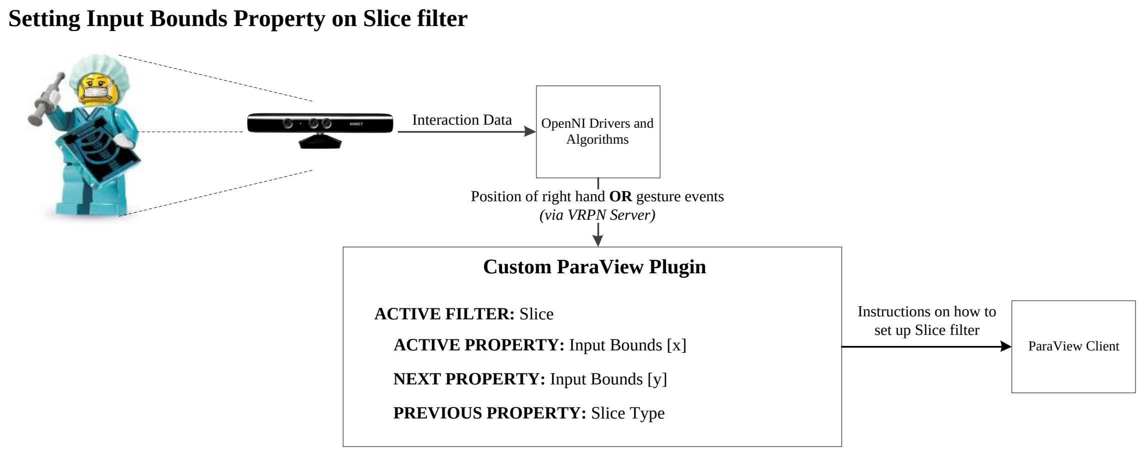

Setting Input Bounds Property on Slice filter
Interaction Data
OpenNI Drivers and Algorithms
Position of right hand OR gesture events (via VRPN Server)
Custom ParaView Plugin
ACTIVE FILTER: Slice
ACTIVE PROPERTY: Input Bounds [x]
NEXT PROPERTY: Input Bounds [y]
PREVIOUS PROPERTY: Slice Type
Instructions on how to set up Slice filter
ParaView Client

Figure 4.3 - Potential plugin configuration
Representation of how user inputs from a medical professional are obtained by the Kinect camera, processed by the OpenNI Drivers and Algorithms then broadcast to a custom ParaView Plugin via the VRPN server. The internal design of the plugin allows the user to cycle between the properties of the Active Filter and set numeric or categorical inputs to the filter. Gestures, recognized by the OpenNI Algorithms, signal to the plugin when to cycle between the previous and next filter properties. When all properties are set, a complete set of implementation instructions is sent to the ParaView client which then caries them out.

Potential Roadblocks and Solutions: There is an exhaustive list of potential filters built into ParaView and VTK that may be applied to the data. In order to provide users with access to all filters, it will likely become necessary for the developer to establish a standardized method of accessing, displaying and populating the properties of the filters. The system designer must also develop some means of determining the expected input type to each property. For example, the **Slice Type** property of the Slice filter must be set by one of three possible inputs; numerical input is not accepted. It would be necessary for the developer to recognize this special input type and provide some means of allowing the user to select one of the possible input values.

Developing a preliminary plugin that creates adapted versions of a select number of standard ParaView filters (and therefore can support the special cases more directly) may be an alternative to developing an all-encompassing plugin. By limiting the number of filters supported by the plugin, development will be less complicated by developing a gesture-based user interface, and more on the technical details of integrating the gesture and position data into the filter properties.

4.1.2 Display Technology

We have shown results in IVEs, projector-based visualization systems and desktop computer monitors, each of which rely on a multiplexing technique to achieve the stereoscopic representation of the depth dimension. Current state-of-the-art technology points towards a progression in displays to adopting autostereoscopic technology. With this technology users are no longer required to wear any kind of special eyewear. The technology used to achieve generally falls into one of two categories. The first, known as parallax barrier, uses LCD pixel elements placed between the viewer and the picture source. By selectively occluding the picture source LCDs, each of the viewer's eyes sees a slightly offset image, thus creating an image with depth information [23]. The second technology, lenticular lens, places a specialized lens atop the picture source. The lens selectively refracts light from the pixels into slightly offset trajectories, directing pixel data meant for the left eye towards the viewer's left eye. There are several variations of the lenticular lens concept, but most function under this basic premise [23].

Using an autostereoscopic display in place of a time-multiplexing display as shown in Section 3.6 eliminates the need for the user to wear specialized eyewear. In many cases this provides an advantage, as the user can appreciate the stereoscopic effect without having to don or remove the specialized glasses. In an operating room setting this may be especially advantageous, whereby a surgeon can review the stereoscopic visualization content on-demand by doing nothing more than looking at the monitor.

Coupled with the gesture-based control system described previoulsy, this suggests a promising technological tool for use in the surgical setting.

4.1.3 Applications in Medical Education

The concept of Virtual Reality can be extended to the field of medical education by creating virtual environments that mimic many aspects of a real clinical setting. As seen in Figure 4.4, a virtual hospital room can be created to a realistic scale for the purpose of training nursing students in a realistic, yet simulated, environment. In addition to the realistic graphics, content of the virtual environment can be generated to emulate real-world scenarios. In the scenario pictured in Figure 4.4 the virtual heart rate monitor on the wall displays real, time-varying heart rate data and audio feedback in the form of audible alarms for abnormal heart rates and rhythms. By training in such an environment, students can gain experience and confidence in handling complex situations with many forms of input, distraction and decision making factors.

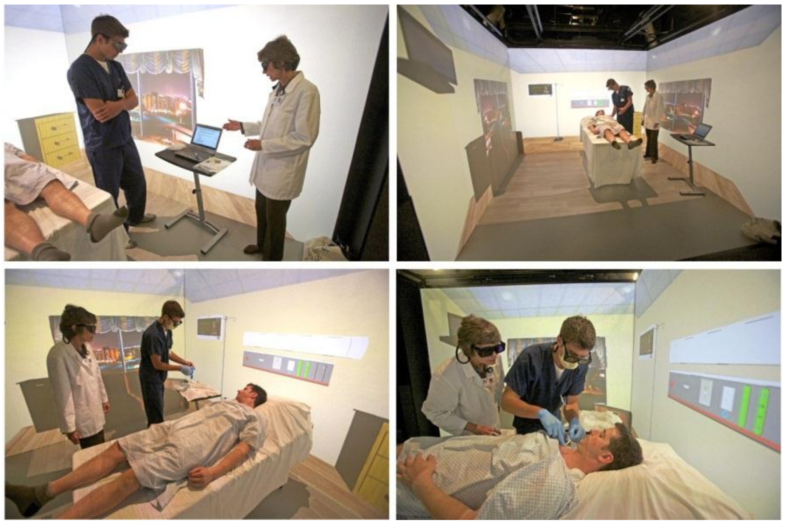


Figure 4.4 – Virtual environments used for educational purposes
A student and instructor wear active stereo glasses and perceive the objects projected onto the walls in stereoscopic 3D. The audio data embedded in the simulation provides students with clinical cues which they must incorporate into their clinical decision making.

4.2 Conclusions

The field of visualization has developed as a result of the 1987 NSF Report titled "Visualization in Scientific Computing" by DeFanti *et al.* [11]. In this report, the authors outline a broad definition of visualization and its ties to computer. More than 20 years later, the field of visualization is still experiencing rapid growth, especially due to the increased interest in visualization, the data deluge and advancements in high performance computing [8,10,93]. The scientific and non-scientific communities alike produce massive amounts of data that makes it difficult to process and analyze in full. Visualization is well-positioned to serve as a tool to alleviate this bottleneck.

Aim 1 described how advanced visualization allows users the ability to visualize biomedical and scientific data in stereoscopic, immersive environments. We showed that the use of an IVE presents many opportunities for enhanced understanding through direct interaction with data and allows a greater level of spatiotemporal knowledge of simulation results. We described the development of a method to rapidly produce immersive VR content from scientific and biomedical data. We demonstrated the flexibility of our method by applying it to the LCX coronary and carotid arteries, which were imaged with OCT and MR respectively. The ability for IVEs to display large quantities of data in a manner that is understandable is a viable solution to the problem of the data deluge. The retention of the depth-dimension allows a greater amount of spatiotemporal data to be displayed and used to establish new relationships between vascular form and function.

In Aim 2 we described an advanced visualization system that uses hand gestures to control the stereoscopic visualization of medical imaging data. The use of hand-gestures provides many advantages, including the ability to maintain a sterile field within an operating environment and a more intuitive HCI than can be afforded by keyboards or mice. The hand gestures were used to control the visualization of 3D, stereoscopic renderings of medical imaging data. This data was processed with an algorithmic framework designed to create high quality, useful visualizations with open-source visualization software packages. The system was designed to serve as a pre-surgical planning tool for cardiovascular surgeons. The use of open-source software packages reduces the costs associated with the software and may allow additional users to adopt the use of advanced visualization.

As the data deluge continues to grow with the increases in computational capacity predicted by Moore’s Law, it will become increasingly necessary to use tools like visualization to understand large sets of data. By using tools like those developed here, we have taken the first steps towards integrating advanced scientific visualization with biomedical research and established how it can aid in the understanding of large datasets.

BIBLIOGRAPHY

[1] Subcommittee on Research and Science Education, The State of Research Infrastructure at U.S. Universities, (2010).

[2] A. van Dam, A.S. Forsberg, D.H. Laidlaw, J.J. LaViola, R.M. Simpson, Immersive VR for scientific visualization: a progress report, IEEE Computer Graphics and Applications. 20 (2000) 26-52.

[3] C.A. Stewart, R.Z. Roskies, S. Subramaniam, Opportunities for Biomedical Research and the NIH through High Performance Computing and Data Management, 2003.

[4] R.G. Baraniuk, More is less: signal processing and the data deluge, Science. 331 (2011) 717-9.

[5] J.F. Gantz, D. Reinsel, The Digital Universe Decade - Are You Ready?, IDC Corporation. (2010).

[6] J.F. Gantz, D. Reinsel, Extracting Value from Chaos, IDC Corporation. (2011).

[7] R.R. Schaller, Moore’s law: past, present and future, IEEE Spectrum. 34 (1997) 52-59.

