Surface Mesh-Based Ultrasound Simulator For Spinal Interventions

by

Laura Bartha

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
July 2013

Abstract

Purpose: Ultrasound is prevalent in image-guided therapy as a safe, inexpensive, and widely available imaging modality. However, extensive training in interpreting ultrasound images is essential for successful procedures. An open-source ultrasound image simulator was developed to facilitate the training of ultrasound-guided spinal intervention procedures, thereby eliminating the need for an ultrasound machine from the phantom-based training environment. **Methods**: Anatomical structures and surgical tools are converted to surface meshes for data compression. Anatomical data is converted from segmented volumetric images, while the geometry of surgical tools is available as a surface mesh. The pose of the objects are either constants or live measurements from a pose tracking device. Intersection points between the surface models and the ultrasound scan lines are determined with a binary space partitioning tree. The scan lines are divided into segments and filled with grey values determined by an intensity calculation accounting for material properties, reflection, and attenuation parameters defined in a configuration file. The scan lines are then converted to a regular brightness-mode ultrasound image. **Results:** The simulator was tested in a tracked ultrasound imaging system, with a mock transducer tracked by an Ascension TrakStar electromagnetic tracker, on a spine phantom. A mesh model of the spine was created from CT. The simulated ultrasound images were generated at a speed

of 50 frames per second, and a resolution of 820 x 616 pixels on a PC with a 3.4 GHz processor. A human subject trial was conducted to compare the learning performance of novice trainees with real and simulated ultrasound in the localization of the facet joints of a spine phantom. With 22 participants split into two equal groups and each participant localizing 6 facet joints, there was no statistical difference in the performance of the two groups, indicating that simulated ultrasound could indeed replace the real ultrasound in phantom-based ultrasonography training for spinal interventions. **Conclusion:** The ultrasound simulator was implemented and integrated into the open-source Public Library for Ultrasound (www.plustoolkit.org) and SlicerIGT (www.SlicerIGT.org) toolkits **Keywords:** ultrasound; simulation; surface mesh; training

Statement of Co-Authorship

The work described in this thesis was completed under the supervision of Dr. Gabor Fichtinger, and with the guidance of Dr. Andras Lasso. The experiments described herein were conceived and supported by Dr. Tamas Ungi, with validation of the results provided Dr. Zsuzsanna Keri, while Csaba Pinter aided in the revisions and optimization of the code.

Much of the work presented in this thesis has been described in an article recently accepted for publication in the International Journal of Computer Assisted Radiology and Surgery (IJCARS)(Bartha L, Lasso A, Pinter C, Ungi T, Fichtinger G, Open-Source Surface Mesh-Based Ultrasound Simulator, International Journal of Computer Assisted Radiology and Surgery, DOI: 10.1007/s11548-013-0901-z (accepted May 16, 2013))

Acknowledgments

I would like to thank my supervisor, Gabor Fichtinger, for his support and encouragement throughout my graduate work and undergraduate project. For whatever reason, he decided that I was worth the investment, and has coaxed work out of me that I did not believe myself to be capable of.

I would like to thank my colleagues in the Perk Lab for their companionship and mentorship. I am indebted to Andras Lasso for taking on the role of a second supervisor, and his incredible amount of patience in teaching me practical programming skills. Csaba Pinter, Adam Rankin, and Tamas Ungi have always been available to discuss new problems arising through the course of the project.

I would also like to thank my friends, especially everyone on the Queen's Varsity Fencing Team (Sydney Houston-Goudge, Jamie MacDonald, Jimmy Wintle, Kathryn Papke, Sarah Kim, and Julia Meeraker), for having kept me grounded and providing a wonderful atmosphere to destress. I must give a special thanks to Simrin Nagpal for the many venting sessions, and to Anna Belkova, as the three of us started our journey into higher education together six years ago. Additionally, I would like to thank the Disk Jockeys and Danger Zoneians, for a wonderful experience playing ultimate frisbee in hurricane force winds, pouring rain, falling snow, and blinding sun.

I am grateful to Victor Grzeda for putting up with me when times were stressful, for being a constant source of comfort and joy, and for encouraging me to dream big.

Lastly, I would like to thank my parents, Eva and Janos Bartha, and my sister, Eva Bartha-Terracol, my brother-in-law Thierry Terracol and my niece, Victoria Terracol. You have always been there for me, and I thank you very much for all of your support and love throughout my life.

Glossary

1D One-dimensional

2D Two-dimensional

3D Three-dimensional

3D Slicer A software for visualizing medical images

Analgesic Pain relieving/pain numbing

BSP Tree Binary space partitioning tree

C++ A programming language

CT Computed tomography

Facet Joint The joint created where two vertebrae overlap

FEM Finite element model

Fiducial A known, fixed, point usually used for comparison

Filter A componont of an image processing pipeline that alters its input and produces the new data as output

GPU Graphics processing unit

IGT Image-guided therapy

Lumbar Spine The lowest five vertebrae in the spine, just above the pelvis

Mann-Whitney U-Test A statistical test used when the data is not normally distributed to determine whether there is a statistical difference between two groups of data

MHA File Meta image file

MRI Magnetic resonance imaging

MRI Open graphics library

PC Personal computer

Perk Tutor An ultrasond-guided needle placement training system

Phantom A container containing a three dimensional plastic printout of a spine, filled with a tissue simulating plastic

PLUS Public software library for ultrasound imaging research

US Ultrasound

VTK Visualization toolkit

XML Extensible markup language

Contents

List of Figures

List of Code Listings

List of Tables

Chapter 1

Introduction

Minimally invasive surgical procedures have become the surgery of choice in recent years. This is due to the many advantages they have over traditional surgery, including reduced tissue damage to the patient and shorter hospital stays. To make minimally invasive surgery possible, medical images are used to allow the surgeon to see within the body and to aide in surgical navigation. Ultrasound is often used for this purpose, as it does not require either the surgeon or the patient to undergo harmful radiation as a computed tomography(CT) scan would, and does not require the magnetization of magnetic resonance imaging (MRI). This makes ultrasound inherently portable and inexpensive; a natural choice for diagnostic imaging and for image guidance during surgery. However, extensive training is necessary to make use of its full potential. Ultrasound is notorious for providing images with features that are only discernible with experience. This is problematic in that an ultrasound machine, as well as access to a patient on a regular basis, would be needed for every trainee to develop the skills of ultrasound diagnosis and image interpretation. Financially, this training scenario is taxing. Simulation could be the solution to this problem, potentially relieving some of the financial burden associated with obtaining numerous

ultrasound machines and increasing the efficiency of the training process.

1.1 Motivation

Current physically accurate simulators require significant processing power and time, making them poorly suited for image-guided therapy (IGT). Commercial systems are not only expensive, but are lacking in versatility, often being limited to a single application. Both the research community and future trainees could benefit from an open-source implementation of an ultrasound simulator, especially one that is inherently within an ultrasound image processing framework such as PLUS. The end goal is to minimize the cost and maximize the accessibility of ultrasound-guided therapy training, resulting in a greater number of skilled physicians performing these delicate surgical procedures, which in turn increases patient access to ultrasound-guided care. A larger pool of capable physicians could lead to shorter wait times for patients in need of ultrasound-guided interventions.

1.2 Proposed Methodology

The work presented here focuses on ultrasound simulation for spinal intervention training scenarios using phantoms, and hopes to one day eliminate expensive ultrasound machines from such training scenarios. In typical phantoms, soft tissues usually have little or no speckle and, especially in musculoskeletal phantoms, bony structures dominate the view. The proposed ultrasound simulation method will exploit this observation. It can be divided into the following three stages:

1. Simulator Development

2. System Integration

3. Human Operator Study

1.2.1 Simulator Development

The simulator is a component of the Public Library for UltraSound (PLUS) library [1] (www.plustoolkit.org), and is implemented in the C++ programming language as a VTK (Visualization Toolkit) [2] (www.vtk.org) filter to facilitate ease of integration and use within the library. Different methods of simulating a binary, cross-sectional ultrasound image are explored including:

1. Combining existing VTK filters

2. Altering an existing VTK filter

3. A variant of Ray Tracing utilizing VTK filters

The simulation is generated from a surface mesh model of a spine. This surface mesh is obtained from a segmented CT volume. However, a surface mesh obtained from any 3D medical image volume would produce the same effects. This addresses the need for the simulator to reach as many people as possible. In addition to the surface mesh, the simulator also needs a matrix containing the transformations between the surface mesh model and the location of a virtual transducer. This transformation is supplied from either prerecorded data or acquired directly from a tracker. Alternately, virtual transducer positions can also be specified in a configuration file. The simulator produces a vtkImage as its output, which can then be easily incorporated into other components of the PLUS Library.

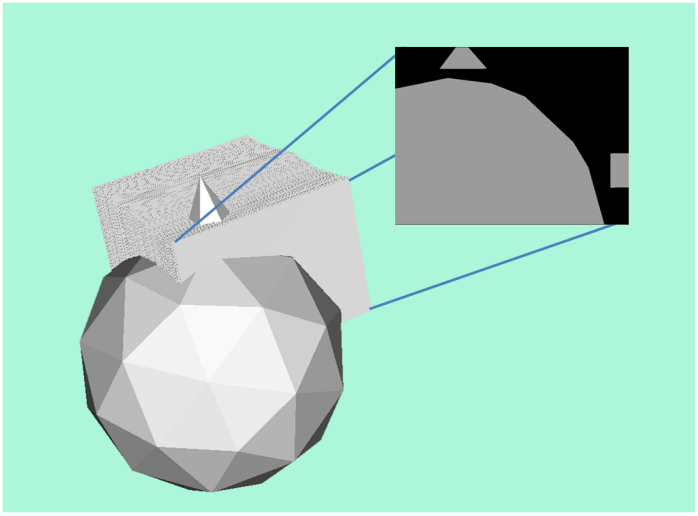

Figure 1.1: Sample of a binary cross-sectional image

1.2.2 Simulator Integration

The simulator is used with other software, PLUS server and 3D Slicer (www.slicer.org) [3], to create a training system where a trainee can move a fake transducer and see the corresponding simulated ultrasound image on screen. This is achieved by displaying the simulated images in 3D Slicer in the proper position. 3D Slicer receives this image and position information from Plus Server, a component of the same PLUS library which the simulator itself is a part of. The next step is to fully integrate the simulator into the Perk Tutor ultrasound-guided spinal injection training system [4], replacing the current ultrasound component.

1.2.3 Human Operator Study

With the simulator functioning in the Perk Tutor environment, its efficacy as a training tool needs to be determined. A set of students, novices to spinal ultrasound, are gathered and divided into two groups. One of the groups learns to perform a

task relating to a spinal procedure using real ultrasound, while the other group uses simulated ultrasound. After the allotted time for learning has elapsed, the trainees are assessed. Statistical analysis of the results follow once the study is complete, with validation of the results by a trained physician. Details of this study are provided in Chapter 4.

1.3 Thesis Objectives

The thesis objective can be divided into 4 goals:

1. Generate a binary, cross sectional image from a surface mesh and vtkTransform
2. Simulate boundary reflection
3. Test effectiveness of this "bare bones" simulation approach with a human operator study
4. Extend simulator frame work to prepare for multiple model simulation in the next generation simulator

1.4 Thesis Contributions

In this thesis, I have:

- Researched existing and previously developed ultrasound simulators
- Created a VTK filter to generate a binary cross-sectional image by:
  - Using existing VTK filters
  - Altering an existing VTK filter employed in the simulator

  - Creating a VTK filter that employs Ray Tracing and finds intersection points with the help of a Binary Space Partitioning Tree

- Simulated reflection at tissue/bone boundaries
- Developed the simulator as a part of the PLUS (Public Software Library for Ultrasound Imaging Research www.plustoolkit.org) toolkit
- Ensured international access to the simulator through the Slicer IGT (www.slicerigt.org) module for 3D Slicer
- Conducted a human operator study to determine the effectiveness of the simulator as a training tool
- Extended the simulator to use a specific class for spatial models, relocating the image intensity calculation

1.5 Thesis Outline

The rest of the thesis is divided into chapters as follows:

Chapter 2, Background: Chapter 2 provides an overview of previous work on ultrasound simulation, including both generative approaches, which model ultrasound propagation, and interpolative approaches which work with pre-acquired volumetric medical data. The interpolative approaches are further examined as those dealing with prerecorded ultrasound volumes and those altering CT volumes. Ultrasound simulation based on CT volumes utilizing the graphics processing unit (GPU) is also examined. In addition, a quick primer on ultrasound physics is given, and a basic overview of relevant components of the Perk Tutor, 3D Slicer, PLUS, and VTK.

Chapter 3, Methods: Chapter 3 details the progression through the three different methods used to achieve a binary cross-sectional image from a surface model and a transformation. In addition, simulating reflection at tissue/bone boundaries is discussed. Images of the output of the functional simulator are also shown.

Chapter 4, Experiments and Results: Chapter 4 elaborates on the testing done to ensure the ultrasound simulator works with the PLUS Library in a 3D Slicer environment. As well, this chapter explains the protocol and setup used during the human operator study, including details pertaining to the statistical analysis of the results.

Chapter 5, Conclusions and Future Work: Chapter 5 presents the conclusions of the work, and focuses on the practical achievement of an open-source ultrasound simulator. Future work is also discussed.

Chapter 2

Background

2.1 Clinical Motivation and Application

Chronic back pain is a symptom affecting millions of people in the United States alone [5]. Often the source of this back pain is the degeneration of the sinovial (facet) joints in the spine. Treatment involves the injections of analgesics into the fluid between the facet joints using a needle [6]. Finding the proper location for needle insertion requires many hours of training and practice, despite the fact that ultrasound is used to visually guide the surgeon. Part of the difficulty is manipulating the needle and the ultrasound probe at the same time; a difficulty that has been addressed in the form of a training simulator [4] utilizing a spinal phantom. The other half of the problem is learning to interpret the noisy and confusing ultrasound images; a problem that is the focus of the work presented in this thesis. The solution presented is the creation of an ultrasound simulator to be integrated into a needle placement navigation system ([4]).

