THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A FRAMEWORK FOR 3D ULTRASOUND IMAGING FOR HIFU TISSUE ABLATION USING OPENGL

HARRISON FESEL
SPRING 2019

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Mohamed Khaled Almekkawy
Assistant Professor of Computer Science and Engineering
Thesis Supervisor

Jesse Barlow
Professor of Computer Science and Engineering
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

The field of medical imaging, specifically ultrasound imaging and therapy, relies heavily on the visualization and proper image reconstruction of ultrasound data. Tools for visualizing ultrasound data and algorithmic methods for image processing have been widely used for medical purposes. Among the vast benefits of ultrasound in medicine is tissue ablation, which is the destruction of diseased tissue cells with heat, delivered using high-intensity focused ultrasound (HIFU). Challenges with this process arise when having to effectively target and ablate a region of tissue under the skin. This thesis sought to utilize computer graphics to achieve a 3D model that depicts the HIFU transducer targeting tissue imaged by B-mode ultrasound. OpenGL is an extensive and low-level graphics application programming interface (API) that was used to achieve this model rendering. The toolkit Qt was used to build a graphical user interface that incorporates Qt's OpenGL library. The application modeled a volume of tissue by compounding B-mode image slices. The focal point of the HIFU beam—the point of highest intensity—was visualized in the model and could be adjusted to choose the site for tissue ablation. Wave propagation was studied, and the finite difference method was considered as a means to solve the acoustic wave equation to incorporate ultrasound wave visualization within the rendered scene in the future. The program has the potential to be scaled to incorporate Computed Unified Device Architecture (CUDA), NVIDIA's graphics processor programming toolkit, which would provide sufficient processing power and parallelization to render the 3D models in real-time. The programming work expanded upon current research with the goal of increasing the accuracy of ultrasound therapy and cancer treatment. This honors thesis explores new methods for accomplishing 3D ultrasound imaging and a means of enhanced *in vivo* visualization for effective HIFU tissue ablation.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to extend my gratitude to my thesis supervisor and mentor, Mohamed Almekkawy, for all of his encouragement and guidance throughout the past two years and for being the individual that sparked my interest in ultrasound therapy. Outside the work of my thesis, I owe thanks to my family for their immense support and for motivating me to exceed my potential. Finally, I would like to kindly thank my close companion, Megan Czekalski, for assisting me by proofreading my writing and not letting me lose sight of my goals.

Chapter 1

Introduction

The intense development of medical imaging technologies has been an area of focus in research of late. Imaging practices such as computed tomography (CT), x-ray, and ultrasound exist as critical diagnostic and therapeutic tools. Whereas CT and x-ray imaging rely on the manipulation of light rays to discover injuries or disease within the body, ultrasound imaging utilizes high-frequency mechanical waves that are produced from a voltage being applied to piezoelectric transducer elements. Ultrasound is minimally invasive and has been largely turned to as a safer alternative than other imaging practices. CT and x-ray methods expose the patient to radiation, but the ultrasound vibrations stimulating the tissue do not produce harmful side effects. Ultrasound waves are typically delivered from an extracorporeal or intracorporeal transducer. For transducers placed directly on the skin, real-time images are displayed by processing the echoes received from anatomical structures inside the body.

Developments in ultrasound continue to occur as more research is conducted and technologies are created to enhance the practice. Some imaging applications for ultrasound, such as visualizing a child in the womb, may be commonly known. The tool can also be used to show blood flow or the structure of internal organs. However, a vast amount of research is currently being conducted in the area of ultrasound therapy. Ultrasound is conducive to heating tissue, and high-intensity focused ultrasound (HIFU) is a means of destroying diseased tissue; such a practice aids in preventing the metastasis of cancer cells in regions of the body.

The focus of this thesis was to develop a software application that increases a physician's accuracy with HIFU tissue ablation procedures by providing a 3D ultrasound model of the tissue and HIFU transducer. While tissue ablation offers a multitude of benefits in practice, challenges exist for accurately targeting tissue and visualizing the expected results. In order to effectively accomplish tissue ablation, proper visualization is pivotal. The bridge of this expectation can be constructed through the integration of computer science, software, and hardware technologies, as was studied and developed in this thesis.