[8] P. Fox, J. Hendler, Changing the equation on scientific data visualization, Science. 331 (2011) 705-8.

[9] I. Foster, Rethinking Cyberinfrastructure for Massive Data, (2012).

[10] N.D. Gershon, C.G. Miller, Environment-dealing with the data deluge, IEEE Spectrum. 30 (1993) 28-32, 42.

[11] T.A. DeFanti, M.D. Brown, Visualization in Scientific Computing, Computer Graphics. 21 (1987).

[12] W. Schroeder, K. Martin, B. Lorensen, L.S. Avila, R. Avila, C.C. Law, The Visualization Toolkit, 4th ed., Kitware, Inc., Clifton Park, NY, 2006.

[13] D. Chung, R. Hirata, T.N. Mundhenk, J. Ng, R.J. Peters, E. Pichon, et al., A New Robotics Platform for Neuromorphic Vision: Beobots, Lecture Notes in Computer Science. 2525 (2002) 325-340.

[14] L. Itti, C. Koch, Computational modelling of visual attention., Nature Reviews Neuroscience. 2 (2001) 194-203.

[15] J.D. Pfautz, Depth perception in computer graphics, University of Cambridge, 2002.

[16] I.K. Fodor, A survey of dimension reduction techniques, Livermore, CA, 2002.

[17] W.B. Thompson, R.W. Fleming, S.H. Creem-Regeher, J.K. Stefanucci, Visual Perception From a Computer Graphics Perspective, 1st ed., CRC Press, Boca Raton, FL, 2011.

[18] E.B. Goldstein, Spatial layout, orientation relative to the observer, and perceived projection in pictures viewed at an angle., Journal of Experimental Psychology: Human Perception and Performance. 13 (1987) 256-266.

[19] P.R. DeLucia, Pictorial and motion-based information for depth perception., Journal of Experimental Psychology: Human Perception and Performance. 17 (1991) 738-748.

[20] N.A. Dodgson, Variation and extrema of human interpupillary distance, in: Proceedings of SPIE, SPIE, 2004: pp. 36-46.

[21] N.S. Holliman, N.A. Dodgson, G.E. Favalora, L. Pockett, Three-Dimensional Displays: A Review and Applications Analysis, IEEE Transactions on Broadcasting. 57 (2011) 362-371.

[22] C. Wheatstone, Contributions to the Physiology of Vision.--Part the First. On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision, Philosophical Transactions of the Royal Society of London. 128 (1838) 371-394.

[23] H. Urey, K.V. Chellappan, E. Erden, P. Surman, State of the Art in Stereoscopic and Autostereoscopic Displays, Proceedings of the IEEE. 99 (2011) 540-555.

[24] W. Rollmann, Zwei neue stereoskopische Methoden, Annalen Der Physik Und Chemie. 166 (1853) 186-187.

[25] H. Jorke, A. Simon, M. Fritz, Advanced Stereo Projection Using Interference Filters, in: 2008 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, IEEE, 2008: pp. 177-180.

[26] I. Sexton, P. Surman, Stereoscopic and autostereoscopic display systems, IEEE Signal Processing Magazine. 16 (1999) 85-99.

[27] Y. Galifret, Visual persistence and cinema?, Comptes Rendus Biologies. 329 (n.d.) 369-85.

[28] C. Landis, Determinants of the critical flicker-fusion threshold., Physiological Reviews. 34 (1954) 259-86.

[29] D. Kelly, H. Wilson, Human flicker sensitivity: two stages of retinal diffusion, Science. 202 (1978) 896-899.

[30] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, R.V. Kenyon, J.C. Hart, The CAVE: Audio Visual Experience Automatic Virtual Environment, Commun. ACM. 35 (1992) 64-72.

[31] T.A. DeFanti, G. Dawe, D.J. Sandin, J.P. Schulze, P. Otto, J. Girado, et al., The StarCAVE, a third-generation CAVE and virtual reality OptIPortal, Future Generation Computer Systems. 25 (2009) 169-178.

[32] Nvidia Corporation, NVIDIA 3D Vision 2, (n.d.).

[33] Request for Quotation (RFQ), Amira Visualization Software - Quote for Marquette University, (n.d.).

[34] S. LaScalza, J. Arico, R. Hughes, Effect of metal and sampling rate on accuracy of Flock of Birds electromagnetic tracking system, Journal of Biomechanics. 36 (2003) 141-144.

[35] J.E. Moore, C. Xu, S. Glagov, C.K. Zarins, D.N. Ku, Fluid wall shear stress measurements in a model of the human abdominal aorta: oscillatory behavior and relationship to atherosclerosis., Atherosclerosis. 110 (1994) 225-40.

[36] D.N. Ku, D.P. Giddens, C.K. Zarins, S. Glagov, Pulsatile flow and atherosclerosis in the human carotid bifurcation. Positive correlation between plaque location and low oscillating shear stress., Arteriosclerosis. 5 (1985) 293-302.

[37] G.K. Hansson, Inflammation, atherosclerosis, and coronary artery disease., The New England Journal of Medicine. 352 (2005) 1685-95.

[38] A.M. Malek, S.L. Alper, S. Izumo, Hemodynamic shear stress and its role in atherosclerosis, The Journal of the American Medical Association. 282 (1999) 2035.

[39] D.A.H. Steinman, D.A. Steinman, The Art and Science of Visualizing Simulated Blood-Flow Dynamics, Leonardo. 24 (2007) 71-76.

[40] D. Drascic, P. Milgram, J. Grodski, Learning effects in telemanipulation with monoscopic versus stereoscopic remote viewing, IEEE International Conference on Systems, Man and Cybernetics. (1989) 1244-1249.

[41] A. Chaudhry, C. Sutton, J. Wood, R. Stone, R. McCloy, Learning rate for laparoscopic surgical skills on MIST VR, a virtual reality simulator: quality of human-computer interface., Annals of the Royal College of Surgeons of England. 81 (1999) 281-6.

[42] J. Jordan, A. Gallagher, J. McGuigan, N. McClure, Virtual reality training leads to faster adaptation to the novel psychomotor restrictions encountered by laparoscopic surgeons., Surgical Endoscopy. 15 (2001) 1080-4.

[43] A.S. Forsberg, D.H. Laidlaw, R.M. Kirby, G.E. Karniadakis, J.L. Elion, M. Hospital, Immersive Virtual Reality for Visualizing Flow Through an Artery, in: Proceedings of IEEE Visualization Conference, 2000.

[44] J.F. LaDisa, M. Bowers, L. Harmann, R. Prost, A.V. Doppalapudi, T. Mohyuddin, et al., Time-efficient patient-specific quantification of regional carotid artery fluid dynamics and spatial correlation with plaque burden, Medical Physics. 37 (2010) 784-792.

[45] J. Justice, M. Bergerud, J. Garrison, D. Cafiero, L. Churches, EON Studio, (2011).

[46] M. Aumüller, R. Lang, D. Rainer, J.P. Schulze, A. Werner, P. Wolf, et al., COVISE, (2008).

[47] B.C. Nelson, D.J. Ketelhut, Scientific Inquiry in Educational Multi-user Virtual Environments, Educational Psychology Review. 19 (2007) 265-283.

[48] R.L. Jackson, W. Winn, Collaboration and learning in immersive virtual environments, Proceedings of the 1999 Conference on Computer Support for Collaborative Learning - CSCL '99. (1999) 32-es.

[49] L.M. Ellwein, H. Otake, T.J. Gundert, B.-K. Koo, T. Shinke, Y. Honda, et al., Optical Coherence Tomography for Patient-specific 3D Artery Reconstruction and Evaluation of Wall Shear Stress in a Left Circumflex Coronary Artery, Cardiovascular Engineering and Technology. 2 (2011) 212-227.

[50] W.S. Kerwin, F. Liu, V. Yarnykh, H. Underhill, M. Oikawa, W. Yu, et al., Signal features of the atherosclerotic plaque at 3.0 Tesla versus 1.5 Tesla: impact on automatic classification., Journal of Magnetic Resonance Imaging. 28 (2008) 987-95.

[51] V.L. Yarnykh, M. Terashima, C.E. Hayes, A. Shimakawa, N. Takaya, P.K. Nguyen, et al., Multicontrast black-blood MRI of carotid arteries: comparison between 1.5 and 3 tesla magnetic field strengths., Journal of Magnetic Resonance Imaging. 23 (2006) 691-8.

[52] T.J. Gundert, S.C. Shadden, A.R. Williams, B.-K. Koo, J.A. Feinstein, J.F. LaDisa, A Rapid and Computationally Inexpensive Method to Virtually Implant Current and Next-Generation Stents into Subject-Specific Computational Fluid Dynamics Models., Annals of Biomedical Engineering. 39 (2011) 1423-1437.

[53] D.W. Holdsworth, C.J.D. Norley, R. Frayne, D.A. Steinman, B.K. Rutt, Characterization of common carotid artery blood-flow waveforms in normal human subjects, Physiological Measurement. 20 (1999) 219-240.

[54] B.S. Gow, D. Schonfeld, D.J. Patel, The dynamic elastic properties of the canine left circumflex coronary artery, Journal of Biomechanics. 7 (1974) 389-395.

[55] I.E. Vignon-Clementel, C.A. Figueroa, C.A. Taylor, K.E. Jansen, Outflow boundary conditions for three-dimensional finite element modeling of blood flow and pressure in arteries, Computer Methods in Applied Mechanics and Engineering. 195 (2006) 3776-3796.

[56] D.J. Quam, L.M. Ellwein, H. Otake, R.Q. Migrino, J.F. LaDisa, Mobile Virtual Reality System for Cardiovascular CFD Analysis, in: Biomedical Engineering Society Annual Meeting, 2011.

[57] T.O. Kiviniemi, M. Saraste, J.W. Koskenvuo, K.E.J. Airaksinen, J.O. Toikka, A. Saraste, et al., Coronary artery diameter can be assessed reliably with transthoracic echocardiography., American Journal of Physiology. Heart and Circulatory Physiology. 286 (2004) H1515-20.

[58] M. Williams, A. Nicolaides, Predicting the normal dimensions of the internal and external carotid arteries from the diameter of the common carotid, European Journal of Vascular Surgery. 1 (1987) 91-96.

[59] C.C. Presson, N. DeLange, M.D. Hazelrigg, Orientation-specificity in kinesthetic spatial learning: the role of multiple orientations., Memory & Cognition. 15 (1987) 225-9.

