VIRTUAL TRAINING SYSTEM FOR DIAGNOSTIC ULTRASOUND

by

Daniel P. Skehan

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical and Computer Engineering

October 2011

APPROVED:

Prof. Peder C. Pedersen, Advisor

____

Prof. Edward Clancy

____

Prof. Reinhold Ludwig

____

Abstract

Ultrasound has become a widely used form of medical imaging because it is low-cost, safe, and portable. However, it is heavily dependent on the skill of the operator to capture quality images and properly detect abnormalities. Training is a key component of ultrasound, but the limited availability of training courses and programs presents a significant obstacle to the wider use of ultrasound systems.

The goal of this work was to design and implement an interactive training system to help train and evaluate sonographers. This Virtual Training System for Diagnostic Ultrasound is an inexpensive, software-based training system in which the trainee scans a generic scan surface with a sham transducer containing position and orientation sensors. The observed ultrasound image is generated from a pre-stored 3D image volume and is controlled interactively by the user's movements of the sham transducer. The patient in the virtual environment represented by the 3D image data may depict normal anatomy, exhibit a specific trauma, or present a given physical condition. The training system provides a realistic scanning experience by providing an interactive real-time display with adjustable image parameters similar to those of an actual diagnostic ultrasound system.

This system has been designed to limit the amount of hardware needed to allow for low-cost and portability for the user. The system is able to utilize a PC to run the software. To represent the patient to be scanned, a specific scan surface has been produced that allows for an optical sensor to track the position of the sham transducer. The orientation of the sham transducer is tracked by using an inexpensive inertial measurement unit that relies on the use of quaternions to be integrated into the system. The lack of a physical manikin is overcome by using a visual implementation of a virtual patient in the software along with a virtual transducer that reflects the movements of the user on the scan surface. Pre-processing is performed on the selected 3D image volume to provide coordinate transformation parameters that yield a least-mean square fit from the scan surface to the scanning region of the virtual patient.

This thesis presents a prototype training system accomplishing the main goals of being low-cost, portable, and accurate. The ultrasound training system can provide cost-effective and convenient training of physicians and sonographers. This system has the potential to become a powerful tool for training sonographers in recognizing a wide variety of medical conditions.

Acknowledgements

I would like to acknowledge Professor Peder C. Pedersen for his guidance and support. His constructive discussions and willingness to help me locate any resource to aid my progress were key components in my success.

I would like to thank the Telemedicine and Advanced Technology Research Center (TATRC), for providing the grant under which this research was performed.

Jason Bryan of the Ohio Supercomputer Center was a tremendous help, providing an updated version of his gear library, along with support and assistance.

I would also like to acknowledge Jason Kutarnia for his continuing assistance throughout the research and Jia Wang for his assistance in interfacing with the Anoto technology.

The USB-based Anoto position sensor and support provided by Anoto, AB, and the transducer shells used to construct the sham transducer from by Sound Technology, Inc. were provided free of charge.

Patrick Morrison and Tom Angelotti of the WPI ECE shop were a valuable resource for locating and machining materials used for various drafts of the transducer and scan surface.

I would like to also thank Neil Whitehouse for his efforts in machining material used for the scan surface.

Contents

Table of Figures

Table of Tables

1) Introduction

Diagnostic ultrasound has become a widely used form of medical imaging as well as the main imaging modality in the developing world. When compared to other imaging modalities such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), ultrasound is more portable, safer (non-ionizing), and relatively inexpensive. Lower cost (below $10,000) ultrasound imaging systems are also becoming available. The image quality and diagnostic content produced by ultrasound imaging, however, are heavily dependent on the skill of the operator. This operator dependency makes sonographer training a bottleneck for the use of diagnostic ultrasound imaging. In order to generate a better quality image and make better diagnostic decisions, significant training is required of the sonographer. During his/her training, a sonographer may not encounter, or have limited exposure to, many conditions or pathologies that may be detected by ultrasound imaging. In attempt to better prepare sonographers, as well as maintaining skill sets, computer-based simulations have been developed.

1.1) Commercial Ultrasound Simulation Systems

Currently the main sources of experience for sonographers in training are through training workshops and clinical practice. The clinical practice may occur for a medical student, 2-year training program for sonographers, or during continuing education courses. The workshops use healthy volunteers as human subjects and give little exposure to different pathologies for the trainee. The greatest experience that a trainee receives is during several months of clinical practice where they learn and develop their ultrasound imaging skills. During this period the trainee may still only have exposure to a limited number of conditions. In order to fill this void, training programs will show the trainee images or video clips of different pathologies. This procedure still does not give the trainee the hands-on experience of dealing with these conditions.

In many medical fields, especially surgery and other invasive procedures, simulation has been playing a more prominent role in training. Simulation attempts to fill gaps that normal training cannot typically fulfill. Simulation also engages the trainee, allowing them to become better acquainted with the processes involved with their learning field.

The critical role of simulation in medical training has been well established, as supported by several conferences (e.g., Medicine Meets Virtual Reality) and abundant publications [Satava]. “Simulation provides a controlled and safe practice environment and ease of access to promote learning” [Henriksen]. A range of ultrasound training systems are available and their effectiveness has been proven, e.g. in trauma [Knudson, Petrinec], obstetrics [Maul], interventional [Lövquist]. However, despite their potential, ultrasound simulators are being utilized only to a small extent in the training settings described above. Medical schools with emphasis on medical ultrasound still rely on traditional teaching [Hoppmann].

Studies have shown the effectiveness of learning from ultrasound simulator-based training. Fetal abnormalities rarely occur (overall about 2%), and due to a lack of experience ultrasound screening has not led to better detection [Maul et al.; Merz]. This example shows how only very experienced ultrasound obstetricians may develop adequate skills. However, with the use of ultrasound simulation it is possible to have obstetricians hone their ultrasound scanning skill set for detecting such abnormalities.

There have been previous developments of ultrasound simulation systems, but none of which have become widely used. Due to high costs and large size, commercial ultrasound training systems are mainly found in sonography teaching facilities. The usefulness of ultrasound simulation has however begun to prove itself. As a result of this more widely seen effectiveness, the market for ultrasound simulation has been growing. Several competing companies have produced systems that are readily available for purchase.

*MedSi*m has an ultrasound simulation system named *Ultrasim*. This system requires the purchase of a complete “traditional”, but realistic-looking, ultrasound scanning station and manikin. The system has the ability to load different patient data. This feature allows for the user to have experience with a wide variety of pathologies. The system however only uses specified reference points to scan for the user, causing a lack of realism. These systems range from $35,000 - $40,000 plus an annual fee [MedSim].

Glassworks has developed an ultrasound simulation system named *Heartworks*. This system focuses on trans-esophageal echocardiography ultrasound procedures. The system hardware is comprised of a manikin, trans-esophageal probe, and display. This system is able to display a visualization of anatomy and a synthetic ultrasound image. The system is limited to over the cardiac region via esophagus. The system is also sold for over $100,000 and not sold in the U.S. [Heartworks].

Schallware is another company that has produced an ultrasound simulation system. The *Schallware* system uses multiple dummies with multiple transducers required to scan for different conditions. The company also has provided much planning and documentation for workshops to be used for their system. This system has been estimated to cost $60,000 per module [Schallware].

The *Simulab Corporation* offers several different ultrasound simulation systems including the *SonoMan*. Each of the systems has a sensing probe, a specific manikin, and several different software modules. These systems however, do not require the purchase of additional scanning hardware. The scanning technology with these systems uses unique scan points on the manikins. Unlike the previously discussed system, these systems use a PC which is assumed to be commonly available. The SonoMan system is priced at $7,400 for one procedure. [Simulab]

1.2) Motivation for the Thesis Research

The mission of the Ultrasound Research Laboratory at Worcester Polytechnic Institute is to enable a wider use of medical ultrasound so that it can be used by medical personnel with modest training. This in part focuses on the development of robust, portable ultrasound systems for obstetrics training (including the developing world) first-responders and battlefield medicine. This wider use, however, produces a need to effectively and efficiently train more sonographers. The goal of this training system is to provide realistic, low-cost training opportunities. By limiting the need for real ultrasound hardware and human subjects, training can be accomplished at a faster pace and at a lower cost. This system also eliminates the need for training to take place in a clinical environment, allowing for greater flexibility for the trainee.

There are several shortcomings of the existing approaches. One of the most significant of the shortcomings is the cost of the systems. Systems that require a “traditional” ultrasound hardware station to be purchased increase the costs of the system dramatically. Requiring many different manikins for scanning raises the system costs as well. Many of the systems available use a form of magnetic field tracking for determining the position of the sham transducer on the manikin. This form of position tracking requires the production of expensive manikins and the need for different transducer hardware for different pathologies being studied. These manikin-based systems are also bulky and may be difficult to transport.

Approaches taken in the design of this new ultrasound training simulation system may overcome these limitations. The first approach taken is similar to the *Simulab Corporation* products in the utilization of a commonly available PC. PCs are widely available and by using one as the ultrasound simulation hardware, system costs drop dramatically.

Our system takes a unique approach to the use of scan surface by employing a generic scan surface that is similar to the shape of an average human torso. The training experience will be kept in the virtual domain as much as possible. By allowing for a more virtual simulation, the need for hardware is greatly reduced, which in turn reduces the cost of the system.

Another approach that sets this system apart from others is its position tracking technology. Instead of using costly magnetic tracking, a less expensive form of optical tracking and gyro-based tracking is used. This eliminates any possible need for purchasing multiple pieces of hardware for different simulations as well.

1.3) Thesis Overview

Chapter 2 begins by discussing the first generation ultrasound training system developed at WPI’s Ultrasound Research Laboratory. The chapter continues by discussing the strengths and weaknesses of these initial training systems. This is followed by an overall systems description of the newly developed ultrasound training system. This section is concluded by giving the reader the necessary background for understanding more detailed descriptions of the new system including mathematical concepts of quaternion orientation representations.

Chapter 3 describes the new hardware and its firmware that are used in the new system. Updates from the previously developed software that allow for the use of the new position tracking technology are summarized. The integration of the new position tracking technology along with an Inertial Measurement Unit for orientation tracking into the previously developed software is also discussed.

Chapter 4 describes updates made to the existing software that create an enhanced virtual experience for the user. A set of data used with this system to derive the virtual patient is discussed. The implementation of a virtual transducer applied to the virtual patient is described. The chapter then goes into detail of the coordinate transformations from the physical transducer on the scan surface to the virtual transducer on the virtual patient.

Chapter 5 discusses how the system was evaluated. The system is broken down into several main parts to discuss where error may occur. Each part of the system has its own type of errors defined as well as when they would occur. Several tests were performed and discussed for the evaluation of the image reslicing software.

Chapter 6 provides conclusions of the newly developed system. These conclusions state the accomplishments of this research in regards to the original goal of developing a low-cost, portable, and accurate ultrasound training system. Future work to improve the system is also presented. This future work includes making the system more visually appealing, tailoring the system to best affect students learning, and defining of image data requirements for the system.

2) Background

2.1) Development of Ultrasound Simulation Training System

The Ultrasound Research Laboratory has previously carried out research involving ultrasound simulation. In particular, research has been led by Christian Banker and Jason Kutarnia. Resulting from this research are projects that have developed software which has provided a foundation for this ultrasound training system. The following sections summarize relevant contributions from previous ultrasound simulation systems as well as give a development background of this ultrasound simulation.

2.1.1) First Generation System

One notable project was Chris Banker's Interactive Training System for Medical Ultrasound, which used a manikin as a scanning surface [Banker]. One of the goals of the research was to create a software ultrasound simulation. The software would track a sham transducer's position and orientation by use of a system called *trakSTAR* from *Ascension Technologies*. The *trakSTAR* system used magnetic field signal technology which required its own hardware for transmitting and receiving [Banker]. A portion of this tracking hardware was embedded into a lifelike manikin. Embedding the hardware into the manikin allowed for position and orientation (six degrees of freedom) of the transducer relative to the scanning surface. The user would observe an ultrasound image generated from a pre-stored 3D image volume [Banker]. A systems diagram of the system is depicted in Figure 2.1. The foundation of the software used in this thesis is based on Chris Banker's implementation of his research.

One of the core functionalities of the software is to render a 2D image through a 3D volume. The volume renderer is based on a previously open-source library named GEAR, which is written in C++ and based on OpenGL. OpenGL is the most widely adopted graphics standard that has come to be used in a wide range of industries and research [OpenGL]. OpenGL is exclusively a rendering language and purposely does not implement any features relating to windows or any other specific sources in order to be platform independent. The independence of OpenGL causes it to be often used with the *OpenGL Utility Toolkit* (GLUT). GLUT is a toolkit for OpenGL programs and a windowing API for OpenGL [OpenGL].

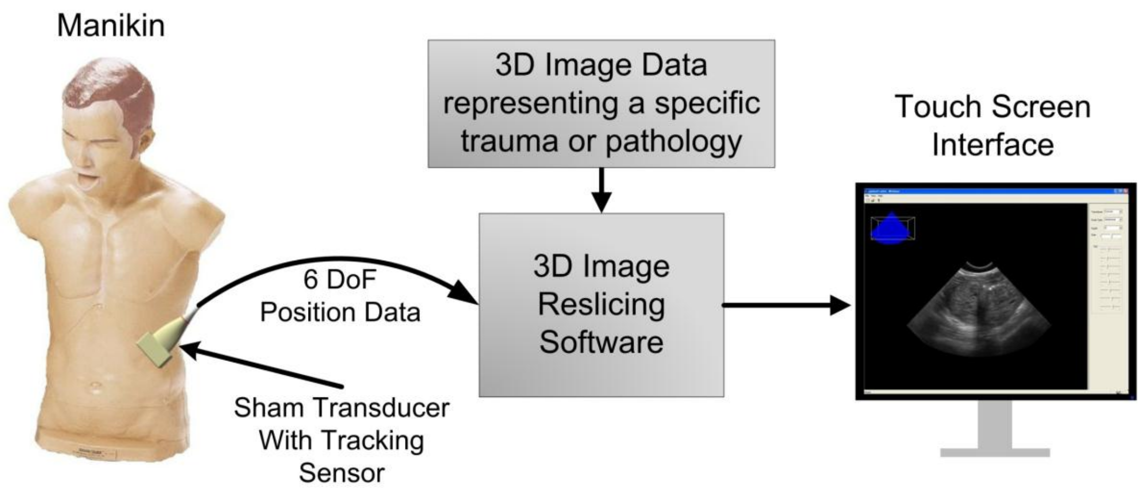

Manikin
3D Image Data representing a specific trauma or pathology
Touch Screen Interface
6 DoF Position Data
3D Image Reslicing Software
Sham Transducer With Tracking Sensor

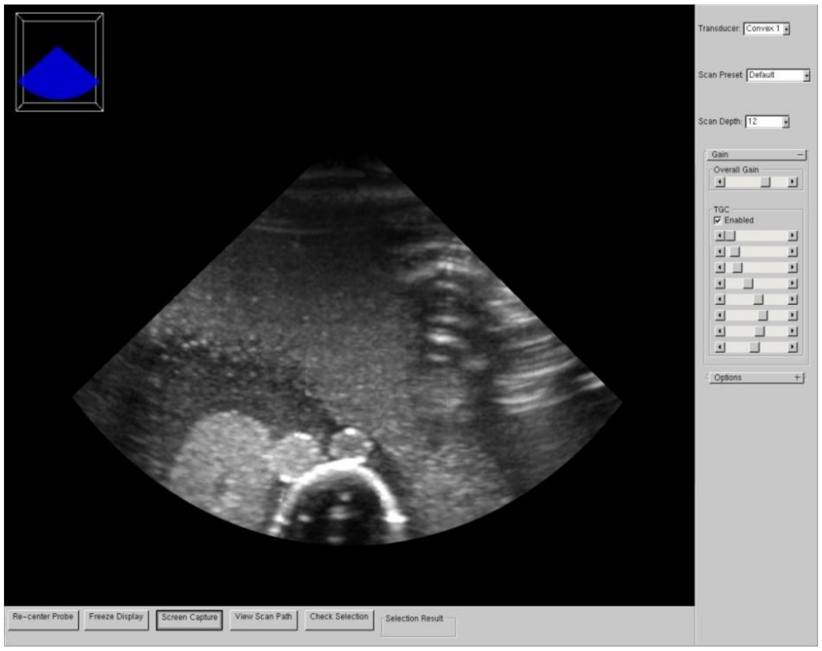
Figure 2.1 – Systems Diagram of First Generation System [Banker]

The first generation system software is capable of arbitrarily reslicing a 3D image volume and producing the corresponding 2D slice in real-time. This functionality is made possible by having the 2D slice be of small thickness while not thin enough to be transparent. “The specific range that is used is -1.0 to -0.95, which are unitless values represented in OpenGL coordinates” [Banker].

By default, Gear tessellates the image volume using enough texture bricks to represent the entire volume. Due to the large image volumes needed for this application, the number of bricks needed can be quite large and require a large amount of computation. This posed a potential problem of running the application in real-time. After consulting with Jayson Bryan, the author of the *Gear* library, it was decided that in order to fix this issue a limit was to be placed on the number of bricks rendered to four per image rendering. This limit was implemented by using the image volume shader object’s method *setTesselateFunc*.

After applying these changes, the program rendering software was reported to have been reslicing the image volume and rendering at 25 to 35 frames per second (FPS). These rates were reported while running the application on a Desktop PC with 2.0 GHz dual-core Xeon processor, 4GB of memory, and a NVIDIA Quadro FX 550 graphics card [Banker]. These rates are comparable to ultrasound imaging frame rates and should be acceptable for this application. The frame rates, however, are dependent on the amount of available processing power. Therefore, the higher the computational performance that a computer offers, the higher the frame rate can be adjusted to be.

The graphical user interface (GUI) used in this software was created with the open source library named *OpenGL User Interface* (GLUI). GLUI operates with GLUT and is easily incorporated into OpenGL applications. GLUI provides user controls by means of drawing OpenGL objects to the screen. [GLUI] The controls that are featured in the GUI were design to resemble actual ultrasound GUI software and include Time Gain Compensation (TGC), overall gain control, scan depth, freeze display, screen capture and selection of transducer type. An image of the GUI can be seen rendering a portion of the 3D image volume in Figure 2.2.


Transducer: Convex 1

Scan Preset: Default

Scan Depth: 12

TGC ☑ Enabled

Figure 2.2 - Graphical User Interface of Image Reslicing Software during the First Generation System Development [Banker].

2.1.2) Initial Second Generation Development of Virtual Training System

Christian Banker's work on developing the first generation ultrasound training system was continued by Jason Kutarnia. Jason focused on the development of a second generation of the ultrasound training system that was less expensive and could also be portable for trainees to use at their leisure. This research utilized the imaging software from Chris Banker's ultrasound simulation and integrated an easily transportable planar surface. A systems diagram of the system can be seen in Figure 2.3.

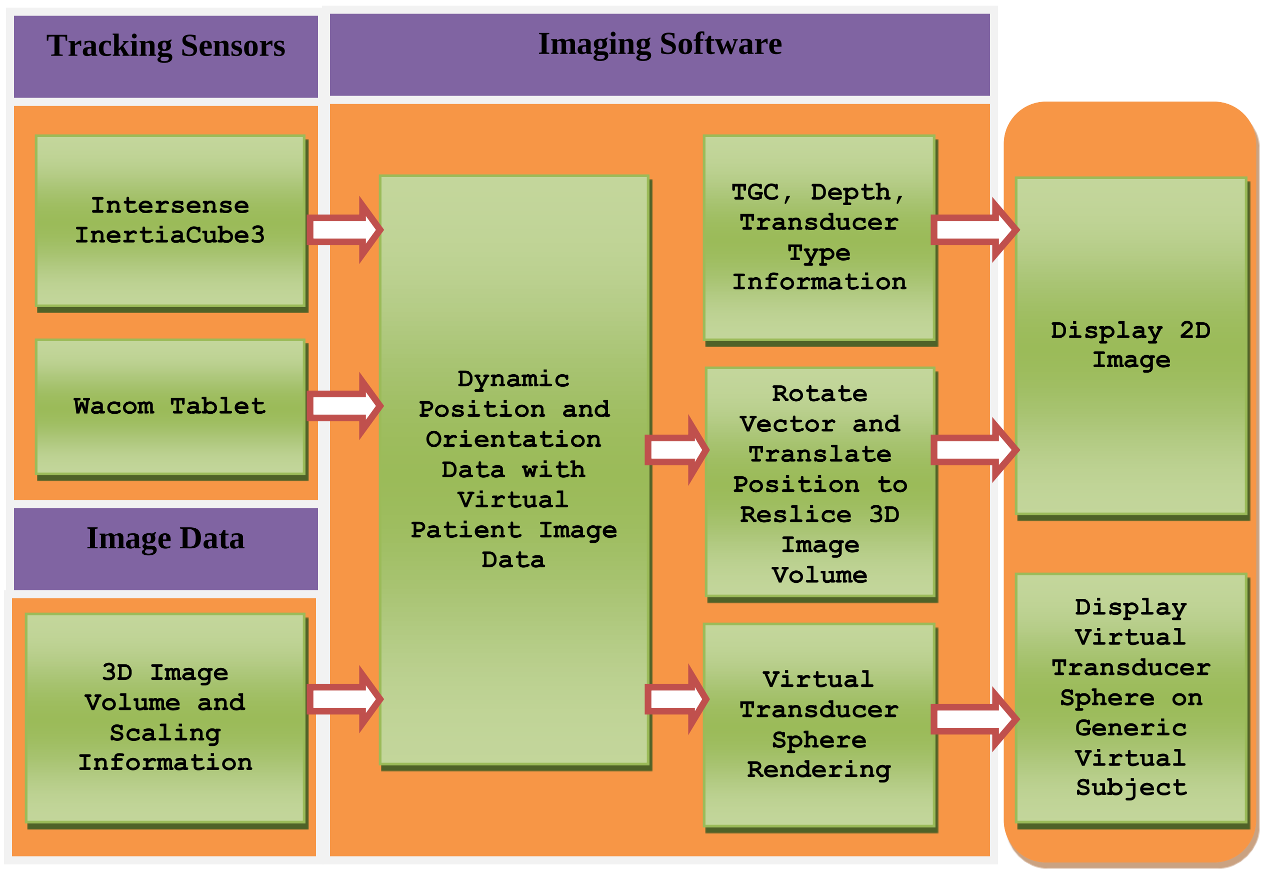

Tracking Sensors
Imaging Software
Intersense InertiaCube3
Wacom Tablet
Image Data
3D Image Volume and Scaling Information
Dynamic Position and Orientation Data with Virtual Patient Image Data
TGC, Depth, Transducer Type Information
Rotate Vector and Translate Position to Reslice 3D Image Volume
Virtual Transducer Sphere Rendering
Display 2D Image
Display Virtual Transducer Sphere on Generic Virtual Subject

Figure 2.3– Systems Diagram of Virtual Ultrasound System [Kutarnia].

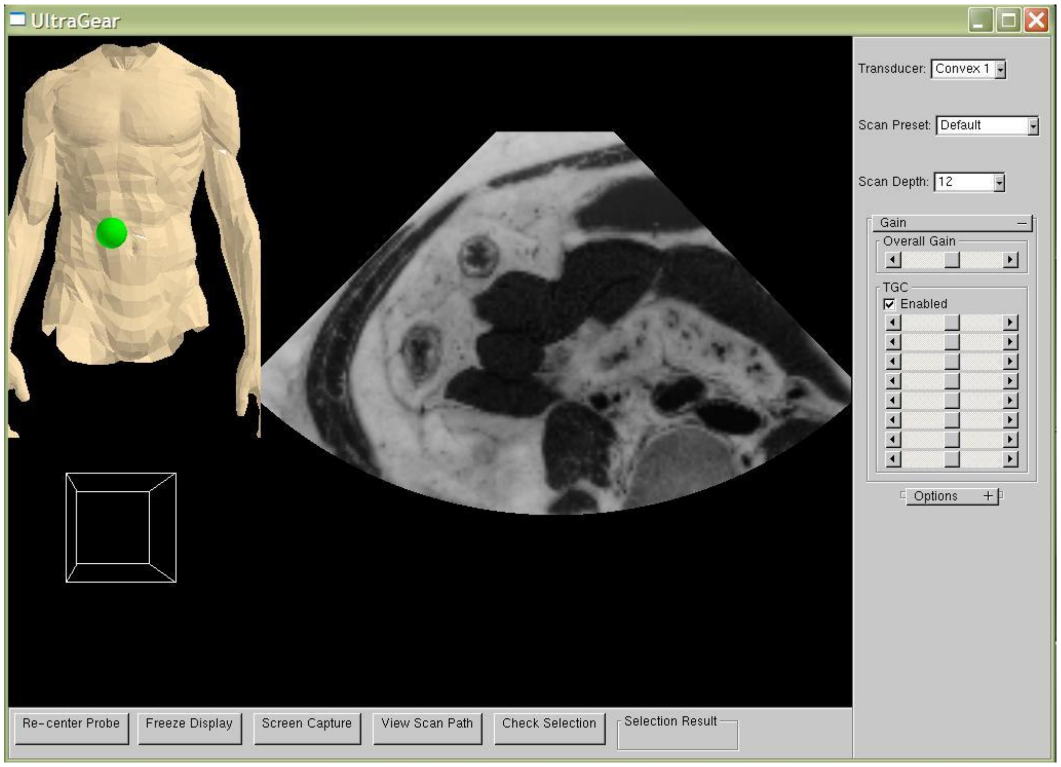
The initial hardware used included an electronic Wacom tablet. The Wacom tablet is an electronic device used for real-time translation of writing on the tablet surface to digital format on a PC. When a specific electromagnetic pen is applied to the surface of the tablet, the position of the pen tip can be tracked. The Wacom pen was integrated into the shell of an ultrasound transducer in order to track the sham transducer's movements relative to the tablet surface. The tablet connected to a PC through a USB connection for both power and data purposes. The tablet was also supplied with an API to allow for code development. The API allowed for the querying of information such as if the pen is present on the tablet surface and the position of the pen tip on the surface of the tablet.

An additional hardware component integrated into the sham transducer was an Intersense InertiaCube 3 Inertial Measurement Unit (IMU). The IMU contained three gyroscopes, three accelerometers, and three magnetometers [Intersense]. The data from these sensors are fused using a Kalman filter algorithm in order to produce accurate orientation data [IC3 datasheet]. The IMU had been attached to the exterior of the transducer shell in order to track the orientation of the transducer. The Intersense IMU also provided an API for code development. This allowed for the integration of the transducer orientation data with the imaging software. Due to the gravitational and magnetic references of the IMU sensors, a reorientation function was needed in order to properly align the physical transducer orientation with that of the virtual transducer. The Intersense API has a reorientation function that was used in this version of the image reslicing software.

A contribution to the software was the addition of a virtual patient and sphere representing the transducer rendered on the PC screen. The transducer sphere was created by using an OpenGL primitive sphere shape. The location of the virtual transducer sphere moves in accordance to the position movement of the physical transducer. The virtual patient added realism to the simulation for the aided learning of the user. A screen shot of the GUI with the virtual patient and transducer sphere can be seen rendering a portion of the 3D image data in Figure 2.4.

“The virtual subject is created from a triangular mesh that is loaded at the same time as the image volume” [Kutarnia]. The virtual patient used is a generic human model. The position data from the Wacom tablet was scaled appropriately for different types of anatomical data used. This allowed the fitting of the data to the generic virtual patient. Some of the anatomical data used for this system was “flattened” in order to produce a more anatomically correct display while using position and orientation data from a planar surface. A surface mesh of a data set used in this implementation can be seen in Figure 2.5.

The *Software Library for Image Detection* (SOLID) allowed for the virtual transducer to smoothly track along the surface of the virtual patient. SOLID was also used to ensure that the transducer did not penetrate the surface of the virtual subject.



Transducer: Convex 1

Scan Preset: Default

Scan Depth: 12

Figure 2.4 - Graphical User Interface of Image Reslicing Software during the Initial Phases of the Second Generation System Development [Kutarnia].