In order for clinicians to understand ultrasound images and reap their benefits, it is imperative for them to have a thorough understanding of what they see. Image reconstruction is a crucial

aspect of ultrasound imaging. The transducer emits sound waves into the body and processes them once they reflect from internal structures. The time between reflections and the position of the wave indicate certain properties about the anatomy of the patient. Specifically, brightness mode (B-mode) ultrasound imaging involves scanning a plane of the body and rendering the results as a 2D image. This is perhaps the most commonly used type of ultrasound image formation, but due to the information restriction of 2D imaging, this thesis considers the possibility of a 3D model. Reconstructing the radio-frequency data as a 3D model in real-time would allow for an updated, comprehensive representation of a patient's body. Although the 3D model was achieved, real-time imaging was not accomplished in the scope of this thesis and will be discussed as a future goal of the project. Still, a volumetric interpretation provides greater detail concerning the anatomy and relative location of structures being observed; such an understanding would greatly enhance HIFU tissue ablation preparations and procedures.

Although a 3D rendering provides greater detail, many complications may arise. 3D real-time rendering requires significant computational power that is only accessible through a multi-core machine or the graphics processing unit (GPU). Spatial information about the transducer must also be precisely captured. Only by recognizing the transducer's location in physical space can a 3D anatomic model be reconstructed from the ultrasound data. The methods for overcoming such obstacles will be reviewed along with possible algorithms that efficiently accomplish image reconstruction. Achieving a 3D real-time rendering of ultrasound data would greatly benefit the field of medicine, especially when targeting tumors or lesions for HIFU ablation.

Modern graphics tools can be utilized to effectively visualize the 3D anatomy and focal point targeting for a transducer array of any shape. OpenGL, an open source graphics library, interfaces with the GPU to render scenes on a computer screen. OpenGL has been used in medical practices and in reconstructing ultrasound images, and its application programming interface (API) has the low-level capabilities needed for this thesis that allow for customization. This work implemented OpenGL's API in C++, and B-mode ultrasound data was reconstructed in a way that interfaces with a cylindrical and spherical transducer array. Rendering techniques and processing technology

were studied alongside the development of this 3D visualization. A graphical user interface (GUI) was designed in Qt for the manipulation of the 3D models to provide a physician with increased usability and detail. The ultimate goal of this work was to develop a simulation model using C++ and OpenGL for 3D diagnostic ultrasound imaging and HIFU simulation. Literature that was considered for this work is reviewed in the background chapter. An entirely noninvasive 3D simulation model for HIFU therapy that relates to physical constraints has not yet been developed. Thus, the work of this thesis is very critical and could provide immense benefits for the field of ultrasound therapy. The focus of this thesis resides in developing software and optimizing performance to provide a fully comprehensive, simple, and effective tissue ablation modeling application.

Chapter 2

Background

2.1 HIFU and its use for tissue ablation

To understand the context of the thesis work and its purpose, an overview of the application of this work will be provided in this background chapter. While the primary focus of this work was to visualize a tissue ablation model using a HIFU transducer array and captured ultrasound images, understanding the purpose of HIFU therapy and the benefits that would result from real-time modeling of this procedure elucidate the significance of this research area.

2.1.1 HIFU functionality

A prominent area of research for HIFU is the destruction, or ablation, of cancerous tumors or lesions. Invasive surgical procedures are often used for the treatment of cancer tissue; however, such operations may result in negative side effects. Therapeutic ultrasound has the convenience of being delivered as extracorporeal or endocavitary, methods that minimize disruptiveness. Chemo- and radio-therapy are often dangerous for patients undergoing such treatments for cancer [1]. The ability of HIFU to induce thermal damage to tissue was first investigated in 1942 by Lynn *et al.* [2][3]. They conducted tissue ablation tests on *ex vivo* liver tissue as well as the brains of animals, which notably did not cause any harm to the scalp [2][3]. In the 1950s, the use of HIFU was explored for thermal lesion creation for treating disorders such as Parkinson syndrome by Fry *et al.* [4][5]. A thermal lesion is an area of cell destruction under the skin of the patient. As time passed, HIFU began to be seriously considered as a means for ablating benign and malignant tumors via image guidance by the 1990s [2]. The use of ultrasound-guided HIFU therapy will be discussed in the next section, as it relates to the computer programming work pursued for this thesis. In order for effective tumor ablation to take place, the region must be heated at a high enough temperature to destroy the cells; the tissue temperature should be quickly increased to a cytotoxic temperature to prevent tissue vasculature from halting cell death [1]. Ultrasound imaging used for diagnostic purposes, such as B-mode imaging, can have a time-averaged intensity of 720 mW/cm2 [1]. However, HIFU intensities are much greater and assume a range of 100-10000