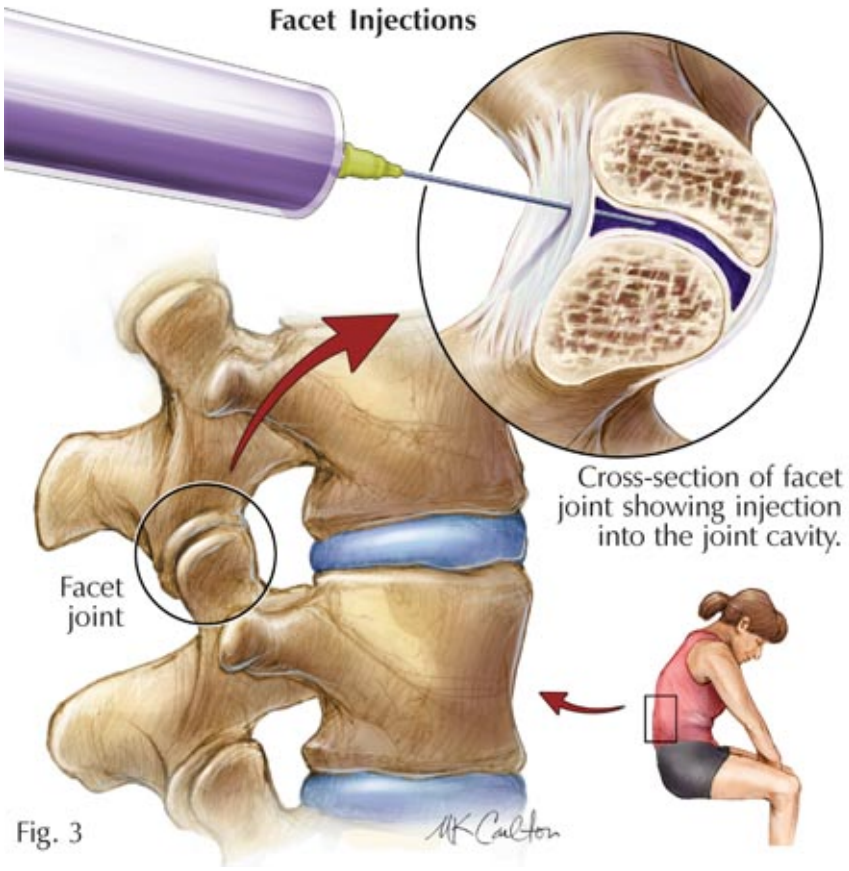

Figure 2.1: Facet joint injection in the lumbar spine
Source: Spine Institute Northwest
https://www.spineinstitutenorthwest.com/treatments/interventional-spinal-injection-techniques/

2.2 Basic Ultrasound Physics

Ultrasound physics can become very complicated very quickly. This work sought only to account for two basic properties, attenuation, and reflection at tissue boundaries, which appear to be the most important and dominating aspects of skeletal ultrasound imaging. Bone was assumed to be a hard surfacec with uniform density throughout.

2.2.1 Reflection

As a sound wave encounters a large interface, a portion of the wave is reflected back towards the source. The reflections cause the outlines of organs and bones in ultrasound images. Interfaces occur between two neighbouring regions of tissue with different acoustic impedance values, that is, different resistance to sound passing through the region. Acoustic impedance is a product of the density of the tissue and the velocity of sound in the specific tissue [7].

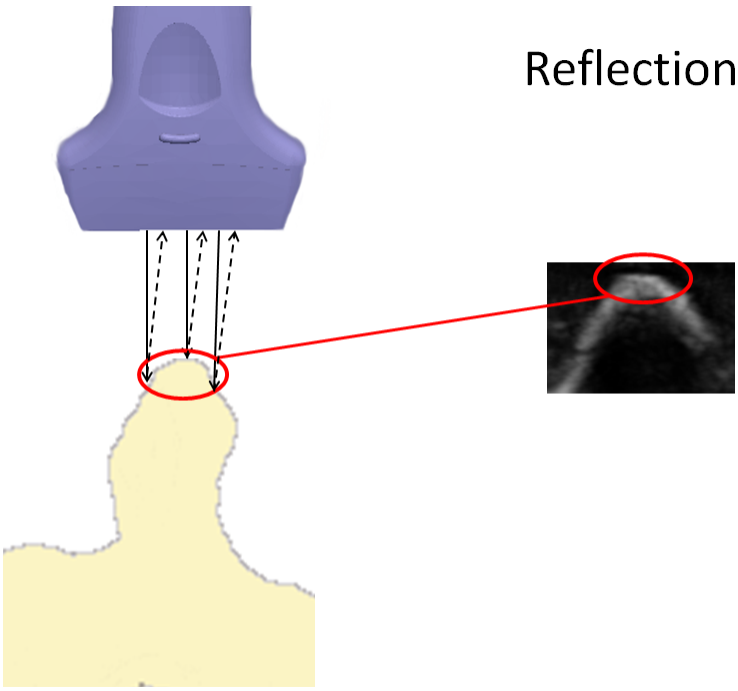

Figure 2.2: Ultrasound waves emitted from the transducer are reflected by the bone boundaries back to the transducer, resulting in an ultrasound image

2.2.2 Attenuation

Attenuation is dependent on the frequency of the ultrasound wave, and the distance the wave has traveled, in addition to the physical properties of the material the wave is traversing. Attenuation includes both scattering and absorption as it describes the reduction in amplitude of the ultrasound wave. Scattering is also known as nonspecular reflection, and occurs when sound waves are reflected in all directions by very small interfaces. Scattering is responsible for the texture seen in internal organs. Absorption transforms the energy from the ultrasound wave into alternate forms, usually heat. It is the only process where the ultrasonic energy is not just redirected [7].

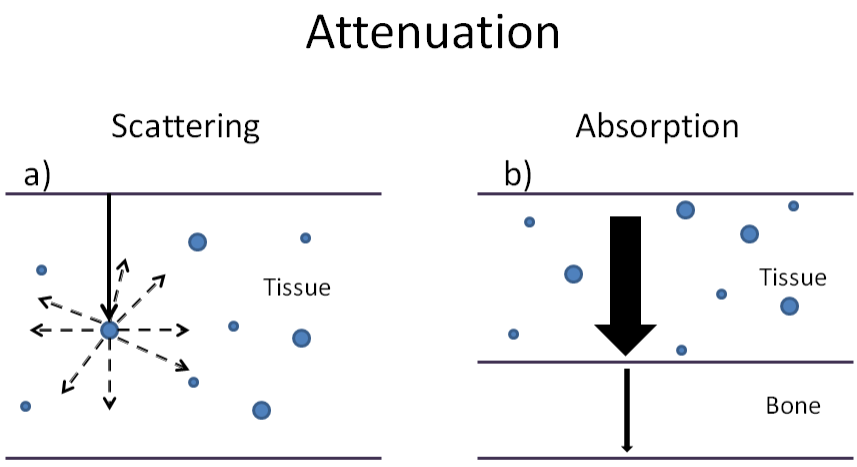

Figure 2.3: a) Details the workings of scattering : an ultrasound wave hits a small interface and is deflected in all directions, the new rays reduced in intensity. b) Details the workings of absorption: The magnitude of the intensity is decreased as it traverses the tissue/bone boundary

2.2.3 Appearance of Tissue in Ultrasound

Tissue has many different inhomogeneities, along with varying impedance values throughout, depending on the type of tissue (e.g. muscle vs liver), which causes scattering and many small reflections. Areas with denser tissue also absorb more of the energy of the ultrasound wave, resulting in an image with darker grey values where the wave is traveling from an area of lower to density to an area of higher density. Obtaining the most detailed and accurate image of tissue deeper in the body can prove to be challenging. Using higher frequency ultrasound results in more detail, but greatly increases the attenuation. Thus, lower frequencies must be used to image deeper tissue.

2.2.4 Appearance of Bone in Ultrasound

Bone has a very high density, especially when compared to the surrounding tissue. Such a high density difference entails a strong, bright reflection due to the difference in acoustic impedance at the bone-tissue interface. Any of the ultrasound wave still remaining is quickly diminished due to attenuation, again because of the dense nature of the bone. As a result, the ultrasound image is completely black along the scanline following such an intense reflection.

2.3 Overview of Ultrasound Simulators in the Literature

Ultrasound simulation has been studied since the 1970s. The first approaches sought to model the propagation of the ultrasound wave based on physical models. Of such generative methods, Jensens FIELD II simulator [8] , [9] is well known. Developed in the 1990s, it remains an influential piece of work and free to use, but the source

code is not openly available. Almost all of the contending simulators use a pre-existing three dimensional image volume, acquired from a real ultrasound, CT or MRI scan, which is an approach inherently very different from the Field II simulator.Simulators focusing solely on MRI volumes were not found in the literature. The first simulators to make use of pre-acquired CT scans were introduced by Hostettler et al. [10] and Magee et al. [11]. More recent CT-based simulators often make use of a method developed by Wein et al. [21] that employs a transfer function to infer acoustic impedance values with the use of the Hounsfield units in the CT. However, very high resolution CT volumes are needed if fine details are to be preserved, and two areas that appear the same on a CT could have very different acoustic properties. Recently, the graphics processing unit (GPU) has facilitated the use of more complex algorithms [12] [13] [14] [15] in the processing of the 3-D image volumes, allowing for the use of more complicated methods while maintaining image generation at a speed comparable to clinical practice. The GPU has even been used for more updated generative approaches to simulation, such as Karamalis et al. [14] in 2010, who modeled ultrasound wave propagation and interaction with different media using the Finite Difference scheme to solve the Westervelt equation . Gjerald et al. [12] also employed the GPU, using it to numerically convolve the point spread function with the tissue model based on the fast Fourier transform.

2.3.1 Jensen's Field II Simulator

Jensen's work, embodied in the FIELD II ultrasound simulator, has been widely used. The first version of the ultrasound field calculator was written around 1990, which was generalized and interfaced with MATLAB five years later, retaining its C core [9].

The simulator is based on spatial impulse calculation. Spatial impulse calculation is very similar to impulse response calculation in linear systems, with the exception that impulse responses are dependent on the distance between the transmitter and receiver. A homogenous medium is assumed to allow for linear wave propagation. Impulse responses are calculated from a summation of all spherical waves originating from the aperture area. The simulator even incorporates a homogenous attenuation. The result is a linear ultrasound field. The FIELD II program is available for download, but it is not open-source. The MATLAB files are easily accessible, but the actual processing occurs in the C files, which are not readily obtainable. Therefore, it remains free to use, but the source code is not openly available. Simulation time is also quite extensive. In the 2004 paper describing the FIELD II simulator, an elaborate dataset was used for testing, which took 391 hours to run [9]. The FIELD II simulator provides far more detail than is necessary for a medical training simulator, and as such, is also slow. The restricted access to the source code also prevents any alterations that could have tailored FIELD II to a platform for medical training simulation projects.

2.3.2 Ultrasound Simulation Using Existing Image Volumes

A significant amount of work has been published on simulating ultrasound from previously acquired medical images. The focus of more current research has shifted from generating accurate tissue simulation, with proper speckle and other physical parameters, to implementing only a slicing of the original volume with minimal processing. These works focused on other aspects of simulation. The goal of Zhu et al. [16] was to realistically simulate a needle in the ultrasound image, which can be challenging, as the appearance of a needle changes depending on the angle it is imaged from. The

authors acquired a plurality of ultrasound images from a needle in different poses and interpolated between them using a hybrid nearest neighbour and linear interpolation method [16]. Bo et al. [17] worked with MATLAB and both CT and ultrasound volumes. The volume was aligned with a tracked mannequin using fiducial registration, and during simulation, the appropriate cross-section was selected from the image volume and displayed [17]. In the case of the CT, some processing was added using a previously published method for visual appeal. The work of Goksel and Sacludean centered on simulating deformation in previously recorded ultrasound images of tissue phantoms [18]. Skehan in 2011 [19] built upon a previously existing system which used pre-aquired 3D volumes to extract a 2D slice. The system was modelled upon a volume renderer that utililzes the OpenGL library. The second generation of the system incorporated an electronic writing tablet for positioning information, and a red sphere registered to an anatomical mesh visualizing the position of the stylus used for writing on the tablet [19]. The latest additions to the system included the changing of the sphere to a model of a transducer probe and adding surface rendering to the mesh depicting the patient. The works of the different groups mentioned above are heavily based on pre-acquired ultrasound volumes, which require access to an ultrasound machine. Many do not account for the angle dependent variations throughout the image.

2.3.3 Simulation Using CT Volumes

Simulating ultrasound specifically from CT volumes is quite different from using other imaging modalities. Wein et al. [20] recognized that the relationship between the intensity of the CT image, expressed in Hounsfield units, and acoustic impedance in

ultrasound, are proportional. This provides a means to simulate large scale ultrasound reflection at tissue boundaries, as well as shadowing effects at bone interfaces. However, the unique speckle pattern of each tissue type visible in ultrasound, that is, the echogenicity, cannot be calculated from this relationship. To remedy this, Wein et al. developed an intensity mapping on a narrow soft tissue range [20]. This method has been highly influential and many later works that simulate ultrasound from CT use Wein's method. Such an example is the simulation developed by Gill et al. [21]. They decided on modelling the ultrasound passing through the CT volume as a ray, calculating the reflection and transmission using the equations developed by Wein et al. [20] at each voxel. The authors incorporated a threshold, to simulate bone, above which the rest of the ray is set to black. For the final voxel intensity, they took into consideration the reflection calculated previously and the mapped CT intensity, both using Wein's method, as well as a bias term, which was comprised of the averaged intensity of the soft tissue region. The drawback with Wein's method is that the proportional Hounsfield unit to acoustic impedance relationship does not apply to all tissue, and the method cannot be easily expanded to other imaging modalities. In general, CT-based simulation methods are very limited to the cases that they can train, as each new scenario would require a new CT.