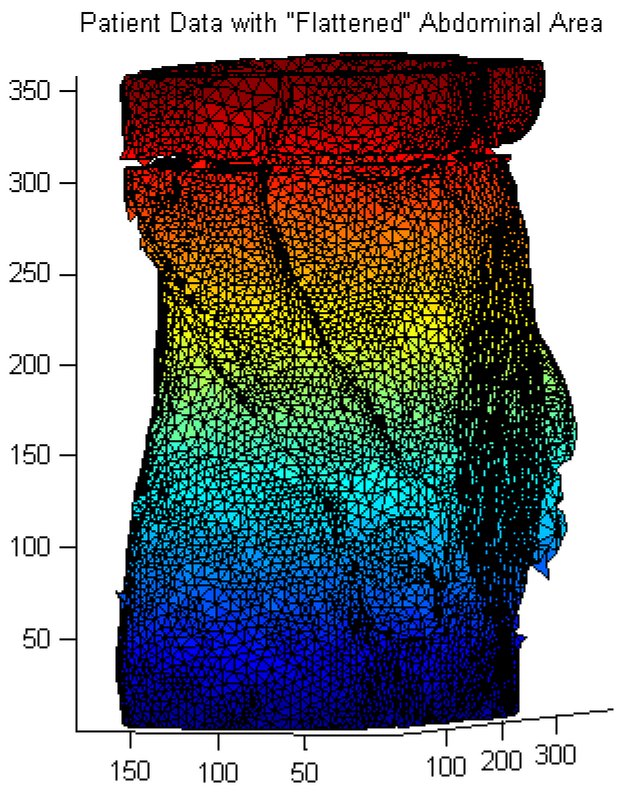


Figure 2.5 – Surface Mesh Generation of Patient Data with the Abdominal Area Flattened

2.1.3) Strengths and Weaknesses of Previous System Development

The first generation system developed by Banker had several strengths. One strength of the system was that it accomplished the main goal of creating a realistic training experience for the user. The system had a display and feel that was similar to many ultrasound scanners and the sham transducer had the look and feel of a real transducer. The system also used a manikin for an enhanced realism as well. The position and orientation tracking of the sham transducer was very accurate. The system also began development of outcome assessment tools.

There were several areas of the first generation system that could be improved upon or changed, which is what inspired the second generation system. The system was composed of several large pieces that were inconvenient to transport. Another issue that was considered after the first generation system was created was the cost. The system had several pieces of hardware, with a total component cost on the order of $6000 plus the cost of a PC. This high cost is one aspect that required some attention when designing the next generation system.

The initial development of the second generation system had several improvements compared to the first generation. The main advantage of the second generation system was its portability. One could now transport a small flat scan surface and physical transducer to use at their convenience. This convenience is in contrast to the first generation system requiring a manikin and additional hardware for tracking the user's transducer movement.

The system now had minimal hardware by being adaptable to any PC, which are commonly available. This utilization of a PC instead of application specific hardware not only allowed the system to be easily portable, but also reduced the cost of the system. Since the cost of any system will always be an issue, the lowering of the cost was greatly beneficial to the system as well.

The initial development of the second generation system had its own set of weaknesses as well. One weakness was the replacement of the lifelike manikin with the planar tablet as a scan surface. This replacement removed a great deal of realism from the scanning portion of the system. This weakness was partially compensated for, however, by a virtual patient on the screen of the image reslicing software that gave the user a sense of where the transducer was on the patient they were examining.

Although the virtual patient that was added to the initial development of the second generation system aided in the user's experience, the virtual patient was a generic model. The first generation's manikin model was generic as well. When a generic virtual model is used, the scanning of the patient does not relate directly to the image data displayed. This is a weakness both developments suffered from.

2.2) Design of Second Generation System

In this section, the reasons for designing a new ultrasound training system will be discussed. This new design will be approached by first identifying the shortcomings of the previous system. Once the shortcomings are identified, improvements for a second generation system will be described. This section will conclude with a systems description of the improved second generation system.

2.2.1) Shortcomings of the initial second generation system development

Beginning with the hardware of the initial development of the second generation system, the Wacom tablet used for position tracking is a planar surface. The human body however is a curved surface with varying contours. Creating a curved scan surface will be more desirable by giving the user a more realistic training experience.

Another shortcoming with the hardware is with the InertiaCube3, the Inertial Measurement Unit (IMU) used for the sham transducer orientation tracking of the user. Although this device gives more than adequate performance for this application, it comes with a price tag of over $2000. A different IMU solution that can provide adequate accuracy for a lower cost would be desirable.

Moving on to the shortcomings of the software, there is no surface rendering of the virtual subject that corresponds to the image volume used to produce the main display for the user. A goal of this training system is to give the user a realistic and accurate experience. Having a virtual patient that correctly corresponds to the data shown would be of great benefit to the user.

Another obvious software shortcoming is the rendering of a virtual transducer. In the initial development, a virtual transducer is represented by a green sphere that is positioned in an approximate area of where the user should be. In order to add more realism to the experience, rendering a more realistic virtual transducer that reorients and repositions according to the user's movements of the sham transducer on the generic scan surface would be highly desirable.

2.2.2) Second generation system improvements

As mentioned in the previous section, there are several areas in need of improvement with the initial development of the second generation system. These areas, discussed below, are given the most consideration in the further development of the second generation system. The first improvement that is made is the method of scanning the surface used for position tracking. The further development of the second generation used Anoto scanning technology that makes use of a mathematically unique dot pattern and optical IR sensor. The Anoto scanning technology allows for a pattern to be applied to a surface with curvature. This scanning technology will allow for a scan surface design more similar to a human torso than a planar surface used in the initial development.

The other hardware improvement for the further development of the second generation system is replacing the IMU with a less expensive solution. The IMU used is the PNI Spacepoint Fusion. The Spacepoint Fusion offers the necessary accuracy for $100 [Pedersen], greatly reducing the cost of the system. Results from a performance analysis of the Spacepoint Fusion can be found in Appendix A.

One software improvement that will be made is the rendering of a virtual patient that is modeled after the body surface associated with image data used to create the ultrasound display. This will be done by preprocessing the data to generate a mesh to be loaded with the image data into the image reslicing software. This allows for the user to observe a more accurate representation of the patient they are examining.

The last main improvement for the further development of the second generation system is the rendering of a more realistic virtual transducer. The improved virtual transducer is more realistic in appearance as well as reorient and reposition by means of accurately mapping the user's movements to the image data. By using this more realistic virtual transducer representation and mapping, the user will be presented with a more accurate understanding of the situation.

2.2.3) Systems description of improved second generation system

This section will outline the software, hardware, and interface challenges presented by implementing the discussed improvements to the second generation system. This process will begin by developing an Anoto system to track the sham transducer's position coordinates on the scan surface. The new scan surface for the system must be designed such that it is a more realistic human torso, allowing for a better experience for the user. The scan surface must also be designed such that the coordinates of the Anoto system will map well to the virtual patient data.

The PNI Spacepoint Fusion Inertial Measurement Unit (IMU), chosen based on excellent performance for this application at a more affordable price, has several differences from its InertiaCube3 predecessor. One difference is that the Spacepoint Fusion provides a quaternion output instead of Euler angles. The quaternion form of orientation representation is more convenient for rotations that will be required in the improved second generation system image reslicing software. This orientation representation, however, is more complicated than Euler angles, and the previous versions of the image reslicing software did not use them.

Another difference between the Spacepoint Fusion and the InertiaCube3 is that the Spacepoint Fusion does not have a reorienting feature. The Spacepoint Fusion IMU always reports orientations in reference to magnetic north. In order to use the Spacepoint Fusion a reorienting feature must be implemented into the image reslicing software. Without any form of reorientation, the physical transducer and scan surface coordinate systems will not be aligned.

The coordinate transformations from the new more realistic scan surface coordinate system to the patient data must be taken into account. The design of the improved scan surface, as well as the data, is no longer a planar surface. The new curved scan surface must be mapped to the data with minimal error. An algorithm must be designed to properly perform this coordinate transformation. The algorithm must transform the scan surface position data and IMU orientation data to the virtual patient data within the image reslicing software with 5 Degrees of Freedom (DoF). The sixth DoF that is absent in the system is the pressure applied to the surface.

The virtual patient that appears on the system screen also must no longer be generic. The virtual patient that appears must be unique for each image volume. Each virtual patient must also be representative of the data contained within the image volume.

Finally, a realistic virtual transducer must be created for the image reslicing software. The virtual transducer is to be moved on the virtual patient, controlled by the position and orientation of the sham transducer with 5 DoF. The control is intended to give the user an experience that is similar, if not exact, to the use of a real ultrasound transducer. A systems diagram of the improved second generation system can be seen in Figure 2.6.

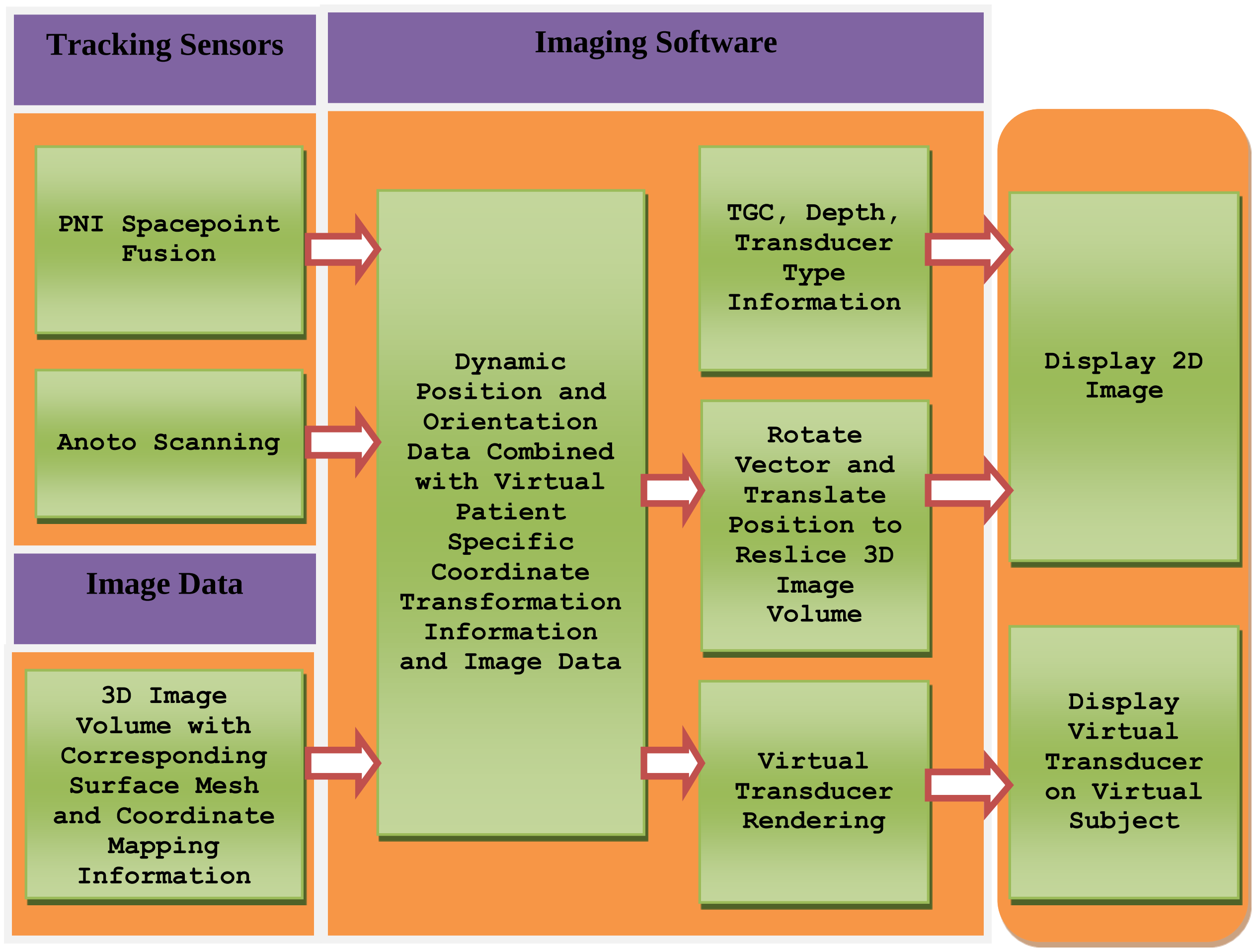

Tracking Sensors
Imaging Software
PNI Spacepoint Fusion
Anoto Scanning
Image Data
3D Image Volume with Corresponding Surface Mesh and Coordinate Mapping Information
Dynamic Position and Orientation Data Combined with Virtual Patient Specific Coordinate Transformation Information and Image Data
TGC, Depth, Transducer Type Information
Rotate Vector and Translate Position to Reslice 3D Image Volume
Virtual Transducer Rendering
Display 2D Image
Display Virtual Transducer on Virtual Subject

Figure 2.6 - Systems Diagram of Improved Second Generation System

2.3) Mathematical Concepts used in Second Generation System

This section will be used to give the reader a brief theoretical background to understand the design of the second generation system. Methods of orientation and rotation representations are extensively covered. As discussed in section 2.2.3, the scan surface is shaped as a cylindrical segment in order to be more representative of a human torso than a planar surface. In order to give a more realistic experience, the position mapping of the scan surface will now require the use curvature and the mathematics that accompany it. Also, as mentioned in the previous section, the orientation tracking will now use a more appropriate method for mapping rotations by use of quaternions. A section is dedicated to the fundamentals of the quaternion orientations and rotations. The information of both the curvature angles from the scan surface and the orientations from the Inertial Measurement Unit (IMU) will be required for the display of the virtual patient and calculating the image volume reslice.

2.3.1) Cylindrical Curvature

The process of designing the scan surface as well as transforming its coordinates into the virtual coordinate space required the knowledge of cylindrical curvature. The use of the Anoto technology on a cylindrical surface allows for the measurement of arc length. By utilizing this measurement and knowledge of other parameters of a cylindrical surface, several important calculations can be made.

A key concept used in this design uses the calculation of angular displacement from arc length and radius as seen in equation 2.1.

$$\theta = \frac{l}{r} \qquad 2.1$$

In this equation θ represents the angular displacement (radians), *l* represents the arc length, and *r* represents the radius of the curve. An example can be seen in Figure 2.7.

The use of a chord of an arc is also utilized. A chord may be calculated by using equation 2.2 where *r* represents the radius of the curve and θ represents the angular displacement.

$$chord = 2 * r * \sin(\frac{\theta}{2}) \tag{2.2}$$

These equations were key components in the design of the scan surface, combining orientation and position data, and for coordinate transformations as well. An example can be seen in Figure 2.7.

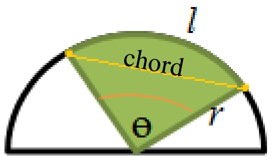


Figure 2.7 - Diagram of Cylinder Segment Base and Corresponding Measurements

2.3.2) Quaternions

Throughout the design a method of orientation representation called quaternions are used. A quaternion is a typical form of three dimensional rotations that consists of a scalar and 3D vector component. The information stored within a quaternion is an angle-axis pair. The vector component, defined by (x y z), contains the axis information that the rotation is to be revolved about. Each of the elements are a function of the angle rotation, θ, about the vector. Each of the quaternion elements contains information regarding the angle to be rotated. An example quaternion can be seen in equation 2.3 and Figure 2.8.

$$q = [w\ (\mathbf{x}\ \mathbf{y}\ \mathbf{z})] = \left[\cos\left(\frac{\theta}{2}\right)\ (\sin\left(\frac{\theta}{2}\right) n_x\ \sin\left(\frac{\theta}{2}\right) n_y\ \sin\left(\frac{\theta}{2}\right) n_z\ )\right] \tag{2.3}$$

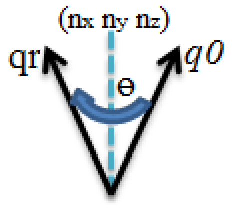


Figure 2.8 - Visual Example of a Quaternion Rotation from orientation *q0* reoriented by a quaternion containing theta and *(n_x n_y n_z)* to the new quaternion orientation *qr*.

When dealing with rotations, quaternions are not as intuitive as Euler angles. However, Euler angles have an inherit problem of Gimbal lock, which occurs when pitch is $\pm 90^o$. In this case a degree of freedom is lost because yaw and roll both rotate about the vertical axis. In a mathematical perspective, Gimbal lock occurs because of the map from Euler angles to rotations. This mapping is not a covering map. That is, it is not one-to-one mapping and therefore at some points the rank becomes less than three, at which point gimbal lock will occur. Euler angles provide a numerical description of any rotation in three dimensional space using three numbers, but not only is this description not unique, but there are also changes in rotation which cannot be represented by Euler angles.

Quaternions avoid Gimbal lock by using four terms to express the rotation [Dunn]. The fourth term used in quaternions allows for a complete mapping of all rotations. It is because of the avoidance of Gimbal lock and complete mapping of all rotations that all rotations involving the reorientation of the IMU are performed by using quaternions.

In order to apply a quaternion rotation to a quaternion orientation, one must multiply the two quaternions. The product of the two quaternions represents the new orientation. The process of multiplying two quaternions can be seen in equation 2.4. It is important to note that the magnitude of the product quaternion is equal to the product of the magnitudes of the quaternions multiplied to obtain this result. This magnitude product is the reasoning for having each quaternion magnitude used to be equal to one. This ensures that the resulting quaternion magnitudes will not decay or grow.

$$(w_1 + x_1 i + y_1 j + z_1 k) \times (w_2 + x_2 i + y_2 j + z_2 k) =$$

$$\begin{bmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - y_1 z_2 \\ + (w_1 x_2 + x_1 w_2 + z_1 y_2 - y_1 z_2) i \\ + (w_1 y_2 + y_1 w_2 + x_1 z_2 - z_1 x_2) j \\ + (w_1 z_2 + z_1 w_2 + y_1 x_2 - x_1 y_2) k \end{bmatrix} \qquad 2.4$$

Quaternions in this system are constructed from Euler angles by means of the following equations:

$$w = \cos\left(\frac{pitch}{2}\right) * \cos\left(\frac{roll}{2}\right) * \cos\left(-\frac{yaw}{2}\right) + \sin\left(\frac{pitch}{2}\right) * \sin\left(\frac{roll}{2}\right) * \sin\left(-\frac{yaw}{2}\right) \qquad 2.5$$

$$x = \cos\left(\frac{pitch}{2}\right) * \sin\left(\frac{roll}{2}\right) * \cos\left(-\frac{yaw}{2}\right) + \sin\left(\frac{pitch}{2}\right) * \cos\left(\frac{roll}{2}\right) * \sin\left(-\frac{yaw}{2}\right) \qquad 2.6$$

$$y = \sin\left(\frac{pitch}{2}\right) * \cos\left(\frac{roll}{2}\right) * \cos\left(-\frac{yaw}{2}\right) - \cos\left(\frac{pitch}{2}\right) * \sin\left(\frac{roll}{2}\right) * \sin\left(-\frac{yaw}{2}\right) \qquad 2.7$$

$$z = \cos\left(\frac{pitch}{2}\right) * \cos\left(\frac{roll}{2}\right) * \sin\left(-\frac{yaw}{2}\right) - \sin\left(\frac{pitch}{2}\right) * \sin\left(\frac{roll}{2}\right) * \cos\left(-\frac{yaw}{2}\right) \qquad 2.8$$

Euler angles are also constructed from quaternion data as seen in the following equations where q_0, q_1, q_2, q_3 are components of the quaternion [Heckbert]:

$$\text{roll} = \frac{180}{\pi} \times \arctan((2 * (q_0 * q_1 + q_2 * q_3)) / (q_0^2 - q_1^2 - q_2^2 + q_3^2)) \qquad 2.9$$

$$\text{pitch} = \frac{180}{\pi} \times \arcsin(-2 * (q_0 * q_2 - q_3 * q_3)) \qquad 2.10$$

$$\text{yaw} = \frac{180}{\pi} \times \arctan((2 * (q_1 * q_2 + q_0 * q_3)) / (-q_0^2 - q_1^2 + q_2^2 + q_3^2)) \qquad 2.11$$

3) New System Hardware and Firmware

3.1) Sensing Hardware for Position Tracking

New technology has become commercially available since the research of the first generation and initial development of the second generation of the ultrasound simulation training system. These technologies include new concepts for accomplishing similar tasks as wells as new manufacturing techniques that have led to reduced size and costs in comparison to previous sensors. The following sections discuss a new concept used for surface position tracking.

3.1.1) Anoto Technology

Anoto technology was originally created to be used with a digital pen in order to digitally record handwriting. This was done using a unique dot pattern on a sheet of paper and an IR camera plus IR light source within the pen that scans the portion of the dot pattern that the pen's tip was placed upon. Based on the portion of the Anoto dot pattern from the camera's scans, an absolute position on the dot pattern can be determined within 0.3/8 mm = 37.5µm [Anoto SPCD]. This technology with use of a pen scanning an Anoto pattern is portrayed in Figure 3.1.

This scanning technology, within the digital pen form, is dependent on a pressure sensor on the tip of the pen. When the pen is applied to a surface, the pressure sensor activates an IR light, IR camera, and the image processing unit. The IR light and camera are located within the pen near the tip. Once the scanning process is activated by the pressure sensor, images are taken by the IR camera. The pen then processes the images and applies an Anoto pattern algorithm. This algorithm determines if the pen has been applied to an Anoto pattern or not. If the pen determines that it is being applied to an Anoto pattern, the algorithm then determines the location of the portion of the pattern that has been scanned. The position data that the pen calculates are then formatted and transmitted to a PC. A driver on the PC then allows access to the data for applications. The process of an Anoto pen determining position data can be seen through a systems diagram in Figure 3.2.

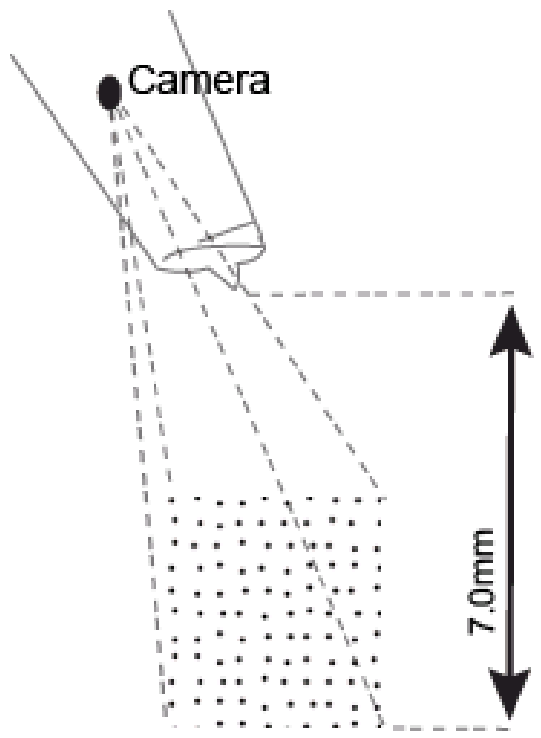


Figure 3.1 - Illustration of digital pen scanning an Anoto pattern [Anoto Functionality].

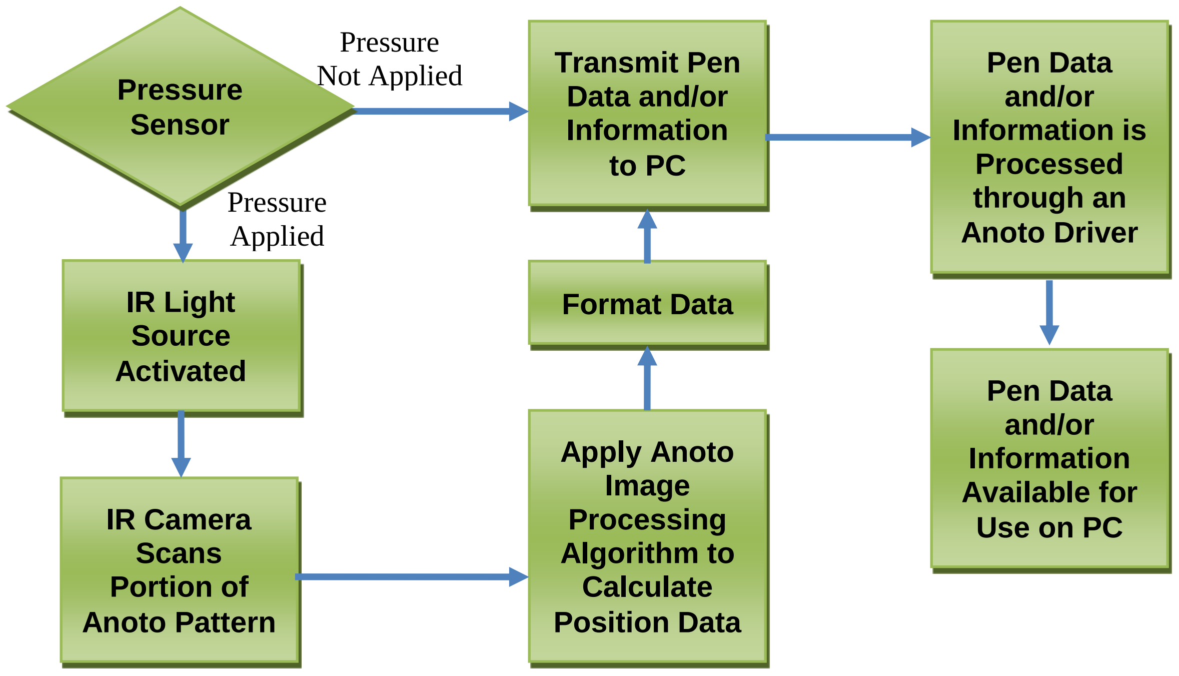

Pressure Sensor
Pressure Not Applied
Pressure Applied
IR Light Source Activated
IR Camera Scans Portion of Anoto Pattern
Apply Anoto Image Processing Algorithm to Calculate Position Data
Format Data
Transmit Pen Data and/or Information to PC
Pen Data and/or Information is Processed through an Anoto Driver
Pen Data and/or Information Available for Use on PC

Figure 3.2 – Systems Diagram of Anoto Scanning System

3.1.2) Anoto Pattern and Operating on a Curved Surface

Previous to this research, the Anoto pattern had been applied to planar surfaces, typically to track the handwriting on paper. For use in the ultrasound simulation research, the performance of an Anoto device positioning on a curved surface was of interest. The ability to have a device that can position itself on a curved surface allows for the creation of an ultrasound simulator with a scanning surface that is more representative of a human torso or other bodily area with curvature. Results from these tests can be visually seen in Appendix B.

Positioning on Curved Anoto Pattern

This section discusses the test methods used to determine the operability of the Anoto pattern and digital pen on a cylindrical surface with different radii of curvature. Three different curvatures were chosen for this experiment. Each of the experiments was conducted within the WPI Ultrasound Laboratory and performed with non-interfering illumination of the fluorescent lights from the ceiling of the laboratory.

Each test used one sheet of paper with the Anoto pattern printed on it. The paper was placed on the outer surface of a plastic cylinder. This placement created a consistent curvature of the Anoto pattern. The digital pen was held in a position normal to the surface of the cylinder. The pen was then used to draw a variety of lines, curves, and symbols on the paper. This process was performed using several cylinders, each cylinder having a different diameter. The diameters used were 6, 4, and 2 inches.

The overall appearances of the written portions were obtained correctly by the digital pen and can be seen in Appendix B. The digital pen was able to correctly locate its position on the particular page and its relative position on the page. However, the digital data obtained from each of the tests contain very small offsets from their corresponding written portions. This effect is more noticeable as the diameter of the cylinder is reduced. For the purposes of the ultrasound device, the location accuracy on the curves proved to obtain sufficient accuracy for the sham transducer surface positioning.

Pen Angle Relative to Surface Operability

This section discusses the test methods used to determine the operability of the Anoto pen when positioned at different angles from the surface. Several different angles were chosen for this experiment. Each of the experiments was conducted on a flat desk surface within the WPI Ultrasound Laboratory and performed with non-interfered illumination of the fluorescent lights from the ceiling of the laboratory.

Each test consisted of the pen user holding a protractor against the edge of the pen while on the writing surface. A constant angle was kept between the writing surface and the center of the diameter dimension of the pen. The pen was then used to draw a line. This process was repeated for several angles until the pen could no longer draw a legible line. The first section tested the pen angled in a way such that the infrared camera was below the extrusion of the pen tip from the base of the pen. The second section tested the pen angled in a way such that the infrared camera was above the extrusion of the pen tip from the base of the pen.

For each of the angle tests, the pen was able to draw a line until the pen was angled approximately 55 degrees away from normal. When the pen was angled more than 55 degrees, a clear line could not be drawn due to the pressure sensor of the pen unable to be triggered to turn on the infrared camera. For each of the tests that could successfully draw, the pen was able to accurately identify where it had drawn a mark. This indicates that the pen should be successful in detecting positions on an Anoto pattern when angle from 0 to 55 degrees relative to surface normal.

3.1.3) Anoto Firmware

The Anoto pen operates using firmware that has been preloaded onto the pen during production. The firmware is responsible for the IR camera scanning, position calculating, communication, and other pen functions. The scanning process is activated by the pressure sensor on the tip of the pen. Once the scanning process is activated, the pen will periodically scan and calculate the tip’s relative position on the Anoto pattern. Once the pen has calculated or sensed new information, the pen will transmit data reporting the information.