W/cm2 [1]. The HIFU waves rapidly heat the targeted tissue when the acoustic energy is absorbed. When a temperature between 60 and $95\,^{\circ}\mathrm{C}$ is reached at the focal point, coagulative necrosis occurs, which is cell death caused by "cooking" the tissue [6]. Although a high temperature must be reached to achieve thermal ablation, overheating must be avoided due to the risk of boiling and creating a thermal lesion of unpredicted size [6]. This mechanism is the main method of using HIFU to destroy cancer cells within a patient. Some of the various application and targeted tumors will be further discussed in the next section.

2.1.2 Applications for HIFU

The treatment of breast cancer via HIFU is an area of non-invasive ultrasound therapy that has seen successful results. In a study conducted in 2005, 22 patients with breast cancer were subjected to HIFU therapy as a means of completing non-invasive breast-conserving treatment [7]. After ablating the tumor in the patients and performing chemo-therapy or other treatment modalities, most of the patients saw a consistent reduction of tumor size and regression in the following months [7]. Another advantage of using HIFU tissue ablation for breast cancer is that the location of the breast suits the procedure, since the path of the ultrasound beam would not be obstructed by any bones or vital organs [8]. Much research has gone into the use of HIFU therapy for the treatment of pancreatic cancer as well. Individuals with pancreatic cancer have a severe mortality rate. Less than 1% have a 5-year survival rate when they are diagnosed, and about 80% of patients are at a stage where the cancer will not regress [9]. HIFU treatment for pancreatic cancer has been tested in China, and the studies suggested that the therapy can reduce the size of the tumor without prompting an onset of pancreatitis, and as a corollary, increase the life expectancy of the patient [1][9]. Hyperthermia HIFU treatment also has the potential of enhancing the effect of chemotherapy drugs such as doxorubicin, which by itself is relatively ineffective at penetrating the fibrotic, hypovascular stroma of the pancreatic tumor [8][9]. Experiments on rodents have shown that releasing doxorubicin from a temperature-sensitive liposome inside the tumor increases the

tumor's absorption of the medicine [8]. HIFU treatment can bring abdominal pain caused by the tumor in the pancreas under control as well [4].

The use of HIFU for the treatment of prostate cancer was proposed in 1992 by Chapelon *et al.* [10]. Ultrasound waves with amplified intensities were used to ablate a cancerous prostate tumor located in living rat subjects, and a large percentage of rats experienced complete tumor destruction from the experiments [10]. Such research led to the pursuit of HIFU therapy for practical prostate cancer treatment. More modern implementations for prostate therapy involve transrectal ultrasound-guided procedures [11]. Lastly, HIFU's applications toward the treatment of brain diseases will be mentioned. Some of the first research for applications of HIFU therapy was conducted in the 1950s and involved targeting areas under the skull, and the work explored the possibility of treating disorders like Parkinson syndrome [2]. Although the safety and promising results of HIFU treatment for brain cancer such as glioblastoma has heightened the research in this field, the obstacle of effectively transmitting acoustic waves through the skull and effectively guiding the beam with imaging still poses to be a challenge [12]. After reviewing these applications for HIFU, the purpose of the thesis work can be realized. Effectively simulating tissue ablation with a 3D ultrasound model using modern rendering strategies and technology could greatly improve the accuracy and efficiency of noninvasive cancer therapy. The study of image-guided HIFU and what it accomplishes in terms of accuracy will be discussed in the next section, especially the application of ultrasound imaging since it is integrated into the work of this thesis.