2.3.4 Simulation Using CT Volumes on the GPU

Ultrasound simulation from CT volumes in more recent papers often takes advantage of the graphics processing unit (GPU). Nevertheless, Wein's method has remained highly influential, such as in works by Kutter [22], Karamalis [13], and Sutherland [23]. Vidal [15], however, makes full use of the image processing capabilities inherent in

the GPU. Kutter et al. still use a ray-based model and draw upon the Hounsfield unit-acoustic impedance proportionality [22]. The authors segment out the bone and soft tissue to allow for calculation of reflection coefficients in both air-tissue and bone-tissue interfaces. To account for non-specular reflection at tissue interfaces, the Lambertian scattering model is employed [22]. During image generation, a virtual probe is positioned within the space of the CT volume, and the casting of an ultrasound beam from each element in the virtual probe is simulated. Using the GPU, multiple rays can be produced in parallel. The equations for intensity calculation are computed inside a fragment shader, resulting in a measure of the acoustic intensity received by the transducer element [22]. This processing is implemented with the use of OpenGL and is independent of probe geometry. The framing of the simulation is very similar to ray casting, the difference being that sample values are stored along the ray as opposed to just one returning value [22]. Ray sampling is also employed by Karamalis et al. [13] . Very much like Kutter, a ray is calculated that propagates through the CT volume beginning at each transducer element. CT volume intensities are sampled at equidistant sampling points. Next, scan simulation is employed [13]. The previously obtained sampled values are used to generate the reflection, transmission, and the echogeneity image, based on the model proposed by Wein et al. [20]. The echogenity image is obtained by using a 1D lookup texture on the GPU [13]. Once calculated, the different images are combined and then scan converted as needed. This is done by assigning a coefficient for each image and one for brightness. During scan conversion individual scan lines are mapped to the image-based on the transducer design [13].

Sutherland et al [23], on the other hand, created an augemented reality training system, for spinal needle insertion procedures, with haptic feedback based on a Finite

Element Mode (FEM). Ultrasound simulation was not a large part of the work, and they implemented a GPU version of a previously published ray tracing algorithm, that draws heavily from the concepts of Wein's work [20]. However, they did have an expert group evaluate the training system, yielding positive feedback.

Vidal et al. [15] used a very different methodology. They were also working on visualizing needle insertions in ultrasound. Once they obtained a 2D oblique image from parallel CT slices, they calculate the needle position and replace those pixels in the oblique image with bright pixels simulating the reflection produced by the metallic needle [15]. A shadow mask is then created to simulate gas and bone by processing the oblique image. A 3D texture is created from noisy 2D slices. Finally, the image is computed from this texture, making use of the probe position and orientation. All of the generated images are combined using multi-texturing in OpenGL. Bright reflections are added by enhancing the vertical lines discovered after using the gradient direction, found by using the Sobel filter. All processing is done in real-time, implemented with GLSL (OpenGL Shading Language), which computes all of the operations in a single pass, with the aid of the GPU [15]. The disadvantage associated with methods employing the GPU is that it becomes difficult to repeat performance on different machines. The disadvantages of ultrasound simulation from CTs, mentioned above, also applies.

2.4 Software Platforms

The simulator makes use of existing software to communicate with hardware and to process information, as described below.

2.4.1 Perk Tutor

The Perk Tutor [4] is a training system for image-guided needle placement procedures developed by the Perk Lab(http://perk.cs.queensu.ca/). It consists of PLUS and 3D Slicer working together with a tracked ultrasound machine and a spine phantom. These components are explained in more detail in the Experiments and Results section, where the system was set up to use simulated ultrasound in place of real ultrasound. The goal of the system is to train subjects in minimally invasive spinal procedures (i.e. facet joint injections), by allowing the subjects to see a virtual spine model and its spatial relation to the needle and ultrasound image, updating live on screen, as they are inserting a needle.

2.4.2 3D Slicer

3D Slicer(www.slicer.org) is a software package for medical image visualization [3]. It is open-source and has an active community of developers constantly adding modules and improvements. Two modules used significantly by the Perk Tutor are the Perk Tutor module and Slicer IGT (image-guided therapy). SlicerIGT allows for the collection of fiducials using information from a tracker, making fiducial based registration of virtual models and physical phantoms possible. Additionally, it also is capable of taking ultrasound snapshots and displaying them, while displaying live ultrasound coming from a tracked probe. The Perk Tutor module contains components that can record the transformations of certain tools coming from a tracker, as well as replay previously recorded workflows. One of the vital capabilities of 3D Slicer is its ability to communicate via the OpenIGTLink [24] protocol, allowing it to display live ultrasound images and any other tracked object in the correct position. Such a

scene can contain multiple objects, including surface models of anatomy and surgical instruments alongside the live ultrasound.

2.4.3 PLUS

The goal of PLUS (Public Software Library for Ultrasound Imaging Research www.plustoolkit.org) is to make the transfer of ultrasound-guided intervention systems developed in laboratory settings to clinical research as seamless as possible [1]. To this end, PLUS focuses on software for use in calibrating, acquiring, processing, and transferring tracked ultrasound. Tracked ultrasound involves synchronizing position information from a tracker with ultrasound images acquired from an ultrasound transducer. To reach as many users, and thus to achieve the greatest possible impact on the field of image-guided therapy, the software package is available for download under an open-source license. As well, support is offered for many different types of trackers, again expanding the usability of the system. PLUS also contains the ITK and VTK libraries, making access to these resources much simpler for developers using the PLUS library, as they do not have to integrate these extra libraries into their development framework.

2.4.4 VTK

VTK stands for the Visualization Toolkit, which is a templated library developed by Kitware, freely available with an open-source license [2]. It contains various data structures to increase the speed at which software requiring data visualization can be produced. As well, it contains the infrastructure necessary to create a data visualization pipeline, including various classes that read in the data, perform operations

on the data (or "filter" the data), map the data to a scene, and ultimately render it. The code for 3D Slicer makes heavy use of VTK.

2.4.5 VTK Structures

The following structures are used often in the simulator:

vtkMatrix4x4 is used to represent homogenous matrices used in transformations. This object has functions to easily transpose, invert, and find the determinant of the matrix it represents. Additionally, a function is provided for matrix multiplication.

vtkTransform is an object that contains within it a vtkMatrix4x4. It contains more functions to facilitate a full range of linear coordinate transformations, including translations, rotation, and concatenations.

vtkPolyData is a dataset used to represent geometric structures, which are composed of various other more primitive elements, such as vertices, lines, polygons, and triangle strips. The models used by the simulator often consist predominantly of triangle strips. VtkPolyData is comprised of many cells, each of which has a type corresponding to one of the primitive elements mentioned above. It also contains special data traversal and manipulation functions.

vtkImageData is used to represent a topologically and geometrically regular array of data. It is a highly effective container for images, as it contains functions to set the image dimensions, spacing, and origin, and to traverse the data.

Chapter 3

Methods

From conception, the simulator was designed to take advantage of the Visualization Toolkit (VTK) [3], [2] and to be integrated into the Public Library for Ultrasound (PLUS) software toolkit [1], offering easy access to a wide range of functions for image, surface mesh, and pose information acquisition and processing. As well, it allows for integration with 3D Slicer [2]. The interactions of these separate systems are depicted in Figure 3.2. The input to the simulator is a surface mesh model and a linear transformation representing the spatial relationship between the surface mesh and the simulated ultrasound image. This allows the model to be transformed to the image coordinate system, and the intersecting region to be identified and displayed as a constant intensity binary image. The linear transformation can either be specified as a constant, in a configuration file, or come from a tracker. The surface mesh, if anatomical, can be generated easily from any segmented volumetric image, such as a CT or MRI scan. The pose, that is, the linear transformation mentioned above is obtained either from a live tracker or from a configuration file. Transducer parameters, such as radius, imaging depth, number of elements and shape (curvilinear versus linear) are also defined in the configuration file read in by the simulator, as are the

material properties associated with each model. The workflow of the simulator is described in Figure 3.1.

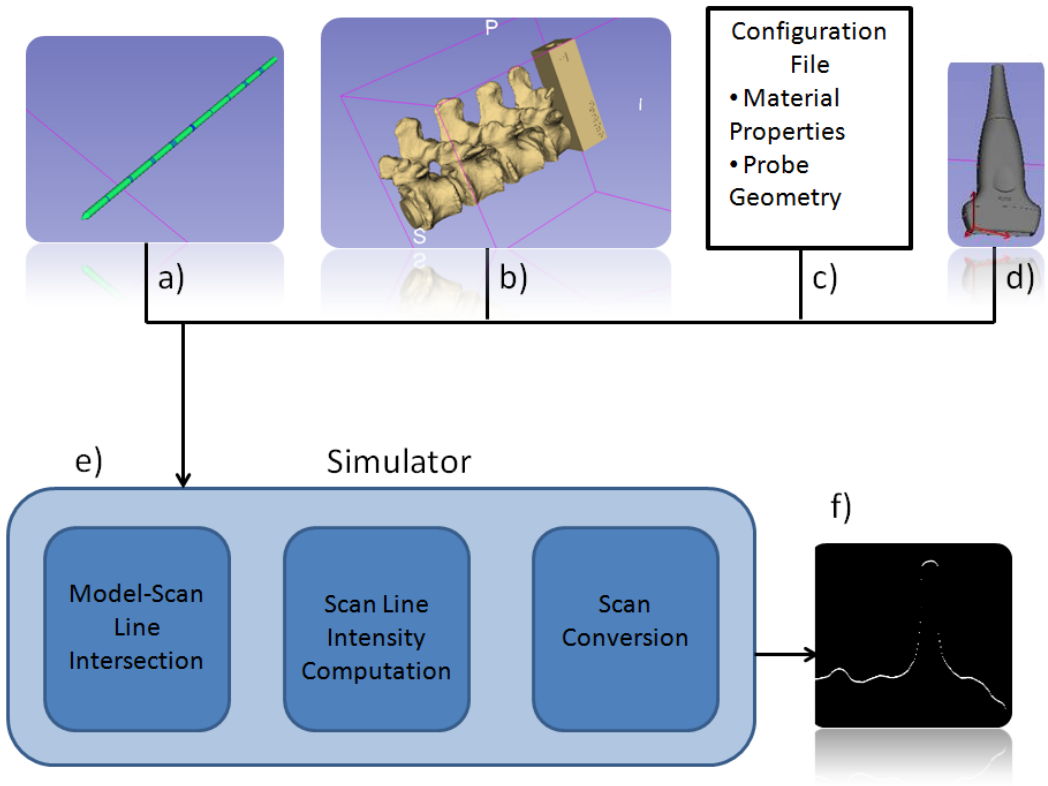

Figure 3.1: Workflow of the simulator; a) shows an example of a surface model to be used in future work, in the form of a needle, b) shows another surface model in the form of a spine, c) represents the configuration file used by the simulator, d) exemplifies the linear transformation representing the probe position in relation to the surface mesh, e) represents the simulator divided into three steps, and f) shows a sample simulated image as the result of the workflow.

The ultrasound simulator was created in three iterations. All of the iterations made use of the same development environment, which included the use of the visualization toolkit and PLUS. The simulator itself was developed as VTK filter, an

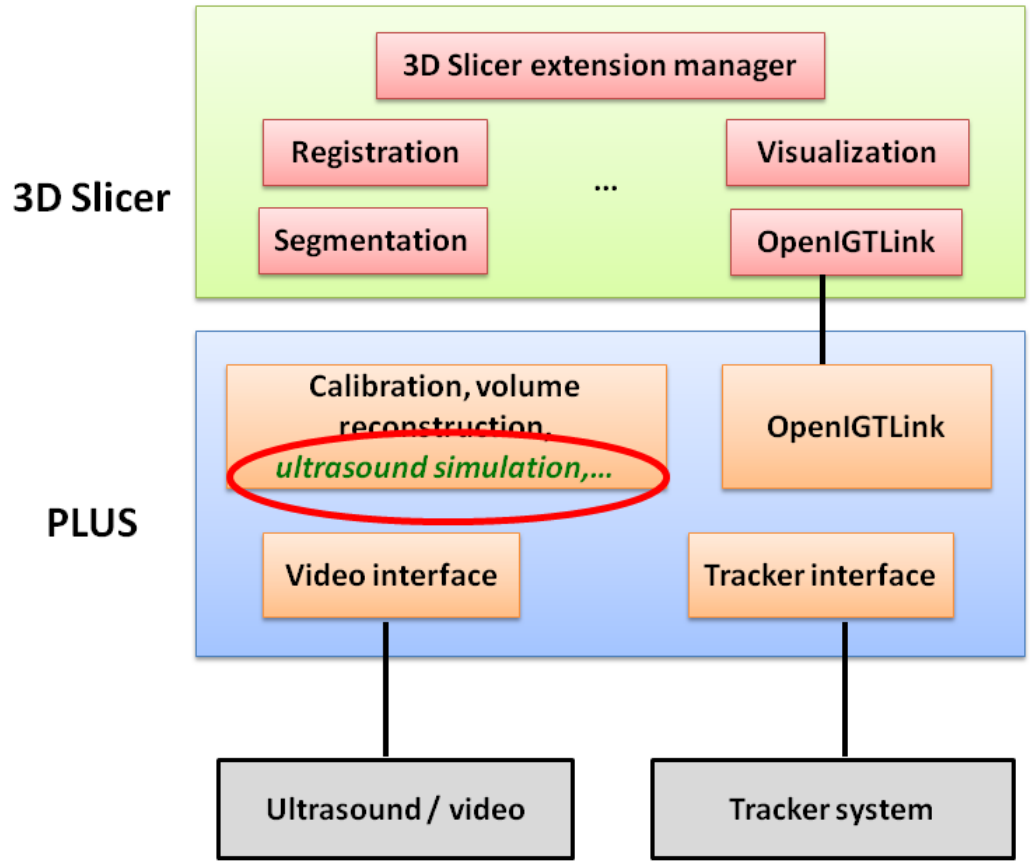

Figure 3.2: Architecture of the Plus-Slicer system including the ultrasound simulator

important aspect in decreasing the specificity of the simulator and increasing its reusability, that is, allowing the simulator to function with any surface mesh model and transformation. The three iterations came about to meet the demands of clinical simulation speed. The first two were simply not fast enough. However, they provide valuable insight into the different methodologies that can be utilized in the creation of an ultrasound simulator.