3.1.4) Anoto API

The pen's firmware can be interfaced using the Anoto API. The API consists of a dynamic-link library (DLL) file and the Streaming Pen Connectivity Driver (SPCD). The SPCD exposes "COM interfaces that can be used from all major development environments on Windows" [Anoto SPCD]. The SPCD also processes the Human Interface Device (HID) protocol data that are transmitted from Anoto pen.

The Anoto DLL file contains a COM object named *AnotoGenericStreamer*. This COM object contains two interfaces. One interface, `IPenManager`, can be used to send commands to the pen. These commands can be used to tell the pen to start and stop its scanning procedure. The other interface, `_IPenManagerEvents`, can be used to handle interrupt signals sent from the pen. The function subscriptions to pen events allow for the receiving of data such as the pen tip's relative position and if the pen tip sensor has been placed on or taken off the pattern.

3.2) Integration of New Position Tracking Hardware

There were two main components to integrating the Anoto technology into the image reslicing software. One component was updating the existing image reslicing software to be compatible with the Anoto API. The other component was the code integration of the Anoto API with the image reslicing software. The following sections discuss the steps that had to be taken in order to integrate the new hardware into the image reslicing software.

3.2.1) Ultrasound Imaging Software Updates

After the research of Jason Kutarnia the image reslicing software was supplied as a Visual C++ project. The project, as well as its dependencies, had been created and built using Microsoft Visual Studio 2003. The project was also compiled with the /MT compiler option and without CLR support. The project had also lacked source code for some of the dependencies. Without the source code for the dependencies, no changes could be made to them.

Visual Studio Versions

All versions of Microsoft Visual Studio contain precompiled functions to be used at run-time. In Visual Studio 2005 and later, some of these precompiled functions that have the same name and perform a similar operation as past functions now have a different function declaration. Therefore, in versions of Visual Studio before 2005 (i.e. Microsoft Visual Studio 2003), functions with identical names and functions as those in the later versions will not be linked correctly due to this difference. Compiling a project in Visual Studio 2005 or later that has dependencies on files that were compiled in Versions of Visual Studio 2003 or earlier will result in linker errors. The common use of *Xran* and *Xlen* functions in particular are responsible for most of these errors.

The Anoto API was compiled using Microsoft Visual Studio 2005 and required the use of COM objects with the .NET framework. Because the Anoto API DLL file came precompiled without access to the source code to recompile it, the DLL file was not compatible with the image reslicing Visual Studio 2003 project. This incompatibility and the inability to recompile the Anoto DLL file led to the updating of the image reslicing software.

Dependencies

The image reslicing software depended on a library called *Gear*. Gear is a C++ OpenGL based library that was used to render images using alpha compositing. Alpha compositing is a method of blending the color data from each selected data point corresponding to the pixels on the screen. The source code of Gear in a Visual Studio 2003 project was available with the image reslicing software. When a version of Visual Studio encounters a project that has been created in a previous version of Visual Studio, a Microsoft conversion process will be performed in order to update the project. However, when the Gear project was opened in versions of Visual Studio later than 2003, the conversion process failed due to its dependencies on code that was no longer supported. There was also an unfinished version of Gear source code in a Visual Studio 2008 project that came with the image reslicing software. This version did not compile and was far from working.

It was during this period that other image renderers were considered for replacing Gear. Among those that were explored was Voreen [Voreen]. Voreen utilized modern graphics cards by applying rendering techniques that required more computation. It was eventually decided that these forms of volume renderers would not suit the needs of the image reslicing software. The methods of rendering used are too computationally intensive to run in the real-time nature that the reslicing software requires.

During the researching of different volume renderers, the author of Gear, Jason Bryan, was contacted. Jason had a private version of the Gear library that compiles and runs using Visual Studio 2008. This newer version however, used code that was owned by Jason's employer. Jason was able to supply the updated working version of Gear with certain portions missing. However the missing code was not required for the use of the image reslicing software. After the removal and altering of references to the missing code, a Visual Studio 2008 version of Gear was available for use.

The source codes of other dependencies were also acquired at this time. One such dependency was the SOLID library. The SOLID library is used for the collision detection between the virtual transducer and patient. Another dependency was the OpenGL Utility (GLUI). This dependency is required for the User Interface (UI) controls of the image reslicing software.

Compiler Settings

After the completion of obtaining the source code for all the dependencies, the project was successfully recompiled in Visual Studio 2008 producing a new updated working version of the image rendering software. However, the image rendering software along with its dependencies had all been using the /MT compiler option. The /MT compiler option enables a static version of the run-time library and forces the linker to use the LIBCMT.lib to resolve external symbols [msdn]. In order to use the Anoto DLL file, the image rendering software and all of the dependencies needed to be recompiled using the /MDd compiler option. The /MDd compiler option enables a DLL-specific version of the run-time library and forces the linker to use the MSVCMTD.lib to resolve external symbols [msdn].

The Anoto API also required the use of COM interoperability. In order to support the use of COM, the project must have *Common Language Runtime* (CLR) support. CLR support is achieved by adding the /CLR compiler option to the project.

The Intersense InertiaCube 3 had been interfaced by means of static DLL interfacing. Microsoft Visual Studio versions presently do not allow the use of static DLL interfacing while the /MDd compiler option is enabled. The InertiaCube 3 API also used C code that ironically the "common language" runtime does not support. The lack of support for static DLL interfacing and the use of C code were additional reasons (beside cost) that led to the removal of the InertiaCube 3. Although the code that interfaces with the InertiaCube3 could be rewritten, it was not merited due to far cheaper, and possibly better performing, alternatives that would be used in its place.

3.2.2) Anoto API integration into Image Reslicing Software

During the initializing phase of the Image Reslicing software, an object, named *AnotoPen*, was created to manage the communications with the Anoto pen. Upon creation of this object, functions were subscribed to handle pen events proceeded by a command sent to the pen to begin its scanning procedure. Upon the destruction of this object, a signal to the pen to stop its scanning procedure is sent. This object's methods and their role in the communication between the Anoto firmware and the image reslicing software are illustrated in Figure 3.3.

One of the functions that was used in this process was *OnNewCoord()*. *OnNewCoord* was subscribed to the event *NewCoord*, which is transmitted every time the pen has calculated the tip's current position on the Anoto pattern. *OnNewCoord* receives the page and (x,y) coordinates, in Anoto units, that correspond to the portion of the pattern that the pen tip is located. Once the (x,y) coordinates are obtained, scaling factors are applied to convert Anoto units into coordinates that the image reslicing software can use to translate the patient image for rendering.

The coordinates produced by the Anoto pen are in Anoto units. One Anoto unit is equivalent to (0.3/8) mm = 37.5 µm, and one 8.5 by 11 sheet of paper with the Anoto pattern has been used for testing purposes. The dimension of the pattern printed on the sheet is approximately 5500 by 7700 Anoto units. The image reslicing software has been configured to take coordinates (x_{tab}, y_{tab}) from the Wacom tablet. The coordinates that the tablet produces are float values ranging from 0 to 1. The coordinates that are returned from the Anoto pen are scaled by these values as seen in equations 3.1 and 3.2.

$$x_{tab} = \frac{Anoto_x}{w} \qquad 3.1$$

$$y_{tab} = \frac{Anoto_y}{l} \qquad 3.2$$

w being the width and *l* being the length of the Anoto pattern. This scaling allows for the coordinate input of the Wacom tablet to be replaced by that of the Anoto pen.

Other functions that are used in this process are *OnPenUp()* and *OnPenDown()*. *OnPenUp* is subscribed to the event *PenUp*, which is transmitted every time the pressure on the pen tip sensor has been released. *OnPenDown* was subscribed to the event *PenDown*, which is transmitted when pressure on the pen tip sensor has first been applied. Each of these functions controlled the values of a Boolean variable called *draw_trans*. The variable *draw_trans* is used in the image reslicing software to control the rendering. The communication process between the Anoto firmware and image reslicing software is further illustrated in Figure 3.3. These functions allow for the image to render only when the pen tip is placed on the Anoto pattern. This functionality is similar to an ultrasound image of a patient only being produced when the transducer is placed on a patient.

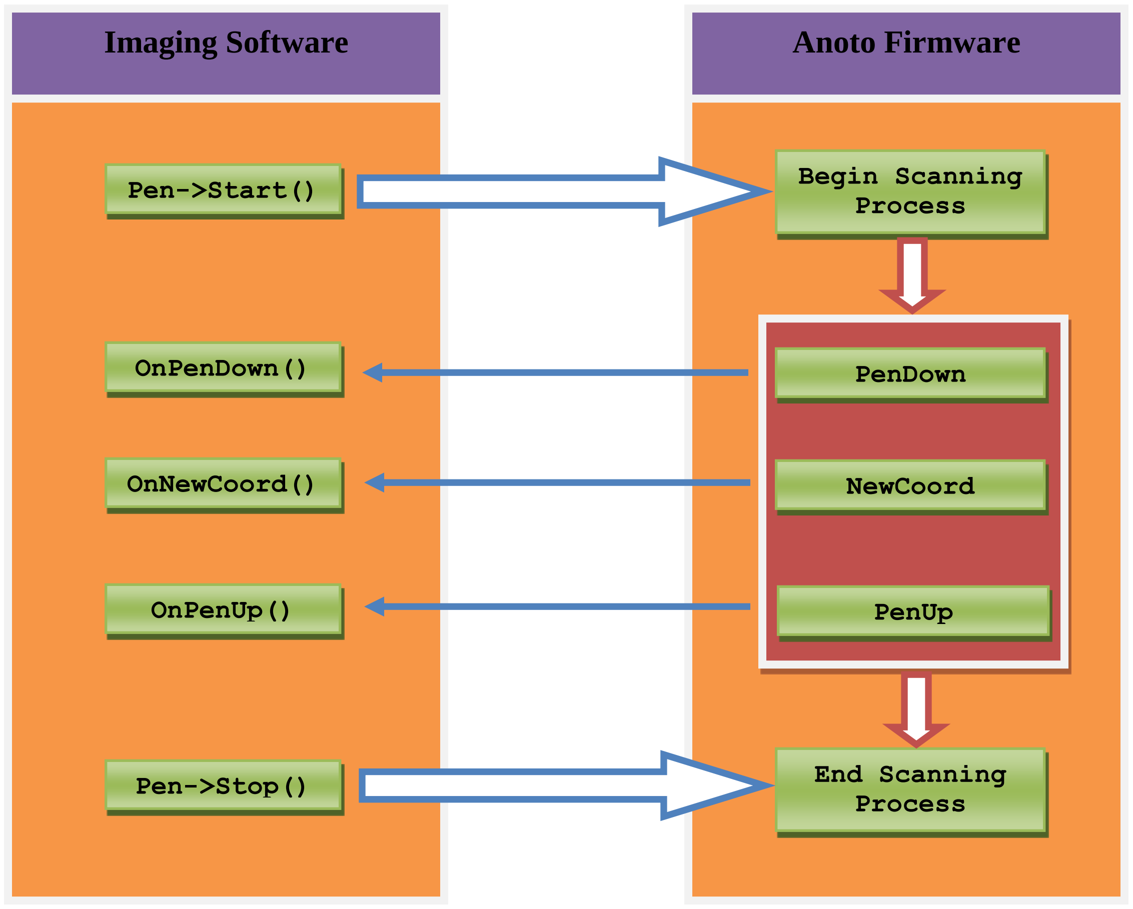


Figure 3.3 - Diagram of high level communications between the image reslicing software and Anoto pen firmware.

3.2.3) Utilizing Multiple Sections of the Anoto Pattern

The Anoto API is packaged with sections of the Anoto pattern in postscript format. Each section of the pattern has a corresponding label that resembles an IP address. The pattern section label information is sent by the Anoto firmware when a new coordinate is sent. This information arrives and is ready for use after each *OnNewCoord* instance. This section label allows for multiple portions of the Anoto pattern to be used within the same application. As will be discussed later in this research, a generic scan surface for ultrasound simulation will typically be larger than one section of the pattern. The allowance of multiple sections to be used in one application yields the ability to use multiple sections to ensure the full coverage of a generic scan surface with the Anoto pattern.

3.3) Inertial Measurement Unit for Orientation Tracking

The PNI SpacePoint Fusion sensor module (PNI Fusion) is a motion tracking device that consists of three sets of MEMS accelerometers, gyroscopes, and magnetic sensors. The module contains three of each type of MEMS sensor. “These 9 outputs are internally processed using PNI’s SpacePoint algorithm to provide calculated orientation data in the form of quaternions”[SpacePoint Fusion]. The internal processing of the module fuses the data of the sensors in order to provide greater accuracy than any one of the sensors could provide as well as compensates for sensor weaknesses such as gyroscope drift.

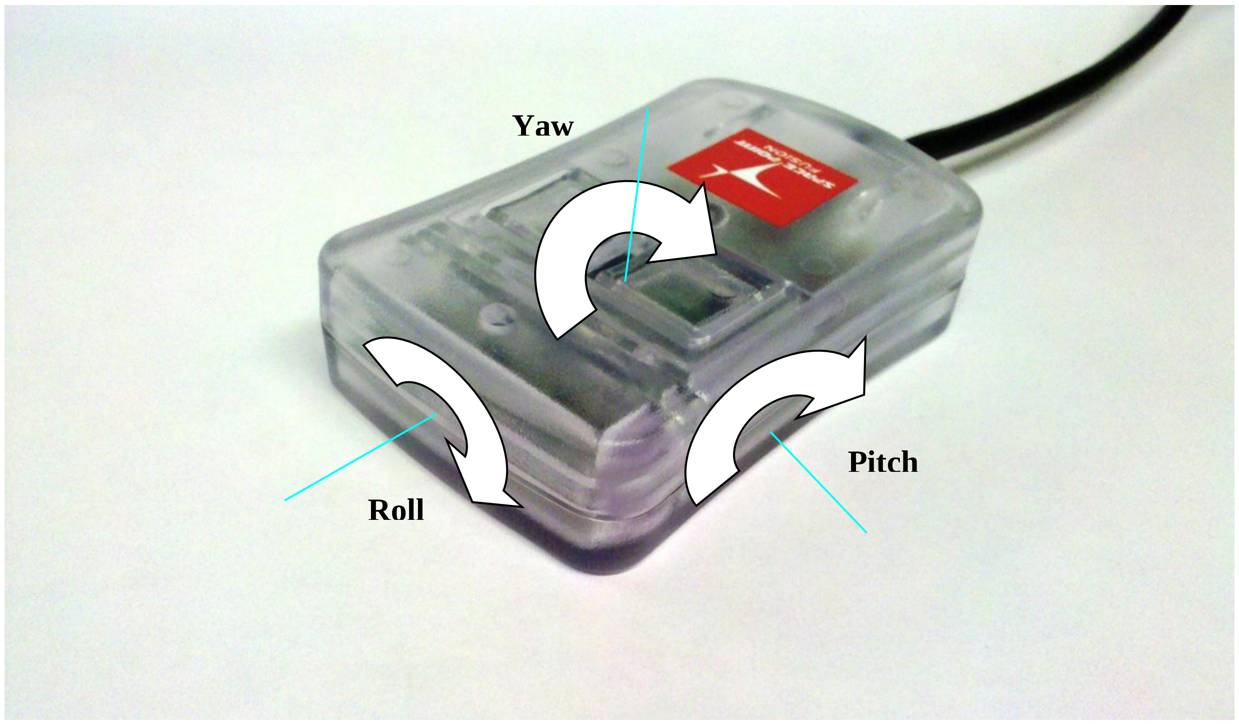


Figure 3.4 - A PNI SpacePoint Fusion sensor module with axis and rotation labels.

3.3.1) Configuration and USB API

The sensor was connected to a PC running the image reslicing software via a USB connection. The PNI Fusion module must first be placed on the motionless surface to allow for calibration. Upon power-up and while the sensor is at rest, the PNI Fusion module requires five seconds to perform its auto-calibration phase [SpacePoint Fusion].

The image reslicing software uses a simple USB Human Interface Device (HID) API to interface with the module. The API provided by PNI consists of a Dynamic-Link Library (DLL) and header file.

The DLL provides the client application an easy method for accessing an HID device via USB link. The acts of reading and writing a USB device under Windows are significantly different and more involved than for other comm devices such as serial comm ports. For this reason, PNI felt it necessary to encapsulate the complexity of the interface within a DLL. This DLL provides all the required functions for accessing the device. The USB link is implemented as a (HID) class function. As such, the DLL is dependent on the following Windows hidclass.sys, hidparse.sys, and hidusb.sys drivers.

The image reslicing software links implicitly with the DLL. By implicitly linking, the application links to an import library (UsbHidApi.lib) and makes function calls just as if the functions were contained within the executable. Two such functions used are the *Open* and *Read*.

Open opens a USB comm link to a HID device. The software opens a specific HID device by providing specifiers for the vendor ID and product ID. If successful, the open function returns "true" and otherwise "false." The open function is used within the *PNIinit()* function called upon the startup of the image reslicing software. Within the *PNIinit()* function, the appropriate checks for *Open*'s success are made. A message is printed in a command prompt upon *Open*'s success or failure.

Read reads from the HID device. The number of bytes read is determined by the input report length specified by the PNI device. The device was opened using a "true" asynchronous parameter, therefore non-blocking reads are performed. On success, the call returns the number of bytes actually read. A negative return value indicates an error. If the number of bytes read is greater than zero the data read are then parsed and used for 3D image volume manipulation.

3.3.2) Parsing of Data

The data are parsed using the *readindata()* function. The function requires three inputs. The inputs are: (*i*) a pointer to the current angles used for 3D image manipulation, (*ii*) the received data, and (*iii*) the number of bytes read. The data read are passed into the function as the variable *indata*. The variable *indata* is then parsed into an array of integer variables named *rawaxes[]*. This parsing is done by taking the appropriate two bytes corresponding to each variable, multiplying the upper byte by 256 (2^8 bits = 256), and finally adding the bytes together to yield a raw data reading for a corresponding raw axes output. This process can be seen in equation 3.3.

$$RawAxes[i] = Least\ Significant\ Byte[i] + 256 \times Most\ Significant\ Byte[i] \qquad 3.3$$

The image reslicing software uses four *rawaxes[]* values corresponding to a unit quaternion. These values are then properly offset by half of the range of the data (32768) in order to properly center the data. The data from the *rawaxes[]* are then multiplied by a scale factor to produce the quaternion values. This process can be seen in equation 3.4.

$$Quaternion\ Element[i] = ScaleFactor \times (RawAxes[i] - 32768) \qquad 3.4$$

3.3.3) Reorient Sensor Heading

By default, the initial transformation used to orient the IMU is that of quaternion *q* seen in equation 3.5. This quaternion rotates PNI sensor module pitch by 90 degrees. This reorientation is performed in order to align the pitch rotation of the PNI sensor to that of the image reslicing software.

$$q = \begin{bmatrix} \sqrt{2}/2 & 0 \\ 0 & \sqrt{2}/2 \end{bmatrix} \qquad 3.5$$

The PNI sensor module uses magnetic sensors. The use of magnetic sensors yields an orientation in reference to magnetic north. Clearly, it is not in the best interest of the user to rotate the training system to magnetic north in order to operate it. In order to eliminate the magnetic orientation annoyance, a reorientation procedure was designed.

For the user, this reorientation procedure is to simply hold the physical sham transducer in an upright position to the side of the scan surface. Hold the thin side of the transducer perpendicular to the scan surface (y-axis) and the IMU facing the user, as seen in Figure 3.5. Once the transducer is in the proper position, the user must press the ‘r’ key on the keyboard of the PC running the image reslicing software.

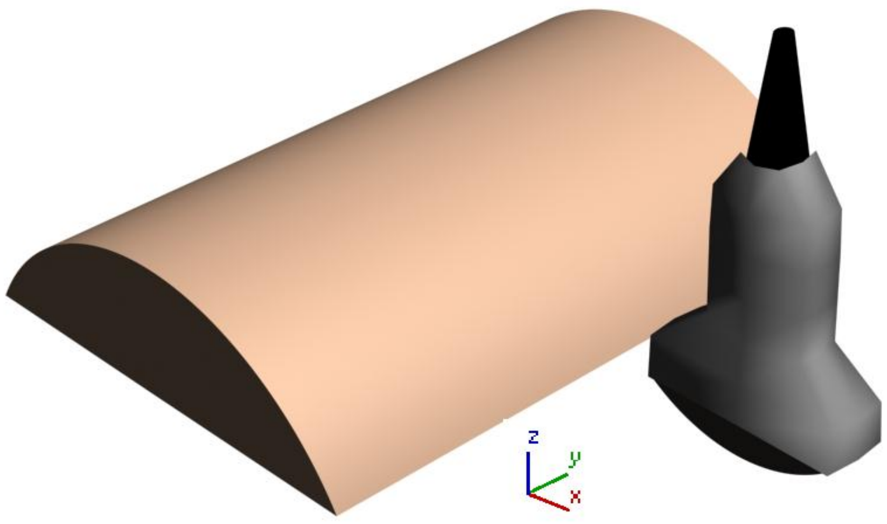


Figure 3.5 – Diagram of Hardware Placement when Performing the Reorientation Procedure

When the ‘r’ key is pressed, a flag is set to get a new heading during the next set of data read from the PNI sensor. The true magnetic north heading offset is stored in a variable passed to the *readindata* function called *zeroheading*. When the reorientation procedure has been called, the current yaw heading data are added to *zeroheading* as seen in equation 3.6. The quaternion orientation reported from *readindata* function has already been multiplied by a reorienting quaternion. Therefore the yaw heading reported will be in reference to the reorientation. This variable is needed in order to save the true offset from magnetic north such that the new reorienting quaternion is properly offset from true magnetic north.

$$zeroheading_{new} = zeroheading_{old} + \ current\ yaw\ reading \quad 3.6$$

The reorientation procedure then constructs a new quaternion to reorient the PNI sensor data based on the zero heading and 90 degree pitch offset. The new reorientation quaternion is formed using the following set of Euler angles:

$$yaw = 0\ , pitch = \frac{\pi}{2}, roll = zeroheading + current\ yaw\ reading \quad 3.7$$

The new quaternion calculated is then used to reorient all future data read from the PNI sensor unless another reorientation procedure is performed. At the end of the reorientation procedure the reorientation procedure flag is reset.

3.3.4) Integrate Data into Image Reslicing Software

The program then converts the reoriented unit quaternion output data of the sensor to Euler angles. The orientation conversions were done with equations 2.9, 2.10, and 2.11. The conversion is performed in order to supply the image reslicing software rotation functions with the necessary data to calculate transformations needed to appropriately rotate the image volume. These transformations will be discussed in section 4.3. Also the direction of the pitch angle is opposite of that desired for the image reslicing software. It is because of this misdirection the pitch angle is negated before being used in the process to rotate the image volume.

4) New System Software

4.1) Virtual Patient Data

This second generation system requires 3D image data representing an actual patient. This image data can come from a variety of sources. For means of demonstration of the functionality of the system, image data from the Visible Human Project was selected. The Visible Human Project is a project overseen by the United States National Library of Medicine. The project has promoted the creation of complete, anatomically detailed, three-dimensional representations of the human body [National Library of Medicine].

The image data obtained is quite detailed and uses a large amount of memory (approximately 40GB) to contain. This very large image volume would be unacceptable to use for both computational reasons and memory usage on a user's PC. For these reasons the image volume has been reduced in size to a more memory and computationally efficient size (approximately 100MB). The image data that was obtained also contained unwanted noise and artifacts within the image volume. A two dimensional slice of the data can be seen in Figure 4.1. These artifacts are unwanted in the process of determining the surface data of the virtual patient.

In order to eliminate the noise artifacts in the image volume, Jason Kutarnia applied a control segment process to the data. This process can be further read in [Kutarnia]. After this process was applied to the data, the resulting image volume had eliminated noise artifacts and defined clear surface boundaries. A two dimensional slice of a post-processed image volume can be seen in Figure 4.2. This process ensures a clean and more accurate surface data extraction.

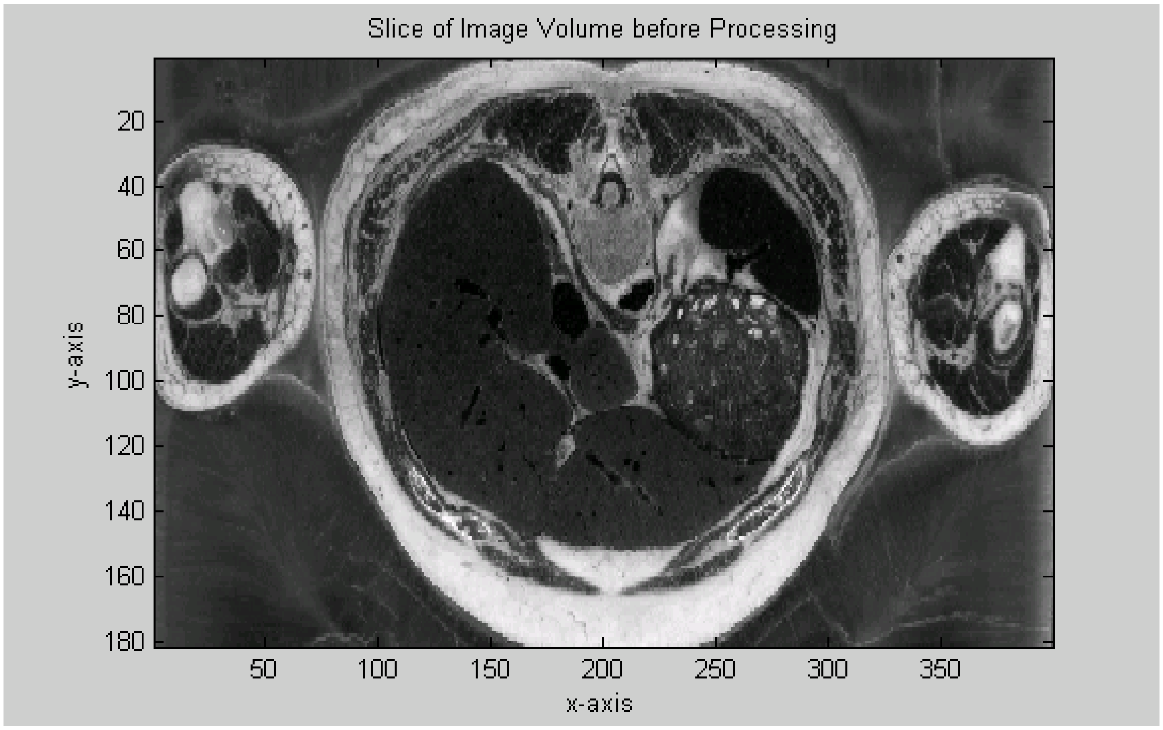


Figure 4.1- Example of two dimensional slice of the original non-processed virtual patient data

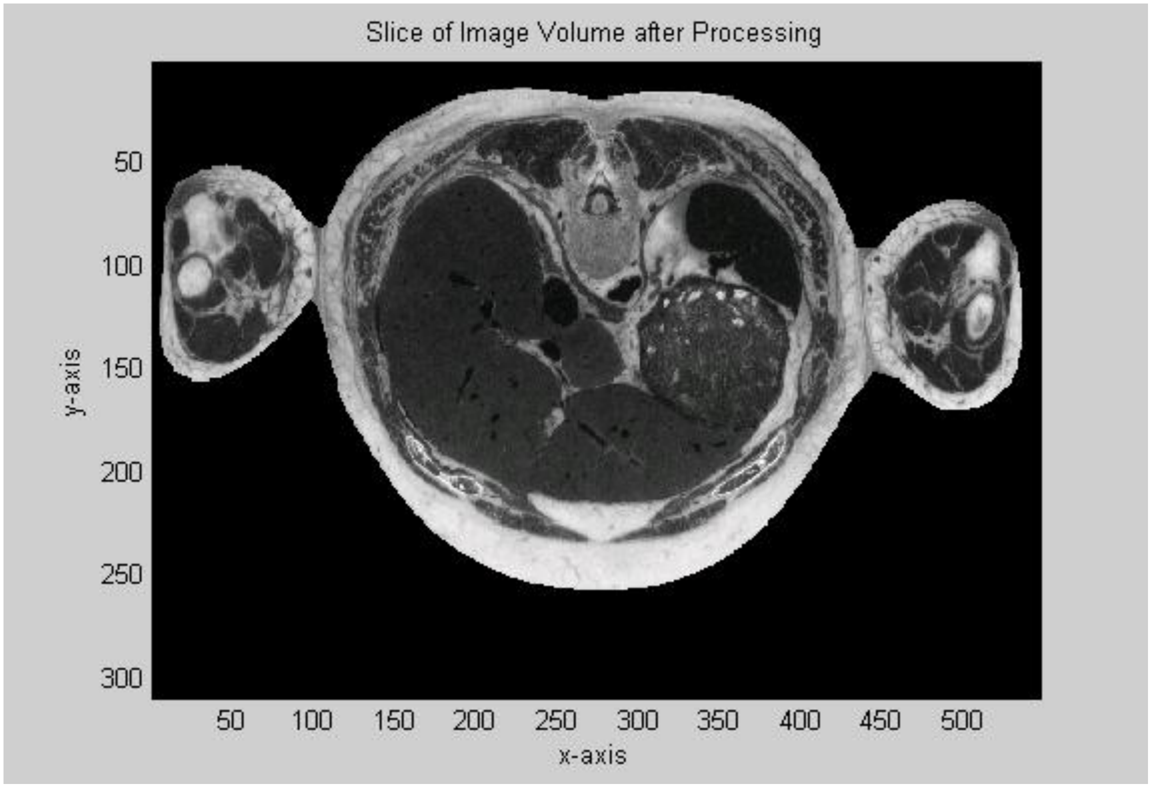


Figure 4.2 – Example of two dimensional slice of the processed virtual patient data

4.2) Addition of a Virtual Transducer

Realism is a key component of the virtual ultrasound training system. The closer a training session can be to a real ultrasound scanning experience, the more effective the training will be. In order to add realism to the experience, a virtual sham transducer was incorporated into the system. This allows a trainee to visualize how the movements of the physical sham transducer are applied to the virtual patient. The following sections discuss the implementation of the virtual sham transducer.