2.1.3 Ultrasound-guided HIFU and ablation simulation

The focus of this research aimed to model HIFU tissue ablation using 3D ultrasound imaging to determine the effects of heating an area under the skin. Corresponding the model to physical constraints in order to accurately target an area for ablation using entirely non-invasive means was another important motivation of the thesis. While the work of this thesis lies within the realm of computer modeling and simulation, image-guided HIFU therapy is an application that occurs

during the practice of real-time ablation. However, real-time imaging for the developed HIFU ablation model is something that will be pursued as a future goal of this project. There exist two main modes for HIFU image guidance: ultrasound imaging and magnetic resonance imaging (MRI) [2]. Magnetic resonance-guided HIFU (MR-HIFU) has been the primary method for image-guidance with HIFU [13]. Although MR-HIFU provides higher resolution and 3D treatment planning, ultrasound-guided HIFU (US-HIFU) is less invasive and significantly more cost-effective [2]. US-HIFU can render images at a high-frame compared to MR-HIFU and has the capability of providing the physician with fast feedback and measurements regarding tissue state during the ablation procedure [13]. Both methods work to image real-time updates of the thermal lesion produced from the ablation procedure. Figure 2.1 represents a depiction of thermal lesions throughout the duration of a HIFU procedure. As the target region becomes whiter, more cells have been destroyed. As US-HIFU is the preferred method over MR-HIFU but lacks the sophistication and accuracy, further research and exploration of its improvement are needed, which pertains to the motivation of this thesis work. Increasing the accuracy of ultrasound imaging for HIFU practices, most notably in 3D space, could provide physicians with greater control over the procedure while enhancing the usability of the relatively low cost and safe equipment.

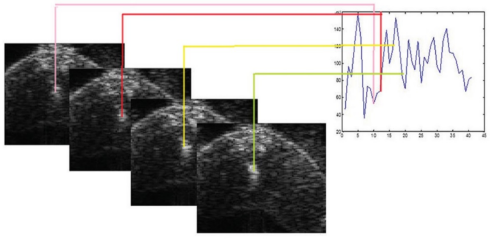


Figure 2.1: B-mode images captured over time as a region of tissue was exposed to continuous emission of ultrasound [14]

2.1.4 Simulating ultrasound emissions by approximating nonlinearity

The final aspect of HIFU visualization and simulation that should be understood for this research is the behavior of ultrasound waves in consistent and changing mediums and how this could affect the expected results of tissue ablation. Although this is a narrowed field of work that is more

related to physics than algorithm development, providing an accurate model of this phenomenon would provide even greater transparency and resolution for the 3D HIFU model constructed as a result of this research. Thus, a high-level description will be provided here, and the approach will be followed-up by a means of deriving a solution to the wave equation for linear and nonlinear modeling in the future work chapter.

The widely used speed of sound in soft tissue is 1540 m/sec [15]. However, when sound travel occurs between two different mediums with different acoustic impedances, reflections could occur as well as attenuation and absorption [15]. A difference in impedance resulting in reflections is a fundamental property that is depended upon for diagnostic ultrasound and assists in providing the composition of the echoes that reveal anatomic information. However, material properties such as speed and pressure inadvertently affect the propagation of the beam. When the amplitude of the ultrasound wave is relatively small, which corresponds to a low-intensity sound, the attenuation due to encountered mediums dampens the acoustic effects on the wave [16]. However, at high intensities, like the ones associated with HIFU, phase velocity, the rate at which a phase of the wave propagates through space, is manipulated in a nonlinear fashion by the pressure and acoustic properties of the material [16]. An example of wave propagation becoming nonlinear is ultrasound brain therapy. The speed of sound in bone is approximately double the speed of that in soft tissue, which results in significant phase aberration and distortion of the beam due to the impedance mismatch [17]. Mathematical methods for approximating the propagation of nonlinear ultrasound waves will be discussed in the future work chapter, as it pertains to the future visualization and simulation of ultrasound waves in soft tissue and changing mediums for the developed model. This will involve using finite difference methods in order to numerically solve the wave equation for sound. HIFU ablation points could be more accurately represented to the clinician if the acoustic wave behavior is represented for the phased array specifications and medium traits. This would be valuable to pursue following the work of this thesis and would expand the work for simulating real-time tissue ablation using an ultrasound-guided model instead of depending on MR imaging data for 3D simulation.