[60] C. Dede, Immersive Interfaces for Engagement and Learning., Science. 323 (2009) 66-9.

[61] J. Fornaro, S. Leschka, D. Hibbeln, A. Butler, N. Anderson, G. Pache, et al., Dual- and multi-energy CT: approach to functional imaging, Insights into Imaging. 2 (2011) 149-159.

[62] D.R. Obaid, P.A. Calvert, J.H.F. Rudd, D. Gopalan, M.R. Bennett, 113 Dual Energy CT improves differentiation of coronary atherosclerotic plaque components compared to conventional single energy CT, Heart. 97 (2011) A64-A65.

[63] C.K. Zarins, D.P. Giddens, B.K. Bharadvaj, V.S. Sottiurai, R.F. Mabon, S. Glagov, Carotid bifurcation atherosclerosis. Quantitative correlation of plaque localization with flow velocity profiles and wall shear stress., Circulation Research. 53 (1983) 502-14.

[64] I. Marshall, S. Zhao, P. Papathanasopoulou, P. Hoskins, Y. Xu, MRI and CFD studies of pulsatile flow in healthy and stenosed carotid bifurcation models., Journal of Biomechanics. 37 (2004) 679-87.

[65] S.Z. Zhao, X.Y. Xu, a D. Hughes, S. a Thom, a V. Stanton, B. Ariff, et al., Blood flow and vessel mechanics in a physiologically realistic model of a human carotid arterial bifurcation., Journal of Biomechanics. 33 (2000) 975-84.

[66] H.G. Bezerra, M. a Costa, G. Guagliumi, A.M. Rollins, D.I. Simon, Intracoronary optical coherence tomography: a comprehensive review clinical and research applications., JACC. Cardiovascular Interventions. 2 (2009) 1035-46.

[67] P.H. Stone, A.U. Coskun, Y. Yeghiazarians, S. Kinlay, J.J. Popma, R.E. Kuntz, et al., Prediction of sites of coronary atherosclerosis progression: In vivo profiling of endothelial shear stress, lumen, and outer vessel wall characteristics to predict vascular behavior., Current Opinion in Cardiology. 18 (2003) 458-470.

[68] I.-K. Jang, B.E. Bouma, D.-H. Kang, S.-J. Park, S.-W. Park, K.-B. Seung, et al., Visualization of coronary atherosclerotic plaques in patients using optical coherence tomography: comparison with intravascular ultrasound., Journal of the American College of Cardiology. 39 (2002) 604-9.

[69] A. Tatu, G. Albuquerque, M. Eisemann, P. Bak, H. Theisel, M. Magnor, et al., Automated Analytical Methods to Support Visual Exploration of High-Dimensional Data., IEEE Transactions on Visualization and Computer Graphics. (2010).

[70] C.A. Taylor, M.T. Draney, J.P. Ku, D. Parker, B.N. Steele, K. Wang, et al., Predictive Medicine: Computational Techniques in Therapeutic Decision-Making, Computer Aided Surgery. 4 (1999) 231-247.

[71] J. Wortham, With Kinect Controller, Hackers Take Liberties, The New York Times. (2010).

[72] IHS iSuppli, The teardown, Engineering & Technology. 6 (2011) 94-95.

[73] A. Shpunt, Z. Zalevsky, Depth-Varying Light Fields for Three Dimensional Sensing, U.S. Patent 8050461, 2011.

[74] B. Freedman, A. Shpunt, M. Machline, Y. Arieli, Depth Mapping Using Projected Patterns, U.S. Patent Application 0118123, 2010.

[75] Microsoft Research, Kinect for Windows SDK Programming Guide, (2011).

[76] OpenNI Organization, OpenNI, (2011).

[77] A. Gordon, The COM and COM+ Programming Primer, Prentice Hall PTR, Upper Saddle River, 2000.

[78] P. Wimmer, Stereoscopic player and stereoscopic multiplexer: a computer-based system for stereoscopic video playback and recording, in: Proceedings of SPIE, SPIE, 2005: pp. 400-411.

[79] Kitware Inc, ParaView, (2012).

[80] A. Rosset, L. Spadola, O. Ratib, OsiriX: an open-source software for navigating in multidimensional DICOM images., Journal of Digital Imaging. 17 (2004) 205-16.

[81] Kitware Inc, VolView, (2011).

[82] D. Maupu, M.H. Van Horn, S. Weeks, E. Bullitt, 3D Stereo Interactive Medical Visualization, IEEE Computer Graphics and Applications. 25 (2005) 67-71.

[83] Planar Systems Inc., Planar SA2311W Specification Manual, (n.d.).

[84] J.P. Wachs, M. Kölsch, H. Stern, Y. Edan, Vision-based hand-gesture applications, Communications of the ACM. 54 (2011) 60.

[85] Hometheater.com, Samsung Consumer Electronics 2012 Product Briefing, (2012).

[86] Amazon.com, (2012).

[87] Vicon Bonita Camera System, (2012).

[88] B. Temkin, E. Acosta, A. Malvankar, S. Vaidyanath, An interactive three-dimensional virtual body structures system for anatomical training over the internet., Clinical Anatomy. 19 (2006) 267-74.

[89] Z. Sun, CT virtual endoscopy and 3D stereoscopic visualisation in the evaluation of coronary stenting, Biomedical Imaging and Intervention Journal. 5 (2009).

[90] M.D. Ford, G.R. Stuhne, H.N. Nikolov, D.F. Habets, S.P. Lownie, D.W. Holdsworth, et al., Virtual angiography for visualization and validation of computational models of aneurysm hemodynamics., IEEE Transactions on Medical Imaging. 24 (2005) 1586-92.

[91] A. Persson, Postmortem Visualization: The Real Gold Standard, in: J. Steele, N.P.N. Iliinski (Eds.), Beautiful Visualization, 1st ed., O'Reilley Media, Sebastopol, CA, 2010: pp. 311-28.

[92] R.M. Taylor, T.C. Hudson, A. Seeger, H. Weber, J. Juliano, A.T. Helser, VRPN, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST ’01, ACM Press, New York, New York, USA, 2001: p. 55.

[93] G. Bell, T. Hey, A. Szalay, Beyond the data deluge, Science. 323 (2009) 1297-8.

COMPUTER SOURCE CODE

All relevant computer source code involved in this project has been submitted in hard and soft copy form to Dr. John LaDisa. Requests for copies should be directed to David Quam at +1 (612) 520-1357 or Dr. LaDisa at the following address.

Department of Biomedical Engineering
Marquette University
PO Box 1881
Milwaukee, WI 53201-1881

Appendix A: Aim 1 Implementation Details

A.1 Use of Individual Scripts for Application of Post-Processing Methods

There are two possible mechanisms with which to apply the post-processing methods described in Section 2.4. The first is to run each script separately using the steps described below. This workflow indicates the order of operations, expected inputs and outputs at each step and special notes on the use of the scripts. It also relates these to the steps indicated in Figure 2.1 on page 26. However, applying the post-processing methods with this technique presents challenges that have been mitigated with our second mechanism, a GUI written in MATLAB (see next page).

Important Note about Formatting: In this section, text that appears in `this font` refers to the names of scripts or the files produced by those scripts.

Table A.1 – Input/Output information for individual post-processing steps

Step In Flowchart	**Sub-step**	**Expected Input**	**Name of Script**	**Expected Output**	**Overall Purpose/Function**	**Special Notes**
G – Resample 3D Model H – Transform to meet VR environment scale	1	Series of files with convention foo_mesh#.vis	`gui_visMeshParser`	foo_mesh_nodes.vis foo_mesh_connectivity.vis	Calculates/extracts the connectivity between the points	***foo_mesh_nodes.vis*** *describes the XYZ location of the nodes of the CFD mesh* ***foo_mesh_connectivity.vis*** *describes how the nodes are connected*

	2†	Series of files with convention foo_mesh#.vis	`gui_visFileParser`
			Note: this script calls
			`gui_visResultsParser`
				[quantity]_foo_res#.vis
				Where 'quantity' is one of: WSS, velocity, traction, displacement, pressure
					Separates the quantity data from each foo_mesh#.vis file into a quantity file	*Each* ***foo_mesh#.vis*** *file describes all hemodynamic data at the # point in time*

	3†	[quantity]_foo_res#.vis	`gui_visFileRename`	foo_res#_[quant].vis *Where 'quant' is one of: wss, vel, tract, press, disp*	Renames input files to "foo_mesh#_[quant].vis" convention	*Each* ***foo_mesh#_[quant].vis*** *file describes a single hemodynamic quantity for all nodes at the # point in time*
	4	foo_mesh_nodes.vis foo_mesh_connectivity.vis	`gui_vessel_location_pre p`	Trans_Rot_Operations.mat	Scales and rotates the models and data such that the flow is parallel to the IVE floor	*The script opens 3 dialog boxes for the user to input rotation amounts and to visually verify results* ***Trans_Rot_Operations.mat*** *describes the translation and rotations that are to be applied to all parts of the model*

	5	foo_mesh_nodes.vis
		wall.ebc
		Trans_Rot_Operations.mat
		
		wall.ebc *is produced by the CFD simulation*
			`gui_wall_mesh_creator`
			Note: this script calls
			`gui_TransRot`
				vessel_wall_pts.txt
				vessel_wall_conn.txt
				vessel_wall_norm_txt
					Extracts vessel surface; removes redundant points	***vessel_wall_pts.txt*** *describes the XYZ location of each node of the mesh; where 'vessel' is the name of the region (e.g. carotid, coronary)*
						vessel_wall_conn.txt *describes how the nodes are connected*
						vessel_wall_norm_txt *describes the normal to the surface described by the nodes/connectivity*

	6‡	foo_mesh_nodes.vis
		stent.ebc
		Trans_Rot_Operations.mat
		
		stent.ebc *is produced by the CFD simulation*
			`gui_stent_mesh_creator`
			Note: this script calls
			`gui_TransRot`
				vessel_stent_pts.txt
				vessel_stent_conn.txt
				vessel_stent_norm.txt
					Extracts surface of stent structure; removes redundant points	***vessel_stent_pts.txt*** *describes the XYZ location of each node of the mesh; where 'vessel' is the name of the region (e.g. carotid, coronary)*
						vessel_stent_conn.txt *describes how the nodes are connected*
						vessel_stent_norm_txt *describes the normal to the surface described by the nodes/connectivity*