3.1 Development Environment

The simulator was developed in a Windows 7 environment using Visual Studio 2008, as seen in Figure 3.3. The simulator was designed as a part of the library section of PLUS, PlusLib as illustrated in Figure 3.5. PLUS is comprised of PlusApp and PlusLib, dividing the software toolkit, respectively, into a set of applications and library classes. The repository for PLUS is stored on an Assembla workspace (Fig-

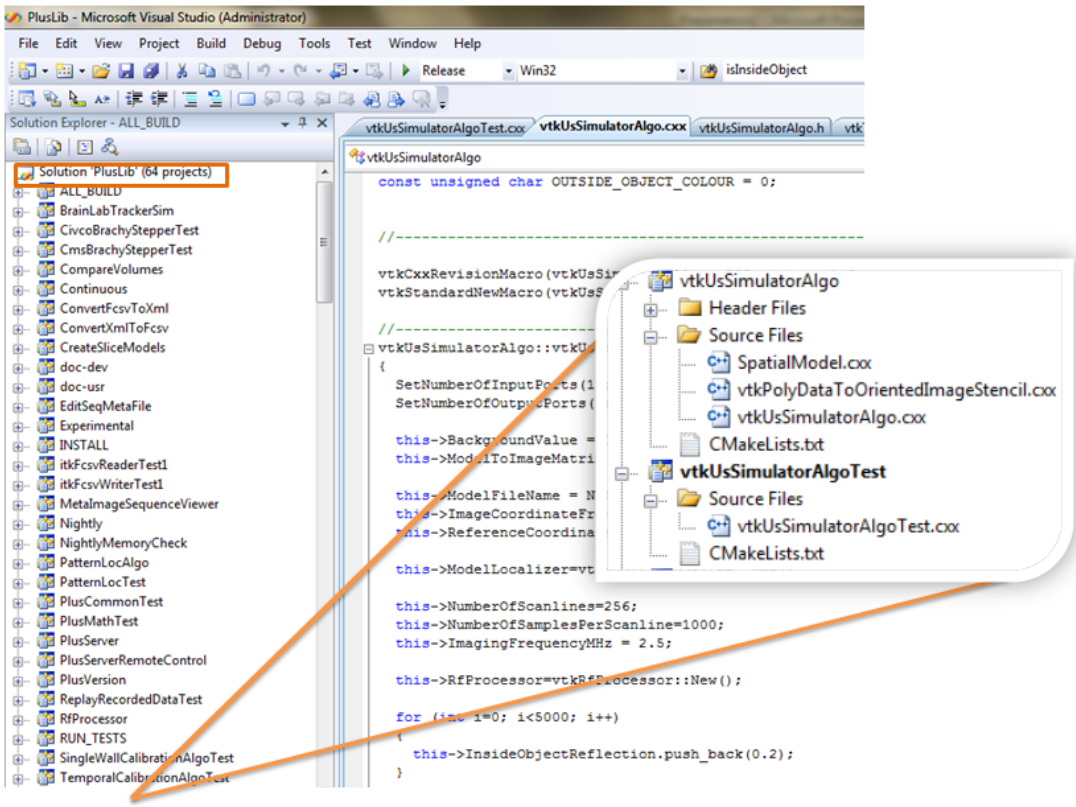

Figure 3.3: The ultrasound simulator in the PLUS environment

ure 3.4 www.assembla.com). The workspace includes source-control in the form of

Subversion (SVN) repositories for the maintaining the lab's projects, including PLUS. Bug fixes, as well as new developments, are tracked using a ticket system inherent to Assembla. The simulator itself was assigned in the form of a ticket, and all updates pertaining to the simulator are made under the same ticket number. In addition, nightly tests are run, again hosted by Assembla, through the C-Dash interface. Before a commit is made, PLUS development protocol requires the tests to be run, ensuring code integrity.

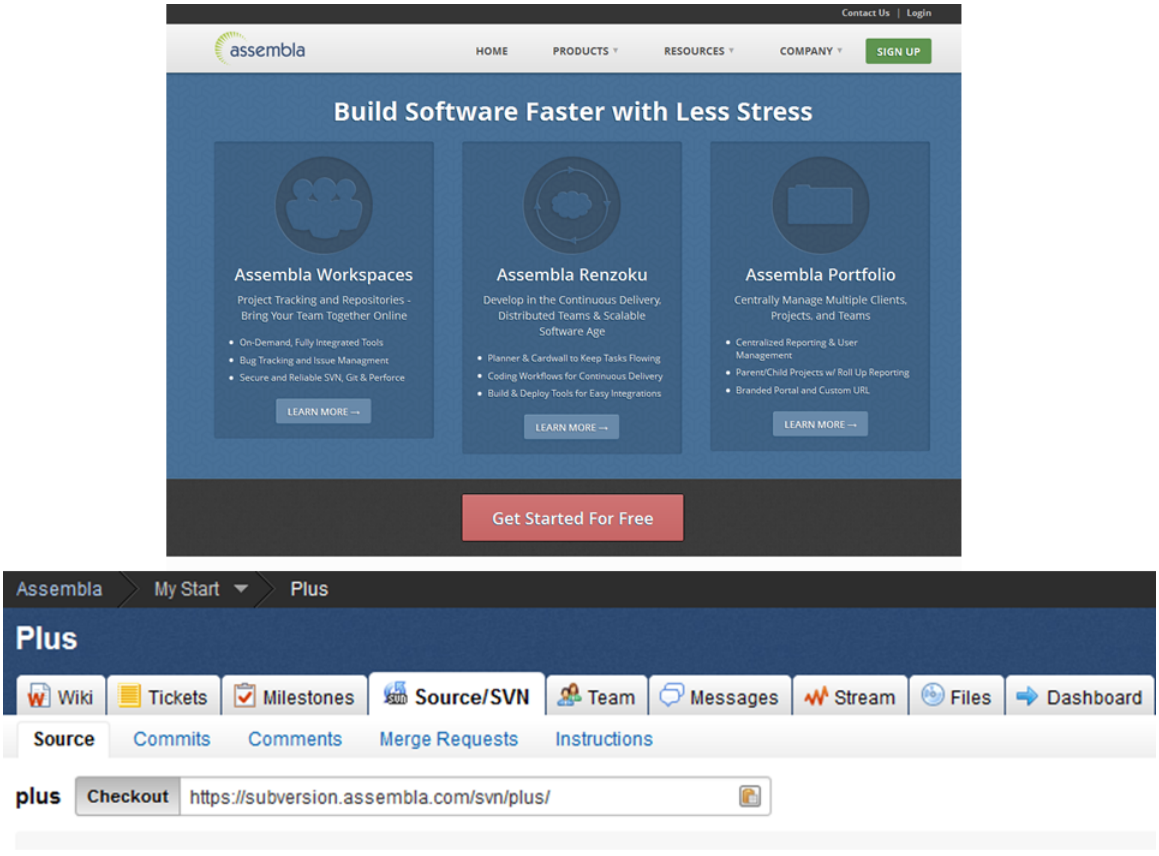

Figure 3.4: Assembla homepage and taskbar

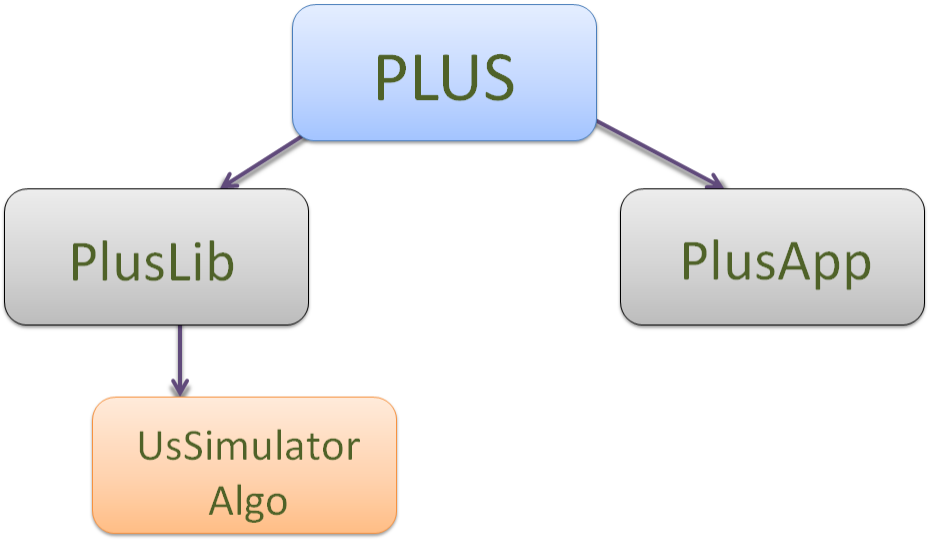

Figure 3.5: The simulator's location within Plus is shown, under the Plus Library (PlusLib).

3.1.1 Plus Components Vital to the Ultrasound Simulator

The simulator can easily make use of, and be used by, the rest of the PLUS library. Vital to the functioning of the simulator is the processing of input data. These are stored in the form of meta image (MHA) files, which, most importantly for the simulator, contain the transformations between the position of a virtual ultrasound probe and the spine phantom. Without the use of PLUS, processing each of these frames and calculating the proper transformation would require a great deal of work, involving much frustration and confusion throughout the process. However, PLUS contains a tracked frame list structure, and transform repository structure. The tracked frame list structure reads in the MHA file, containing the ultrasound sequence including

transformation information, and organizes it into an easily usable list, correlating the appropriate image and transformations with each time stamp present in the file. The transform repository must be loaded with initial transformations not included in the MHA file that complete the transformation path from the surface mesh to the final simulated ultrasound image, that is, the output of the simulator (Figure 3.6). During the processing of each frame, the transformations for that particular frame are added to the transform repository. The different transformations and coordinate systems that comprise a generic experimental setup are illustrated in Figure 3.7.

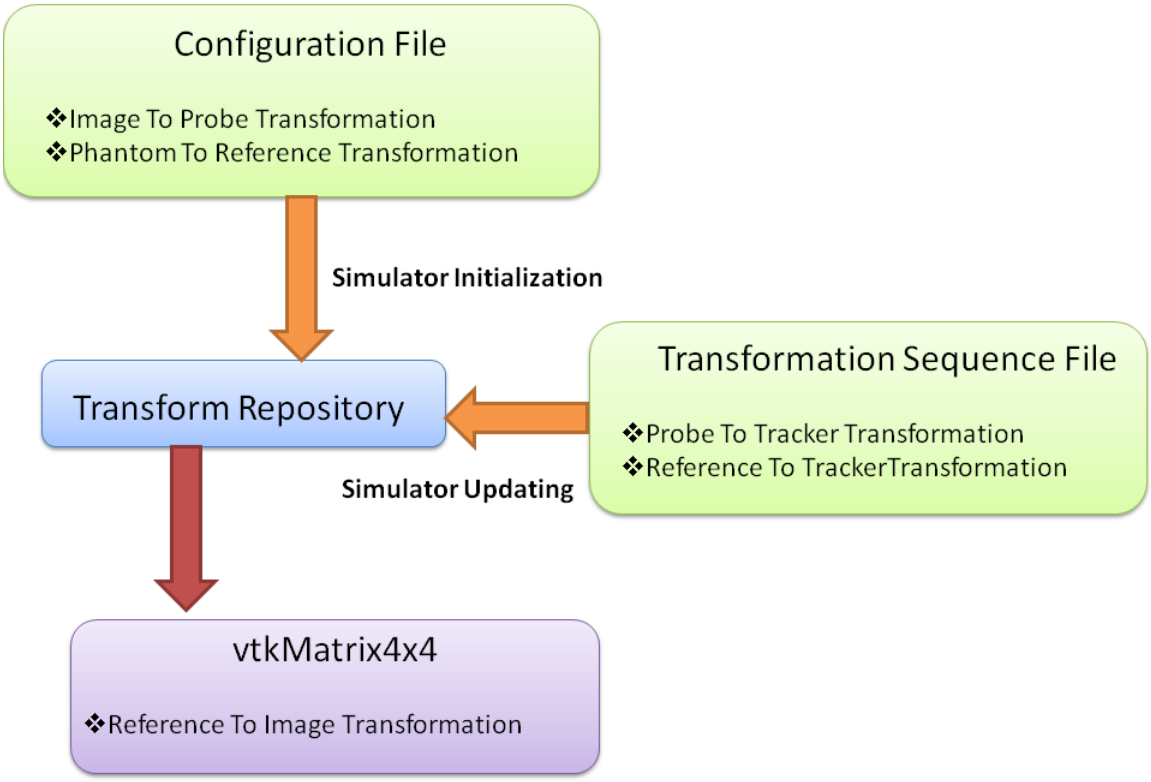

Figure 3.6: The figure showcases how the ultrasound simulator uses the transform repository

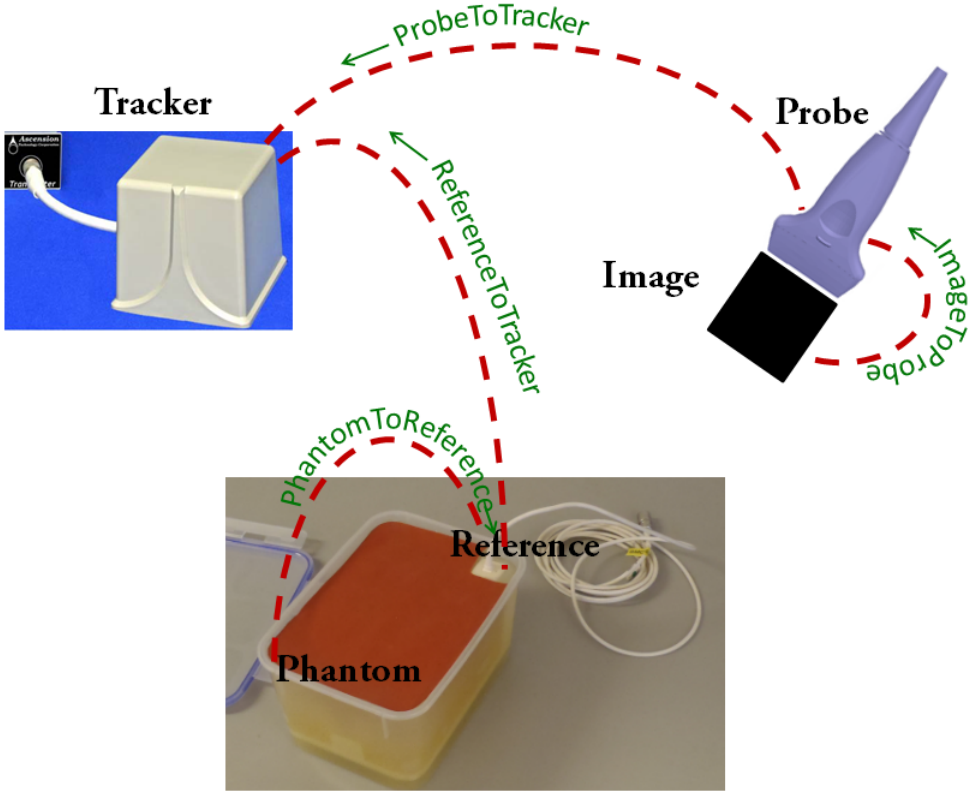

Figure 3.7: The necessary coordinate systems and the transformations between them, with the transformations being represented by green text and the dotted lines

3.1.2 VTK Uses

PLUS greatly benefits from its incorporation of the VTK library, and it is used heavily throughout the simulator, especially for reading and writing data. Many of the parameters of the simulator are stored in extensible markup language (XML) configuration files (Appendix A), and these are read using the XML readers and elements in VTK. Additionally, all major graphical structures and operations, such as images, models, points, and transformations are stored in VTK containers (e.g.

vtkImageData, vtkPolyData, vtkTransform) which were described in Chapter 2.