4.2.1) Creating the Virtual Sham Transducer 3D Model

The model of the virtual transducer was created using a trial version of Autodesk 3DS MAX 2011. The model was created by use of manipulating several polygons until a desired model of an ultrasound sham transducer was created. The model as seen in 3DS MAX can be seen in Figure 4.3.

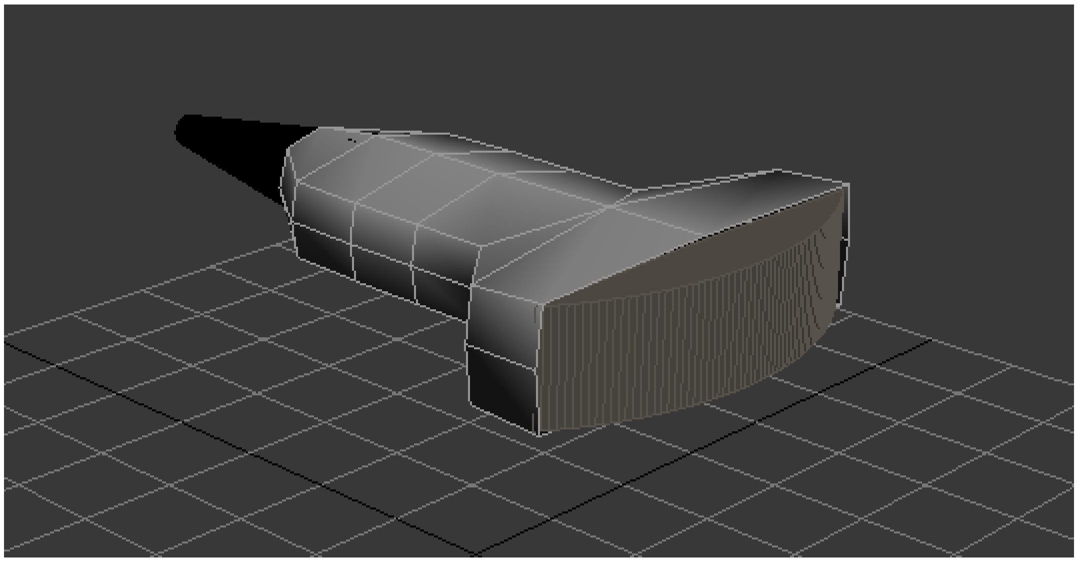


Figure 4.3 - 3D model of ultrasound sham transducer as created in 3DS MAX.

4.2.2) Loading the Virtual Sham Transducer into the Image Reslicing Software

The image reslicing software was already using the OpenGL Mathematics (glm) library to load an object file (.OBJ) of a triangular mesh to represent the virtual patient. The virtual sham transducer was then loaded in a similar manner.

The 3D model of the sham transducer created in 3DS MAX was then exported as a triangular mesh to an object file. By default the export process adds information not required or recognized by glm. For this reason the exported virtual transducer object file had several pieces of information and comments removed. This removal of information allowed for proper parsing of the file by the function *glmreadOBJ*.

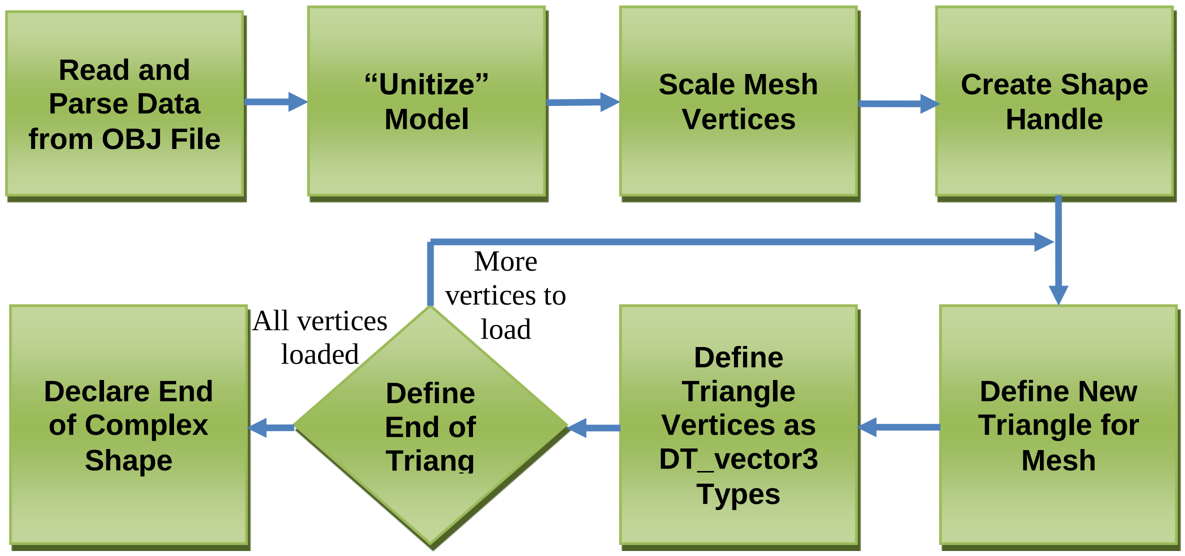
The *glmreadOBJ* function was used to read and parse the transducer object file. The *glmreadOBJ* function added the data to a *GLMmodel* object and returned a pointer to the object. The *GLMmodel* object contains the number of triangles and their corresponding vertices of the triangular mesh.

The *glmUnitize* function was then used to "unitize" a model into a format for OpenGL. This process consists of translating the model to the origin and scaling it to fit in a unit cube centered at the origin. The function returns the scalefactor used.

The *glmScale* function was then used to rescale the *GLMmodel* object data. A scale factor was emperically determined to give the virtual sham transducer proportions to the virtual patient similar to the proportions of the physical sham transducer to the curved scan surface.

The SOLID libraries that were previously added for collision detection were used to detect the collisions between the virtual patient and the virtual sham transducer. In order to utilize the SOLID libraries, *DT_shapehandle* objects must be made for both virtual patient and transducer. The *DT_shapehandle*s are created by loading the pointers to each of the post-scaled vertices of each triangle in the objects particular triangular mesh. This loading was done by looping through each of the triangles stored in the *GLMmodel* objects. Each loop iteration consisted of three parts. The first part was to use the *DT_Begin* function to declare the beginning of a new triangle of the triangular mesh to be loaded into the *DT_shapehandle* object. The second was to define the *GLMmodel* vertices of the particular triangle to be loaded into

DT_vector3 types. Once all three vertices were defined as *DT_vector3* types, the *DT_end* function was used to signify the end of the triangle being loaded. Once all of the iterations had completed, the *DT_EndComplexShape* function was used to signify the end of the triangular mesh. This loading procedure created a handle to be used for collision detection functions from the SOLID library. This procedure is further illustrated in Figure 4.4.



Read and Parse Data from OBJ File
"Unitize" Model
Scale Mesh Vertices
Create Shape Handle
More vertices to load
All vertices loaded
Declare End of Complex Shape
Define End of Triang
Define Triangle Vertices as DT_vector3 Types
Define New Triangle for Mesh

Figure 4.4 – Block Diagram of the Importing and Scaling Prodecure for the Virtual Transducer into the Image Reslicing Software

4.2.3) Controlling the Movement of the Virtual Sham Transducer

The virtual training system is designed such that the physical scan surface is representative of an actual patient lying down facing up and a virtual patient that is upright. The data from the PNI sensor module is required to control both the orientation of the virtual sham transducer and the manipulation (i.e. reslicing) of the 3D image volume. In order to use the data from the PNI sensor to manipulate both the image volume and the orientation of the virtual sham transducer, a 90 degree offset must be accounted for. The 90 degree offset of the virtual patent from the physical scan surface and 3D image volume manipulation was accounted for in the orientation of the 3D model of the virtual sham transducer. By creating a model of a transducer that is 90 degrees offset from the virtual patient, the orientation data from the PNI sensor that is applied to the 3D image volume can now be applied directly to the virtual transducer as well. The orientations can be seen in Figure 4.5.

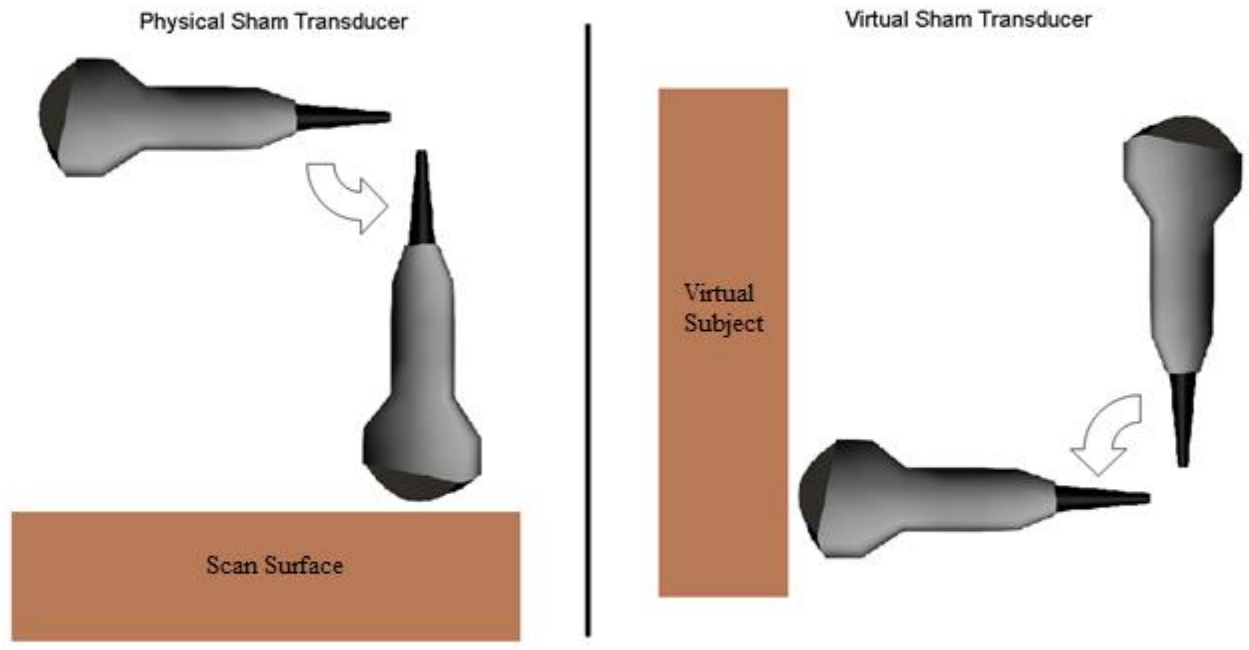


Figure 4.5 - Display orientations of scan surface relative to sham transducer, and virtual subject relative to virtual transducer.

The procedures for rendering the appropriate virtual sham transducer are only executed when the Anoto sensor is applied to the scan surface. The Anoto sensor being applied to the scan surface is represented in code by a flag set called *draw_trans*. The setting of *draw_trans* is discussed in Anoto section. When the *draw_trans* flag is set, the following sections will occur.

The previously created handles for the virtual objects are used to manipulate their position and orientation. The position manipulations are performed using the *DT_SetPosition* function. *DT_SetPosition* uses the object handle and a vertex to reposition the object such that the center of the object is at the vertex that is passed to the function. The orientation manipulations are performed using the *DT_SetOrientation* function. *DT_SetOrientation* uses the object handle and a quaternion to reorient the object.

The reorientation of the virtual sham transducer is controlled by the reoriented quaternion produced from the PNI sensor module. After the virtual transducer has been passed through *glmUnitize*, however, the model is centered at the origin. This centering performed by *glmUnitize* means that any reorientations of the virtual sham transducer will be about the origin. Based on the physical transducer we know this not to be true. The physical transducer is rotated about the position where the transducer comes into contact with the scan surface.

The rotation about the origin is accounted for by applying a shift in position based on the rotation angles that are being applied. By knowing the distance from the origin to the tip of the virtual sham transducer model and applying trigonometry as in equations 4.1 and 4.2,

$$Y_{shift} = R * \sin(pitch) \tag{4.1}$$

$$X_{shift} = R * \sin(roll) \tag{4.2}$$

an appropriate shift vector is calculated. The shift vector was then applied to the position of the transducer as determined by the Anoto data. The shifting process can be seen in Figure 4.6.

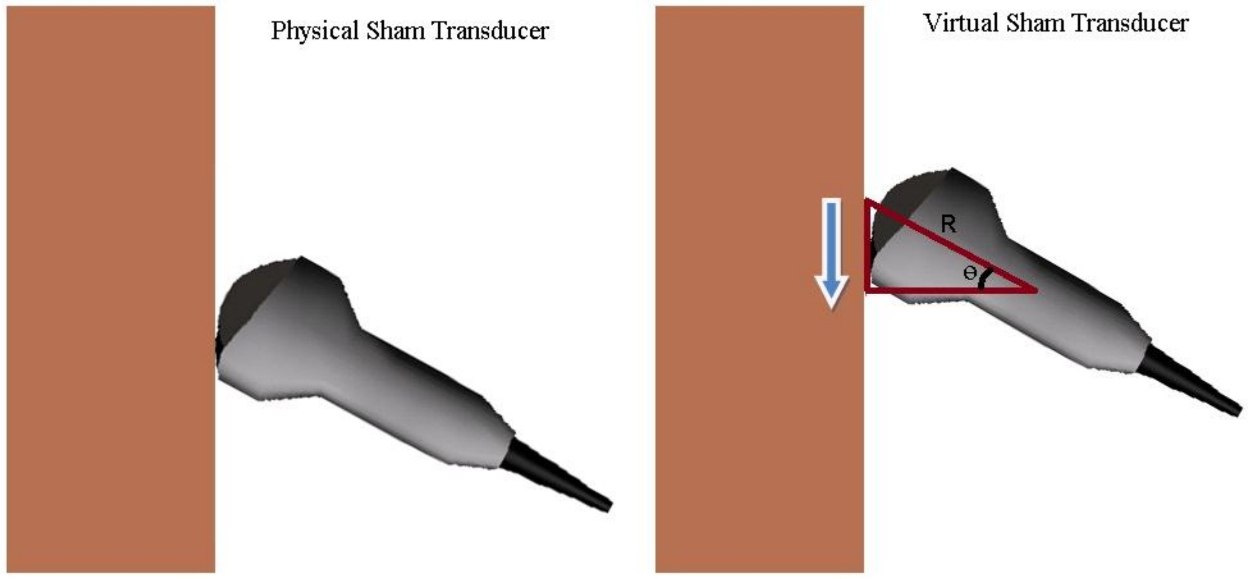


Figure 4.6 - Diagram showing the position shifting required for the x and y position components due to the rotation about the center instead of the tip of the sham transducer.

Once the virtual sham transducer has been properly oriented and positioned, the software then searches for the closest vertex pair between the triangular mesh of the virtual transducer and the virtual patient surface. The overall procedure of positioning and orienting the virtual transducer, given the transformed position and orientation data, can be seen in Figure 4.7. The transformations involved with calculating the position and orientation data will be discussed in section 4.3.

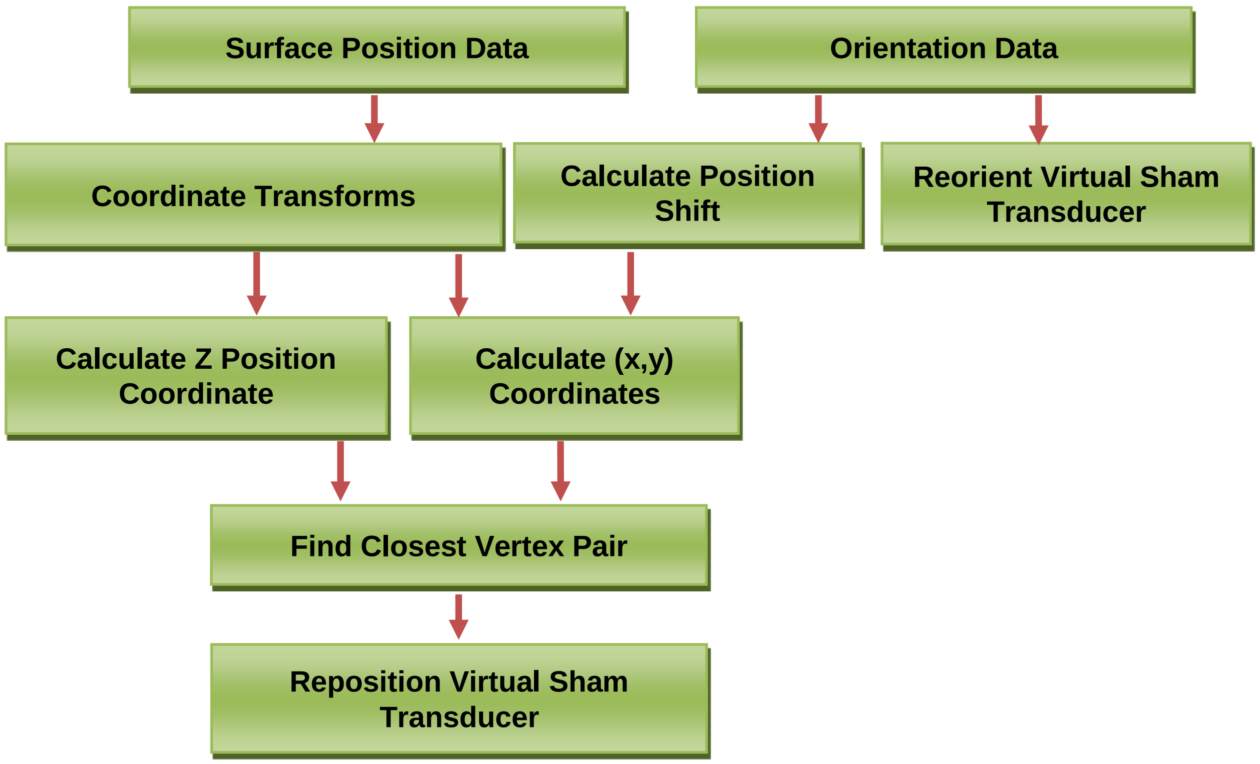

Surface Position Data
Orientation Data
Coordinate Transforms
Calculate Position Shift
Reorient Virtual Sham Transducer
Calculate Z Position Coordinate
Calculate (x,y) Coordinates
Find Closest Vertex Pair
Reposition Virtual Sham Transducer

Figure 4.7 - Diagram showing the virtual sham transducer position and orientation procedure.

At the end of the virtual sham transducer rendering procedure, a color is applied to the surface of the triangular mesh. The color is added using the GLUT function *glColor3ub*. A red-green-blue combination was determined and passed to *glColor3ub* to yield a gray color similar to that of plastic shell of the physical sham transducer.

4.3) Coordinate Transformations from the Physical Surface to the Virtual Environment

This section discusses the theory and implementation of the coordinate transformations between the physical scan surface and the virtual environment. The process first describes the thinking behind the design of the physical scan surface. The section will then continue with the discussion of the data extraction of the patient image data used within the image reslicing software. This is followed by the integration of these data into the image reslicing software to transform the physical coordinates into the virtual environment.

4.3.1) Design of scan surface

The scan surface on which the user moves and angles the physical sham transducer was designed to be representative of a portion of a generic human torso. To allow for the similar shape of a human torso as well as the ability to accurately track position on the surface, the scan surface was designed to be in the shape of a portion of a cylinder. With this type of design, the position coordinates fit perfectly with a cylindrical coordinate system and easily map to different coordinate system. One of the goals of the ultrasound training system is to accurately map position and orientation from the scan surface to the virtual patient. The two following sections outline how to determine the best fit cylindrical coordinate system map from the scan surface to the scanned region of the virtual patient. This process is necessary in order to map the physical scan surface position coordinates to the region of interest of the virtual patient data with minimal error.

4.3.2) Image data manipulation

A selected image volume to be used by the image reslicing software is contained within a raw binary file. This file contains a grayscale 3D image volume of a chosen segment of the virtual patient. The image volume must first be pre-processed using MATLAB in order to extract data needed for the physical position data mapping to the region of interest on the virtual patient.

The image volume is first opened into MATLAB using a *fopen* command. The file is then read into MATLAB by using a *fread* command within a loop. The dimensions (width, height, depth) of the image volume are known and are used to set the number of iterations of the loop. Once this process is complete the image volume will have been imported into MATLAB in the form of an unsigned 8-bit integer three dimensional matrix. Each element in the matrix is a data node which represents the intensity of the grayscale at the corresponding location within the image volume.

Once the image volume has been imported into MATLAB, a surface region of interest must be defined to correspond to the scanning surface. This process was performed using the ISO2MESH MATLAB toolbox. This toolbox contains various functions for extracting data from image volumes and creating meshes based on image data. The *vol2surf* function is used to extract the nodes of the exterior of the patient by using a three dimensional edge detection algorithm. During this process a resampling of the image data occurs. The resampling is a function of a user defined value for the maximum radius of a Delaunay sphere. This process is performed using Delaunay triangulation, where there is no point within a chosen radius distance of another point. Therefore, the larger the radius of the sphere, the smaller the number of nodes used.

Once the surface nodes and elements have been calculated, the *surf2mesh* function is used to produce a triangular mesh. By creating a triangular mesh, the calculated nodes and faces can be passed to the MATLAB command *plotmesh*. The *plotmesh* command will yield a three dimensional plot of the triangular mesh surface data. Once this plot is created, a region of interest can be determined and measured using the plot's data cursor. The region of interest is defined by assigning boundaries within the image volume. The boundaries consist primarily of upper and lower bounds for each axis. An example of a virtual subject and surface mesh with chosen boundaries are represented by the lines plotted in Figure 4.8.

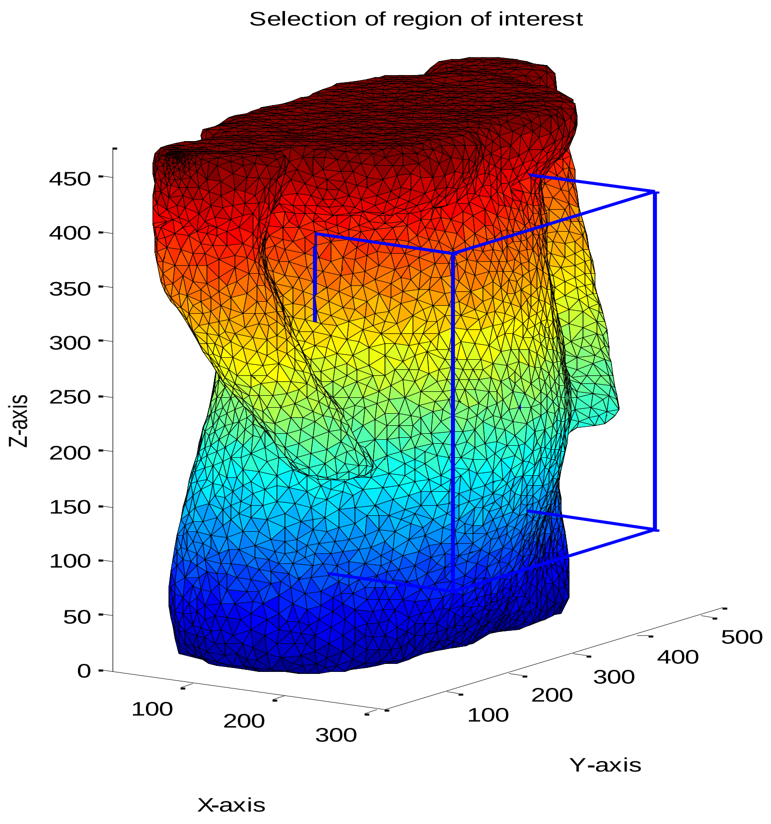


Figure 4.8 – Surface Mesh of Patient Data with Boundary Lines for the Region of Interest

After the selection of these bounds, a new object consisting of only the data within those bounds is created. Portions of this selected region however may still be unwanted. An example of which may be a portion of a human torso as a selected region. Within the selection a portion of an arm that is just above the torso may still remain. In these instances, the above process of creating a mesh and corresponding three dimensional plot to determine the coordinates of these unwanted areas is necessary. An example of such a plot corresponding to the patient in Figure 4.8 can be seen in Figure 4.9. Once the unwanted areas are defined, simple MATLAB scripts can be used to loop through the regions and set each of the data points equal to zero (number representing empty space). After this process is performed, only the region of interest remains. The result of this process for the data seen in Figure 4.8 and Figure 4.9 can be seen in Figure 4.10.

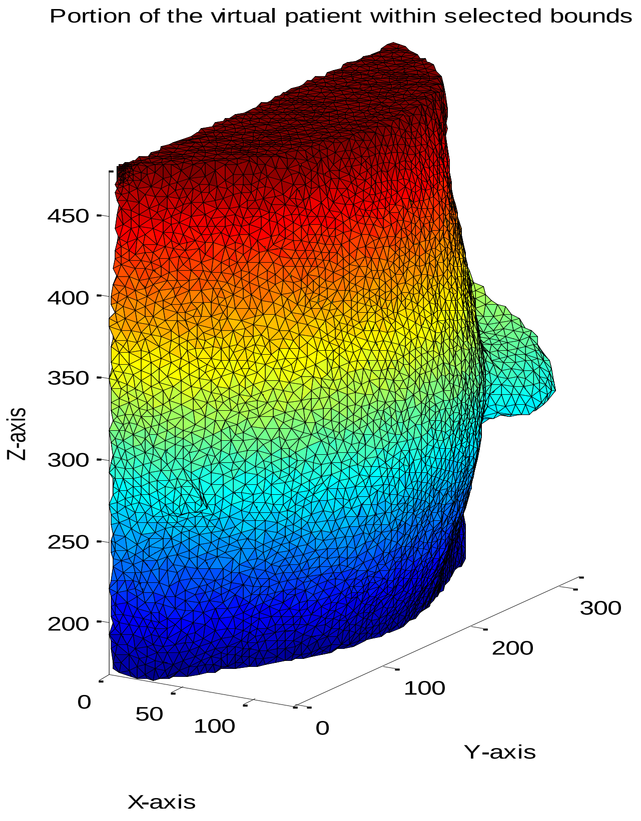


Figure 4.9 - Portion of the virtual patient within the selected bounds to be used as a region of interest.