	7	Trans_Rot_Operations.mat
		RGBColorMap.mat
		foo_mesh#_wss.vis
			`gui_wss_eon_prep`
			Note: this script calls
			`gui_PolyDataScalars`
				eon_wss_vert_color_#.txt
				eon_wss_time_avg_vert_color.txt
					Interpolates the hemodynamic data (WSS) onto the vessel wall and exports data understandable by	*One* ***eon_wss_vert_color#.txt*** *file describing entire vessel WSS values for a single point in time created for*

Step In Flowchart	Sub-step	Expected Input	Name of Script	Expected Output	Overall Purpose/Function	Special Notes
		RGBColorMap.mat *is a file describing 60 colors in Red/Green/Blue colorspace*			EON	*each point in cardiac cycle* ***eon_wss_time_avg_vert_color.txt*** *describes TAWSS of entire vessel in single file*
	8	Trans_Rot_Operations.mat RGBColorMap.mat OSI VTK file wall.ebc foo_mesh_connectivity.vis	`gui_osi_eon_prep` *Note: this script calls* `gui_PolyDataScalars`	eon_osi_vert_color.txt	Interpolates the hemodynamic data (OSI) onto the vessel wall and exports data understandable by EON	***eon_osi_vert_color.txt*** *describes OSI as a single, time-invariant quantity*
J – Generate custom 3D content	1	Trans_Rot_Operations.mat RGBColorMap.mat foo_mesh#_vel.vis files	`gui_velocity_eon_prep`	eon_node_locations.txt OrientMagCol_#.txt	-	*A single* ***OrientMagCol_#.txt*** *file create for each velocity vector; this file describes the orientation and magnitude of the vector for each point in the cardiac cycle* **eon_node_locations.txt** *describes location of all vectors in single file*
	2	PressHist.dat	`gui_pressure_plotter`	pressure_coordinate.txt pressure.png	Creates a plot of pressure whose time scale is normalized to the length of the cardiac cycle; outputs a text file describing the XY coordinate of the pressure plot	*When the pressure plot appears on the screen, the user needs to manually save the file as a PNG file; no other image format will work*
K – Prepare VR environment	1	vessel_wall_pts.txt vessel_wall_conn.txt vessel_wall_norm_txt	`MeshScript.vbs`	-	Transfers the MATLAB coordinates of mesh nodes, connectivity and normal into the EON mesh format	*Must be run explicitly by pressing 'INSERT' key in EON* *Need only be run once*
	2	vessel_stent_pts.txt vessel_stent_conn.txt vessel_stent_norm.txt	`StentScript.vbs`	-	Transfers the MATLAB coordinates of mesh nodes, connectivity and normal into the EON mesh format specific to the stent	*Must be run explicitly by pressing 'Z' key in EON* *Need only be run once*
	3	eon_node_locations.txt OrientMagCol_#.txt	`VelocityVectors.vbs`	-	Creates velocity vectors and associates the time-varying data so that they can change during the course of the cardiac cycle	*Must be run explicitly by pressing 'HOME' key in EON* *Need only be run once*
L – Medical image registration with 3D model	OCT data 1	normals_points-[time].dat final_ul_vectors.dat reg_points_trans.dat	`eon_oct_orient`	eon_parameters.txt	Calculates the Cartesian coordinates of the image center using the heading and plane locations provided Outputs a text file where each line contains image number, XYZ image center and image normal	***normal_points-[time].dat*** *describes the normal vector to each image* ***final_ul_vectors.dat*** *describes the heading of the Upper Left corner of each image* ***reg_points_trans.dat*** *describes the origin of each image*
	OCT data 2	eon_parameters.txt OCT images	`EON_OCT_Orientation.vbs`	-	Creates 2D planes on the 3D model, translates, orients and	*The OCT images need to be numbered with the same convention in the image*

Step In Flowchart	Sub-step	Expected Input	Name of Script	Expected Output	Overall Purpose/Function	Special Notes
					associates the image with the plane	*file and the **eon_parameters** file.*
	MRI data 1	Segmented MR images Image slice numbers File name root	`mri_prep`	Cropped MR images	Crops the MR images to a region surrounding the carotid artery	*This step is not essential to the function of the system, but it does make it easier to appreciate co-location of plaque components with hemodynamic data*
	MRI data 2	Cropped MR images	`MRIScript.vbs`	-	Places 2D MR slices at 2mm slice intervals along the vessel	-

NOTES

† *These steps are not required if the user has already applied the existing "parse_vis_file.exe" application to the data*

‡ *This step is only applied if a stent model is present in the simulation*

A.2 Use of MATLAB GUI for Post-Processing Methods

The following MATLAB GUI incorporates the workflow described in the previous section and automates much of the file input/output that makes the previous mechanism challenging. Also, the GUI automatically creates the necessary file structure that is expected by the IVE software. In order to use the MATLAB GUI (seen in Figure 2.2) the user completes the following series of commands in the GUI.

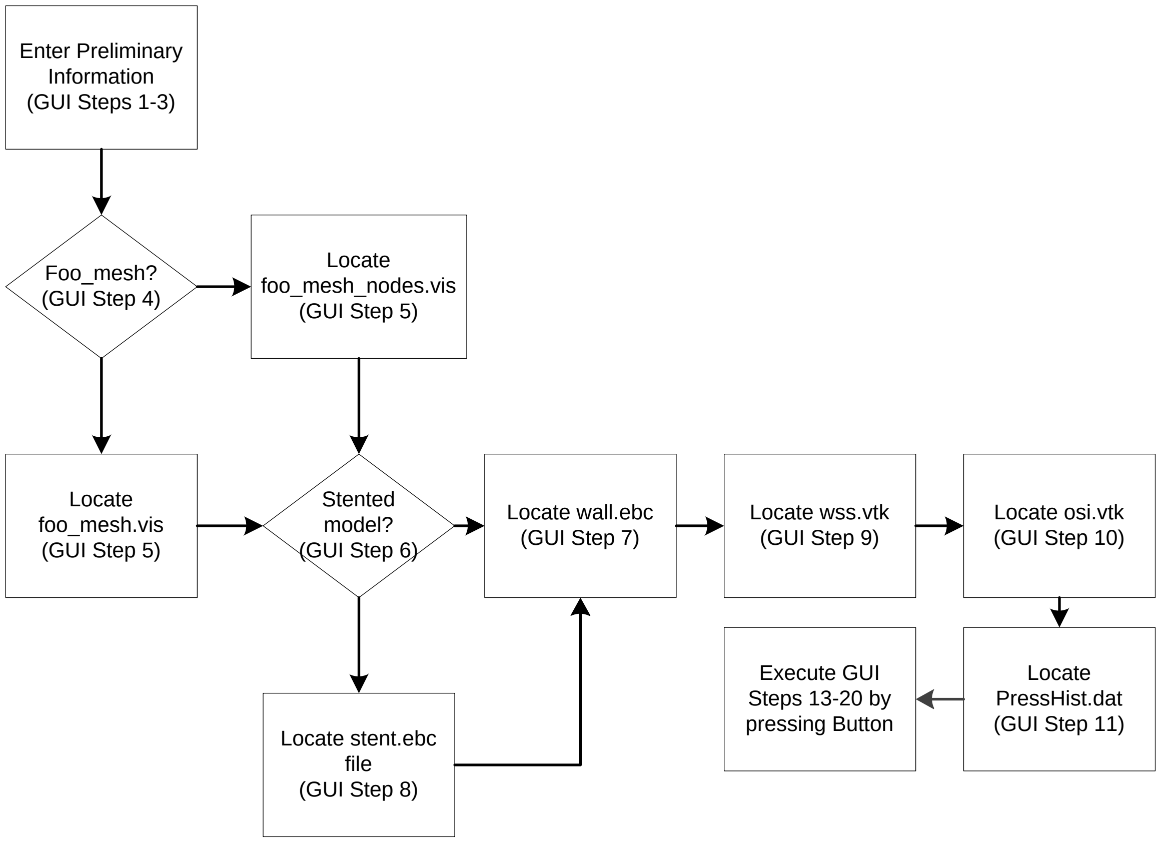

Enter Preliminary Information (GUI Steps 1-3)
Foo_mesh? (GUI Step 4)
Locate foo_mesh_nodes.vis (GUI Step 5)
Locate foo_mesh.vis (GUI Step 5)
Stented model? (GUI Step 6)
Locate wall.ebc (GUI Step 7)
Locate wss.vtk (GUI Step 9)
Locate osi.vtk (GUI Step 10)
Locate stent.ebc file (GUI Step 8)
Execute GUI Steps 13-20 by pressing Button
Locate PressHist.dat (GUI Step 11)

Figure A.1 – Required steps in using the MATLAB GUI.
The final step indicated above ("Steps 13-20") is simple to execute. The user need only press the corresponding button on the GUI. These steps are best performed separately to prevent errors in early scripts from causing fatal errors in subsequent scripts.