3.2 VTK Filter Development

The VTK toolkit is built around the construct of a data pipeline, beginning with a data source and ending with the display of the data on the screen. Data sources can either be readers, as is the case in the pipeline pertaining to the simulator, or independent sources that generate data based on parameters. VTK filters are the next step in the pipeline and are responsible for making modifications to the data. In the case of the simulator, it is a VTK filter with a vktPolyData, being the surface mesh, as an input from the pipeline and a vtkTransform as an input set separately. Since the VTK library is composed of many filters, typical tutorials for VTK focus on the coding of pipelines and the use of the filters, not the actual design of them. The filter differs from a typical object class in three important ways. The filter must have a "fillinputport"and a "filloutputport"function, which respectively set the input and output type of the filter. Additionally, the body of the filter must be placed in a function named RequestData. The rest of filter development involves the utilization of objects specific to VTK and is discussed in more detail below as it relates to the different methodologies.

3.3 Binary Cross-Sectional Filter Development

The first functional goal in the development of the ultrasound simulator was to simulate a cross-sectional image from the surface model and the linear transformation (Figure 3.8) to serve as a basis for further image processing. However, finding a way to generate such an image in real time required delving into three different approaches.

Originally, the simplest approach, from a design perspective, was to implement and test a combination of existing VTK filters to create a mini pipeline. When this proved to be far too slow, a new version of one of the VTK filters, the vtkPolyDatatToImageStencil filter, was implemented. However, this still did not meet the speed requirements needed for a real time simulator, and thus the third method, using ray tracing, was implemented.

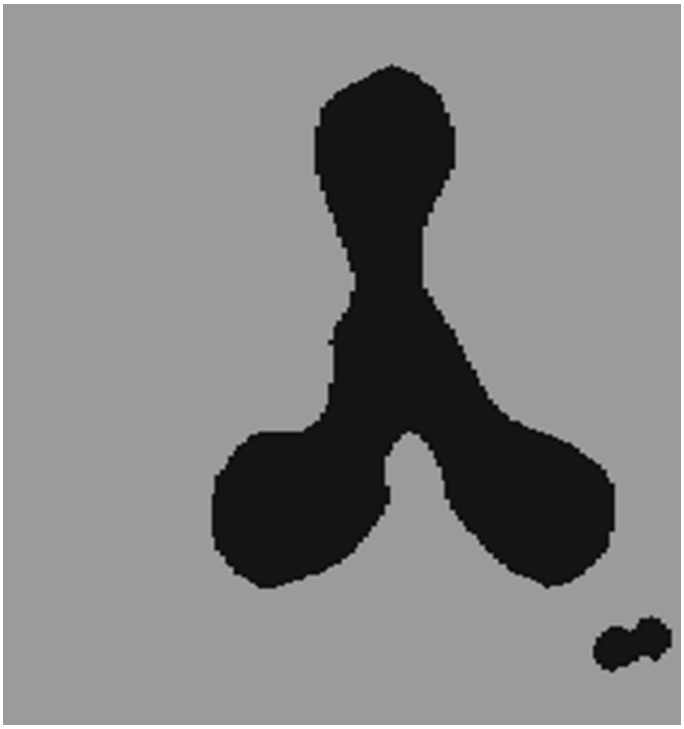

Figure 3.8: Sample binary image

3.3.1 Approach Using Existing Filters

The first method employed to generate the preliminary binary image involved the creation of a new VTK filter comprised of existing filters. The new filter performed a number of preprocessing steps, such as converting the surface mesh into triangle

strips and transforming the model into the image coordinate system, to prepare it for the vtkPolyDataToImageStencil filter. The vtkPolyDataToImageStencilFilter clips an image with a polydata, which in the case of the simulator entails the resultant simulated image clips the surface model, and outputs a vtkImageStencilData type. The vtkImageStencilData still needs to be combined with a blank background image to produce the final simulated cross-sectional appearance using the vtkImageStencil filter. Once the filtering process is complete, a stencil is produced in the shape of a cross-section of the model, with the cross-section being calculated at the position of the imaging plane. Functionally, using the existing filters worked very well, however it was far too slow, needing approximately 40 minutes to process 77 frames. The poor performance was due to the fundamental problem of transforming the dense model every time the position of the image plane, which corresponds to the position of the ultrasound probe, changed. The simple fix of transforming the image plane to the model coordinate system is not possible in the VTK framework, as images have no orientation.

3.3.2 Modifying an Existing Filter

In the second method, the filter of filters construction was retained, and a new version of the vtkPolyDataToImageStencil filter was developed. The modified filter was able to support images in any orientation with the use of vtkPlanes to store the relevant transformations. To further increase the speed, a binary space partition (BSP) tree was used to find all intersecting positions of the image plane and bone model. The BSP tree recursively subdivides a space into convex sets using hyperplanes. The VTK implementation used here, vtkModifiedBSPTree, differs from a conventional BSP Tree

in that it stores cells straddling the dividing plane in a separate container, adding another child node to each parent node in the tree; effectively creating a ternary tree. The modified filter approach allowed for processing 4 frames per second, which was an improvement compared to the first method, but it was still inadequate for use in a real-time clinical training system.

3.3.3 Ray Tracing and Binary Space Partitioning Tree

In third and presently used method, the previously used filter of filters construction was abandoned altogether for ray tracing. The input still consists of the surface mesh and the transformation. The simulated ultrasound image is generated in two steps: first a scan line computation and then a scan conversion. Scan line positions are determined in the image coordinate system using the desired size for the simulated ultrasound image, a parameter that is specified in a configuration file. These positions are then transformed to the model coordinate system, and a BSP tree (vtkModifiedBSPTree) is used to determine the points intersecting with the model located along the scan line. The intersection points divide the scan line into segments, which are then filled with a grey value, the result of an intensity calculation based on the material properties also defined in the configuration file. The process of dividing a scan line into such segments is depicted in Figure 3.9. The scan lines are then converted to a regular brightness-mode ultrasound image using the RF Processing module of PLUS. The RF Processing module interprets the information in the scanlines and applies the appropriate spacing to formulate an ultrasound image. The scan line method allows for relatively straightforward computation of the grey values of the generated image, facilitating more complicated intensity calculations based on the physical properties

of ultrasound.

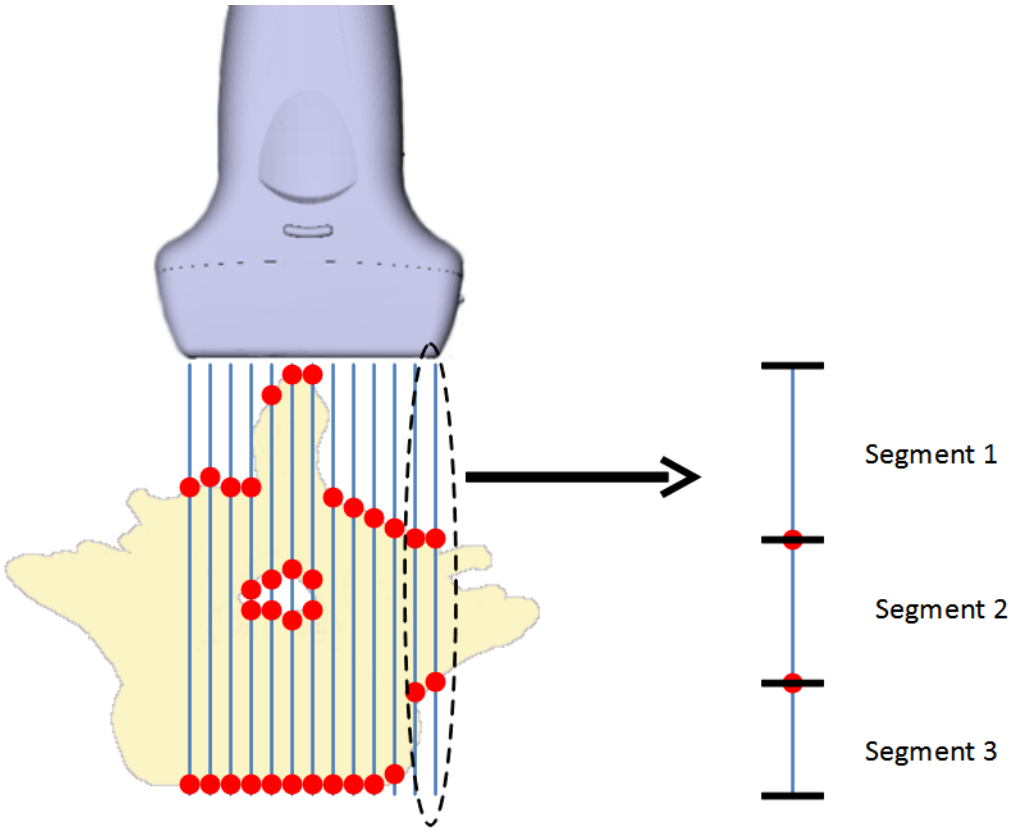

Figure 3.9: The large dots represent the intersection points between the image plane and the surface model. Straight lines are the scan lines divided into different segments by the intersection points that fall along the scan line.

3.4 Simulating Reflection and Absorption

Currently two modes exist, the simple binary image, and a simulation of bone. During bone simulation, the surface model specified is assumed to be representing bone. When ray casting is simulated, the grey value intensity calculation is altered to create strong reflections from the bone and background surface, as well as fast absorption

in the bone. Once an intersection point is detected, a short series of bright pixels, rapidly decreasing in intensity, are written to that segment of the scan line and the rest of the scan line is filled with black, representing the absorptive properties of bone. The result from this type of simulation is shown in Figure 3.10

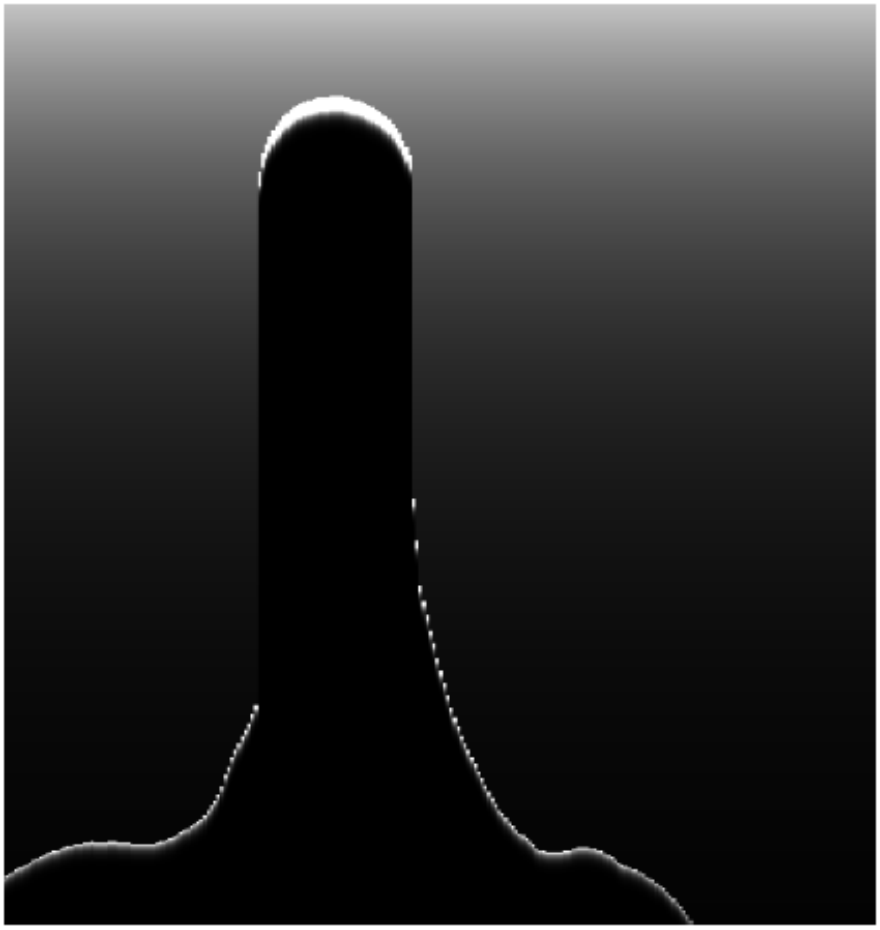

Figure 3.10: Sample bone simulation

3.5 Simulator Classes Structure

The simulator, as it is now, is comprised of a class called vtkUsSimulatorAlgo, it's corresponding header file, and the vtkUsSimulatorAlgoTest. The vtkUsSimulatorAlgo

is the filter itself, and produces one output image for each linear transformation and polydata. The linear transformation must be set; however, the simulator is capable of reading and loading the polydata. The vtkUsSimulatorAlgoTest is responsible for setting the proper inputs to vtkUsSimulaorAlgo. It reads the input parameters, as well as the configuration and MHA files, creates the transform repository and the tracked frame list mentioned above, creates an instance of the simulator filter (vtkUsSimulatorAlgo) and iterates through all of the frames in the tracked frame list, updating the simulator each time. The class is even capable of displaying the polydata model and the frame positions of the slices. When the simulator is used in an experimental environment, the class vtkUsSimulatorVideoSource is used in place of vtkUsSimulatorAlgoTest, to allow for processing of live tracker information, and for sending of the ultrasound images to the other components of the setup (3D Slicer) for display.