2.2 OpenGL and GPU-based rendering models

2.2.1 Scan-conversion algorithms in practice

A vast amount of research has been conducted to increase the speed and efficiency of visualization algorithms. In the field of ultrasound imaging, challenges exist with interpreting the raw data received from the transducer. In their fundamental form, the piezoelectric transducer elements receive sound waves that are echoed from the medium that the original wave was emitted to. Unlike A-mode ultrasound, which reads an amplitude signal from a single array element, B-mode ultrasound consists of data accumulated by the entire transducer array, thus allowing the information to be recompiled into a 2D representation. A single element transducer needs to be moved manually by the operator in order to attain the data for imaging, but multidimensional phased array transducers can electronically maneuver the beam [18]. The data are gathered in the form of image planes or "slices" of information. Each plane's cross-section corresponds to the depth of the imaged medium. Although the acquired data are collected in polar coordinates, using angle and radius parameters, most rendering tools require the data to be referenced using Cartesian coordinates, a system of orthogonal basis vectors [19]. The process of transforming the ultrasound data into Cartesian coordinates for visualization is referred to as scan conversion, and this is needed in order to display the data as they are captured. Under the condition that the sampling rate for the data acquisition is fixed and no over-sampling occurs, a one-to-one mapping between the B-mode capture and the rendered image does not exist due to the direction and proximity of the emitted acoustic waves [20]. This phenomenon necessitates an interpolative approach for image reconstruction for real-time visualization, which involves presenting the data in a form that can be understood by the operator.

Although scan conversion techniques are often computationally heavy, the process is largely handled by software rather than hardware systems, but allowing for programmable hardware components facilitates scalability and customization. The low-level interaction with the system may also result in smarter memory usage while achieving parallel computing, running processes simul-

taneously. A variety of algorithms exist for efficiently handling scan conversion in real-time, and a few such as nearest-neighbor, linear interpolation, and bilinear interpolation will be discussed in this section. These methods will closely relate to the decisions made for the rendering techniques and algorithms used for the programming portion of the work that will be delved into. Other methods for image reconstruction have been pursued in research that did not involve systematic interpolation. A research group at the University of Washington developed a means of achieving ultrasound image reconstruction on a media-processor-based system to deliver high computational throughput and a flexible framework [19]. These researchers stored the information for a polar to Cartesian coordinate mapping in a table architecture to save computational cost [19]. This may be a fast method for real-time scan conversion, but specific hardware (e.g. a media-processor) and technology is required. Although some studies have utilized hardware that is not readily available, increased reconstruction speed can still be achieved via the aforementioned algorithmic approaches with standard computer hardware.

Nearest neighbor is the simplest of the rendering approaches and requires little computational power. This method does not involve interpolation since the pixel that is colored on the imaging screen is given the color of the nearest ultrasound data point. If a pixel is located at a screen position (x, y), the color value assigned to that pixel will correspond to the color of a data point from the B-mode frame located at polar coordinates (θ, r) [20]. Therefore, the only computations performed during this process are discovering the closest adjacent data point in relation to the Cartesian pixel point (x, y). An image is shown below in Figure 2.2 that represents this method. While the nearest neighbor method is simple, it does not optimally color the pixels and can often result in low resolution.

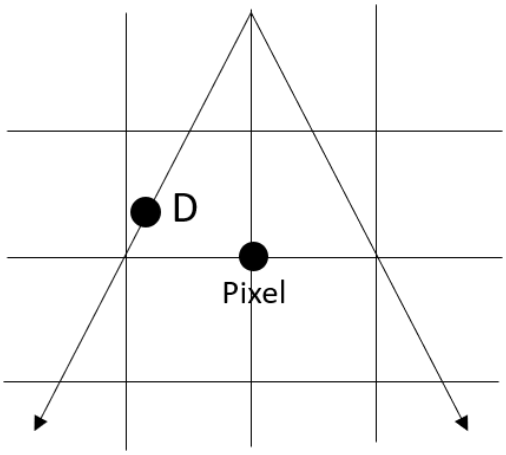


Figure 2.2: The sample data point (D) will be the value of the indicated pixel per the nearest neighbor algorithm [20].