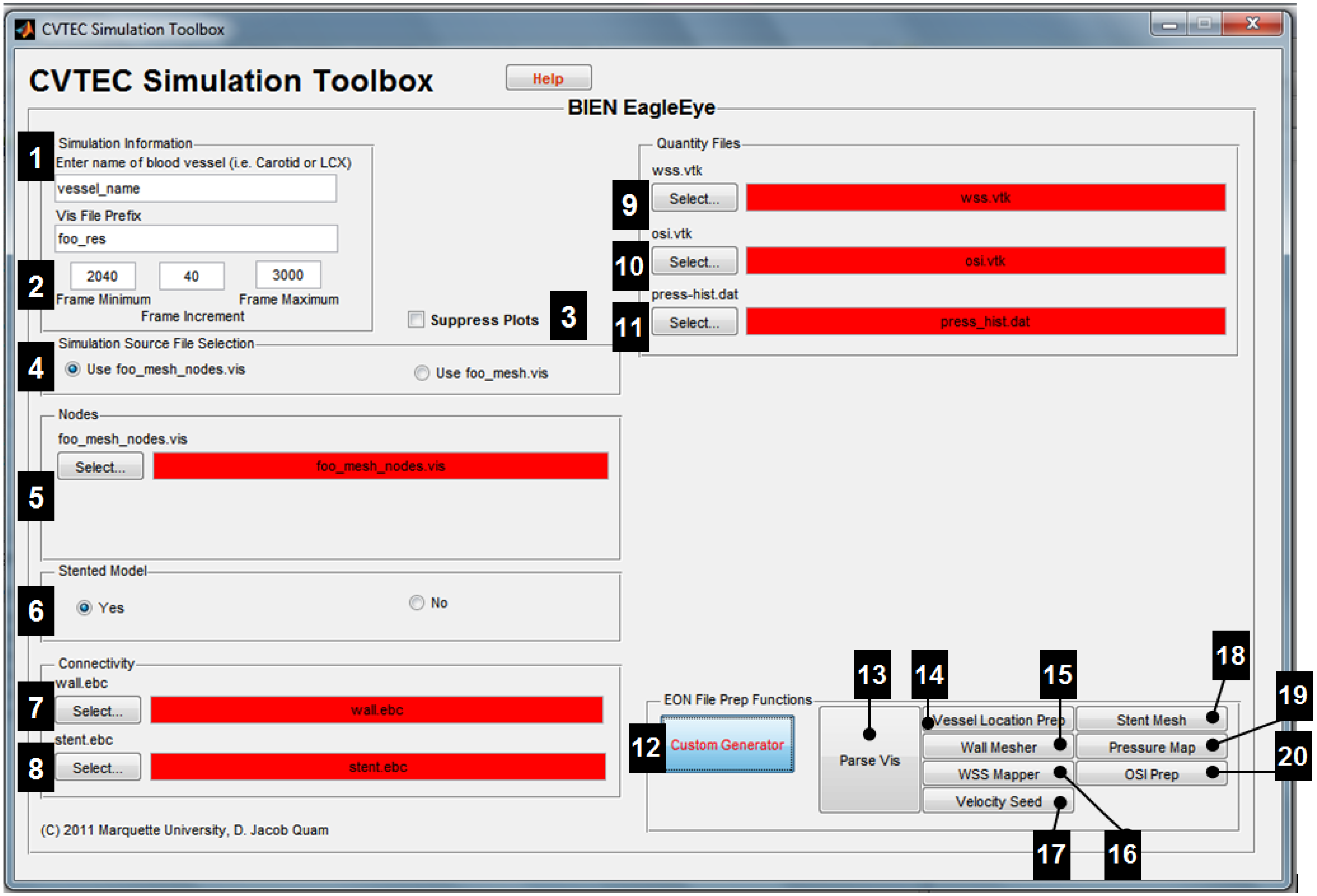

CVTEC Simulation Toolbox
CVTEC Simulation Toolbox
Help
BIEN EagleEye
1
Simulation Information
Enter name of blood vessel (i.e. Carotid or LCX)
vessel_name
Vis File Prefix
foo_res
2
2040
40
3000
Frame Minimum
Frame Increment
Frame Maximum
Suppress Plots
3
4
Simulation Source File Selection
Use foo_mesh_nodes.vis
Use foo_mesh.vis
Nodes
foo_mesh_nodes.vis
Select...
foo_mesh_nodes.vis
5
Stented Model
6
Yes
No
Connectivity
wall.ebc
7
Select...
wall.ebc
stent.ebc
8
Select...
stent.ebc
(C) 2011 Marquette University, D. Jacob Quam
Quantity Files
wss.vtk
9
Select...
wss.vtk
osi.vtk
10
Select...
osi.vtk
press-hist.dat
11
Select...
press_hist.dat
EON File Prep Functions
12
Custom Generator
13
Parse Vis
14
Vessel Location Prep
15
Wall Mesher
WSS Mapper
Velocity Seed
18
Stent Mesh
19
Pressure Map
20
OSI Prep
17
16

Table A.2 – User information for MATLAB GUI

Location Number	Description	Overall purpose/function	Typical Information Provided	Related Steps from Table Above
1	Vessel Information	Allows the user to name the vascular region being modeled; this name is used in the naming of files for better file management	Typical names include: `lcx_coronary` `converged_carotid` *Do not use spaces, instead use '_' character* Vis file prefix is usually 'foo_mesh' but be sure to verify	N/A
2	Frame Increments	Indicates the numerical part of filenames	The `foo_mesh#.vis` file with the smallest numerical component indicates frame minimum	N/A
3	Allow pop-up graphs and plots	The series of scripts used in the post-processing methods generates several output plots; these can be suppressed (not shown) at the user's discretion	It is suggested that the user allow the plots to be shown as they provide useful information	N/A
4	Data source selection	If the user has already processed the `foo_mesh.vis` file, `foo_mesh_nodes.vis` is already present and the user should select this option **If `foo_mesh_nodes.vis` is not available, the user should select `foo_mesh.vis` and the `foo_mesh_nodes.vis` will be generated by the "Parse Vis" button (Step 13)**	The user uses the radio button to indicate if the scripts should look for `foo_mesh.vis` or `foo_mesh_nodes.vis`	N/A
5	Data source location	The user presses the "**Select…**" button to locate the file of interest *The Red color will disappear when the proper file has been located*	-	N/A
6	Stented model?	Indicates whether or not the model contains a stent	-	N/A
7	Wall.ebc file location	Use the "**Select…**" button to locate the `wall.ebc` file	-	N/A
8†	Stent.ebc file location	Use the "**Select…**" button to locate the `stent.ebc`	-	N/A

		file *This box will **not** be visible if Step 6 is set to "**No**"*		
9	Wall shear stress file location	Use the "**Select…**" button to locate the `vessel_wss.vtk` file	-	N/A
10	OSI file location	Use the "**Select…**" button to locate the `vessel_osi.vtk` file	-	N/A
11	PressHist.dat file location	Use the "**Select…**" button to locate the `PressHist.dat` file	-	N/A
12	Show/Hide functions	Shows/Hides buttons related to Steps 13-20	-	N/A
13	Parse Vis Files	Reads `foo_mesh.vis` and separates into `foo_mesh#.vis` and `foo_mesh#_quant.vis`	Requires that **Steps 1-5** have accurate information*	Step G/H, Sub-Steps 1-3
14	Vessel Location Prep	Allows the user to rotate the vessel such that flow is parallel to the floor of the IVE	Requires that **Steps 1-8** have accurate information*	Step G/H, Sub-Step 4
15	Wall Meshing	Resamples the mesh and removes the surface and redundant points	Requires that **Steps 1-8** have accurate information*	Step G/H, Sub-Step 5
16	WSS Mapping	Removes redundancies in the WSS data and maps it to match the nodes of the wall mesh	Requires that **Steps 1-9** have accurate information*	Step G/H, Sub-Step 7
17	Velocity Prep	Using spacing parameters from the user, places	Requires that **Steps 1-8** have accurate information*	Step J, Sub-Step 1
18†	Stent Meshing	Resamples the mesh and removes the surface and redundant points	Requires that **Steps 1-8** have accurate information*	Step G/H, Sub-Step 6
19	Pressure Mapping	Reads the PressHist.dat file and creates a plot and exports a Pressure Versus time file describing the proper location of the marker	Requires that **Steps 1-11** have accurate information*	Step J, Sub-Step 2
20	OSI Prep	Removes redundancies in the OSI data and maps it to match the nodes of the wall mesh	Requires that **Steps 1-10** have accurate information*	Step G/H, Sub-Step 8

NOTE

**The system checks for the necessary files and will flag an error if not all requisite information is available*
†This step should only be employed if a stented model is being used

Appendix B: Aim 2 Implementation Details

B.1 Technical Details of Real-Time GUI

The Stereoscopic Control application is written as a multi-threaded, C# Windows Form GUI. C# is used because of its built-in memory management and integration with Microsoft’s .NET framework. This reduces the amount of overhead that must be written in order to make the software operable on a Microsoft Windows computer. C# also supports syntax enforcement whereby the arguments to a method are validated by variable type and location; this effectively performs error-checking as the code is being written. This feature speeds development because errors are caught before attempting to run the code, and provides the developer with a stronger ability to locate bugs in the code. The code’s methods and fields are broken into 9 classes, associated with a single namespace (`PanelViewer`).

Important Note about Formatting: When text appears in this $\boxed{\texttt{font()}}$ it is referring to a specific method (software function) described in Figure B.1. When text appears in this `font` without “`()`” parenthesis it is referring to a class referenced in Table B.1. In some cases methods are identified along with the class to which they belong in this fashion $\boxed{\texttt{class\_name::method\_name()}}$.

B.1.1 Class Structure

Table B.1 – Class and Inheritance listing

Class	Inheritance†	Function	No. Methods	No. Fields	Lines of Code
`Program‡`	-	Opens and continuously updates the GUI created by `ViewingPaneForm`	1	0	22
`config_dlg`	Form	Creates the dialog box in which the user adjusts the parameters for gesture recognition	7	55	81
`About_Diag_Box`	Form	Creates the "About" dialog box and associated control methods	4	9	108
`ConfigurationVars`	-	Sets the default gesture recognition parameters and provides a global set of variables	0	12	161
`Confirm_Exit`	Form	Creates the dialog box that confirms the user's wish to exit the software	5	5	30
`control_functions`	-	Contains the Methods used to send commands to the Stereoscopic Video Player	14	9	328
`Error_result`	Form	Creates the dialog box that informs the user of errors	4	6	20
`ViewingPaneForm`	Form	Creates the main GUI	38	52	766
`Select_MultiFile_Format`	Form	Creates the dialog boxes needed to open stereoscopic video files	7	7	42
Total	**-**	**-**	**80**	**155**	**1558**

‡This class is automatically created by the Windows Form template.
†Inheritance describes the process by which a lower-level class (such as the ones created for this Aim) inherits the methods and fields of an existing, higher-level class. For example, the `About_Diag_Box` class inherits from the `Form` class. This means that all the methods and fields that exist in the `Form` class are automatically available to the `About_Diag_Box` class. The `Form` class inherits from 7 other classes, which provide a large number of available methods and fields to the Classes developed in Aim 2. Inheritance is illustrated below:

```
        namespace PanelViewer
        {
            partial class About_Diag_Box : Form
        …
        }
```

The `COM Server` used to send commands to the Stereoscopic Player is not treated as a class of the Stereoscopic Control Application code. Instead, we simply use the methods that are made available through server. The GUI displays a real-time depth image from the Kinect camera via the `ViewingPaneForm::OnPaint()` method, actively monitors hand position for gestures and maintains a COM connection to the Stereoscopic Video Player. Table B.2 below describes the primary functions that are carried out by the software the threads on which they are executed.