3.6 Spatial Models Representation

Once the simulator was functional in the two different modes, the SpatialModels class was added (Figure 3.11). The goal of having separate spatial models is to expand the simulator to be able to handle a theoretically unlimited number of surface mesh models, allowing for the generation of an ultrasound image containing the spine and needle, and perhaps even muscular tissue and skin. The spatial model structure would read in relevant parameters, such as the acoustic impedance of the particular material, and be responsible for the intensity calculation, producing intensities one ray segment at a time. Currently, the infrastructure for the interaction of the SpatialModels class and the vtkUsSimulatorAlgo class is in place, the intensity calculation methods still

need tweaking.

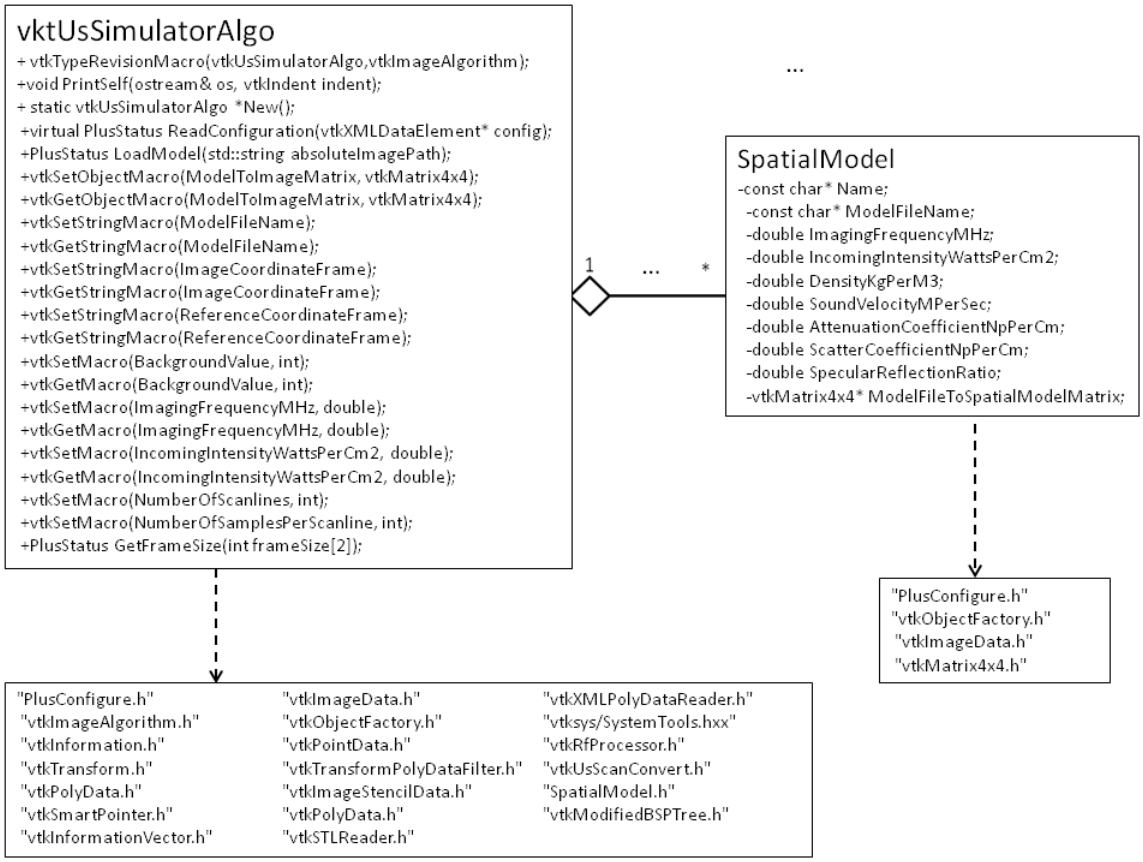

Figure 3.11: Class Diagram of the Ultrasound Simulator using Spatial Models

3.7 Surface Meshes

A surface-based geometrical model is used for the simulation, which enables the production of simulated images that clearly show the shape of anatomical structures. Moreover, the surface model can be easily altered computationally in any design program allowing for the study of an infinite variety of pathological cases or to show time-varying models. Surface meshes cannot describe the internal texture of the structures, but it is argued that this noise texture may not be always necessary and

could even prove to be a distraction during the learning process in certain cases, especially if the objective is the recognition of the shapes of anatomical structures. The surface mesh based simulator is implemented entirely at the level of application software in a system-independent manner, and hence it is particularly amenable to multi-platform open-source implementation, and thereby widespread use. The surface mesh, if anatomical, can be generated easily from any segmented volumetric image, such as a CT or MRI scan. However, the segmentation itself can be a time-consuming process. In our case, the surface mesh was of a spine originally from a CT scan. Surface meshes of surgical tools are readily available with such medical image software, and surface meshes in general are a compact form of data representation when compared to volumetric models.

Chapter 4

Human Operator Study

Once the development of the simulator was complete, including reflection and absorption simulation for a bone model, the simulator was tested in a 3D Slicer environment, similar to the Perk Tutor, which will be elaborated upon below. When functionality at real time was demonstrated, that is, the ultrasound updated without a noticeable wait time, a human operator study was organized to assess the effectiveness of the simulator as a training tool.

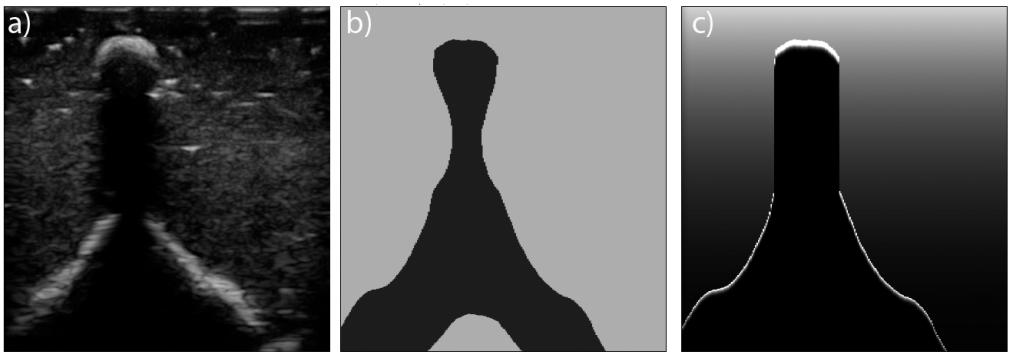

Figure 4.1: Images of a spinous process in the spine phantom: a) acquired with an ultrasound scanner, b) corresponding simulated cross-section, and c) final simulated bone image.

4.1 Visual Performance Tests

The simulator was used in the Perk Tutor ultrasound-guided spinal injection training system [4]. The anatomical object (spine), tools (ultrasound transducer, needle), and simulated ultrasound image were visualized in 3D Slicer.

4.1.1 Perk Tutor Components

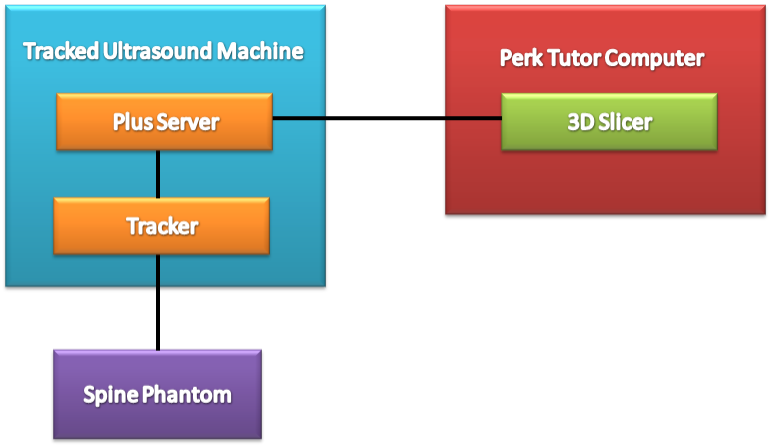

Figure 4.2: Components of the Perk Tutor

The Perk Tutor (Figure 4.2) is a multi-part system consisting of :

- Spine Phantom
  - Created in the lab by previous students and post docs
  - Contains 3D plastic printout of a the lumbar spine, with space for attaching a reference sensor
  - Filled with plastic gel

- Tracked ultrasound machine
  - With built-in electromagnetic tracker
  - Set to research mode, a mode of operation that allows the user to treat the ultrasound machine as a computer (install programs, connect to the internet, etc.), as opposed to clinical mode where only the ultrasound imaging program can be used.
- PlusServer running on the tracked ultrasound machine
  - A component of the PLUS Library
  - Sends images and linear transformations
  - Data is sent via the OpenIGTLink Protocol to a computer running 3D Slicer
- 3D Slicer
  - A multi platform software package for medical visualization
  - Lab-created extensions
  - Visualizes the information coming through the OpentIGTLink connection

4.1.2 Simulator Testing and Results

During testing in the Perk Tutor-like 3D Slicer environment, no ultrasound machine was used, with tracking accomplished by a stand alone ascension TrakStar tracker. Instead of a tracked ultrasound probe providing position information for the simulator, a simple sensor was used. Eventually, for visual appeal, this sensor was taped to an old transducer no longer in use. The spine mesh used to simulate the ultrasound

images, and displayed in 3D Slicer, consisted of over 97,000 points in 195,000 cells. Such cells make up a surface mesh and contain primitive graphical elements including lines, points, triangles, and triangle meshes. The ultrasound images were generated at a speed of 50 frames per second, and a resolution of 820 x 616 pixels on a Windows PC with a 3.4 GHz processor. Figures 4.1 and 4.3 depict ultrasound images of a spine phantom in a) for easy comparison to the images simulated at the same position using the surface model. The b) images are the cross-sections created after the first stage of development, and the c) images are the final images resulting from simulating bone. As shown in Figure 4.4, the simulator was successfully integrated in the 3D Slicer environment. The PLUS Server was used to send the position information of a fake transducer to the simulator, and then to send the simulated image to 3D Slicer where it was displayed using the Volume Reslice Driver module. The simulated ultrasound image correctly followed the motion of the tracked fake ultrasound probe without noticeable delay, as this was visually ascertained by multiple independent observers in numerous series of qualitative visual testing. These observers were a post doctorial fellow, and senior software engineers who volunteered feedback.

4.2 Human Operator Study

To test the effectiveness of the developed simulator as a training tool, a human subject study was undertaken with the Perk Tutor [4]. A sample of 22 students (11 male and 11 female), who had never used an ultrasound machine on the spine, were recruited. The students had varying educational backgrounds and ages. The volunteers were divided into two groups, one receiving training with ultrasound, and the other receiving training with simulated ultrasound. After initial training, the subjects were asked to

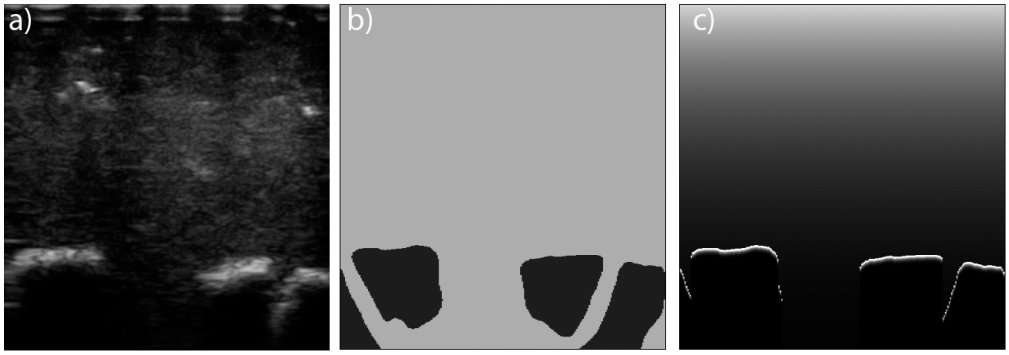

Figure 4.3: Images of facet joints in the spine phantom: a) acquired with an ultrasound scanner, b) corresponding simulated cross-section, and c) final simulated bone image.