Another approach for determining the pixel value is linear interpolation. Instead of assuming the value of the nearest data sample, like with nearest-neighbor, linear interpolation uses two sample points to acquire the color value for a given pixel. These two data points are from a B-mode image in an angular direction, and they lie along a roughly linear path with the pixel in question [21]. The pixel color can be interpolated based on the distance between these two points of the oversampled B-mode image (Figure 2.3). If L is the distance between the pixel in question and point D2, and D1 and D2 are data points along vectors $\theta 1$ and $\theta 2$ respectively, then the formula in equation 2.1 can be used to compute the interpolated pixel value [22]. This method provides greater accuracy than the nearest neighbor method but is still restricted by the bias of two data points when more could be used.

$$Pixel = D_1 * L + D_2 * (1 - L) \tag{2.1}$$

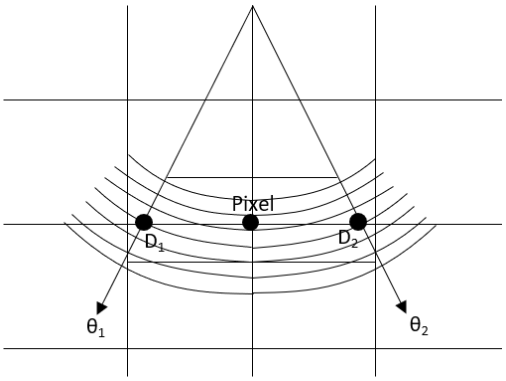


Figure 2.3: Linear interpolation of Pixel from the closest two points given a 4x over-sampling in the radial direction [22].

The final scan conversion method to be reviewed is bilinear interpolation. Bilinear interpolation relies on the data from four sample points instead of two to compute the value of the pixel on the screen. Although this method requires more computations, over-sampling is not needed. The bilinear interpolation algorithm has been implemented for cases of image resampling where scale or resolution is changed and unknown pixel values must be predicted [23]. Figure 2.4 shows the three interpolations that must be performed to calculate the pixel value. A point must be interpolated between D1 and D3 and between D2 and D4 [22]. Then, by interpolating between the two new points, the pixel value can be obtained as shown in the equations below where L1 is the distance between M1 and D3, L2 is the distance between M2 and D4, and L3 is the distance between the pixel and M2 [22].

$$M_1 = D_1 * L_1 + D_3 * (1 - L_1) \tag{2.2}$$

$$M_2 = D_2 * L_2 + D_4 * (1 - L_2) \tag{2.3}$$

$$Pixel = M_1 * L_3 + M_2 * (1 - L_3) \tag{2.4}$$

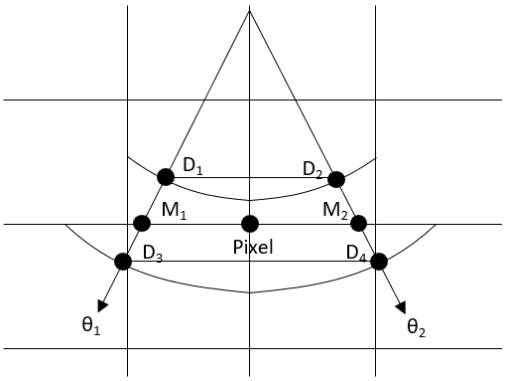


Figure 2.4: A bilinear interpolation of Pixel from the four closest data points from the two adjacent scan vectors [22].