B.1.2 Multi-Threaded Architecture

Table B.2 - Distribution of computing tasks across multiple threads

Description of Function	Primary Thread	Render Thread
Create depth image and actively update		X
Establish connection with OpenNI interface	X	
Establish connection with Stereoscopic Player	X	
Instantiate gesture recognition objects and monitor for gestures	X	
Create State Machine and route data as described	X	

The `Program::Main()` function executes to continuously update the visible GUI (Figure B.1.1). The visible GUI is created by the code contained in the `ViewingPaneForm` class (Figure B.1.2). The `Control_Functions` class provides the means to send commands to the Stereoscopic Player, like the `Control_Functions::togglePlay()` method (Figure B.1.3).

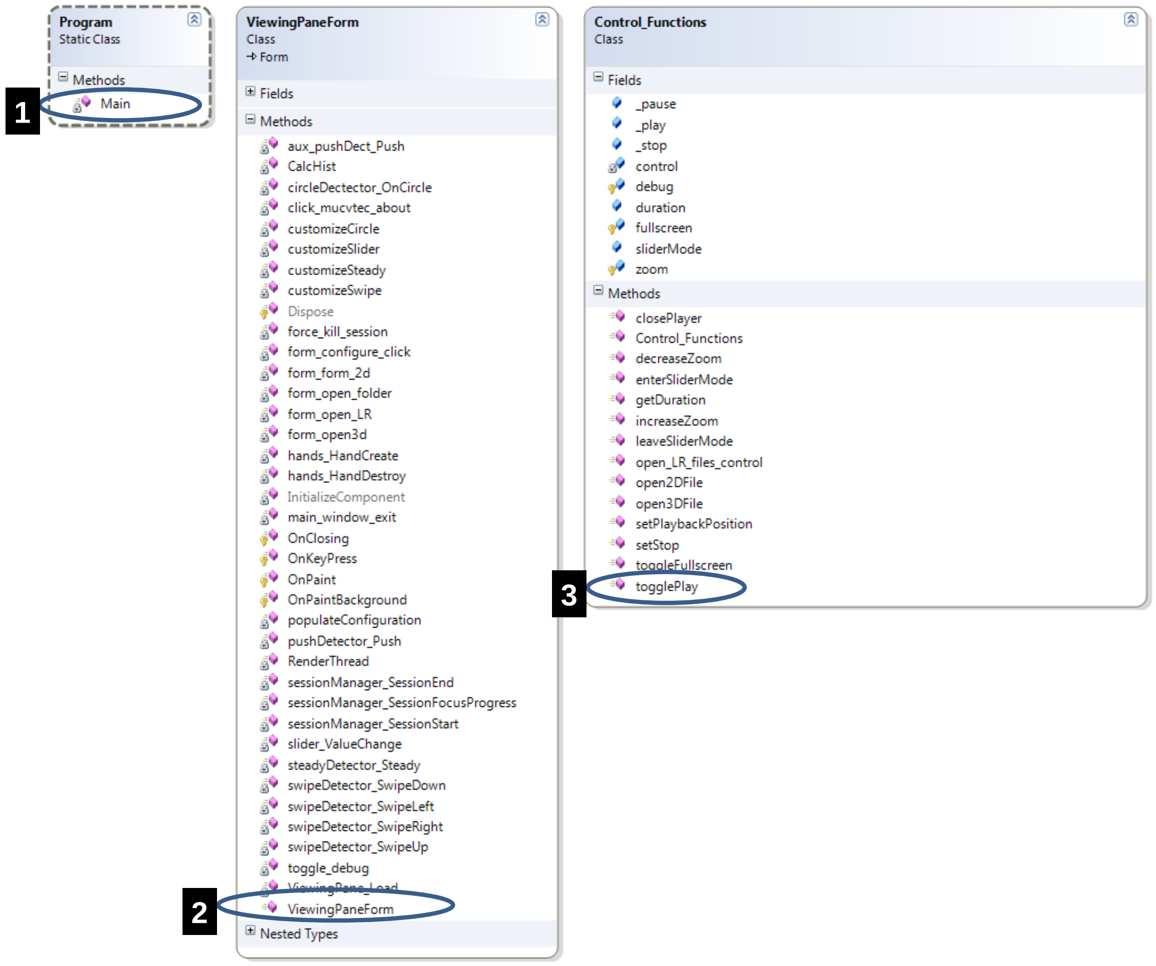


Figure B.1 - Class diagram ofprimary classes of Real-Time GUI
Each class contains fields (single values/variables) and methods (executable functions). Many of the methods contained in the `ViewingPaneForm` *class call methods and rely on fields in the* `Control_Functions` *class.*

B.1.3 Loop Operation

ENTRY POINT: `ViewingPaneForm()` **is called when the form is opened (the method is called once)**

1. Open the GUI window with `Program::Main()`
2. Initialize connections to the Kinect camera, which relies on the `'CVTEC-Tracking.xml'` resource file (a copy of the file is found below)

3. Initialize/start the gesture recognition nodes *(including detectors, broadcasters, router)*

4. Set up the connections between the nodes that define the State Machine (e.g. Flow Router connected to Broadcaster connected to gesture recognition algorithms)

5. Set the default values for the gesture recognition parameters *(e.g. swipe velocity, swipe duration)*

6. Register the gesture events with event handling methods

7. Create the second thread (Render Thread) and connect it to the `ViewingPaneForm::OnPaint()` method described below.

MAIN LOOP: When a new frame arrives to the `ViewingPaneForm`, the following steps are executed (approximately 30 times per second) regardless of the Hand/Slider Mode state of the system

1. Check if the system is *In Session* with `ViewingPaneForm::OnPaintBackground()`. Update the indicator on the GUI if the system state has changed since the last frame.

2. Calculate the elapsed time since the last frame arrived with `ViewingPaneForm::OnPaintBackground()`; display the rate as Frames Per Second on the GUI.

3. Assemble the depth information with `ViewingPaneForm::CalcHist()` and apply a color map and render it on the GUI with `ViewingPaneForm::OnPaint()`.

EXIT POINT: When the user uses the GUI's Menu to select `Exit` (or makes the circle gesture while in `Slider Mode`), the following events are executed (these called once)

1. Join the Render Thread back into the Primary with `ViewingPaneForm::OnClosing()`, which effectively ends the second thread so that the application can be safely closed

2. Disassemble/destroy the State Machine

3. Close the associated instance of the Stereoscopic Player

4. Exit the GUI application

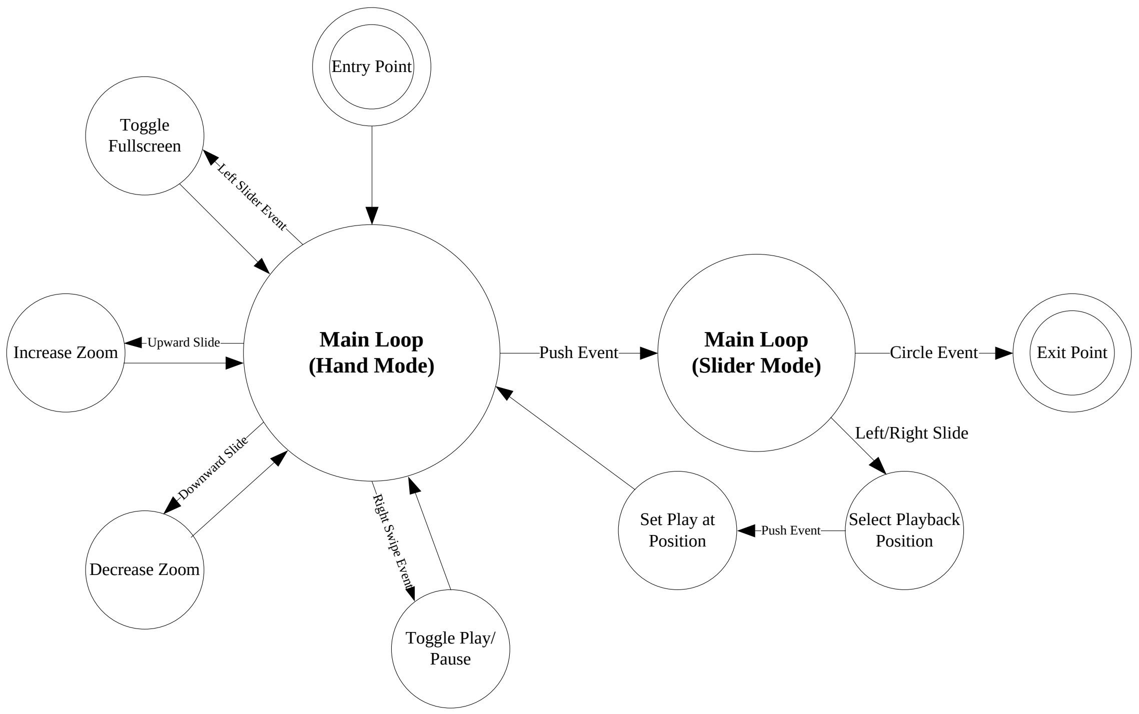


Figure B.2 – Schematic of Loop Structure
After the necessary set-up functions are executed at the ***Entry Point****, the system will run in the* ***Main Loop*** *continuously until the proper command is sent to reach the* ***Exit Point****. The gestures used to achieve specific commands with the Stereoscopic Player are indicated along the arrows.*

CVTEC-Tracking.xml

```
<OpenNI>
        <Licenses>
                <License vendor="PrimeSense" key="insert key here"/>
        </Licenses>
        <Log writeToConsole="false" writeToFile="false">
                <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error (default) -->
                <LogLevel value="2"/>
                <Masks>
                        <Mask name="ALL" on="true"/>
                </Masks>
                <Dumps>
                </Dumps>
        </Log>
        <ProductionNodes>
                <GlobalMirror on="true"/>
                <Node type="Depth" name="myDepth"/>
                <Node type="Hands" name="myHand"/>
    <Node type="Gesture" name="gesture" />
        </ProductionNodes>
</OpenNI>
```

At the **Entry Point** to the Loop the State Machine is created and other set-up functions are executed. The flow of Interaction Data through the Stereoscopic Control Application is controlled by the State Machine. The State Machine is defined in the `ViewingPaneForm` class's constructor method, `ViewingPaneForm()`. The following code from the constructor method connects the major components of the state machine:

```
{...}
        1)  this.InteractionData.AddListener( this.router );

        2)  this.FlowRouter.ActiveListener( this.broadcaster );

        3)  this.broadcaster.AddListener( this.pushDetector );
        4)  this.broadcaster.AddListener( this.swipeDetector );

        5)  this.aux_broadcaster.AddListener( this.circleDectector);
        6)  this.aux_broadcaster.AddListener( this.slider );
        7)  this.aux_broadcaster.AddListener( this.aux_pushDect );
{...}
```

1.