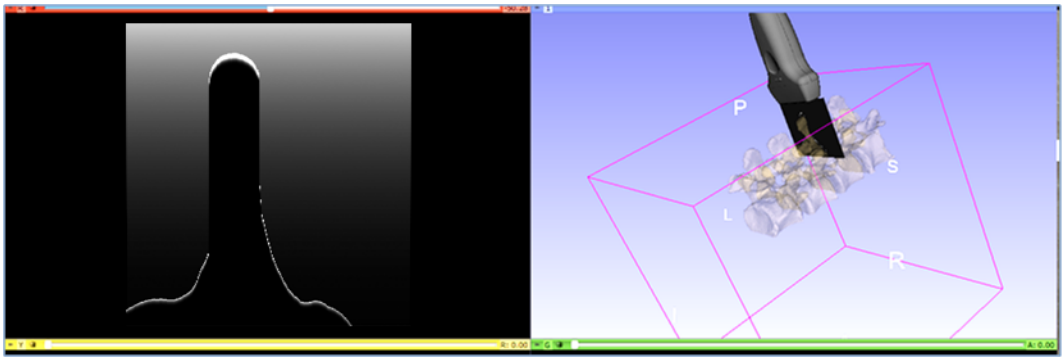

Figure 4.4: Simulated Ultrasound in 3D Slicer with spine

locate the facet joint on an opaque spine phantom using ultrasound. The spine phantom included a replica of a human spine under multiple layers of artificial tissue and skin. The training session entailed the trainee moving an ultrasound transducer connected to a Sonix Touch (Ultrasonix Medical Corp., Richmond, Canada) ultrasound machine across the aforementioned phantom that contained a plastic printout of a spine under artificial tissue. In addition, a 3D visualization of the spine and either

simulated or real ultrasound images were displayed on a computer screen. The ultrasound images were displayed in the same 3D view as the spine as well as in a separate 2D dimensional window. In the 3D view, the position of the ultrasound relative to the 3D spine was representative of the physical location of the ultrasound transducer and phantom, which was achieved by electromagnetically tracking the probe and a reference point on the phantom, and through registering selected points on the phantom with the corresponding points on the virtual spine. After the training session, the trainee was asked to locate 6 facet joints in the phantom using real ultrasound, regardless of which training group they were in. The trainee was instructed to indicate when the facet joint was found, at which time the study coordinator took a snapshot of the ultrasound image that the trainee believed to contain a facet joint. The trainee then placed a marker on the snapshot at the location where he/she believed the facet joint was located. The 6 markers were saved and later assessed by an independent physician who was not involved in the trial. Each trainee was assigned a score on a scale of 0 to 6, indicating the number of facet joints identified correctly.

4.2.1 Protocol

1. Determine whether participant will be in the real ultrasound group or the simulated ultrasound group. This is done by arbitrarily choosing the designation of the first participant and then alternating thereafter.

2. Set up the ultrasound machine, 3D Slicer, PLUS Server, and tracker before participant arrives.

   (a) Open 3D Slicer

   (b) Load saved 3D Slicer scene

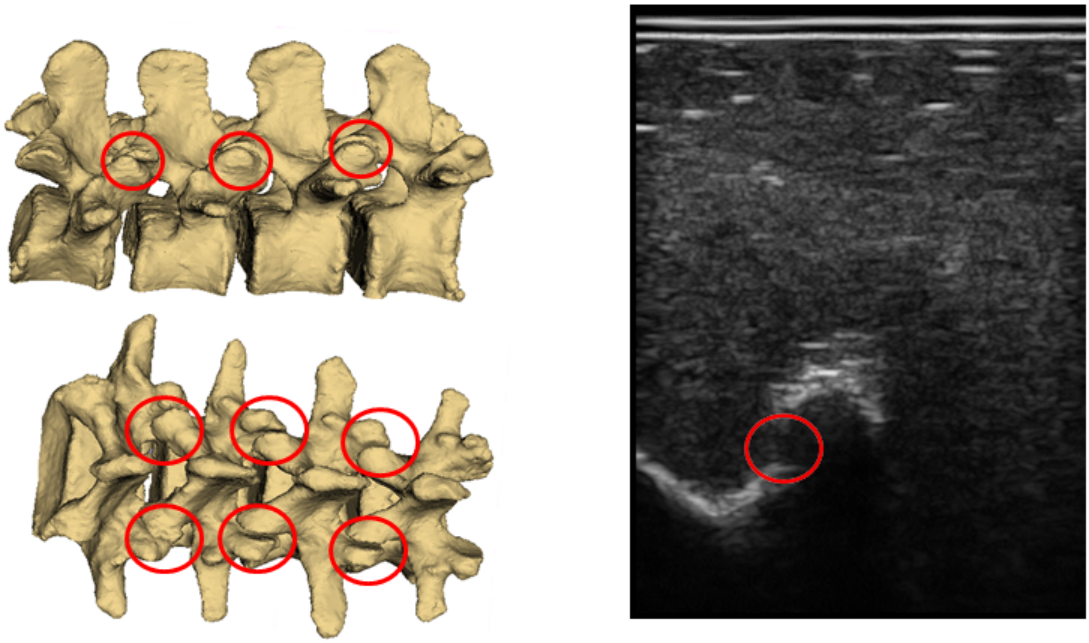

Figure 4.5: Training material shown to trainees to describe a facet joint.

(c) Navigate to the OpenIGTLinkIF module and check the active box

(d) On the ultrasound machine, start PlusServer with the configuration file for either real or simulated ultrasound

3. Give participant consent form to read and sign upon arrival.

4. If participant signs, show them printout/presentation detailing the location of the facet joint and a snapshot of what to look for in the ultrasound.

5. Explain the experimental setup (phantom, tracker, 3D Slicer, ultrasound).

6. Explain to the trainee that they will be given 5 minutes to train with the phantom, with facilitator feedback and then they will be required to find the 6 facet joints in the phantom.

7. Allow the participant to train with facilitator feedback, including confirmation of facet joint localization.

8. Shut off subject's monitor. If participant is using simulated ultrasound, restart PLUS Server to use the real ultrasound with the command line on the ultrasound machine.

9. Set up the Ultrasound Snapshots module on Slicer and start transform Recorder

   (a) Navigate to the Ultrasound Snapshots module

   (b) Set the Ultrasound Image drop down menu to the name of the incoming ultrasound image (ImageRAS)

   (c) Navigate to the Transform Recorder module

   (d) Create a new Transform Recorder by the Module Node

   (e) Set the Transform Selection drop down menu to ProbeToRAS

10. Instruct participant to find the first facet joint, and to inform you once they believe to have located the facet joint, press start when the participant beings to look for the joint.

11. Upon the localization of the facet joint with the ultrasound probe plane, the participant is asked to place a red fiducial in Slicer to mark the location of the facet joint.

12. Add an annotation in the transform recorder module indicating a move to the next facet joint.

13. Repeat steps 10 -12 for all facet joints.

14. Save Slicer scene (with the fiducial list created) and save the annotations file.

15. Thank participant for their time.

4.2.2 Experimental Setup

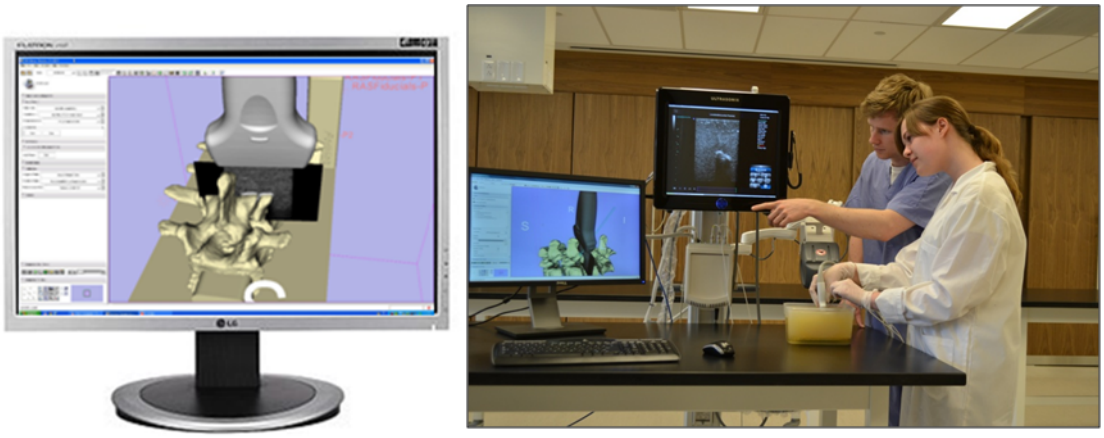

Figure 4.6: Experimental setup during the training phase, showing the phantom as well as the agumented reality training system

The experimental setup during the learning phase of the trainee is depicted in Figure 4.6. The trainee can see the 3D model of the spine, along with the ultrasound, simulated or real depending on the group. The study coordinator can also see the 3D model and the ultrasound on a separate monitor. In the testing phase, Figure 4.7, the trainee is left with only the phantom and the real ultrasound, while out of sight, the study coordinator still has access to the 3D model.

4.2.3 Results

Originally, only five minutes were allowed for the training session. However, both simulated and real ultrasound trained groups were doing poorly, resulting in the

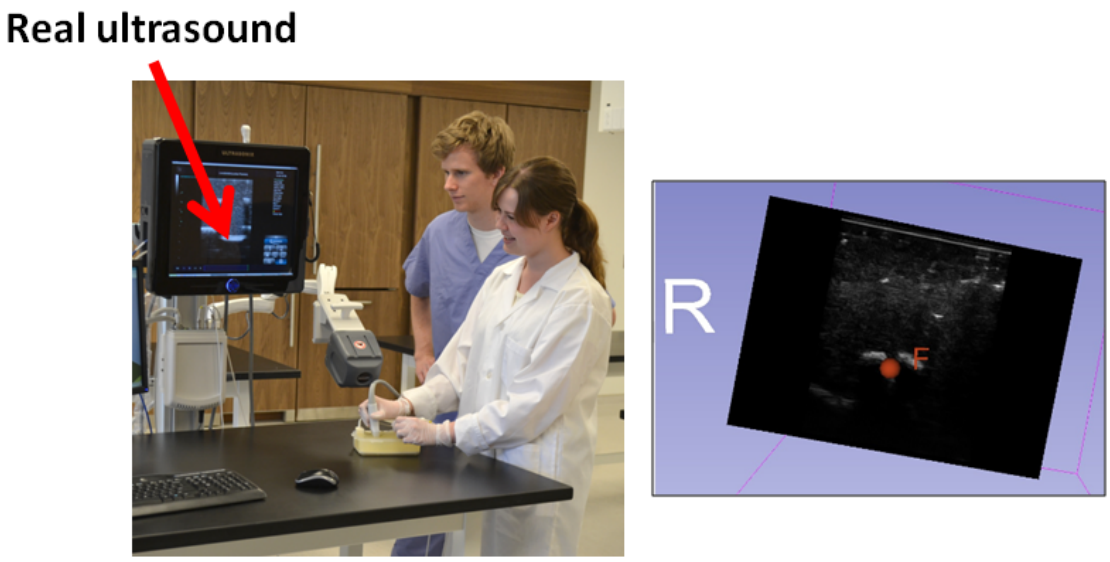

Figure 4.7: Experimental setup during the evaluation phase, emphasizing the lack of augmented reality

raising of the training time to ten minutes after the second day of the study,that is, after the first five participants. The integrity of the study is maintained since both groups improved after the time increase, and they were both scoring poorly before the time increase.

4.2.4 Statistical Analysis

The simulated ultrasound trained group (henceforth called simulated group) was found to have a mean score of 3.7/6 facet joints correctly localized, whereas the real ultrasound trained group (henceforth called real group) had a mean score of 4.8/6. However, the median for the simulated group was 6 facet joints, and the median for the real group was 5. As this data is nonparametric, a Mann-Whitney U Test was performed to compare the trainee performance, resulting in a p value of 0.79. The p value is above the conventional threshold value of 0.05, indicating that at this sample

Table 4.1: Participants trained with real ultrasound

ID Number	Facet joints identified correctly
1	5
3	0
5	2
7	6
9	5
11	6
13	5
15	6
17	6
19	5
21	6
mean:	4.8
median:	5

Table 4.2: Participants trained with simulated ultrasound

ID Number	Facet joints identified correctly
2	0
4	0
6	6
8	1
10	6
12	6
14	6
16	2
18	6
20	6
22	3
mean:	3.7
median:	6

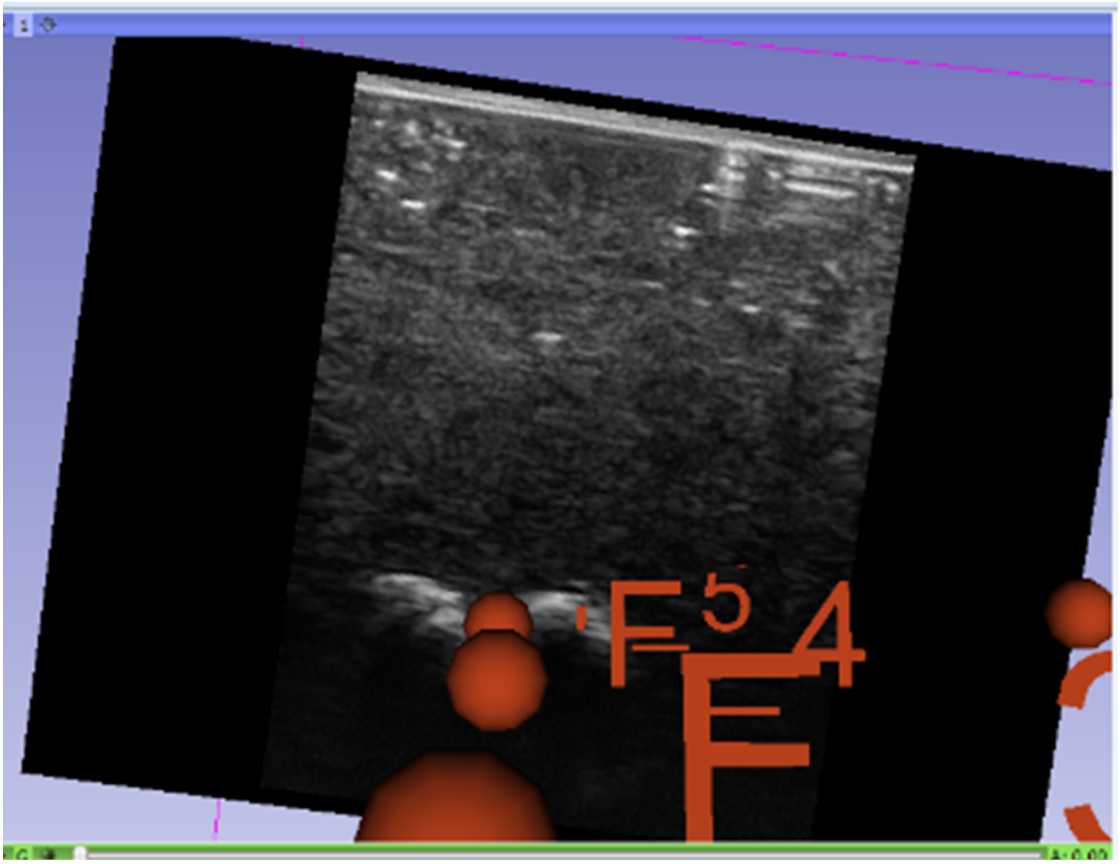

Figure 4.8: 3D Slicer view during evaluation of trainee performance, with fiducials representing where the trainee localized the facet joints

size there is no statistical difference in the scores between the two groups. Despite the limited sample size, the mean and median scores for the two groups only differ by 1 on the scale of 6. This is a promising step toward demonstrating similar efficacy of simulated and real ultrasound as an imaging training tool in spinal interventions.