2.2.2 OpenGL toolkit and GPU manipulation

After discussing some popular interpolation algorithms for ultrasound scan conversion, the benefits of using OpenGL as a tool for rendering ultrasound images can be understood [24]. OpenGL is an open source graphics library API which can be implemented by using a number of toolkits such as Glut, GLFW, and freeglut. For the research conducted in this thesis, a version of GLFW and freeglut were implemented [25][26]. OpenGL is widely used in industry and research for creating 2D and 3D environments through the incorporation of texture mapping and other visual effects. GLSL, or OpenGL Shading Language, is used to provide the GPU with code regarding how to process specific renderings [27]. Writing code for an OpenGL shader allows the programmer to divert attention away from the central processing unit (CPU) for tasks that the GPU can support, such as pixel interpolation. The GPU is capable of performing real-time scan conversion at a faster rate than the bilinear interpolation algorithm conducted on the CPU due to its utilization of parallel processing [20][22]. This conclusion validated the necessity of implementing a GPU-based texture rendering algorithm for this thesis work.

Steelman *et al.* studied the use of OpenGL as a tool for assisting scan conversion to utilize the graphics hardware present on modern PCs [20][22]. Through the manipulation of shaders, OpenGL texture mapping objects, an algorithm used the capabilities of OpenGL to map the raw B-mode data onto a sector shape that represents the appropriate Cartesian coordinate image [22].

This was accomplished by mapping quadrilaterals from the B-mode data to the image sector [20]. OpenGL uses bilinear interpolation to approximate the pixel values from provided image maps [22]. Since OpenGL uses the computational power of the machine's graphics processor, Steelman determined experimentally that the OpenGL adaptation of the algorithm, although using the same bilinear interpolation at a low-level, has increased throughput due to advantages in hardware and parallel processing. This discovery led to the use of bilinear interpolation in this project as well as the incorporation of OpenGL's use of the GPU to execute the algorithm.

The OpenGL API provides low-level rendering capabilities, which were beneficial for this thesis work for tasks such as manipulation of 3D space, user interaction and real-time processing, and texture mapping as previously mentioned. The OpenGL toolkits are typically compiled in C and integrate smoothly with C++ applications [27]. Thus, object-oriented design strategies could be implemented alongside the rendering algorithms.

2.2.3 3D ultrasound imaging

Most representations of ultrasound signal data are displayed to the operator in two dimensions. Specifically, diagnostic tools allow the clinician to view the plane parallel to the transducer array aim or perpendicular to the skin. The acquired image depth and scanning range depend on the frequency of the mechanical waves as well as the shape and type of transducer.

2D ultrasound imaging is most commonly used for *in vivo* diagnoses and can be used to view the heart, blood vessels, eyes, thyroid, brain, and more [28]. However, the qualitative understanding of the produced image is limited to a single plane and does not provide information regarding volume or surrounding tissue. Accomplishing real-time 3D imaging of ultrasound data increases the anatomical understanding of the area of interest by allowing for more accurate diagnostics and visualization, which was desirable for the HIFU tissue ablation model developed in this thesis. As for surgical procedures where ultrasound is used for guidance, 3D visualization—especially in real-time—would give the operator a stronger intuition as to where their probe or needle lies under

the skin. For example, ultrasound is used for guided needle biopsy, and 3D imaging would provide more information to the physician. 3D ultrasound has been investigated as a guidance tool for a prostate biopsy for enhanced real-time vision [29]. There are three main stages of processing that allow for 3D ultrasound imaging: acquisition, reconstruction, and visualization [30].

The challenge of acquisition involves realizing the position of the 2D transducer in physical space. The relative position of the probe must be captured while ensuring reconstruction without errors from involuntary motion or other artifacts [30]. Methods for acquiring data for 3D reconstruction could involve the use of 2D array transducers or mechanical 3D probes [30]. 2D array transducers that collect data relative to the dimensional propagation of their elements can steer the beam electronically in the azimuth and elevation dimensions to achieve a volumetric scan [30]. Another way of accomplishing the data acquisition is through the use of mechanical 3D probes. 3D probes are relatively new in the medical field and retrieve the echoes in a pyramidal volumetric form in order to represent the anatomical position data [31].