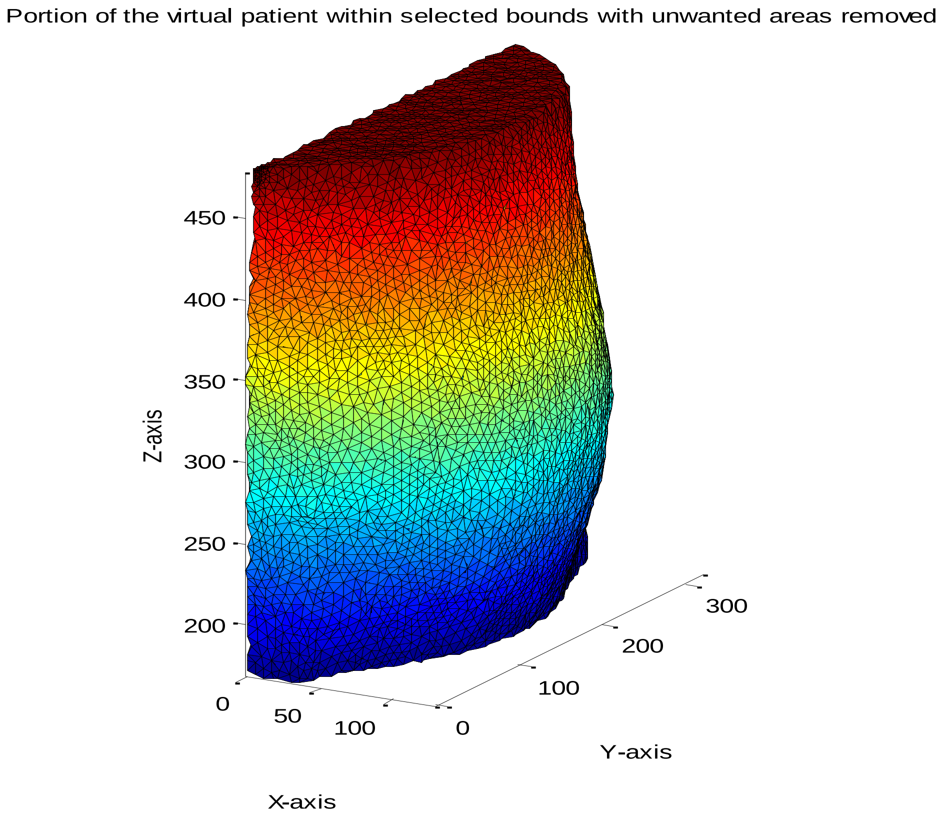

Portion of the virtual patient within selected bounds with unwanted areas removed
Z-axis
450
400
350
300
250
200
0
50
100
X-axis
0
100
200
300
Y-axis

Figure 4.10 – Portion of the virtual patient within the selected bounds to be used as a region of interest with unwanted areas removed.

Once there is an image volume containing only the region of interest, the surface data corresponding to the scan surface must be extracted. This procedure is necessary to avoid the use of surfaces corresponding to the boundaries of the image volume that are not surfaces of the virtual subject. In order to obtain the surface data that relate to only the scan surface without the portion represented by the bottom of the scan surface, the data below the scan surface must be removed. This is accomplished by first finding an axis such that if a vector parallel to this axis is translated anywhere within the image volume, the vector will not penetrate the surface of the virtual subject more than once. In the case of the supplied data, this is the x-axis. Once found, a MATLAB script can loop through each (z, y) pair within the image volume. For each (z, y) pair, the script will loop through each of the x values starting from the bound that does not contain any data (in this case zero values). Each x-value is checked if it is a non-zero value (meaning data are present). If a non-zero value is detected, representing the surface, all of the values after the first non-zero value detected are set equal to zero (no data). After this process the image volume will be left with surface data that corresponds to the scanning surface.

Once the image volume contains only the surface data corresponding to the scan surface, the ISO2MESH toolbox functions will be used again to create a triangular three dimensional mesh of the surface data. This is done for two reasons. One is that this produces an array of nodes that represent only the surface. The second is that this process will resample the data, representing it with fewer data points. Having less data is necessary for MATLAB to be able to handle calculations that will be discussed in the next section.

4.3.3) Least square cylindrical fit

Once the desired surface data has been obtained, the parameters of a cylinder that best fits the data are calculated by means of a least square fit. This process is necessary to best match the cylindrical coordinates of the scan surface to the data. The least square cylindrical fit is performed by using the tolerances and initial estimates below. These estimates were based on observations of the surface data as can be seen in Figure 4.10. The estimated values may be altered depending on the data set used. Position and distances are measured in element spaces of the image data volume.

Estimate of the axis direction vector: $[x_{ae}\ y_{ae}\ z_{ae}]^T = [0\ 0\ 1]^T$
Estimate of the cylinder radius: $r_0 = 200$
Estimate of the point on the axis: $[x\ \ y\ \ z]^T = [-110\ \ 160\ \ 0]^T$
Tolerance for test on step length: $p = 0.01$
Tolerance for test on gradient: $g = 0.01$

As can be seen from [EuroMETROS], the purpose of the least squares cylindrical fit is to minimize the distances of the data from the fitted cylinder. An iterative process is repeated until parameters of a fitted cylinder fall within the defined tolerances.

4.3.3.1) Determining the centroid of the surface data

This process begins by finding the centroid of the surface data for the chosen image volume. The centroid will be required for a transformation in the next step of this procedure. The centroid is obtained by calculating the mean of each the *x*, *y*, and *z* components of the surface data points. Each of the means is calculated using equations 4.3 and 4.4.

$$X_i = \begin{bmatrix} node_{xi} \\ node_{yi} \\ node_{zi} \end{bmatrix} \qquad 4.3$$

$$x_c = \frac{1}{N}\sum_{i=0}^{N-1} X_i \qquad 4.4$$

The value *N* represents the total number of surface data points and each x_i represents a component. The vector containing the corresponding means is the centroid of the data.

4.3.3.2) Translate and rotate data into a standard position

Following this the procedure requires that the data be transformed to a standard position by means of rotation followed by a translation. The standard position is to be in line with a given axis. When the data are aligned with an axis, the calculations for determining the cylindrical parameters become simplified. In this scenario, the standard position is to have the estimated cylinder axis to lie along the z-axis. Both the rotation and the translation are based on the initial information and estimates. The rotation is calculated by solving for R_0 in equation 4.5.

$$R_0 * a_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad 4.5$$

The vector a_0 represents the initial axis direction estimate. In the event that the initial axis estimate is equal to the z-axis, the rotational matrix R_0 is the identity matrix. However, if this is not the case, the following equation could be solved using Givens rotations [EuroMETROS].

The rotational matrix is then applied to the initial estimate of the point on the axis *x0* and the centroid of the data x_c as seen below.

$$x0_r = R_0 * x0 \qquad 4.6$$

$$x_{cr} = R_0 * \ x_c \qquad 4.7$$

This is followed by finding the point on the axis nearest the centroid of the rotated data x_p. This value is then subtracted from the rotated data nodes, axis estimate, and centroid. This procedure can be seen below. Variable x_{2r} is the transformed data set that will be applied to the least squares algorithm.

$$x_p = x0_r + \left(x_{cr}(3) - x0_r(3)\right) * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad 4.8$$

$$x_{2r} = (X * R_0^T) - \mathbf{1} * x_p^T \qquad 4.9$$

$$x_{crp} = x_{cr} - \ x_p \qquad 4.10$$

4.3.3.3) Gauss-Newton algorithm

This portion of the process uses a Gauss-Newton algorithm to find an estimate of the rotation and translation parameters that transform the data such that the best-fit cylinder is one in the standard position. The Gauss-Newton algorithm is a method for minimizing a sum of squared function values. Using the rotated and translated forms of the initial estimates and data, the algorithm performs an iterative process that will be described below. The values to be estimated in this process are contained within the *a* vector as seen below.

$$a_i = [x_i \;\; y_i \;\; alpha_i \;\; beta_i \;\; r_i]^T \qquad 4.11$$

The x_i and y_i terms refer to a point located on the axis of the cylinder. The *alpha*$_i$ and *beta*$_i$ terms represent angles used to rotate the cylinder when performing the best-fit calculations. Finally, the r_i term is an estimate of the radius for the best-fit cylinder.

Gauss-Newton iterations

During this process it is required to calculate a rotational matrix **R** and several of **R**'s derivatives. This rotation matrix is calculated from the estimated angles of rotation (*alpha*, *beta*) for the cylinder from the standard position. The more accurate the angle estimates, the closer the rotation matrix **R** will align the data with the standard position. In order to rotate the estimated axis of the cylinder into the standard position, "it is first rotated about the x-axis so that is now in the YZ plane. Then the axis of the cylinder is rotated about the y-axis so that it is now along the z-direction" [udel]. The calculations of the rotation matrix and its derivatives begin with calculating the planar rotations as seen below.

$$\theta = \begin{bmatrix} alpha_i \\ beta_i \\ 0 \end{bmatrix} \qquad 4.12$$

$$R1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{bmatrix};\; R2 = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \qquad 4.13$$

$$R_i = R2 * R1 \qquad 4.14$$

This is followed by calculating the derivatives of the plane rotations as seen below.

$$dR1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin\theta_1 & -\cos\theta_1 \\ 0 & \cos\theta_1 & -\sin\theta_1 \end{bmatrix};\ dR2 = \begin{bmatrix} -\sin\theta_2 & 0 & \cos\theta_2 \\ 0 & 0 & 0 \\ -\cos\theta_2 & 0 & -\sin\theta_2 \end{bmatrix} \qquad 4.15$$

Then finally we calculate the derivative matrices of **R** as seen below. These derivatives will be needed when calculating the Jacobian matrix as will be discussed later in this section.

$$DR1 = R2 * dR1 * R_0 \qquad 4.16$$

$$DR2 = dR2 * R1 * R_0 \qquad 4.17$$

There are also several intermediate calculations that are required that can be seen below.

$$X_a = (X - \mathbf{1} * [x0_i \quad y0_i \quad 0]) * R_i^T \qquad 4.18$$

$$[xt \quad yt \quad zt\,] = X_a \qquad 4.19$$

$$rt = \sqrt{\mathrm{xt.*\,xt\ +\ yt.*\,yt}} \qquad 4.20$$

$$N_t = [xt./rt \quad yt./rt \quad \mathbf{0}] \qquad 4.21$$

$$A1 = \mathbf{1} * \left(R_i * \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \right) \qquad 4.22$$

$$A2 = \mathbf{1} * \left(R_i * \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \right) \qquad 4.23$$

$$A3 = (X - \mathbf{1} * [x0_i \quad y0_i \quad 0]) * DR1^T \qquad 4.24$$

$$A4 = (X - \mathbf{1} * [x0_i \quad y0_i \quad 0]) * DR2^T \tag{4.25}$$

Then Jacobian matrix $\frac{\partial f(i)}{\partial a(j)}$ is calculated as seen in the following MATLAB code:

```
J(:, 1) = dot(A1, Nt, 2);
J(:, 2) = dot(A2, Nt, 2);
J(:, 3) = dot(A3, Nt, 2);
J(:, 4) = dot(A4, Nt, 2);
J(:, 5) = -1 * ones(m, 1);
```

From here it is required to calculate the distance of the points to the cylinder.

$$\begin{bmatrix} d_x \\ d_y \\ d_x \end{bmatrix} = Ry(beta) * Rx(alpha) * \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x0_i \\ y0_i \\ 0 \end{bmatrix} \right) \tag{4.26}$$

$$F_i(j) = \sqrt{d_x(j)^2 + d_y(j)^2} - r_i \tag{4.27}$$

Calculate update step and gradient

The next segment of the Gauss-Newton algorithm requires the calculation of update step and gradient. The process for calculating these variables begins with a QR decomposition of the matrix **M** seen below, i.e., decomposition of **M** into an orthogonal matrix and an upper triangular matrix.

$$M = [J_i \quad F_i] \tag{4.28}$$

The upper triangular matrix resulting from the QR decomposition of **M** is then defined as ***Rq*** and used in the calculation of the update step *p* and gradient *g* as seen below. The matrix ***Rq*** is split into two matrices where the i^{th} column of ***Rq*** is equivalent to *q* also seen below.

$$Rq = [R \quad q] \tag{4.29}$$

$$p = -\frac{R}{q} \tag{4.30}$$

$$g = 2 * R^T * q \tag{4.31}$$

The parameters of the cylinder that are being estimated are then updated with the update step as seen below.

$$a_i = a_0 + p \tag{4.32}$$

Check convergence

Each iteration of the Gauss-Newton algorithm must be checked against a set of convergence criteria in order to determine whether the estimates are within the user provided tolerances. If the estimated parameters are within tolerance, this means that the algorithm has converged and that the estimated cylinder parameters are accurate enough for the application.

Before the convergence criteria can be applied certain values must be calculated. Similarly to the initial calculations of the Gauss-Newton algorithm, the distance from each data point to the cylinder post-update *f1* must be calculated. The norm of *f1*, *F1*, and the norm of the original distances of *f*, *F0* are calculated. Other calculations needed for the testing of the convergence criteria are seen below. The parameter scale represents a scale for columns of the Jacobian matrix. The value *tolr* is the relative tolerance. The value *scalef* is a scale value for function values.

$$c_1 = \frac{\mathrm{max(abs(p\ ))}}{\mathrm{scalef} \ * \ \mathrm{tolr\^{}(0.7)}} \tag{4.33}$$

$$c_2 = \frac{\mathrm{abs(F0 - F1)}}{\mathrm{scalef} \ * \ \mathrm{tolr}} \tag{4.34}$$

$$c_3 = \frac{\mathrm{max(abs(g\ ))}}{\mathrm{scalef} \ * \ \mathrm{tolr\^{}(0.7)}} \tag{4.35}$$

$$c_4 = \frac{F1}{\mathrm{scalef} \ * \mathrm{eps\^{}(0.7)}} \tag{4.36}$$

$$c_5 = \frac{\mathrm{max(abs(g\ ))}}{\mathrm{scalef} \ * \mathrm{eps\^{}(0.7)}} \tag{4.37}$$

The first set of criteria to judge the convergence of the algorithm is by checking if the step size c_1, change in function value c_2, and the size of the gradient c_3 are all less than one [EuroMetros]. If this is true, the algorithm has converged. If false, the criteria states to check if the sum of squares near zero c_4 is less than one [EuroMetros]. If this statement is true, then the algorithm has converged. If not, the criterion states that the gradient near zero c_5 must be less than one. Similarly, if this statement is true than the algorithm has converged. If none of these criteria were met, then the algorithm has not converged and must perform another iteration.

4.3.3.4) Inverse transformation

After the Gauss-Newton algorithm has converged, some of the estimated cylinder parameters need to be transformed from the rotated and translated standard position used in the estimation process back to that of the pre-transformed data. Two of the estimated parameters from the algorithm, within the vector *a*, are used to find the rotation to align the cylinder with the data. This process is also similar to the rotational matrix calculation at the beginning of the Gauss-Newton algorithm producing a rotational matrix ***R*** and its derivatives. Using these matrices the transformations can be computed as seen below.

$$r_n = a_n(5) \tag{4.38}$$

$$a_n = R_0^T * R_n^T * \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{4.39}$$

$$p = R_n * \left(x_{crp} - \begin{bmatrix} a_n(1) \\ a_n(2) \\ 0 \end{bmatrix} \right) \tag{4.40}$$

$$x0_n = R_0^T * \left(x_p + \begin{bmatrix} a_n(1) \\ a_n(2) \\ 0 \end{bmatrix} + R_n^T * \begin{bmatrix} 0 \\ 0 \\ p(3) \end{bmatrix} \right) \tag{4.41}$$

The term *x0*$_n$ is the point on the axis in the plane containing the centroid, *an* is the estimated axis, and *a(5)* is the estimated radius of the cylinder. A cylinder created using these parameters with a triangular mesh of the corresponding region of interest surface data can be seen in Figure 4.11. It can be seen that even with the best fit cylinder there are areas where there is noticeable error from (or below) the surface data. Compensation techniques to better account for this error will be discussed in the upcoming section.

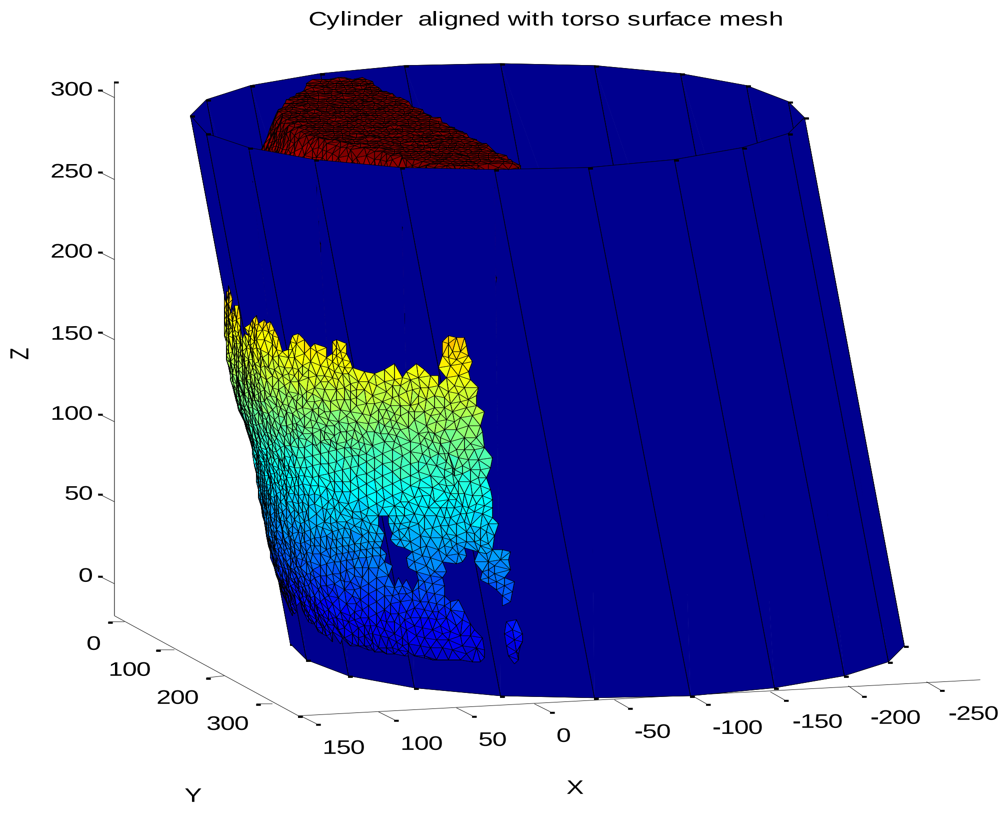


Figure 4.11 – Estimated size, position, and orientation of cylinder to fit surface data of region of interest

4.3.4) Integrating position and orientation data into reslicing software

Once the parameters for the best cylindrical fit to the surface data of the area of interest have been determined, the transformation of scan surface data can be converted into a corresponding position and orientation of the image volume. This section discusses the transformation from surface scanning data to the image volume coordinates. Later in the section the altering of image volume position and orientation data to be used with the movement of the virtual transducer will be discussed.

4.3.4.1) Converting between different cylindrical coordinate systems

The two coordinate systems can be seen in Figure 4.12. Each of the coordinate systems has a radius, arc length, and ɵ value.

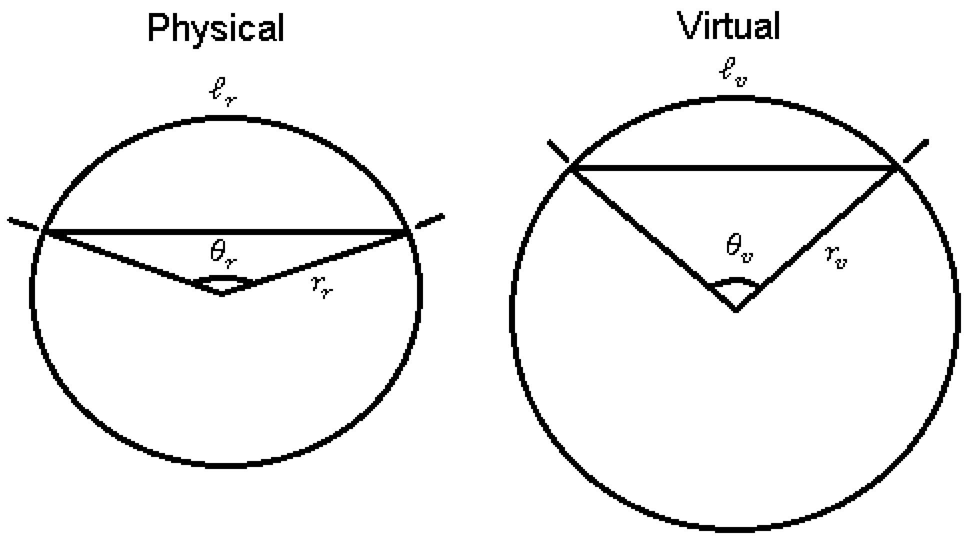


Figure 4.12 – View of both physical and virtual coordinate systems perpendicular to their cylindrical axis.

For the virtual scan surface system, the value of θ_v is calculated based on the radius and point on the axis estimates from the least square cylindrical fit discussed in the previous section. By knowing that the point on the axis will be below the chord of the arc, elementary trigonometry is applied to solve for θ_v as seen below.

$$\theta_v = 2 * \cos^{-1}\frac{x0_{n_1}}{r_n} \qquad 4.42$$

For the physical scan surface system, the value of θ_v is known to be 2π/3. This value of ɵ is determined due to the original material used for the creation of the scan surface was a cylinder that was cut into three equal pieces. The arc length of the physical system is known to be the largest possible Anoto unit value of the Anoto pattern applied to the scan surface. For reasons of simplicity, ease of integration, and to allow for easy adjustments to the code in the event a different Anoto pattern is desired, the arc length is normalized by the maximum Anoto unit in the corresponding direction. The post-normalization Anoto position reported, which should now be between 0 and 1, is subtracted by ½. This subtraction allows for zero centering of the arc on the center of the physical (and virtual) cylindrical portion. This implementation is illustrated in Figure 4.13.

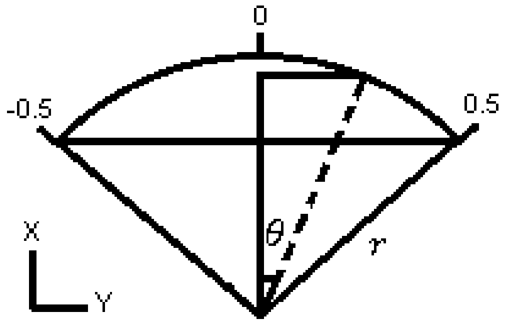


Figure 4.13 – Diagram of the arc length of the virtual cylindrical coordinates post Anoto scaling and centering

For every angle calculated based on the reported real-time data θ_{ri}, a proportional virtual angle θ_{vi} is calculated using the following equation.

$$\theta_{vi} = \theta_{ri} * \left(\frac{\theta_v}{\theta_r}\right) \quad 4.43$$

Through substitution it can be shown that the real-time virtual angle can be calculated directly as a function of the Anoto position.

$$\theta_{vi} = \ell_{ri} * \left(\frac{\theta_v}{\ell_r}\right) \quad 4.44$$

After the calculation of θ_{vi}, the displacement from (x_0, y_0) is calculated as seen below. The displacement is then summed with (x_0, y_0) to yield a position that best corresponds to a cylindrical coordinate system while still along the z-axis. This scenario is also depicted in Figure 4.14.

$$y_i = r_v * \sin\theta_{vi} \qquad 4.45$$

$$x_i = r_v * \cos\theta_{vi} \qquad 4.46$$

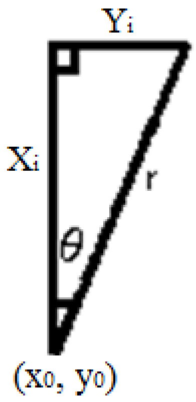


Figure 4.14 - Illustration of displacement from (x_0, y_0) as a function of θ_{vi}

The Anoto positional measurement aligned with the z-axis is a simple linear transformation. An offset is calculated for the transformation that aligns the middle of the scaled Anoto position data with the center of the virtual subject's region of interest.

Following the above calculation of the (x_i, y_i, z_i) coordinates, the points must be rotated such that the z-axis of the cylindrical coordinates is in accordance with the point and vector estimated using the least square cylindrical fit.

4.3.4.2) Applying coordinate transformation of virtual sham transducer

The OpenGL mathematics (glm) library unitizes the triangular mesh of the virtual subject. During this unitization process, the model is scaled and centered at the centroid of the data. These transformations require that the virtual transducer be additionally shifted and translated after the calculation of the (x, y, z) coordinates.

The virtual transducer is also centered at the centroid of its own triangular mesh. Due to the nature of the shape of the virtual transducer, the bottom of the transducer is approximately a set scalar value b away from the centroid. This requires the addition of b to the x value used to translate the transducer, allowing for the bottom of the virtual transducer to align with the physical transducer.

4.3.5) Determining transducer orientation in the virtual environment

A desired feature of the ultrasound training system is to have the physical transducer's angular displacement from the vector normal to its position on the physical surface have an equal virtual transducer angular displacement from the vector normal to its position in the virtual environment. This feature allows for the user to have a more realistic experience when the transducer surface position remains stationary. The implementation of this feature is discussed below.

The angle normal to the physical surface, θ_{ri}, is derived in the previous section. In the virtual environment, the normal angle is rotated such that the best fit cylinder axis is aligned with the 3D image data. The difference between the original normal angle from the physical surface and the IMU data is calculated. This difference in orientation from the physical scan surface is then applied to the rotated normal angle in the virtual environment. This process best simulates the desired feature of unaltered transducer deviations from the normal angle. A block diagram of this procedure can be seen in Figure 4.15.

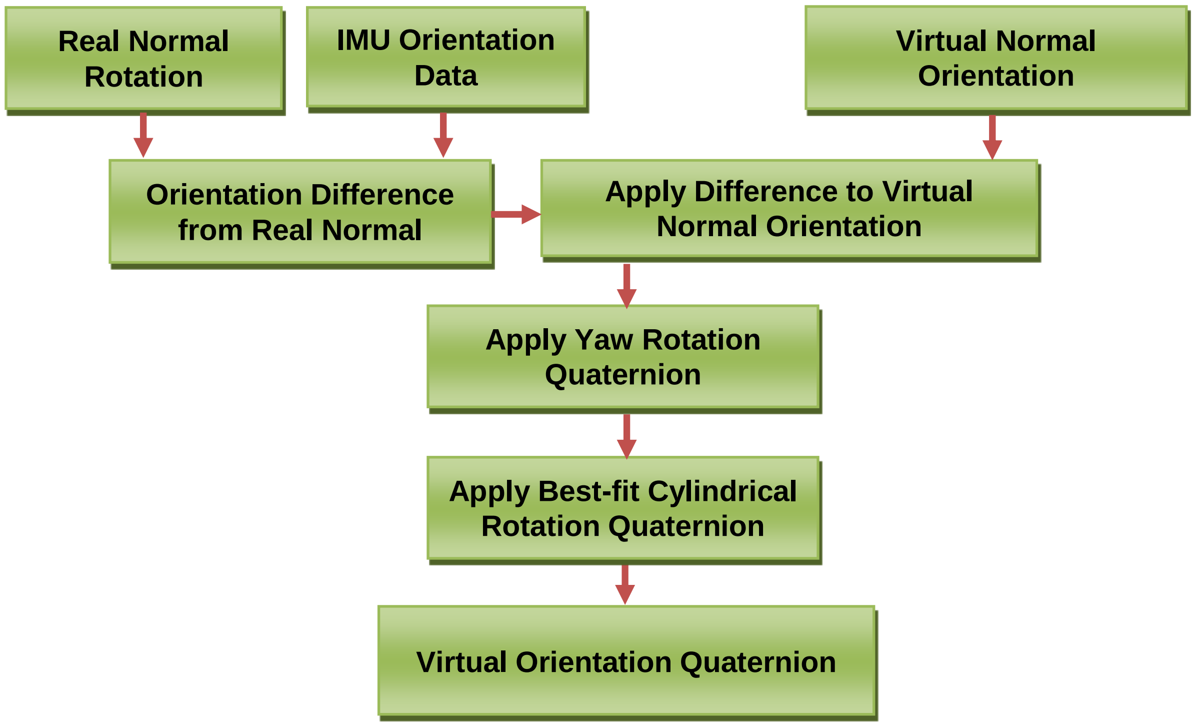

Real Normal Rotation
IMU Orientation Data
Virtual Normal Orientation
Orientation Difference from Real Normal
Apply Difference to Virtual Normal Orientation
Apply Yaw Rotation Quaternion
Apply Best-fit Cylindrical Rotation Quaternion
Virtual Orientation Quaternion

Figure 4.15 - Block Diagram of Determining the Virtual Transducer Orientation

This process begins by first determining the normal orientation of the physical sham transducer. The normal orientation of the physical sham transducer is calculated by first using the Anoto position data to calculate the normal angle of the physical transducer on the scan surface. This angle (θ_{ri}) is used as a rotation in the roll direction. Orientation data from the Spacepoint Fusion IMU is then used to calculate the pitch ($\text{pitch}_{\text{IMU}}$) and roll ($\text{roll}_{\text{IMU}}$) difference from the scan surface normal. This calculation can be seen in equations 4.47 and 4.48.

$$roll_{diff} = -roll_{IMU} + \theta_{ri} \qquad 4.47$$

$$pitch_{diff} = -pitch_{IMU} \qquad 4.48$$

The normal orientation of the virtual transducer must also be calculated for this process. The normal angle of the virtual transducer from the virtual cylinder axis (θ_{vi}) has been described in the previous section. After the angle normal to the axis of the virtual cylinder is calculated, the difference from normal, as seen above, is applied. This calculation can be seen in equations 4.49 and 4.50.

$$roll_{normal} = roll_{diff} + \theta_{vi} \qquad 4.49$$

$$pitch_{normal} = pitch_{diff} \qquad 4.50$$

This process yields an orientation pointing from the scan surface that is proportional to that of the user. From here, the yaw data from the IMU in quaternion from is applied. This calculation produces the orientation normal to the virtual cylinder axis. This process is illustrated in Figure 4.16.