4.2.5 Discussion

The results of these two experiments showed two separate things. The visual performance tests indicate that the simulator is fast enough to integrate into a clinical

environment. The usual workflow associated with ultrasound training is not interrupted by waiting for the simulator to update, the imaging appears seamless. The seamlessness has demonstrated that the simulator is fast enough to be usable. Usefulness of the simulator and its applications is demonstrated in the second experiment. Despite the simulator not simulating the backscattering and noise often associated with ultrasound, trainees using only the simulator did not perform worse than those using real ultrasound with noisier images. With such positive results, the study questions whether simulating more than the basic reflection and absorption is necessary, particularly when training minimally invasive spinal procedures on phantoms. The extra information could prove to inhibit the trainee, as it makes the shape of the bones more difficult to discern for novices. Nevertheless, adding tissue-specific noise patterns to the simulated images is achievable with a straightforward extension of the proposed method: by defining a procedural texture for each structure and superimposing this texture on the scanline pixels. As an example, we added Perlin noise texture to the generated image as shown in Figure 4.9

For any future studies, there is much to consider. Before any participants are recruited, a training video should be recorded. This would ensure that all participants receive exactly the same instruction and do not have any misconceptions regarding the shape they are looking for. The participants should also be reminded not to move the phantom itself, as this could cause inaccurate imaging. The inaccurate imaging results from a change in the distance between the magnetic field generator and the phantom. The farther the phantom from the generator, the more misaligned the image. Concerning participants, it could prove useful to study groups of people

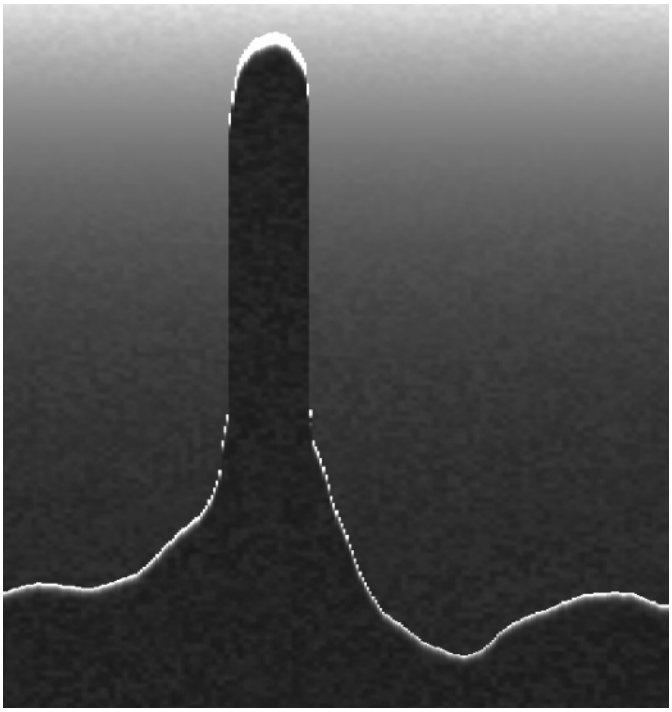

Figure 4.9: Image of a spine phantom with simulated speckle using Perlin noise

with the same educational background, and perhaps compare their results to others of different backgrounds. Engineering graduate students, especially those with significant experience with a virtual environment, whether through computer aided design courses or personal interests, could skew the results in favour of the simulated ultrasound. The next study should focus primarily on medical students, as they are the target audience for the training platform. The training time allotted should be fixed for the duration of the study. During testing, all participants should be told to localize the facet joints in a predetermined order. The tracked movements of the participants during each facet joint localization should be recorded, as should the time required to localize each joint. Statistical analysis should be conducted separately on the simulated ultrasound and real ultrasound groups. The test performed should be a one sided T-test, the results of which can then be compared.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Throughout this thesis, the development of an ultrasound simulator has been explored. The main challenge facing such a simulator was achieving image generation at a speed that the user would perceive as real time. This goal was achieved on the third implementation of the simulator. Using existing VTK classes proved to be simply too slow.

The goal of the simulator has always been to transcend the scope of a laboratory demo. The first steps towards actually being able to use this simulator in a relevant training environment have been taken. With the study conducted on novice trainees, it has been shown that there is potential for the simulator to be an effective training tool and replace real ultrasound.

As the simulator was developed as part of the PLUS (Public Software Library for Ultrasound Imaging Research,www.plustoolkit.org) toolkit, it is immediately accessible to research groups worldwide. This also allows it to easily be used in 3D Slicer, which entails a rapid transition to clinical testing scenarios. It is part of a

development framework that is aimed at putting medical imaging tools in the hands of physicians quickly.

5.2 Future Work

The simulator in its current state holds significant promise for future work. The basic framework to extend the simulator to a multiple model simulation is in place. This would allow the simulator to be applied in more complex and more realistic training scenarios, creating an environment where both the spine and the needle can be visualized. With multiple models, the training is no longer limited to the spine or other bony areas. Theoretically, any number of models can be implemented, but the effect of multiple models on simulation time will also need to be accounted for. If the simulator is expanded to include organ models as well, more complex ultrasound simulation may be necessary. However, with the SpatialModels class controlling intensity calculation, and the configuration files specifying a number of complicated parameters, the code is already there to facilitate this transition. A result of such developments in the simulator itself brings with it a plethora of opportunities for human operator studies. The teaching efficacy of each new advancement would need to be validated on different subjects, ranging from novice students to more specialized medical personnel. In short, the work detailed in here is a gateway to many fascinating projects.

Bibliography

[1] A. Lasso, T. Heffter, C. Pinter, T. Ungi, and G. Fichtinger, "Implementation of the plus open-source toolkit for translational research of ultrasound-guided intervention systems," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2012) Workshop- Systems and Architectures for Computer Assisted Interventions*, (Nice, France), pp. 1–12, The MIDAS Journal, October 2012.

[2] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*. Kitware Inc., 2004.

[3] S. Pieper, B. Lorensen, W. Schroeder, and R. Kikinis, "The NA-MIC kit: ITK, VTK , pipelines, grids and 3D slicer as an open platform for the medical image computing community," in *3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro*, pp. 698–701, 2006.

[4] T. Ungi, D. Sargent, E. Moult, A. Lasso, C. Pinter, R. McGraw, and G. Fichtinger, "Perk tutor: An open-source training platform for ultrasound-guided needle insertions," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 12, pp. 3475–3481, 2012.

[5] G. Andersson, "Epidemiological features of chronic low-back pain," *The Lancet*, vol. 354, no. 9178, pp. 581–585, 1999.

[6] E. Moult, T. Ungi, M. Welch, J. Lu, R. McGraw, and G. Fichtinger, "Ultrasound-guided facet joint injection training using perk tutor," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–6, 2013.

[7] W. Hedrick, P. David L. Hykes, and D. Starchman, *Ultrasound Physics And Instrumentation*. Mosby, 2005.

[8] J. A. Jensen, "Field: A program for simulating ultrasound systems," in *10th Nordicbaltic Conference on Biomedical Imaging, Supplement 1*, vol. 4, pp. 351–353, 1996.

[9] J. A. Jensen., "Simulation of advanced ultrasound systems using field ii," in *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, vol. 1, pp. 636 – 639, april 2004.

[10] A. Hostettler, C. Forest, A. Forgione, L. Soler, and J. Marescaux, "Real-time ultrasonography simulator based on 3d ct-scan images," *Studies in Health Technology and Informatics*, vol. 111, pp. 191–193, 2005.

[11] D. Magee, Y. Zhu, R. Ratnalingam, P. Gardner, and D. Kessel, "An augmented reality simulator for ultrasound guided needle placement training," *Medical & Biological Engineering & Computing*, vol. 45, pp. 957–967, Oct 2007.

[12] S. U. Gjerald, R. Brekken, T. Hergum, and J. D'hooge, "Real-time ultrasound simulation using the gpu," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 59, no. 5, pp. 885–892, 2012.

[13] A. Karamalis, "Gpu ultrasound simulation and volume reconstruction," Master's thesis, Technischen Universität München, 2009.

[14] A. Karamalis, W. Wein, and N. Navab, "Fast ultrasound image simulation using the westervelt equation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010*, pp. 243–250, Springer, 2010.

[15] F. P. Vidal, N. W. John, A. E. Healey, and D. A. Gould, "Simulation of ultrasound guided needle puncture using patient specific data with 3d textures and volume haptics," *Computer Animation and Virtual Worlds*, vol. 19, pp. 111–127, May 2008.

[16] M. Zhu and S. Salcudean, "Real time ultrasound needle image simulation using multi-dimensional interpolation," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2010*, vol. 6362 of *Lecture Notes in Computer Science*, pp. 429–436, Springer Berlin / Heidelberg, 2010.

[17] L. E. Bø, S. U. Gjerald, R. Brekken, G. A. Tangen, and T. A. N. Hernes, "Efficiency of ultrasound training simulators: Method for assessing image realism," *Minimally Invasive Therapy & Allied Technologies*, vol. 19, no. 2, pp. 69 – 74, 2010.

[18] O. Goksel and S. E. Salcudean, "B-mode ultrasound image simulation in deformable 3-d medium," *IEEE Transactions on Medical Imaging*, vol. 28, no. 11, pp. 1657–1669, 2009.

[19] D. Skehan, "Virtual training system for diagnostic ultrasound," Master's thesis, Worcester Polytechnic Institute, 2011.

[20] W. Wein, A. Khamene, D.-A. Clevert, O. Kutter, and N. Navab, "Simulation and fully automatic multimodal registration of medical ultrasound," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2007*, pp. 136–143, Springer, 2007.

[21] S. Gill, P. Abolmaesumi, G. Fichtinger, J. Boisvert, D. Pichora, D. Borshneck, and P. Mousavi, "Biomechanically constrained groupwise ultrasound to ct registration of the lumbar spine," *Medical Image Analysis*, vol. 16, no. 3, pp. 662 – 674, 2010.

[22] O. Kutter, R. Shams, and N. Navab, "Visualization and gpu-accelerated simulation of medical ultrasound from ct images.," *Computer Methods and Programs in Biomedicine*, vol. 94, no. 3, pp. 250–66, 2009.

[23] C. Sutherland, K. Hashtrudi-Zaad, R. Sellens, P. Abolmaesumi, and P. Mousavi, "An augmented reality haptic training simulator for spinal needle procedures," *IEEE Transactions on Biomedical Engineering*, 2011.

[24] J. Tokuda, G. Fischer, X. Papademetris, Z. Yaniv, L. Ibanez, P. Cheng, H. Liu, J. Blevins, J. Arata, A. Golby, *et al.*, "Openigtlink: an open network protocol for image-guided therapy environment," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 5, no. 4, pp. 423–434, 2009.

Appendix A

Configuration File Example

A.1 Configuration File Example

Listing A.1: First Generation Single Model Configuration Files

```
<PlusConfiguration version="2.2">
  <!-- Used by vtkUsSimulatorAlgoTest -->

  <CoordinateDefinitions>
    <Transform
      From="Image" To="Probe"
      Matrix="0.0       0.084     0.0       12.5
              -0.087    0.0       0.0       51.0
              0.0       0.0       0.087     -4.3
              0         0         0         1"
      Error="1.0" Date="022412_110829" />
    <Transform
      From="Phantom" To="Reference"
      Matrix="-0.0217475 0.999619 0.0170018  74.7552
              0.005450511 -0.016888  0.999843  23.1582
              0.999749   0.021836 -0.00503577 36.8398
              0          0         0          1"
      Error="0.965947" Date="022412_105333" />
  </CoordinateDefinitions>

  <vtkUsSimulatorAlgo
    BackgroundValue="155"
    ImageCoordinateFrame="Image"
    ReferenceCoordinateFrame="Phantom"

    NumberOfScanlines="256"
    NumberOfSamplesPerScanline="1000"

    ModelFileName="SpinePhantom2Model.stl"
    ModelToReferenceTransform="
      1 0 0 0
      0 1 0 0
      0 0 1 0
      0 0 0 1" >
    <RfProcessing>

      <ScanConversion
        TransducerName="Ultrasonix_L9-4/38"
```

```
39          TransducerGeometry="LINEAR"
40          ImagingDepthMm="60.0"
41          TransducerWidthMm="40.0"
42          OutputImageSizePixel="476 689"
43          OutputImageSpacingMmPerPixel="0.084 0.087"/>
44
45        <!-- Example for a curvilinear transducer: -->
46        <xScanConversion
47          TransducerName="BK_8848-axial"
48          TransducerGeometry="CURVILINEAR"
49          RadiusStartMm="5.0"
50          RadiusStopMm="60.0"
51          ThetaStartDeg="-60.0"
52          ThetaStopDeg="60.0"
53          OutputImageStartDepthMm="2.0"
54          OutputImageSizePixel="1200 800"
55          OutputImageSpacingMmPerPixel="0.084 0.087" />
56
57      </RfProcessing>
58    </vtkUsSimulatorAlgo>
59
60 </PlusConfiguration>
```