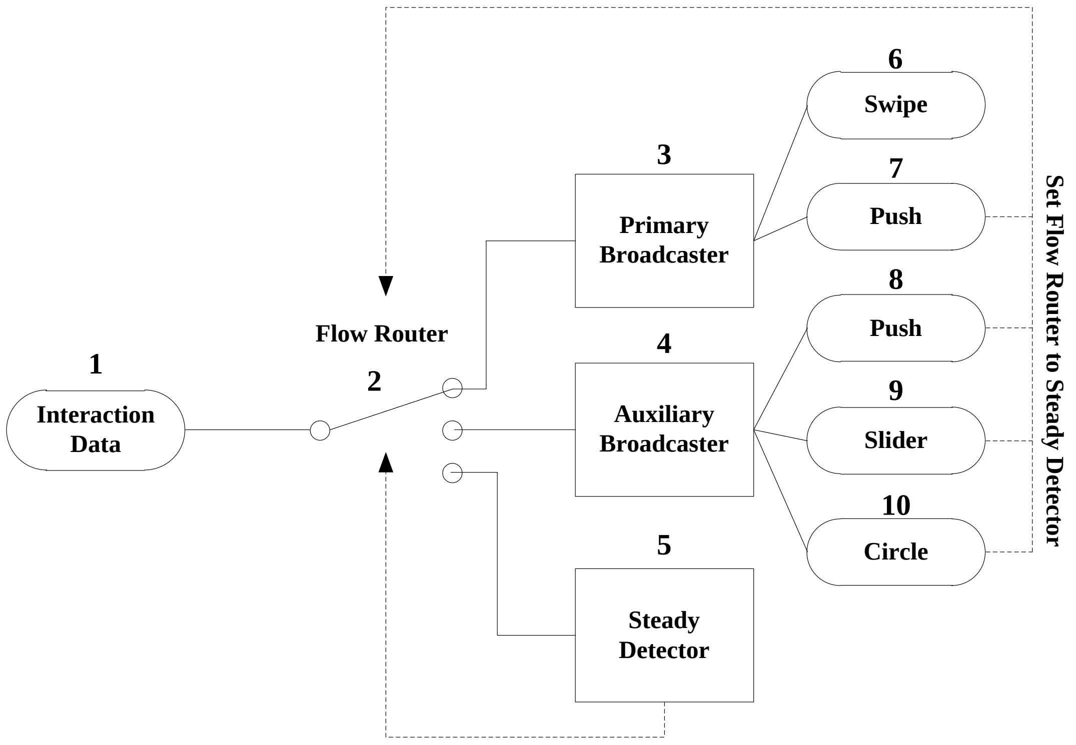


Figure B.3- Alternative representation of State Machine

Table B.3 State Machine connections control flow of Interaction Data

Line of Code	Connects Parts of Figure 3	Function
1	1 to 2	Connects Interaction Data to the Flow Router
2	2 to 3	Routes all Interaction Data to the Primary Broadcaster
3	3 to 6	Connects Primary Broadcaster to Swipe recognition algorithm
4	3 to 7	Connects Primary Broadcaster to Push recognition algorithm
5	4 to 10	Connects Auxiliary Broadcaster to Circle recognition algorithm
6	4 to 9	Connects Auxiliary Broadcaster to Slider recognition algorithm
7	4 to 8	Connects Auxiliary Broadcaster to Push recognition algorithm

Table B.3 describes the initial set-up of the State Machine that controls the flow of Interaction Data about the system. When the Push gesture is recognized `ViewingPaneForm::pushDetector_Push()` toggles the system State, and the connection between the Flow Router and the broadcasters is changed such that:

- When the system is in Slider mode, Flow Router is connected to Auxiliary Broadcaster
- When the system is in Hand made, Flow Router is connected to Primary Broadcaster.

When a gesture is recognized, the Flow Router is set to the Steady Detector. When the Steady Detector criteria have been satisfied, the Flow Router is set back to the Primary Broadcaster via `ViewingPaneForm::steadyDetector_Steady()`.

The flow of Interaction Data within the **Main Loop** can be described in these 3 steps, or as illustrated in Figure B.3.

1. Data from the Kinect camera arrives at the `ViewingPaneForm` class where the gesture recognition algorithms are located. These algorithms continually process the camera data on the Primary thread. When gestures are recognized, events contained in the `ViewingPaneForm` class call methods in the `Control_Functions` class. Methods in `Control_Functions` depend on methods and data from the `COM Server`, as indicated by the thick arrow connecting the two boxes in Figure B.4.
2. The `COM Server` provides access to the Stereoscopic Player's functions, which are exposed to the `Control_Functions` class. This class of functions queries the current state of the player and determines if the player is ready to receive commands.
    a. If the player is not ready to receive commands, no command is sent through the COM server, even if a gesture has been recognized; such commands are ignored.

b. If the player is ready, the logic contained in the methods determines the proper command to send. For example, if the player is currently in the `Play` state, the `Control_Functions::togglePlay()` function will set the state to `Pause`. If the player is currently in the `Pause` state, the `Control_Functions::togglePlay()` function will set the state the `Play`.

3. If the user changes the gesture recognition parameters using the dialog box created by the `config_dlg` class, the new parameters are sent to the `ViewingPaneForm` class via the `configurationVars` variables class and subsequently to the gesture recognition algorithms.

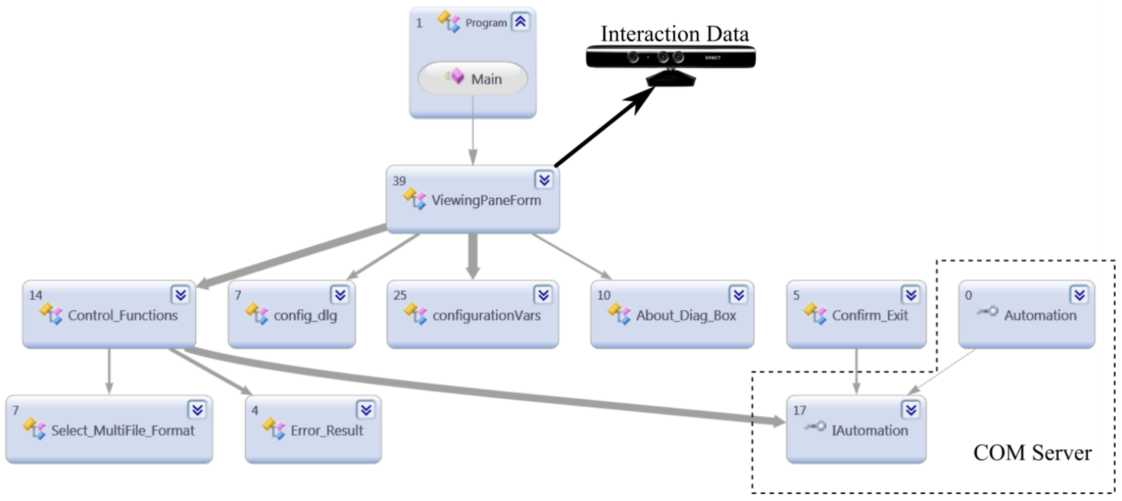


Figure B.4 - Figure illustrating the flow of data between the classes.
*The thickness of the arrow communicates how much data flows; the direction of the **arrow points towards the source** of the data. The* `Control_Functions` *class relies heavily on the methods defined in the COM Server.*

B.2 User/Camera Orientation and Gesture Recognition Algorithms

The swipe gesture detectors are less sensitive to the orientation between the user and the camera than the push detector. This is because the swipe gesture detectors ignore the depth dimension (*z*) and examine only (*x*,*y*) position. As seen in Figure B.5 (right) the *x* component of the swipe gesture vector is still visible to the camera, while the z component is ignored. The system is capable of determining the swipe gestures ***error free*** for $0° < \varphi < 28°$. When $28° < \varphi < 45°$, the system accurately detects the swipe gesture approximately 90% of the time. These values were acquired when the hand was 48” from the camera.

The push gesture detector relies heavily on the *z* component of user hand movement. For this reason, the angular orientation, γ, between the camera and the user’s hand is influential in the camera’s ability to interpret movements (see Figure B.5, left). If the *x* component of the motion vector is too large, the z component will be ignored, and the movement interpreted as a swipe instead of a push. The system is capable of detecting the push gesture accurately for $0° < \gamma < 35°$. For $35° < \gamma < 45°$ the accuracy of the system will vary depending on the displacement and velocity of the motion, as well as the “angle of attack” – the path of the hand in the YZ plane. For $\gamma > 45°$ accuracy is 0%. These values obtained when the hand was 48” from the camera.

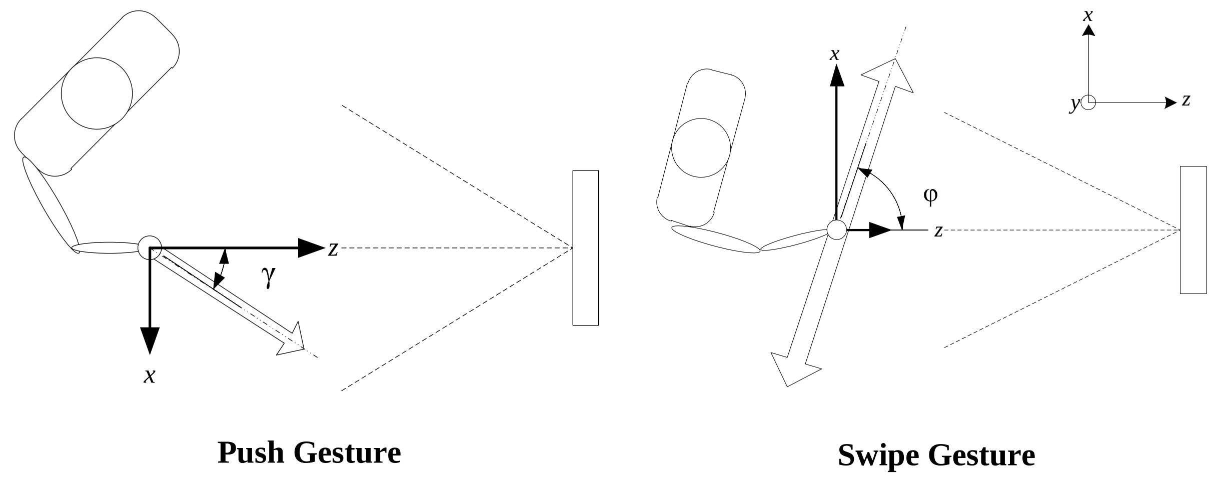


Figure B.5 – User/camera orientation schematic
Interpretation of user hand motions and their directional components. The push gesture detector requires depth information. For $0° < \gamma < 35°$ *the push detector operates accurately, but performance degrades as* $\gamma \to 45°$. *The swipe detector operates sufficiently well for* $0° < \varphi < 45°$.