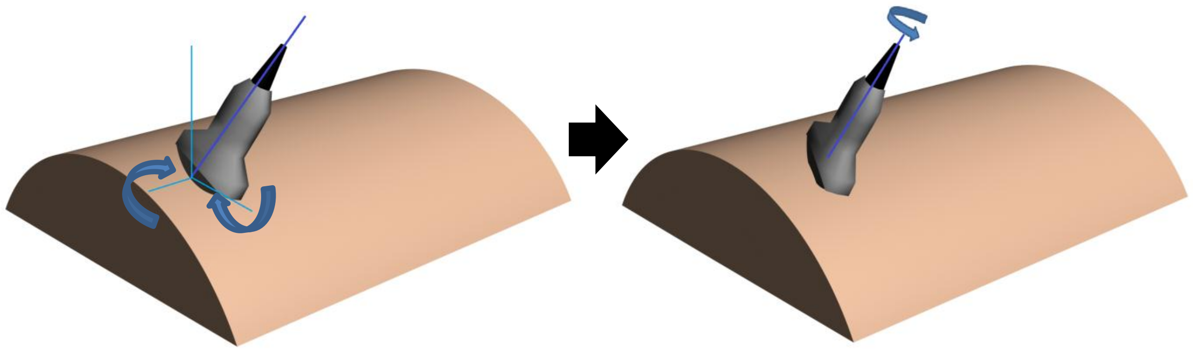


Figure 4.16 – Visualization of the Process to Calculate the Orientation of the Physical Transducer when Normal to the Scan Surface. The Process First Utilizes the Pitch and Roll to Obtain the Virtual Normal. The Process then Applies the Yaw Rotation to the Virtual Normal.

After the orientation normal to the axis of the virtual cylinder is calculated, the rotation of the best-fit cylinder axis must be applied to the normal orientation. As described in the previous section, a rotation matrix is calculated for the best fit of the cylinder to the patient surface data.

This rotation matrix is now converted into a quaternion form and applied to the virtual normal orientation. The conversion process from a rotation matrix (**R** = $R_0^T * R_n^T$) to a quaternion (w x y z components) can be seen below.

$$4w^2 - 1 = \mathrm{R}(1,1) + \mathrm{R}(2,2) + \mathrm{R}(3,3) \qquad 4.51$$

$$4x^2 - 1 = \mathrm{R}(1,1) - \mathrm{R}(2,2) - \mathrm{R}(3,3) \qquad 4.52$$

$$4y^2 - 1 = \mathrm{R}(2,2) - \mathrm{R}(1,1) - \mathrm{R}(3,3) \qquad 4.53$$

$$4z^2 - 1 = \mathrm{R}(3,3) - \mathrm{R}(1,1) - \mathrm{R}(2,2) \qquad 4.54$$

The procedure then continues by determining which of the above equations has the greatest value. The greatest value of the above equations can be represented by the variable *b* and used in Equation 4.55.

$$w = \frac{\sqrt{b+1}}{2} \qquad 4.55$$

$$m = \frac{\left(\frac{1}{4}\right)}{w} \qquad 4.56$$

The procedure then continues based on the largest of Equations 4.51 to 4.54. In the event that Equation 4.51 was the largest value, the procedure continues as seen in the equations below.

$$x = (\mathrm{R}(2,3) - \mathrm{R}(3,2)) * m \qquad 4.57$$

$$y = \left(\mathrm{R}(3,1) - \mathrm{R}(1,3)\right) * m \qquad 4.58$$

$$z = \left(\mathrm{R}(1,2) - \mathrm{R}(2,1)\right) * m \qquad 4.59$$

In the event that Equation 4.52 was the largest value, the procedure continues as seen in the equations below.

$$x = \left(\mathrm{R}(2,3) - \mathrm{R}(3,2)\right) * m \qquad 4.60$$

$$y = \left(\mathrm{R}(1,2) + \mathrm{R}(2,1)\right) * m \qquad 4.61$$

$$z = \left(\mathrm{R}(3,1) + \mathrm{R}(1,3)\right) * m \qquad 4.62$$

In the event that Equation 4.53 was the largest value, the procedure continues as seen in the equations below.

$$x = \left(\mathrm{R}(3,1) - \mathrm{R}(1,3)\right) * m \qquad 4.63$$

$$y = \left(\mathrm{R}(1,2) + \mathrm{R}(2,1)\right) * m \qquad 4.64$$

$$z = \left(\mathrm{R}(2,3) + \mathrm{R}(3,2)\right) * m \qquad 4.65$$

In the event that Equation 4.54 was the largest value, the procedure continues as seen in the equations below.

$$x = \left(\mathrm{R}(1,2) - \mathrm{R}(2,1)\right) * m \qquad 4.66$$

$$y = \left(\mathrm{R}(3,1) + \mathrm{R}(1,3)\right) * m \qquad 4.67$$

$$z = (\mathrm{R}(2,3) + \mathrm{R}(3,2)) * m \qquad 4.68$$

The variables *w*, *x*, *y*, and *z* are then formed into the rotation quaternion. Once the rotation matrix quaternion is calculated, it is then applied to the orientation of the virtual transducer. The normal orientation of the transducer will now be normal to the least square best-fit cylinder to the patient data.

As described in the beginning of this section, the angular displacement from the normal of the physical transducer must be equal to the angular displacement from normal of the virtual transducer. This procedure results in the angular displacement of the virtual transducer to be the same as the angular displacement of the physical transducer. This process gives the user a realistic feel to the movement of the transducer.

4.3.5.1) Display adaptation to surface of image volume

Despite using the least square cylindrical fit to approximate the position of the virtual transducer on the virtual subject, there are still some areas of noticeable displacement for certain portions of the virtual patient surface that deviate from the cylindrical surface. An approach to combat this error is to use the SOLID libraries used for collision detection between the virtual patient and the virtual transducer. Due to the virtual patient mesh loaded into the image reslicing software being derived from the image data, this allows one to make an approximation of the distance from the virtual transducer to the surface of the virtual patient. When the virtual transducer is in contact with the virtual subject, a function called *DT_GetPenDepth* is used to determine the translation vector needed to move the transducer to the closest node of the triangular mesh representing the surface of the virtual patient. When the virtual transducer is not in contact with the virtual subject, a function called *DT_GetClosestPair* is used to determine the distance to the closest node of the triangular mesh representing the surface of the virtual patient. The distance is returned in the form of an (x, y, z) distance vector. By applying the inverse of the model scale factor to the distance vector, an approximate translation of the image volume is applied that may reduce the positional displacement. This additional displacement compensation yields a greater reduction in surface displacement when a portion of the surface of the patient highly deviates from the least square best-fit cylinder. An example of a situation when this additional displacement compensation would be a great asset is shown in Figure 4.17.

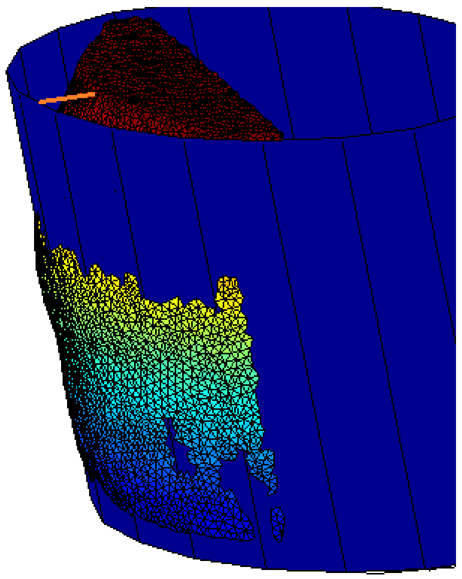


Figure 4.17– Displaying a Scenario when the SOLID Additional Displacement Compensation is greatly needed. Dramatic Displacement from Surface Represented by Orange Line.

5) System Testing and Evaluation

The second generation system is difficult to evaluate for error. This difficulty is partially due to multiple components where error could occur. This difficulty is also caused by the ability to use different virtual subjects corresponding to different image volumes with the system. Different amounts of error will occur with different virtual subjects used. This section discusses the different types of errors and where they originate from in the system. Errors that are a function of the virtual subject data are discussed using a particular virtual subject as an example. This section is divided into three separate portions of the system in which different types of error are discussed.

5.1) Preprocessing Error

One type of error that will most likely arise is the difference in the normal orientation between the real scan surface and the virtual cylinder. This difference in normal orientation is due to the coordinate transformation mapping of the scan surface to the virtual subject data. This difference is related to the total angular displacement that the scan surface maps to the virtual cylinder. In the event that the angular displacements are equal, there would be no error. This event however is rarely, if ever, the case. This difference in orientation has potential to give the user a less accurate experience if the coordinate mapping chosen for the virtual patient highly deviates from the scan surface dimensions.

In the case of one particular virtual subject, the scan surface angular displacement is $2\pi/3$ and θ_v is the corresponding angular displacement mapping. As θ_v approaches $2\pi/3$, the linear scaling of the orientation will approach one. However in this instance, the portion of the best-fit cylinder to the virtual subject had an angular displacement of slightly less than $2\pi/3$. This difference in angular displacement caused the corresponding normal vectors to be closer to the center of the virtual cylindrical segment than normal vector on the scan surface. A visualization of this deviation from the physical angle can be seen in Figure 5.1.

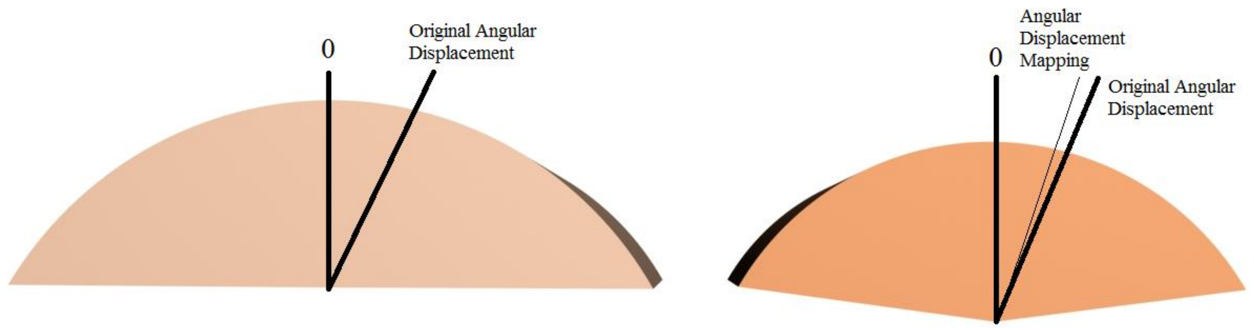

0
Original Angular Displacement
0
Angular Displacement Mapping
Original Angular Displacement

Figure 5.1 - Visualization of the orientation error that may occur based on the difference in angular displacement.

In order to minimize this source of error and have the linear scaling approach one, the y dimension may be adjusted. The adjustment of the y dimension will yield a different result in the coordinate transformation preprocessing. The different coordinate transformations will ideally produce an angular displacement closer to $2\pi/3$. The result of the more ideal angular displacement translates into a more accurate mapping of the scan surface to the virtual subject surface data.

Another type of error that may arise in the preprocessing stage is the coordinate mapping of the length of the virtual subject. It is important to keep the length of the scan surface as directly proportional to the length of the image data selection. By keeping the length of the selected patient equal to that of the scan surface, the coordinate mapping will have little to no error. This selection rule will allow for the user to have a more accurate experience.

5.2) Software Testing

The capabilities of the second generation training system image reslicing software were tested by means of comparing the calculations of position and orientations to higher precision off-line calculations. Deviations from the higher precision calculations are quantified as means for an error metric. This section focuses on discussing the methods, data, and results for the coordinate transformations within the image reslicing software. The section concludes by discussing the potential performance increase gained from the additional error compensation.

5.2.1.1) Methodology

The sensors reporting the data have their own error measurements that are discussed in a later section. The main focus of the software testing was on the performance of the coordinate transformations within the image reslicing software. The coordinate transformations used the reported sensor data to determine the position and orientation of the virtual transducer within the virtual space. Each data within the software is represented in a finite amount of memory. This finite amount of memory results in a lack of precision and accuracy. Each step in the coordinate transformation process yields further potential for error in the position and orientation of the virtual transducer.

The accuracy of the virtual transducer was assessed based on comparison of off-line higher precision calculations. These higher precision calculations will use identical input data reported from the sensors used to position and orient the virtual transducer. The higher precision calculations also use a finite amount of memory and will be subject to the same type of error. However because of the larger amount of memory used allowing for greater precision and accuracy, less of this type of error will be generated. Visual comparisons of the real transducer with plots of the high precision calculated data are used to aid in the confirmation of the correct position and orientation of the virtual transducer.

This process was performed by making minor alterations to the image reslicing software. The alterations to the code made no alterations to the calculation of the coordinate and orientation transformations. This altered version of the image reslicing software was identical to the original version with the addition of the ability to write data to a file for later use in the evaluation process. During the initialization of the software, a file was created and opened with write privileges. At the end of each coordinate transformation, the resulting position and orientation along with the corresponding data used to calculate those values were written to the file. The position data after the additional error compensation were written to the file as well. All of the data and calculations based on the data were conducted using single-precision float memory type.

A procedure was used for confirmation of a correct normal orientation calculation from the image reslicing software. The deviation from the ideal normal orientation results in an error measurement. This error measurement is a metric used to both confirm the correct position and orientations as well as to show any error due to the use of single precision calculations.

This procedure was designed to encompass a variety of positions and orientations that may occur in the system. Six positions were chosen to be tested on the scan surface. Each position is designed to change the situation for at least one input parameter. The test point positions change from a negative normal angle, to a positive normal angle, and to an approximately zero normal angle. The test positions also vary the length position on the scan surface from what would be the lower portion of the torso to the higher portion. These scan surface test positions can be seen in Figure 5.2.

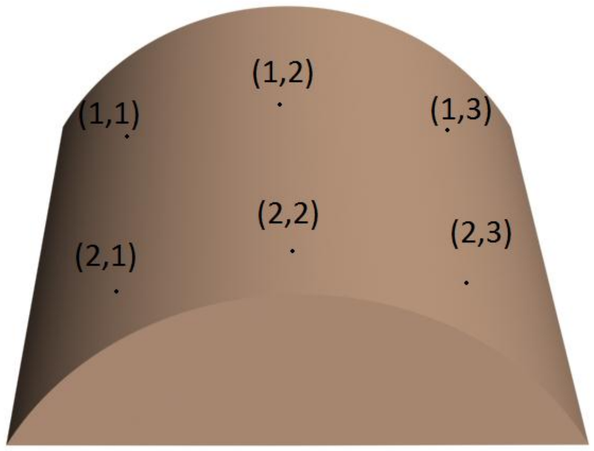


Figure 5.2 - Illustration of the test positions on the scan surface

At each position tested during this procedure, a variety of orientations were tested as well. The approximate normal orientation at both 0 and π/2 yaw angles (rotation about the normal axis). Orientation deviations in both the pitch and roll axis of rotations with various yaw angles are tested as well. These approximate orientation deviations from normal are depicted in Figure 5.3. The transducer was reoriented such that the orientations ranged from no orientation displacement from normal to the maximum deviation from normal while keeping the Anoto sensor on the scan surface.

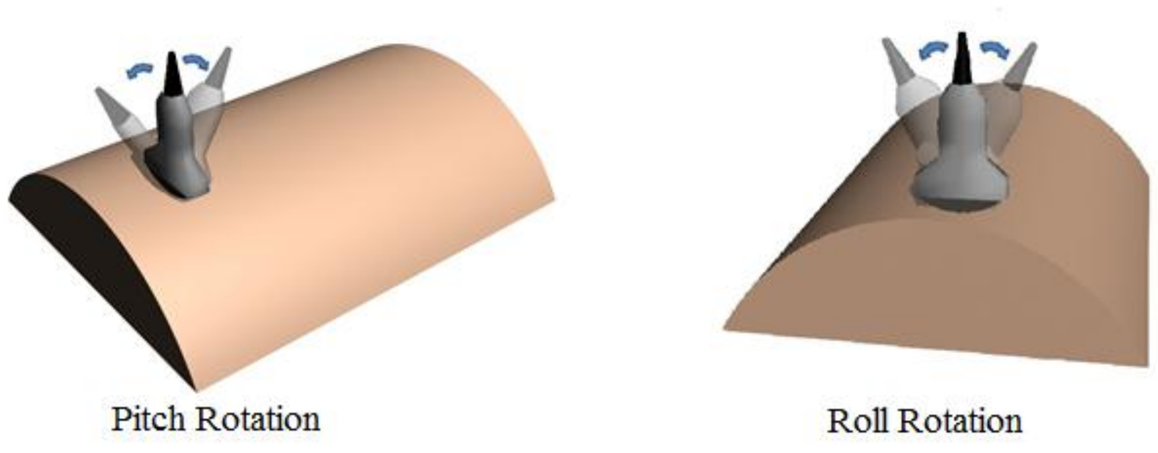


Figure 5.3 – Two orientation deviations from normal used in evaluation of the system

A MATLAB script was created to produce the same orientation and position results, based on a given input, as the image reslicing software. This script allowed an easy to follow high-level process to show the theory behind the coordinate transformation calculations. The script continues to produce various statistics and plots to evaluate the performance of the image reslicing software. The script focused on the position difference, orientation difference, and distance from the resulting virtual transducer position and orientation to the surface data.

The MATLAB script was then used as a comparison to analytic calculation results for the calculation of position and orientation. A set of sensor input data was created such that the system should only vary one degree of freedom at a time. The differences of movements from one position or orientation to another were produced for comparison. No orientation variations occurred during the position testing and position was held constant during orientation variations. The position movements can be seen in Figure 5.4. The movements in orientation are similar to that seen in Figure 5.3. The results of each were then compared to confirm the correct function of the system.

For the calculation of position movement, the comparison was performed using the data just before the final rotation in the calculation. The data were taken at this stage because prior to rotation the data are still aligned with the axes of the coordinate system. This alignment allows for the non-movement of a degree of freedom to be represented by a zero value and simple evaluation.

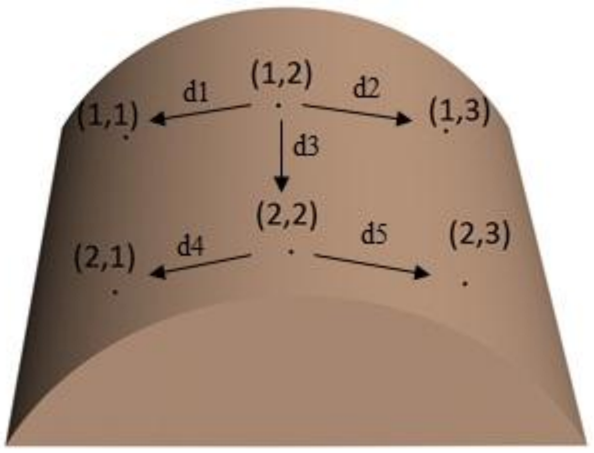


Figure 5.4- Movements from one scan surface position to another used for one degree of freedom change comparison analysis.

The orientation portion held the position stationary at a point with zero angular displacement from the middle of the cylindrical section. This point could correspond to that of (1,2) or (2,2) in Figure 5.4. The orientations were then varied one degree of freedom at a time for pitch and roll rotations by ±45 degrees. These orientation displacements are similar to those seen in Figure 5.3. This process continued with orientation displacements of ±90 degrees in the yaw direction. The data used for this test are the orientations prior to the final cylinder alignment rotation. Similar to the position testing, data collected at this step allow for non-zero values to appear only in the rotated directions.

5.2.1.2) Data Results and Discussion

Tables containing all of the resulting calculations from the MATLAB testing script comparing the image reslicing software can be found in Appendix D. The image volume selected for this test was composed of a data matrix containing 309 elements by 547 elements by 478 elements. Overall, the position coordinate transformation results contained error ranging in magnitude from 10^{-4} to 10^{-5} element spaces. The unit quaternion components contained error ranging in magnitude from 10^{-7} to 10^{-9}. When in comparison to the whole number data increments used in the image reslicing software, these differences can be attributed to the difference in data precision between MATLAB and the image reslicing software. These data confirm the correct functionality of the coordinate and orientation transforms within the image reslicing software.

Tables containing all of the resulting calculations from the MATLAB one degree of freedom movement test script can be found in Appendix E. The position displacement measurements can be seen in

Table 5.1. Overall the positions and orientations performed as expected. Each of the single degrees of freedom displacements resulted in zero values for the non-displaced degrees of freedom. The orientations of the same degrees of freedom that were displaced opposite from each other resulted in identical quaternions with the vector portion negated. These tests further confirm the correct function of the coordinate transformations within the image reslicing software.

Table 5.1– Position displacement measures of one degree of freedom input data in degrees

	delta1	delta2	delta3	delta4	delta5
x	-23.4762	23.47621	0	-23.4762	-23.4762
y	-90.6151	-90.6151	0	-90.6151	90.61506
z	0	0	184.1757	0	0

5.2.1.3) Additional Displacement Compensation

Once the image reslicing software calculates the coordinate transformations of the sensor data, additional displacement compensation is performed. The additional displacement compensation is difficult to quantify in performance. This difficulty is a result of the additional displacement compensation performing differently on each virtual patient. However in order to display the potential of this correction, the error for one particular virtual patient was measured under various conditions.

The same approximate locations on the generic scan surface that were used in the testing of the coordinate transformations of the image reslicing software were used in the additional displacement compensation demonstration. The results of which are also found in Appendix D. The testing on this particular subject resulted in sudden extreme curvature towards the lower part of the scanning area. This curvature is a small area of high deviation from the virtual cylinder. When compared to the upper test points, the test points closer to the lower area of the scanning region resulted in approximately twice the distance from the surface. These results indicate high potential for this form of displacement correction *a posteriori*.

5.3) Hardware Testing

The following section discusses the performance measures applied to the PNI Spacepoint Fusion IMU. The testing involves both static and dynamic testing procedures. Each of these procedures will evaluate different aspects of the Spacepoint Fusion performance. The Static Testing can be found in Appendix F.

5.3.1) Dynamic Testing

The second generation system will require consistent results from an orientation tracker. If a user moves the sensor from a starting orientation, then later returns to the original orientation, the angles reported would be expected to be nearly identical. This section analyzes the values returned from a PNI Fusion in a dynamic situation in order to evaluate its performance.

Test Methods

This section discusses the test methods used to determine the effects of sensor noise outputted by the sensor module when in a dynamic scenario. The sensor was connected to a PC in the Laboratory via a USB connection. The PNI Fusion module was then placed on the flat surface of a desk. Upon power-up and while the sensor is at rest, the PNI Fusion module then took five seconds to perform its one-time auto-calibration phase. [SpacePoint Fusion]

The PNI Fusion was interfaced with using a C++ program. This program was based on the Fusion GamePad Demo. The demo uses a simple USB HID API to interface with the module.

The demo originally consisted of displaying the PNI Fusion output for a set amount of samples. The demo application was expanded in order to convert the quaternion output data of the sensor as discussed in the previous section. The program now wrote the Euler angle data to a text file to be used for later analysis only when a button on the sensor module was initially pressed.

The sensor module was originally placed in a static position (the flat surface of a desk). The sensor button was then pressed for the original data angle to be recorded. The sensor module was then moved to a different orientation and finally returned to the original orientation. At this point the sensor module button was pressed again to record the angle data. The sensor module was moved primarily along the pitch rotational axis and only used the data from the pitch angles. This was due to the shape of the sensor module. The “bottom” of the sensor module could be firmly placed on the flat surface allowing for a static scenario that could be returned to. This experiment was repeated 27 times.

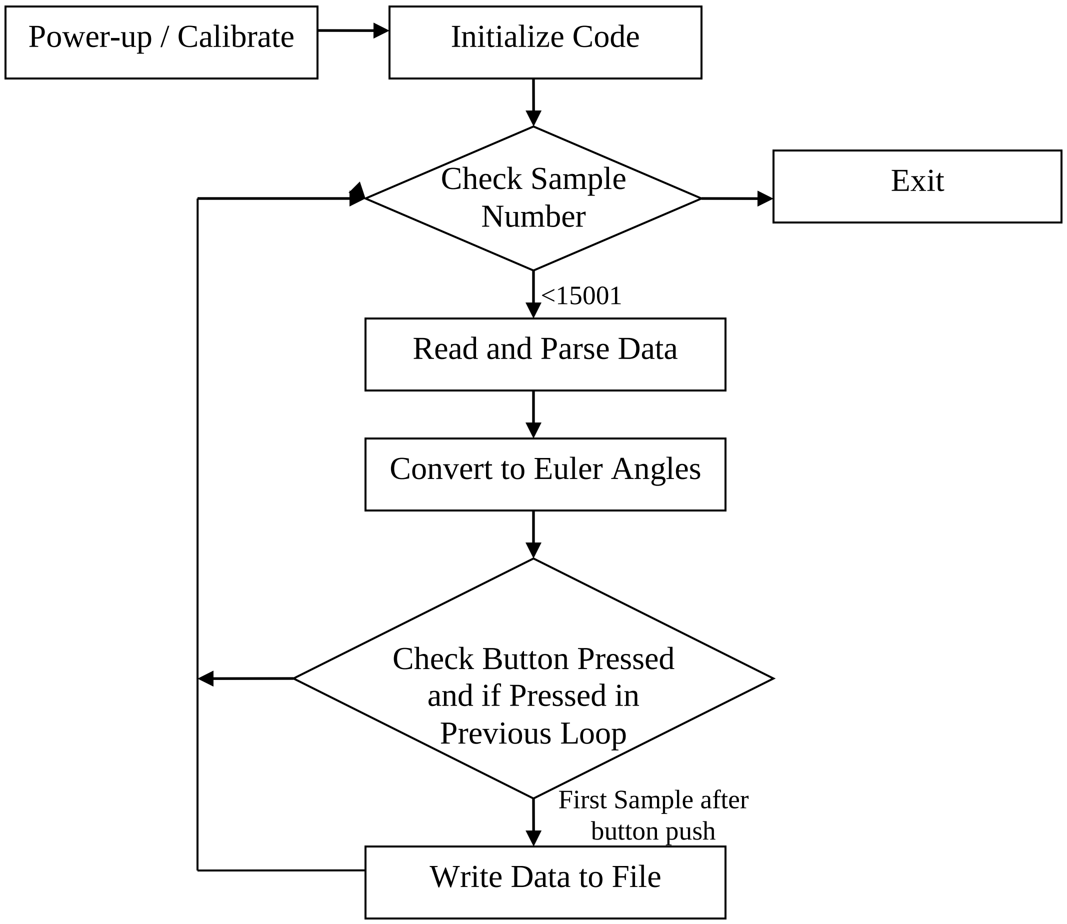


Figure 5.5– PNI SpacePoint Fusion sensor dynamic test procedure code diagram. When running, the experimentalist repeatedly moved the sensor module from a static orientation to new orientation and then back to the original static orientation. Each time the sensor had returned to the original position, the experimentalist pushed the necessary button to affect the test code.

Test Results and Analysis

The orientation data that had been written to a file were then used for analysis. The difference between the original angle outputted of the static position and the second angle were outputted. This difference represents the discrepancy (error) from the set of angles. The resulting error from these trials can be seen in Figure 5.6.

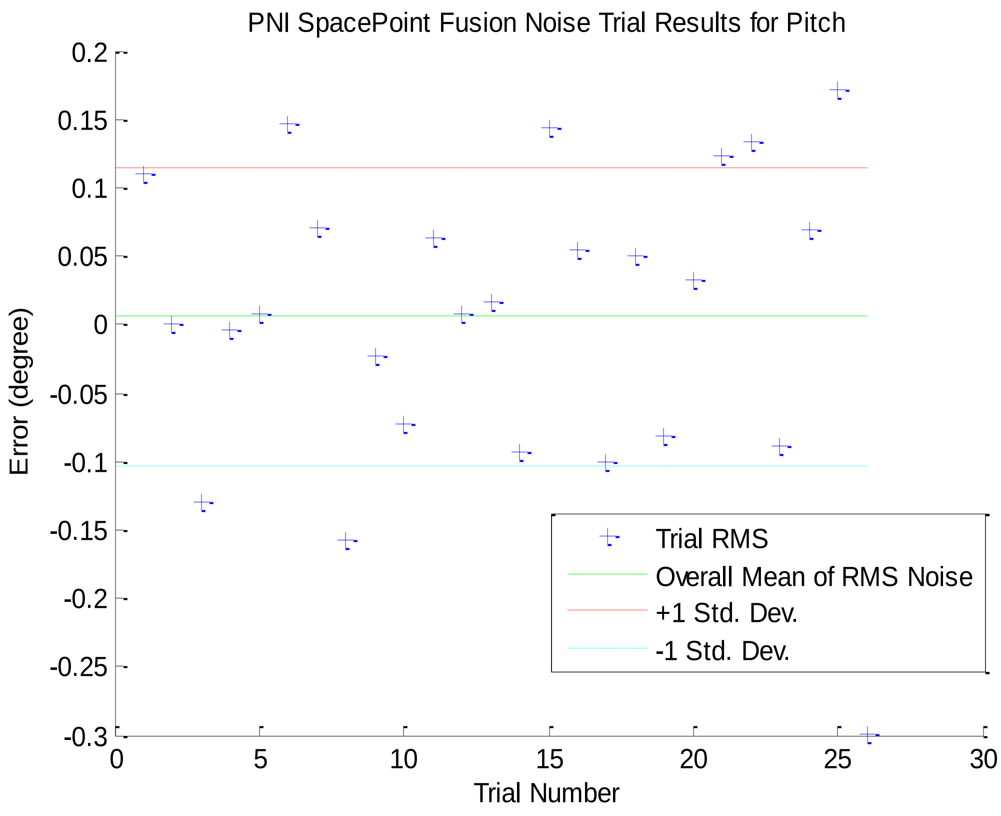


Figure 5.6 – PNI SpacePoint Fusion sensor pitch angle dynamic test position error.

The noise analyzed in this study is that of data from several different sources fused within a military-grade Kalman filtering algorithm [SpacePoint Gaming]. This process is augmented by the high data rate of the sensor, increasing the accuracy of the sensor module [Zarchan]. The fusion of the data from different types of sensors being used to compute common angles has limited the weakness of any one of the sensors. In the case of MEMS gyroscopes, drift has been minimized (if not eliminated).

The error of the dynamic testing shows a low discrepancy between the representations of the same angle. The angular error appears to be below human recognition. Therefore, these results should be acceptable for the purposes of this research. Overall the PNI Fusion sensor module has performed excellently as far as drift in the calculated angular positions.

6) Conclusions and Future Work

6.1) Contributions

Contributions made to the ultrasound training system from this thesis research have included integration of new hardware, modification and addition of software, and the implementation of a new physical scan surface with a method for accurate corresponding coordinate mapping to the virtual patient. The additional hardware added to the system has included an Anoto sensor for position tracking of the transducer and an Inertial Measurement Unit (IMU) for orientation sensing of the transducer. The Anoto technology allowed for the design of a generic curved scanning surface. The IMU integrated into the system allowed for the use of quaternion orientation tracking, a more accurate solution compared to Euler angles.

The software used in this system was based on the image reslicing software from [Banker]. The software was modified and expanded to accommodate the Anoto API, use of quaternion orientations, addition of a virtual subject and transducer, and coordinate mapping involving a least squares cylindrical fit from the physical scan surface to the virtual patient. The coordinate mapping allows for accurate mapping of the physical transducer movements to a subject that is unique to each image volume used.

6.2) Conclusions

This thesis research has successfully developed a prototype of an interactive training system for diagnostic ultrasound. The simulation system permits a trainee to scan a generic scan surface with a sham transducer and view scan planes on the computer screen, updated in real time. This is similar to the previous generation of ultrasound training system developed within the ultrasound research laboratory. The software can be run on a standard PC-based desktop or laptop computer.

This system provides a learning environment without the need to study human patients. The system also eliminates the use of an ultrasound scanner which could be needed for actual medical use. This system provides the opportunity for sonographers to obtain hands-on experience with conditions and pathologies that would otherwise be too rare to occur during training.

This completed prototype system is a major leap in the growing ultrasound training system market. This system has an estimated component price of less than $500 plus the cost of a PC. This system has the potential to gain widespread use due to its relatively low-cost and effectiveness. This system could spearhead its way into a common sonographer's training curriculum. A computer-based training system that is easily portable could make ultrasound much more accessible and encourage its use to become more widespread as well.

6.3) Future Work

Although a working prototype system has been developed, there are still some improvements that could make the system more effective for training and therefore more commercially appealing. Many of these ideas were not original goals of the research, but became obvious opportunities for improvement during the development. Due to the limited scope and time frame of this project, a number of features were unable to be implemented. This section describes these ideas in the hope that they will be considered for future development opportunities.

To add a greater sense of realism, a sixth degree of freedom could be added. The sixth degree of freedom is the displacement (could be detected as pressure) of the transducer on the scan surface. The image volume should accurately compress when pressure is applied to the scan surface. The image should properly reflect the compression to show the shapes of internal organs being temporarily deformed. This pressure being applied could be measured either using a pressure or force sensor placed on the bottom of the transducer.

Another aspect that was not taken into account was the use of ultrasound gel. An improvement to the system could potentially be the detection of ultrasound gel. If the system does not detect ultrasound gel, there should be no image displayed. The presence or absence of ultrasound gel could be detected with the use of a moisture sensor.

An aesthetic detail of the system could be the smoothing of the virtual model. Currently the triangular mesh of the surface data may be "bumpy." This condition can cause a visually unappealing model for the user to look at. This could also result in a possible small unwanted movement in the main image display.

Some image volumes used may contain the arms of the patient. It may be desirable in the future to remove the arms from the final image volume used with the image reslicing software. This would eliminate any possible appearance of a portion of an arm appearing in the image reslice. This process would assure the removal of any non-realistic scans appearing.

The procedure of finding an optimal coordinate transformation between the scan surface and the virtual patient is currently done manually. This process could be easily automated. An automated process could be created with supplied gradient steps and error criterion. The discussed preprocessing could become a subroutine in this automated procedure. The advantage of automating this process would yield rapid data extraction for numerous image volumes with known amounts of error. This initial development of the automated process would yield a great return on investment in the form of working hours required. This decreased work required could increase the progress of the commercialization of the second generation system as well as decreasing the time needed to supply any future updates to the system for the user.

References

"Anoto Functionality: Give Professional Edge in Digital Writing Solutions." *Anoto: Anoto Functionality, Digital Writing Solution with Digital Pen Technology of Anoto*. Web. 13 Feb. 2011. <http://www.karmanya.com/anoto-functionality.html>.

Anoto SPCD 1.3.0 API Documentation. Doc. no: DSS-1-985053. Anoto AB, 2010.

Banker, Christian J. "Interactive Training System for Medical Ultrasound." Thesis. Worcester Polytechnic Institute, 2009. Print.

Dunn, Fletcher, and Ian Parberry. "Chapter 10 Orientation and Angular Displacement in 3D." *3D Math Primer for Graphics and Game Development.* Plano, TX: Wordware,U.S., 2002. Print.

"EuroMETROS - METROS for Europe." Web. 15 Apr. 2011. <http://www.eurometros.org/metros/overview.html>.

"GLUI User Interface Library, V1." *Department of Computer Science, UNC-Chapel Hill*. Web. 20 Feb. 2011. <http://www.cs.unc.edu/~rademach/glui/>.

Goldsmith, Abraham M. *An Inertial-Optical Tracking System for Quatitative, Freehand, 3D Ultrasound*. Thesis. Worcester Polytechnic Institute, 2008. Print.

HeartWorks. Web. 7 July 2011. <http://www.heartworks.me.uk/index.php?page=ttemanikin>.

Heckbert, Paul S. "Chapter III.5 - Euler Angle Conversion." *Graphics Gems IV*. Boston: AP Professional, 1994. Print.

Henriksen KH, Dayton E, "Issues in the design of training for quality and safety." *Qual Safe Health Care*, Vol. 15, pp. i17 - i24, 2006.

Hoppmann RA, Rao VV, Poston MB, Howe DB and Hunt PS *et al.*, "An integrated ultrasound curriculum (iUSC) for medical students: 4-year experience," *Critical Ultrasound Journal*, Vol. 2, pp. 1 – 12, published Online First™, 1 February 2011.

“IC3 datasheet_0908” Intersense, Inc. September 2008.

"InterSense Incorporated | Precision Motion Tracking Solutions | InertiaCube3™." *InterSense Incorporated | Precision Motion Tracking Solutions | Home*. Web. 11 Feb. 2011. <http://www.intersense.com/pages/18/11/>.

Knudson MM and Sisley A.C., “Training Residents Using Simulation Technology: Experience with Ultrasound for Trauma,” Journal of Trauma-Injury Infection & Critical Care, vol. 48, pp 659-665, 2000.

Kutarnia, Jason, Peder C. Pedersen, and Christina Yuan. *Virtual Reality Training System for Diagnostic Ultrasound*. Tech. Print.

Kutarnia, Jason, and Peder C. Pedersen. "Finite Element Method for Whole Body Deformation Using Organ-specific Mechanical Properties." Proc. of Medicine Meets Virtual Reality 18. Print.

Livescribe Platform SDK. Vers. 1.3.0-REV-B. Livescribe Inc., 2007. Program documentation

Lövquist E, O'sullivan O, Oh’Ainle D, Baitson G, Shorten G and Avis N, “VR-Based Training and Assessment in Ultrasound-Guided Regional Anesthesia: From Error Analysis to System Design, *MMVR18*, Newport Beach, CA, Feb. 2011, pp. 304 - 310, 2011.

Maul H, Scharf A, Baier P, Wüstemann M, Günter HH, Gebauer G, Sohn C, “Ultrasound simulators: experience with the SonoTrainer and comparative review of other training systems,” *Ultrasound in Obstetrics and Gynecology*, Vol. 24, pp. 581 – 585, 2004.

Merz E, “Der Ultraschallsimulator – eine ideale Ergänzing zur Erlernung der fetalen Fehlbildingsdiagnostik oder eine Illusion,“ *Ultraschall in Med.*, vol. 27, pp. 321 – 323, 2006.

"MD, /MT, /LD (Use Run-Time Library)." *MSDN | Microsoft Development, Subscriptions, Resources, and More*. Web. 21 Feb. 2011. <http://msdn.microsoft.com/en-us/library/2kzt1wy3.aspx>.

"The National Library of Medicine's Visible Human Project." *National Library of Medicine - National Institutes of Health*. Web. 11 July 2011. <http://www.nlm.nih.gov/research/visible/visible_human.html>.

Pedersen, Peder. Personal Communication. Dec. 2010.

Petrinec K, Savitsky E and Hein C, "Patient-Specific Cases for an Ultrasound Training Simulator," *MMVR18,* Newport Beach, CA, Feb. 2011, pp. 447 - 453, 2011.

"Obtain Anoto Pattern Paper - DeviceLab." Web. 25 May 2010.
<https://wiki.cs.umd.edu/DeviceLab/index.php?title=Obtain_Anoto_pattern_paper>.

OpenGL - The Industry Standard for High Performance Graphics. Web. 20 Feb. 2011. <http://www.opengl.org/>.

Satava RM, "Identification and reduction of surgical error using simulation," *Minim Invasive Ther Allied Technol*. 2005. Vol. 14, pp. 257 - 61, 2005.

Schallware Ultrasound Simulator. Web. 03 June 2011. <http://www.schallware.de/english/index.html?gclid=CKOYyJ2L1KkCFQp75QodqimGPw>.

SpacePoint Fusion. Vers. June 2010. PNI Sensor Corporation, 2010. *User Manual.*

SpacePoint Gaming. Vers. July 20 2010. PNI Sensor Corporation, 2010. *3D sensor for Gaming Data Sheet.*

"Ultrasound Simulation." *EMD Services Ltd - Marketing the MEDEX AD2000 Injector System to UK Hospitals and Their Service Providers.* Web. 03 June 2011. <http://www.emdservices.com/spec.php>.

"Ultrasound Training Simulator." *MedSim | Advanced Ultrasound Simulation*. Web. 03 June 2011. <http://www.medsim.com/ultrasim.html>.

"University of Delaware – Dept. of Health, Nutrition and Exercise Science – Sphere Fitting" Web. 10 Apr. 2011

<http://www.udel.edu/HNES/HESC427/Sphere%20Fitting/LeastSquares.pdf>.

"Voreen – Volume Rendering Engine." Web. 20 Feb. 2011. <http://www.voreen.org>

Zarchan, Paul, and Howard Musoff. "Chapter 16 - Assorted Techniques for Improving Kalman-Filter Performance." *Fundamentals of Kalman Filtering: a Practical Approach*. Reston, VA: American Inst. of Aeronautics and Astronautics, 2005. Print.

A Appendix – IMU Test Reports

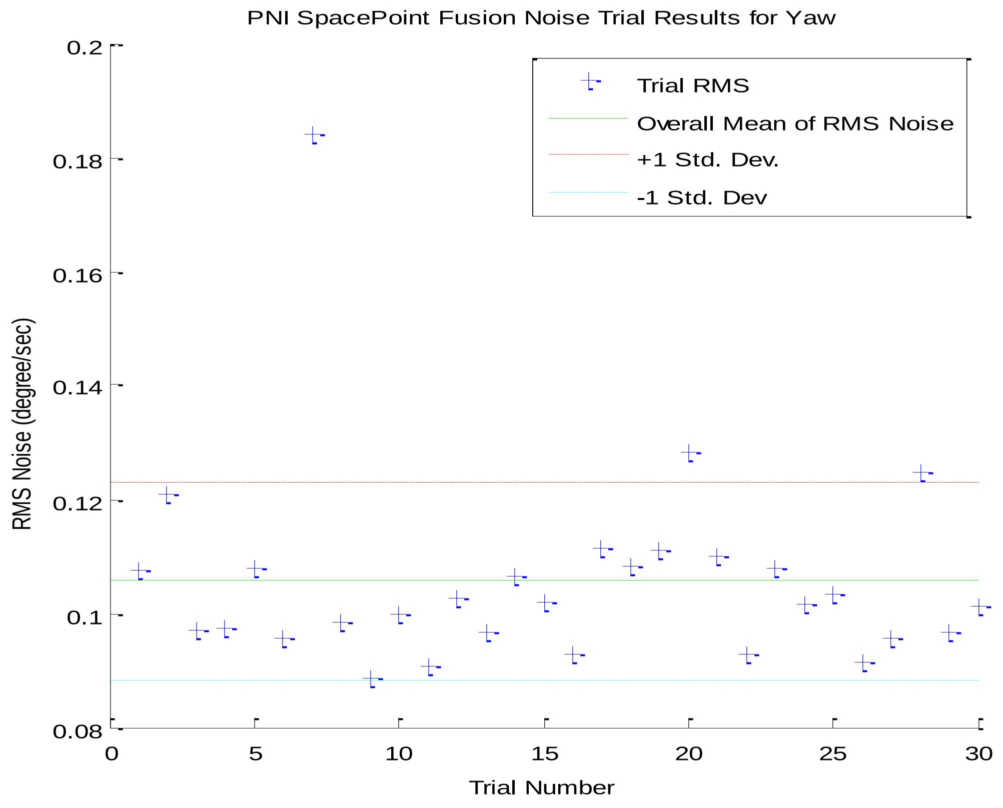


Figure A1 – PNI SpacePoint Fusion sensor static test yaw angle data summary.

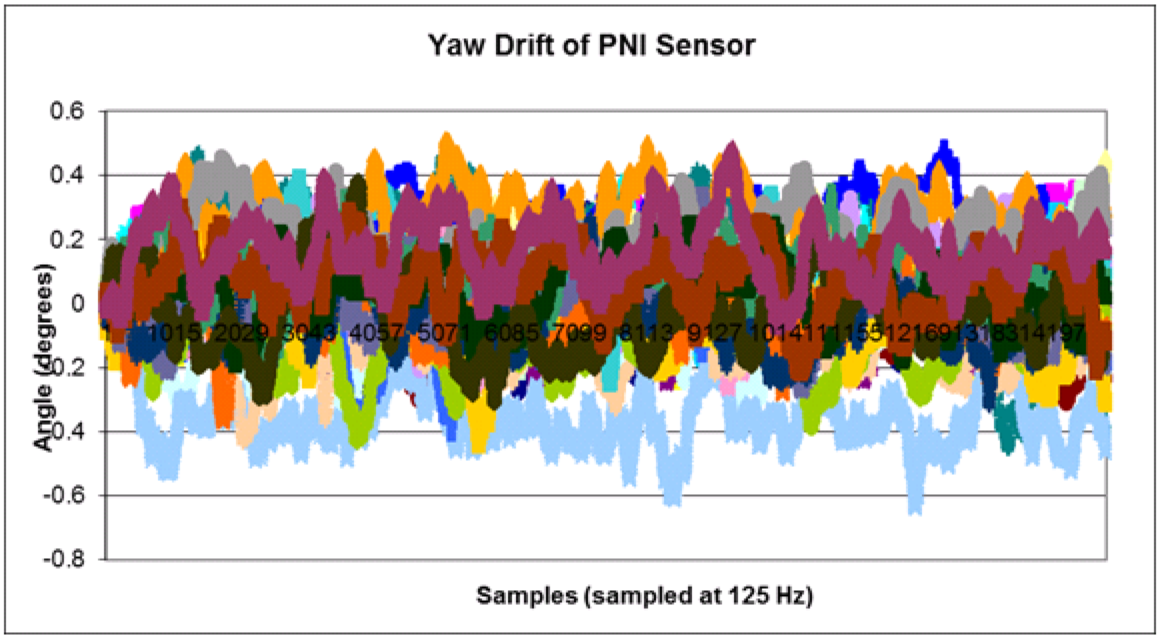


Figure A2 – PNI SpacePoint Fusion sensor module yaw angle static test position data for 30 trials, each over 2 minute testing periods.

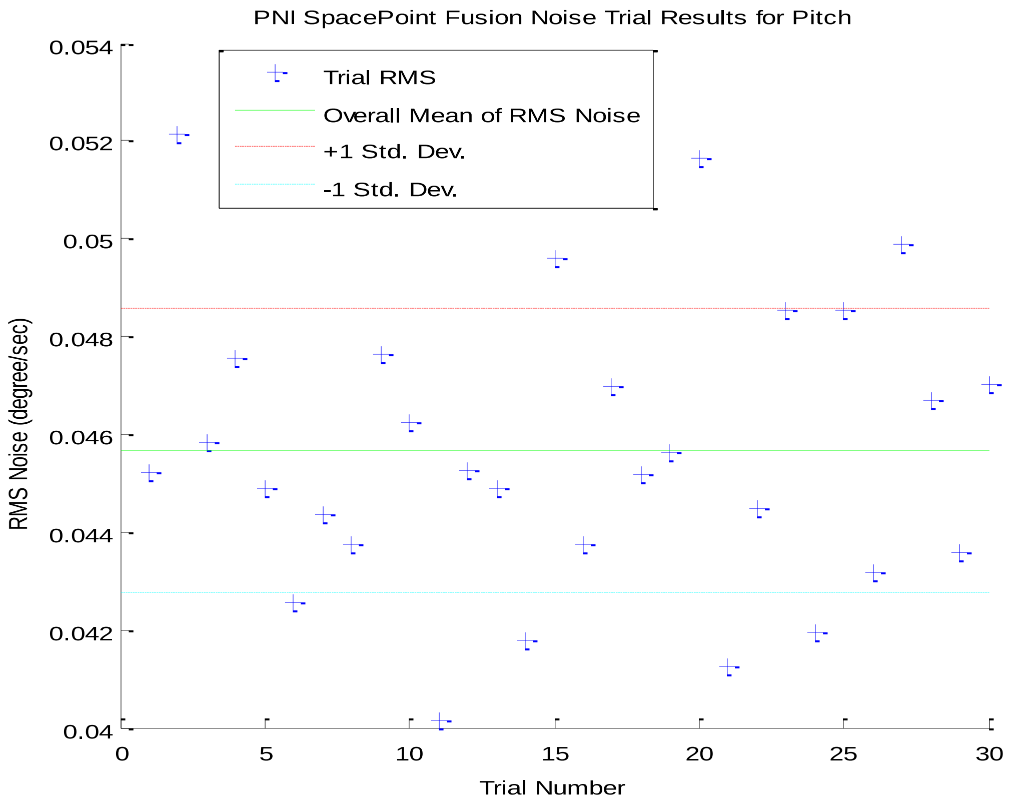


Figure A3 – PNI SpacePoint Fusion sensor static test pitch angle data summary.

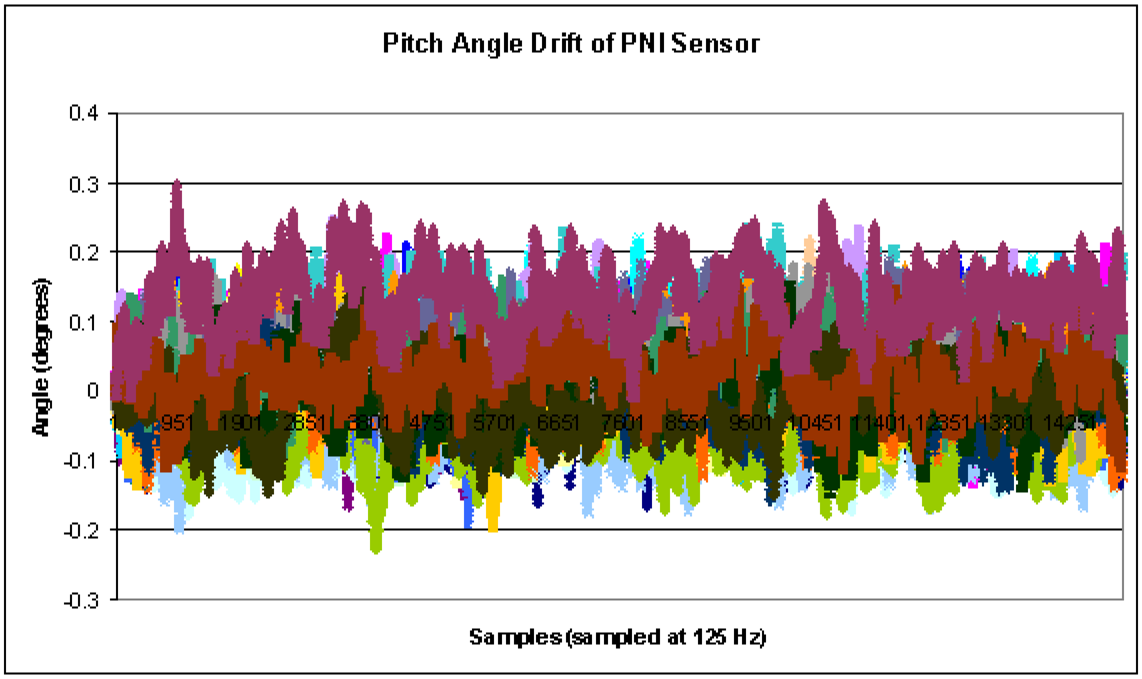


Figure A4 – PNI SpacePoint Fusion sensor pitch angle static test position data over 2 minute testing periods.

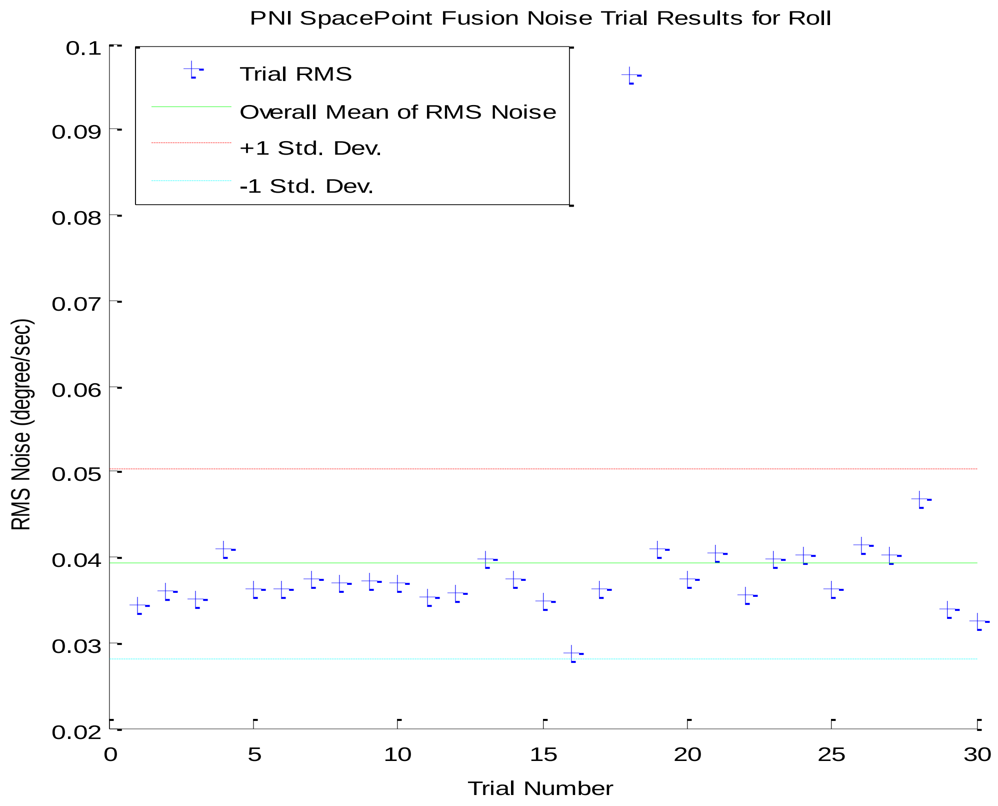


Figure A5 – PNI SpacePoint Fusion sensor static test roll angle data summary.

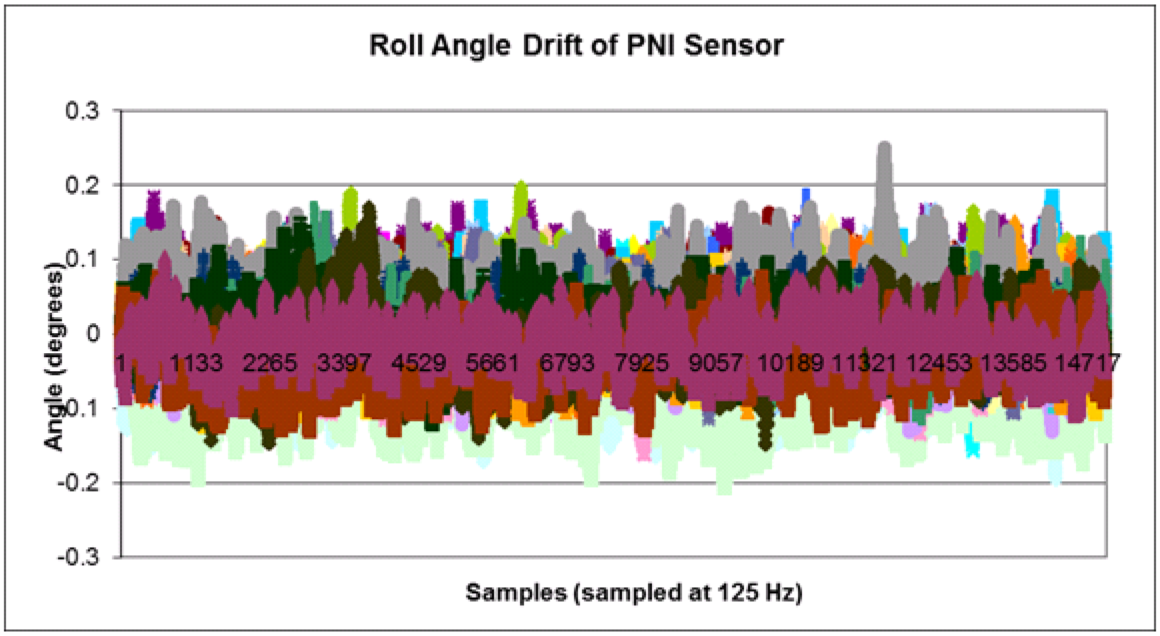


Figure A6 – PNI SpacePoint Fusion sensor module roll angle static test position data for 30 trials, each over 2 minute testing periods.