When starting up, the Real-Time GUI does not immediately track any user motions. The user must perform a training gesture to initialize the hand tracking features of the Real-Time GUI. In its current form, the system relies on the Push gesture (similar to the gesture described above) to initialize hand tracking. It was observed that the Push training gesture was difficult to master without specific instruction. A guide has been supplied with this Thesis to help other users learn the commands use of the system.

It was also observed that the system was able to detect the hand most rapidly (fewest number of attempts) when the user performed the Push training gesture with a γ=0, meaning that the hand's direction of motion is directly towards the camera's sensor (see Figure B.5). Performance was further improved when the user performed the Push training gesture at a modest pace: a full extension and retraction of the arm within the camera's FOV in approximately 0.8±0.04 seconds. At full extension the hand should be no closer to the camera than 24”, as the depth information supplied on objects nearer than this tends to be unreliable if it is even present.

Anecdotal evidence suggests that Push gesture rates that greatly deviate from the 0.8 second average will not cause the system to initiate hand tracking. In such cases the user can attempt the Push gesture again without impact on future performance. Comments from user indicate that the commands are simple and easy to learn.

B.3 User Scenarios and Suggested Settings of Configuration Dialog

In general there are two cases in which the system can be used. The first is a situation when the user is within 60” of the Kinect. Such a situation would arise when the user intends to interact with the system at a desk. The second situation arises when the user interacts with the system from a distance, such as while performing a medical procedure. The following section describes the settings that are best suited for each case.

When the user is within 60” of the Kinect Camera (such as when working at a desk):

The user should leave the parameters at their default setting.

When the user is >60” from the Kinect camera (such as when performing a medical procedure):

The user should set the following parameters; other parameters should be left at their default setting. In additional to recommended settings for these parameters, we offer a discussion on how the parameters influence the performance of the system.

- <u>Minimum Velocity: **0.15**</u>
    - This setting changes how quickly the user must move his hand to trigger the detection

- <u>Swipe Duration: **1500**</u>

    - More deliberate hand gestures are assumed to be of longer duration, where the user's hand is in motion for a longer period of time. By increasing the required duration of hand motions we can ensure that the motions are more deliberate. This allows the user in the scene to make other hand motions without the risk of them being interpreted as gestures.

- <u>Steady Duration: 1**500**</u>
    - The amount of time required for the user to hold his hand steady; increasing this setting allows for more deliberate gestures.

- <u>Angular Deviation: **15**</u>
    - This setting makes the system enforce more horizontal hand motions. The default settings (45º) are more forgiving and allow the user to have a greater vertical component to his hand motions.

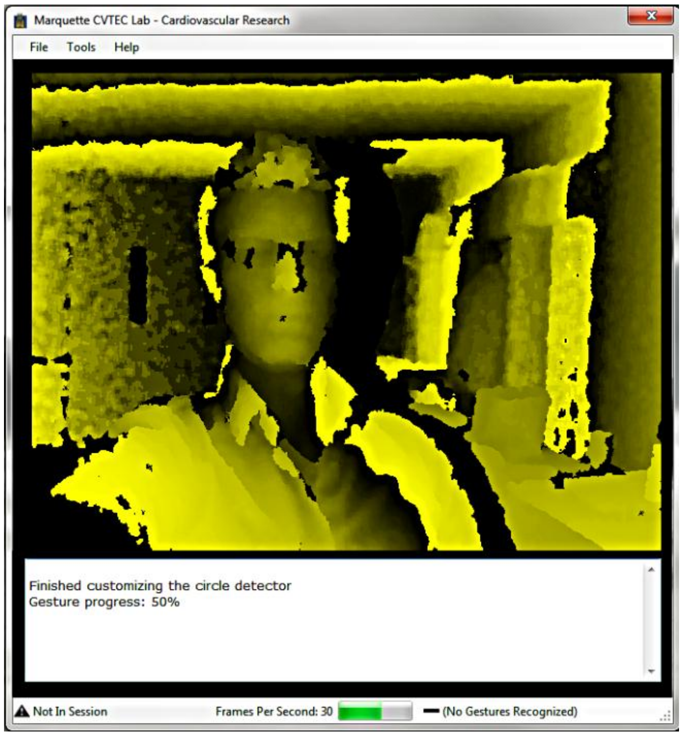
- <u>Slider Width: **200**</u>
    - This setting makes the horizontal scroll bar more narrow and therefore less movement is required to traverse the entire scrolling region.

The user should interpret these proposed settings as suggestions. The values were established during the course of investigation in an averaged size room with some ambient light from natural and artificial sources, and very little obstacles for the infrared tracking light. In the event that the system is used in a less ideal environment, it may be necessary to further refine these parameters in order to achieve optimal results. The discussion provided in the above section was included in order to clarify how each setting may impact performance.

Appendix C: User Guide

C.1 Instructions for Use: Real-Time GUI

These instructions outline the basic operation of the software and gestures used in the Real-Time GUI (Figure 1) component of the Stereoscopic Visualization System.



Finished customizing the circle detector
Gesture progress: 50%

Figure 1 - Real-Time GUI window

To Open and Run the Ream-Time GUI:

1. Run Microsoft Visual Studio 2010 and open the **PanelViewer** solution file
2. Click **Debug** > **Start without Debugging**

To Open a 3D video file for playback with the Stereoscopic Visualization System:

1. Open and run the PanelViewer solution as described above
2. On the Real-Time GUI window select **File** > **Open** > **3D File**
3. You can select a single 3D file video that contains both left and right eye images with the **3D Video File** option
4. You can also select separate left and right eye video files with the **Left and Right Files** option.

To Change the User-Defined Parameters:

1. Open and run the PanelViewer solution as described above
2. On the Real-Time GUI window select Tools > Configure
3. Click **Save** to commit the changes and **Cancel** to clear changes.

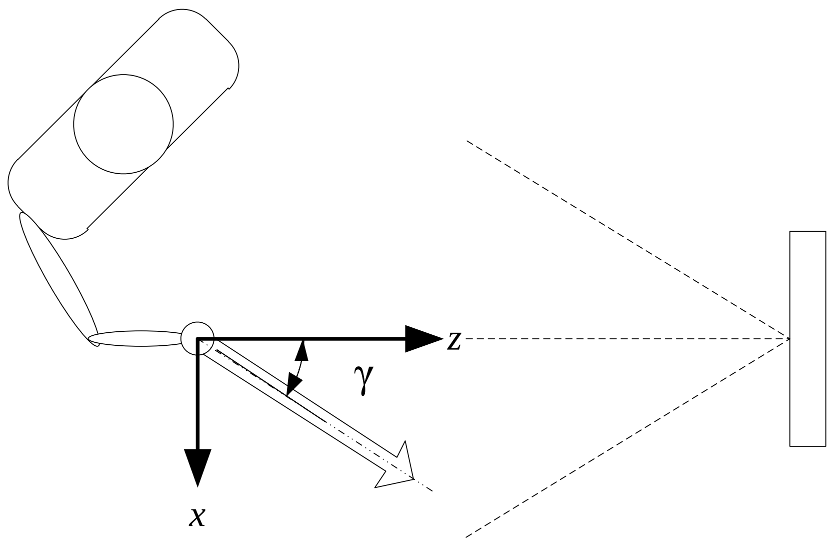


Figure 2 - Bird's eye view of a user performing a push gesture

To Initiate Hand Tracking with the Real-Time GUI:

1. Open and run the PanelViewer solution as described above

   The user should bear in mind that the image displayed on the Real-Time GUI represents the data captured by the camera. The user must ensure his hand is visible on the Real-Time GUI when performing the training gesture.

2. With the hand at least 24" from the camera **perform a Push gesture** such that:
   - the direction of motion of your hand is directly towards the Kinect camera such that γ=0 from Figure 2.
   - the gesture should last approximately 0.8 seconds in which the user fully extends and retracts his arm
2. If the first attempt to initialize hand tracking fails, the user may attempt again 2 seconds after the completion of the first
3. Once the system has recognized the user's hand, the ✓icon will appear in the **lower left** corner of the Real-Time GUI window (see Table 1)
   - At this time the system is ready to recognize other gestures and send the corresponding commands to the Stereoscopic Player

Table 1 - Key of System Icons, Significance and Location

Icon	Significance	Location
✓	System actively tracking a hand	Lower Left
⚠	System not actively tracking any hand points	Lower Left
?	Hand points missing from FOV, waiting for hand to reappear	Lower Left
↓	Downward swipe detected	Lower Right
+	File successfully opened	Lower Right
🔗	Hand is steady, system ready for next gesture	Lower Right
←	Leftward swipe detected	Lower Right
→	Rightward swipe detected	Lower Right
✕	System closing	Lower Right
↑	Upward swipe detected	Lower Right

Considerations for Interacting with the Real-Time GUI with Gestures:

- Attempt to maintain a perpendicular orientation to the camera. While the camera can correct for angular offsets up to ~35º, the most predictable and consistent system performance occurs when the user maintains an angle of <10º between the user and camera lines of sight.

- Use the Real-Time GUI's icon and textual indicators to learn how the system is interpreting your gestures. Watching the Real-Time GUI and observing how your actions are captured and processed will help you to achieve the best system performance.

- Use the User-Defined Parameters to customize the performance of the system to meet your needs. Changes are immediate and can be made any number of times.