B Appendix – Anoto Technology Evaluation

The Ultrasound Research Laboratory obtained a Pulse Livescribe digital pen which uses the Anoto pattern technology to perform its tasks. The technology within this device was considered for use within the second generation ultrasound training system. All tests discussed in this section were performed using the Pulse Livescibe digital pen.

Curvature Test

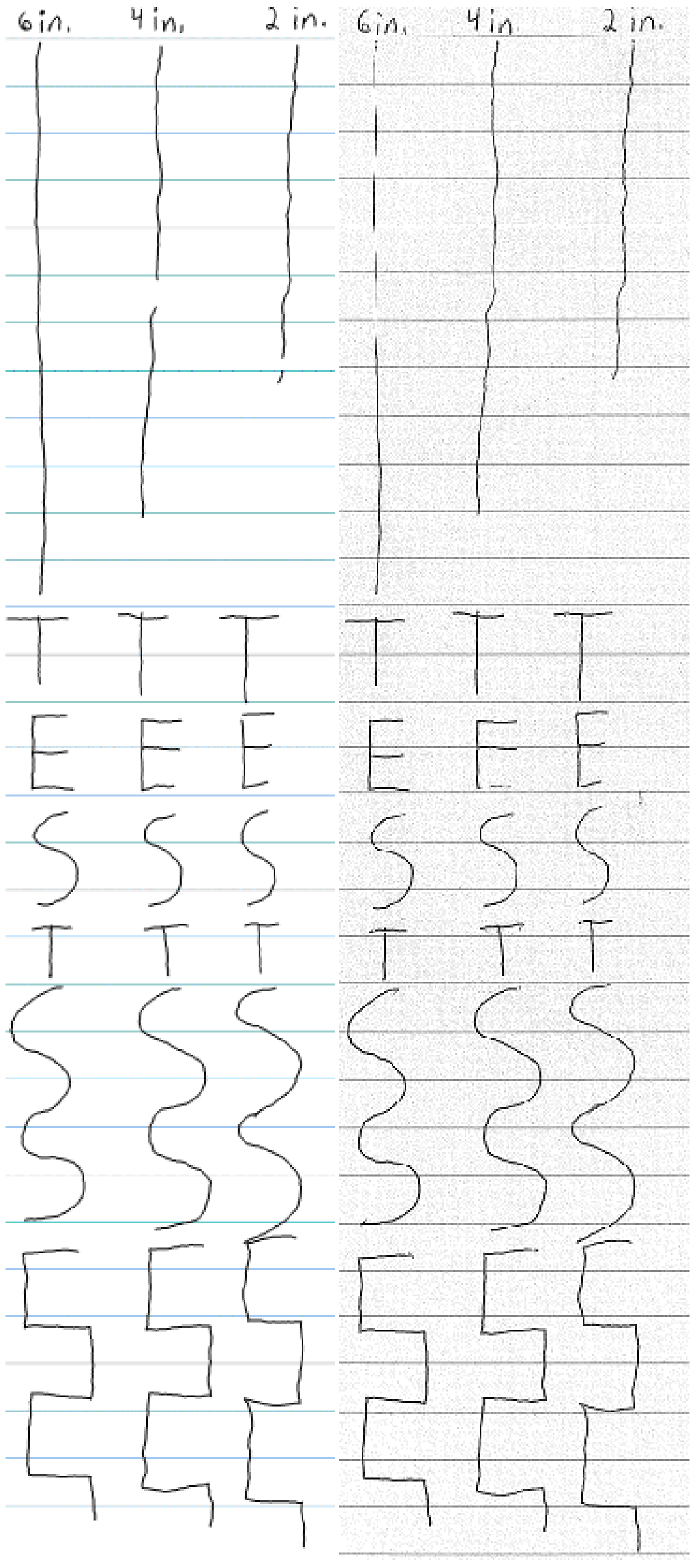


Figure B1 - Right image contains a scanned version of Anoto page that curvature tests used. Left image contains uploaded data from Livescribe pen generated during curvature testing.

Ambient Light Test

This section discusses the test methods used to determine the operability of the Anoto pattern and digital pen under different lighting conditions. Three different lighting conditions were chosen for this experiment. Each of the experiments was conducted on a flat desk surface within the WPI Ultrasound Laboratory.

The first lighting condition was the non-interfered illumination of the fluorescent lights from the ceiling of the laboratory. Also all doors were closed in order to prevent external light from entering the laboratory. This lighting condition will provide optimal lighting condition for the use pen's optical sensor.

The second lighting condition was identical to the first lighting condition with one exception. This exception was the use of a cardboard surface to obstruct the direct path of the light from the ceiling. The cardboard surface was held approximately one half inch from the top of the digital pen when held in the writer's hand. This scenario created a less than optimal lighting condition of dim light for the pen's optical sensor.

The third lighting condition was similar to the first lighting condition, but with the laboratory lights turned off. This created a worst-case scenario for the pen's optical sensor. The pen's optical sensor had minimal to no external light available.

The results of the discussed tests can be seen in Figure B2. The leftmost writing column contains the test results from the first condition. The middle writing column contains the test results from the second lighting condition. The rightmost writing column contains the test results from the third lighting condition.

Each test appeared to be independent of the lighting conditions applied to them. This may suggest that the pen contains its own infrared lighting source. This may be an indication of this sensor having high performance when applied to a surface with minimal light available.

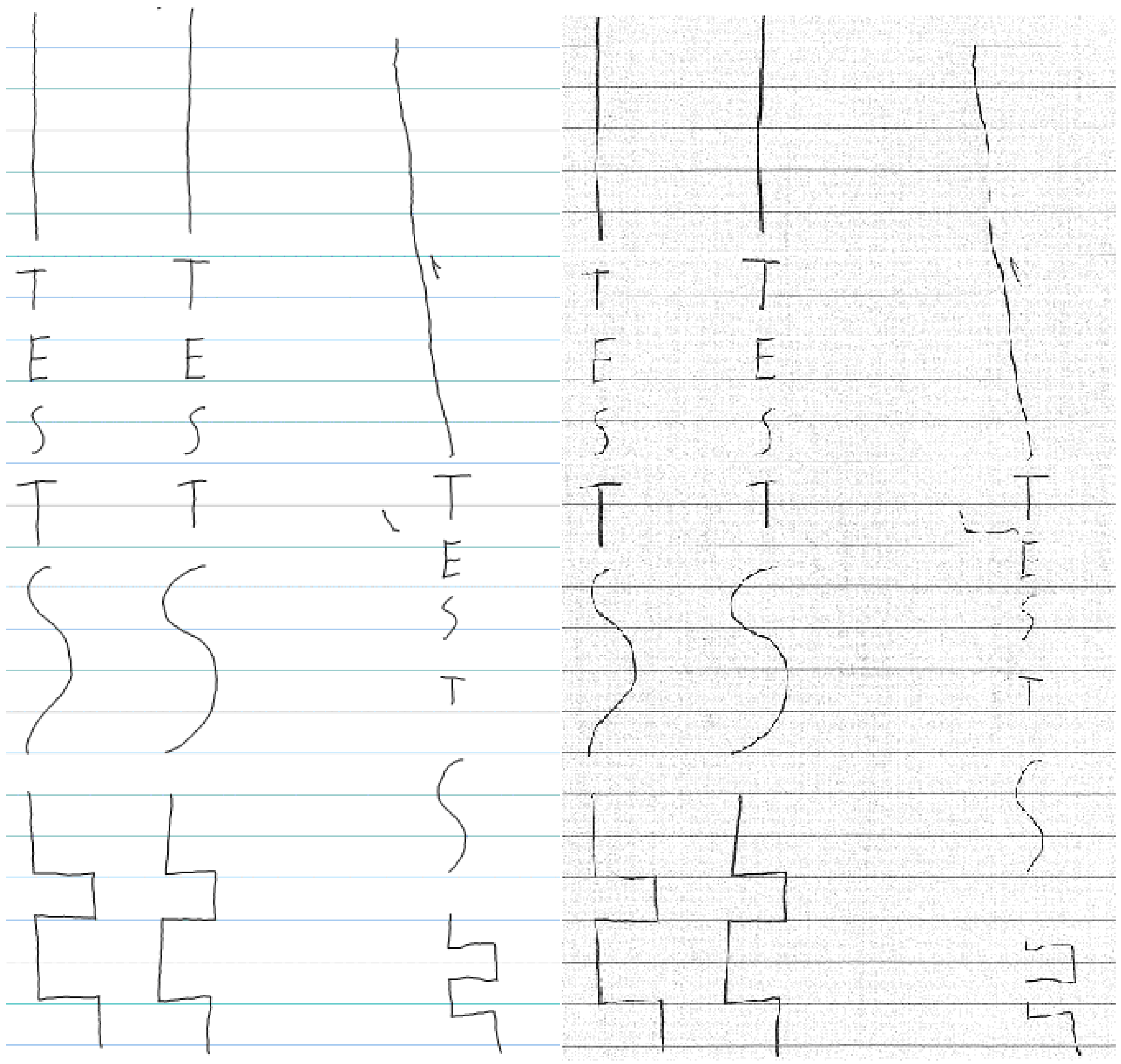
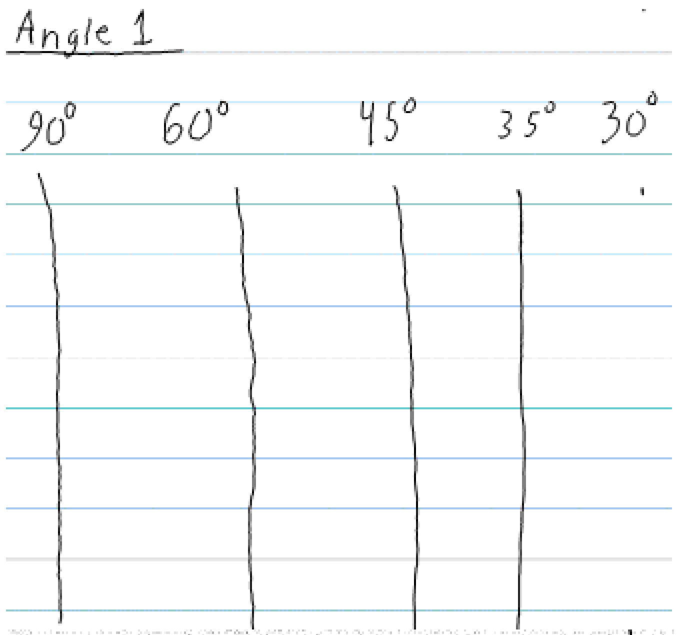


Figure B2 - Right image contains a scanned version of Anoto page that ambient lighting tests used. Left image contains uploaded data from Livescribe pen generated during ambient light testing.

Pen Angle Relative to Surface Test



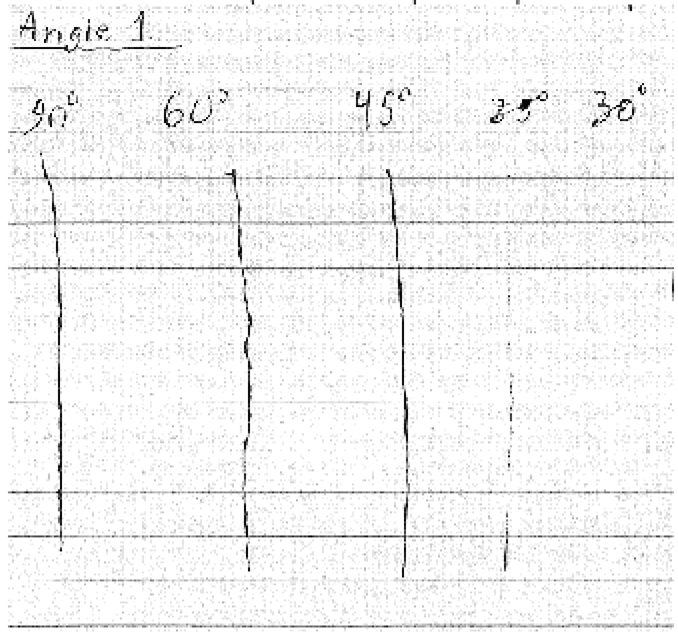


Figure B3 - Top image contains uploaded data from Livescribe pen generated during the pen angle testing for angle one. Bottom image contains a scanned version of Anoto page that was used during pen angle testing for angle one.

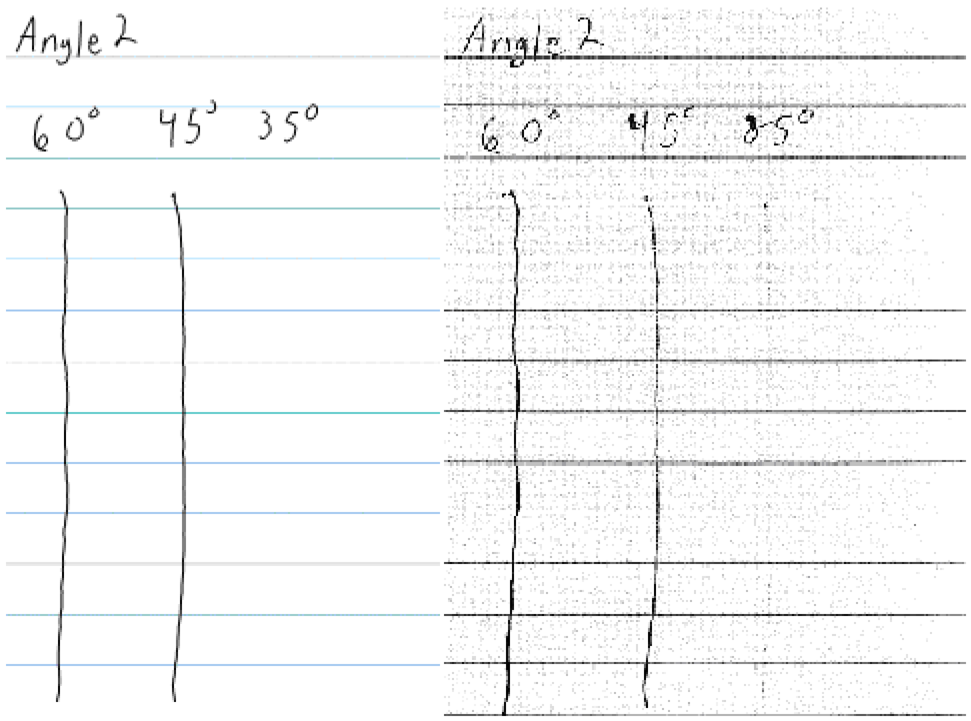


Figure B4 - Top image contains uploaded data from Livescribe pen generated during the pen angle testing for angle two. Bottom image contains a scanned version of Anoto page that was used during pen angle testing for angle two.

C Appendix – Printing of an Anoto Pattern

When the printed pattern is crucial to the application, an accurate form of printing is needed. The printing of the Anoto pattern is possible, but may prove to be difficult. There are many variables involved with the printing of the pattern. Some of the variables involved are the DPI and image processing of the printer.

One variable which needs to be considered is the ability to print at a resolution of 600 DPI (or a multiple there of). [Device Lab] Many modern printers allow a resolution option that may be changed to 600 DPI. This will most likely be the easiest variable to overcome.

Another variable that must be accounted for is the image processing that many printers perform. Many printers will automatically perform alterations in order to enhance text or pictures. Scaling and offsets of the image are examples of image manipulation that may occur. [Livescribe] Any image processing performed on the Anoto pattern could alter the pattern and render the pattern useless. This possible alteration is why the printer used must not perform any image processing to the pages printed. One must use a printer that does not perform any image processing or have an option to turn off any additional processing.

A point worth noting is the amount of time for printing that is to be expected. The large number of dots that is the Anoto pattern may cause a severe increase in printing duration. Livescribe recommends using postscript in order to print the pattern faster. [Livescribe]

D Appendix – System Evaluation Data Report

These results are referenced in section 5.2.1.2.

Position (1,1):

Table D.1 – Position Error Statistics for System Evaluation in Position (1,1) using OpenGL units

	mean	median	mode
x	4.8E-05	2.23E-05	0.000464
y	-8.2E-05	-7.6E-05	-0.00048
z	-5.6E-05	-1E-06	3.38E-05
magnitude	0.000525	0.00052	0.000665

Table D.2 – Orientation Error Statistics for System Evaluation in Position (1,1) in Unit Quaternion Components

	mean	median	mode
q0	8.58E-08	5.76E-08	-2.4E-06
q1	4.24E-08	4.33E-08	-3E-06
q2	-2.7E-07	-2.5E-07	-2.4E-06
q3	9.78E-08	1.26E-07	-6.2E-06

Table D.3 – Additional Displacement Compensation Position Error Statistics for System Evaluation in Position (1,1) using OpenGL units

	mean	median	mode
x	-9.53783	-10.9099	-11.2314
y	4.614208	5.14681	5.14681
z	9.703868	10.7246	10.7246

magnitude	14.38552	16.36004	16.36004

Position (1,2):

Table D.4 – Position Error Statistics for System Evaluation in Position (1,2) using OpenGL units

	mean	median	mode
x	-5.5E-05	-0.00014	-0.00019
y	6.57E-05	6.9E-05	0.000313
z	-3.6E-05	6.83E-05	-0.00044
magnitude	0.000492	0.000551	0.000571

Table D.5 – Orientation Error Statistics for System Evaluation in Position (1,2) in Unit Quaternion Components

	mean	median	mode
q0	-4.6E-08	-5E-08	-1.1E-06
q1	-4.8E-07	-4.9E-07	-1.4E-06
q2	2.72E-07	2.5E-07	-7.1E-07
q3	-1.1E-07	-1.2E-07	-1E-06

Table D.6 – Additional Displacement Compensation Position Error Statistics for System Evaluation in Position (1,2) using OpenGL units

	mean	median	mode
x	0.006234	-0.05293	-0.37554
y	2.584433	2.59488	3.19131
z	7.648897	7.79866	7.40173
magnitude	8.117985	8.268991	8.069145

Position (1,3):

Table D.7 – Position Error Statistics for System Evaluation in Position (1,3) using OpenGL units

	mean	median	mode
x	-6.5E-05	-0.00013	-0.00032
y	8.94E-05	5.23E-05	5.23E-05
z	-3.3E-05	-9.4E-05	-0.0002
magnitude	0.000479	0.000458	0.00038

Table D.8 – Orientation Error Statistics for System Evaluation in Position (1,3) in Unit Quaternion Components

	mean	median	mode
q0	-1.3E-07	-1.5E-07	-1.1E-06
q1	4.02E-07	3.85E-07	-4.5E-07
q2	-1.8E-07	-2E-07	-1.2E-06
q3	1.99E-07	1.42E-07	-5.9E-07

Table D.9 – Additional Displacement Compensation Position Error Statistics for System Evaluation in Position (1,3) using OpenGL units

	mean	median	mode
x	9.283445	11.1446	12.4336
y	6.305241	6.55466	6.55466
z	12.73367	13.8414	14.3321
magnitude	17.94837	20.07404	20.07404

Position (2,1):

Table D.10 – Position Error Statistics for System Evaluation in Position (2,1) using OpenGL units

	mean	median	mode
x	6.61E-06	-4.9E-07	-0.00019
y	-3.4E-06	-5.4E-05	-0.00012
z	8.74E-07	-2.6E-05	0.000334
magnitude	0.000433	0.000433	0.000401

Table D.11 – Orientation Error Statistics for System Evaluation in Position (2,1) in Unit Quaternion Components

	mean	median	mode
q0	-3.7E-07	-3.7E-07	-1.3E-06
q1	-4.6E-07	-4.2E-07	-1.8E-06
q2	2.08E-07	2.21E-07	-9.6E-07
q3	5.96E-09	4.11E-08	-2.4E-06

Table D.12 – Additional Displacement Compensation Position Error Statistics for System Evaluation in Position (2,1) using OpenGL units

	mean	median	mode
x	-22.2328	-22.4776	-19.8226
y	-5.27807	-4.33504	-4.33504
z	28.31522	27.5813	24.2284
magnitude	36.47652	36.40221	31.6029

Position (2,2):

Table D.13 – Position Error Statistics for System Evaluation in Position (2,2) using OpenGL units

	mean	median	mode
x	-3.1E-06	-3.4E-05	-3.4E-05
y	5.5E-05	0.000129	0.000392
z	-1.1E-05	-5.9E-05	0.000194
magnitude	0.000461	0.000439	0.000439

Table D.14 – Orientation Error Statistics for System Evaluation in Position (2,2) in Unit Quaternion Components

	mean	median	mode
q0	2.1E-07	2.08E-07	-6.4E-07
q1	4.03E-07	4.08E-07	-6.8E-07
q2	4.67E-08	5.84E-08	-8.4E-07
q3	-4.8E-08	-4.2E-08	-9.8E-07

Table D.15 – Additional Displacement Compensation Position Error Statistics for System Evaluation in Position (2,2) using OpenGL units

	mean	median	mode
x	1.776904	1.94405	2.41788
y	-2.20857	-1.8287	-1.63936
z	31.93989	31.83605	30.9832
magnitude	32.11906	32.02172	31.12061

Position (2,3):

Table D.16 – Position Error Statistics for System Evaluation in Position (2,3) using OpenGL units

	mean	median	mode
x	2.75E-05	9.91E-07	0.000402
y	7E-05	0.000141	0.000141
z	5.49E-06	-8.9E-05	-9.2E-05
magnitude	0.000468	0.00046	0.000436

Table D.17 – Orientation Error Statistics for System Evaluation in Position (2,3) in Unit Quaternion Components

	mean	median	mode
q0	1.7E-07	1.76E-07	-5.2E-07
q1	9.77E-08	8E-08	-1E-06
q2	3.69E-07	3.77E-07	-7.1E-07
q3	7.04E-08	8.16E-08	-7.4E-07

Table D.18 – Additional Displacement Compensation Position Error Statistics for System Evaluation in Position (2,3) using OpenGL units

	mean	median	mode
x	7.725222	-1.18974	-1.75185
y	-23.8576	-32.6527	-32.6606
z	24.98264	19.7546	15.2562
magnitude	40.22753	39.77585	36.09066

E Appendix – One Degree of Freedom Evaluation Results

Table E.1 – Position displacement measures of one degree of freedom input data

	delta1	delta2	delta3	delta4	delta5
x	-23.4762°	23.47621°	0°	-23.4762°	-23.4762°
y	-90.6151°	-90.6151°	0°	-90.6151°	90.61506°
z	0°	0°	184.1757°	0°	0°

Table E.2 – Orientation displacement measures of one degree of freedom input data with no yaw rotation

	pitch -45°	pitch 45°	roll -45°	roll 45°	yaw -90°	yaw 90°
X (Quaternion Component)	0	0	0	0	-0.8509	0.850904
Y (Quaternion Component)	-0.48717	0.487175	0	0	0	0
Z (Quaternion Component)	0	0	0.487175	-0.48717	0	0
W (Quaternion Component)	-0.8733	-0.8733	-0.8733	-0.8733	0.525322	0.525322

Each MEMS sensor within the PNI Fusion returns a value comprised of a value used to calculate a rotational angle of the sensor. Each of these sensors' data also contains an additional noise component. The noise component is important to understand in order to evaluate the performance of a sensor. This noise component can be better understood when the sensors remain in a static position. This section analyzes the values returned from a PNI Fusion in a static position in order to evaluate its performance.

Test Methods

This section discusses the test methods used to determine the effects of sensor noise outputted by the sensor module when in a static position. The sensor was connected to a PC in the Laboratory via a USB connection. The PNI Fusion module was then placed on the flat surface of a desk. Upon power-up and while the sensor is at rest, the PNI Fusion module then took five seconds to perform its one-time auto-calibration phase. [SpacePoint Fusion]

The PNI Fusion was then interfaced with using a C++ program. This program was based on the Fusion GamePad Demo. The demo uses a simple USB HID API to interface with the module. The demo originally consisted of displaying the PNI Fusion output for a set amount of samples. The demo application was expanded in order to convert the quaternion output data of the sensor to Euler angles and then write the Euler angle data to a text file to be used for later analysis. The data were recorded for 15000 samples (approximately 2 minutes). The orientation conversions were done with equations 2.9, 2.10, and 2.11. [Heckbert] The test procedure for each trial is illustrated in Figure E. This experiment was repeated 30 times.

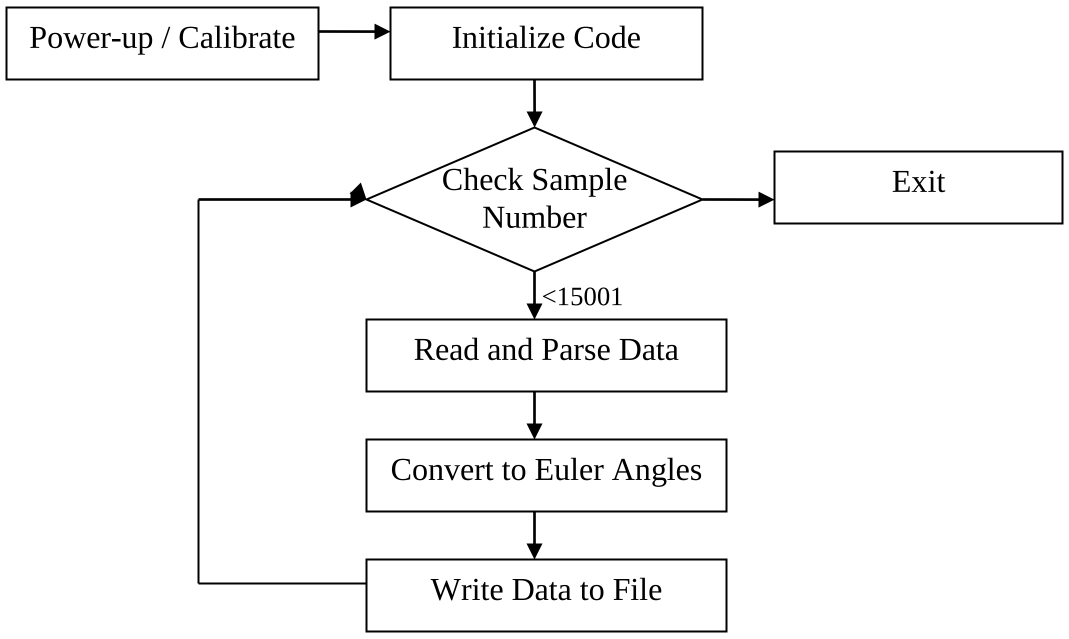


Figure E.1– PNI SpacePoint Fusion sensor static test procedure code diagram.

F Appendix – IMU Static Testing

Test Results and Analysis

The orientation data that had been written to a file were then used for analysis. The auto-calibration of the sensor module prior to the tests allows the sensor to have no transient startup within the tests. The values of the first samples for each trial were then used as an offset subtracted from each of the remaining data for their corresponding trial. The subtraction of these original values allowed for the viewing of deviation from the original output static position angles. The deviation from the original values, while the sensor was in a static position, is the noise (error) of the reported angles from the sensor module. This noise was then analyzed by computing Root-Mean Squares.

The Root-Mean Squared (RMS) noise for each of the tests was calculated using Equation F.1 [Goldsmith]. An angular degree value is represented by ω_{ci}. The average of all angular velocities used from a trial is represented by $\overline{\omega_c}$. N is the number of samples used.

$$RMSnoise = \sqrt{\frac{1}{N}\sum_{i=N}^{N}(\omega_{ci} - \overline{\omega_c})^2} \qquad \text{F.1}$$

All averages were calculated using Equation F.2. The average is represented by m. N is the number of samples used. An angular degree value is represented by ω_i.

$$m = \frac{1}{N}\sum_{i=1}^{N}\omega_i \qquad \text{F.2}$$

The results of the discussed tests can be seen in the following Figures and Table. Table numerically summarizes the results of the sensor module. The full results of the PNI Fusion output data with respect to the yaw of the sensor are shown in Figures A1 and A2 of Appendix A. The results of the PNI Fusion with respect to the pitch of the sensor are shown in Figures A3 and A4 of Appendix A. The results of the PNI Fusion with respect to the roll of the sensor are shown in Figures A5 and A6 of Appendix A. A diagram depicting the orientations of yaw, pitch, and roll in reference to the PNI Fusion module for this study can be seen in Figure 3.4.

Table F.1 – Numerical results from PNI SpacePoint Fusion sensor testing.

	Yaw Angle Output	Pitch Angle Output	Roll Angle Output
RMS Noise Average (deg/sec)	0.1057	0.0457	0.0392
Worst RMS Noise (deg/sec)	0.183877	0.052131	0.096415
Drift Rate Average (deg/min)	0.016769	0.020171	0.001429
Maximum Angle Deviation (deg)	0.514	0.29836	0.249606