The reconstruction portion of the 3D ultrasound imaging is handled by software and ultimately determines the efficiency of the process. In order to effectively reconstruct the voxels, the 3D version of pixels, from the acquired data in real-time, an algorithm that utilizes parallel processing must be incorporated. Real-time reconstruction algorithms for 3D voxels can be grouped into three categories: Voxel-Based Methods (VBMs), Pixel-Based Methods (PBMs), and Function-Based Methods (FBMs) [30]. Consider the space in which the ultrasound volume will be rendered as a 3D grid [32]. For VBMs, each voxel or cube in this grid is assigned a brightness value based on the acquired B-mode scans [32]. When deciding the value of the voxel, the interpolation methods discussed earlier can be used. The 2D planes of ultrasound data with two data points nearest to the center of a voxel are used to get the interpolated color value for that voxel [33]. This interpretation can be understood from Figure 2.5. The 2D grid that is displayed represents the center of an array of voxels, and the points on the line represent the centers of pixel data captured from a B-mode scan (Figure 2.5). The normal vectors of the adjacent B-mode scans that intersect with the center of a voxel indicate the data points from the B-mode images that will be used for interpolation

[30]. Thus, bilinear interpolation determines the color of the voxel according to the four points surrounding the intersection point of the normal vector and the B-mode scans.

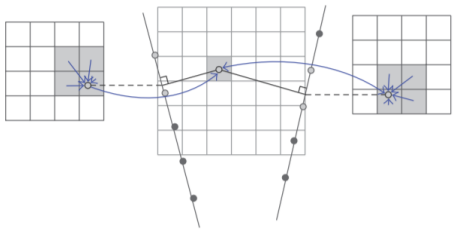


Figure 2.5: An interpolated voxel using VBM from the two closest surrounding images where a normal to each image is calculated [30].

The second method type to be discussed for 3D reconstruction is PBM. In PBMs, the B-mode scans are traversed, and the brightness value for each pixel is assigned to one or more voxels [30]. The voxels that they are assigned to lie in between the scans. Therefore, low scanning iterations or small voxel sizes can cause gaps in the voxel filling, which would result in an incomplete image [30]. In order to eliminate this expected issue, two stages occur for PBMs: a distribution stage (DS) and a gap-filling stage (GFS) [33]. DS involves a more complicated pixel interpolation, while GFS can be performed through bilinear interpolation or the application of a kernel [30]. The last mechanisms for reconstruction that will be discussed are FBMs. Although FBMs were not considered for the work of this thesis, this approach is a unique way of accomplishing voxel interpolation. FBMs map the relative positions and values of pixels from the B-mode images to a specific function like a polynomial, and the coefficients are determined for the best match [33]. Then, the function can be used to compute values by evaluating the function at regular intervals for the rectangular voxel array [33].

The final component of 3D ultrasound imaging is visualizing the computed voxel array data, specifically in real-time. Although this portion of the process is not mathematically intense, volume rendering (especially at a high resolution) requires significant processing power. There exist three primary methods for volume rendering: slice projection, surface rendering, and volume rendering [30]. Arbitrary image planes are captured for slice projection and are displayed in real-time in a

way that provides information about the perpendicular axes of the volume being imaged [34]. This algorithm for 3D imaging is straightforward and allows for the transformation of each planar view but requires the operator to mentally understand their 3D relation [34][30]. Another approach for combining slices involves acquiring multiple 2D ultrasound images at varying degrees of freedom, and the slices are overlaid on their intersection lines in 3D space to be visualized as a 3D object [35]. Surface rendering assumes information about the shape of the region or object of interest and infers a simplified geometry [30]. This technique allows for quick rendering by incorporating volume approximating algorithms such as marching cubes but has the risk of yielding false positives or lacking certain feature details due to over-simplification [34]. Lastly, volume rendering effectively recreates the anatomy of the scanned object by incorporating translucency [30]. In order to achieve more detail for enhanced diagnostic capabilities, ray-casting is used, but mapping voxels directly onto the screen is computationally expensive and requires more processing power [34].

The tactics applied for this thesis work include a combination of slice projection and surface rendering to optimize the 3D understanding of the object's volume, computational efficiency, rendering simplicity, and feature extractability. While this background section highlighted some approaches to 3D ultrasound imaging, the specifics of the reconstruction and rendering techniques utilized will be covered later in this document. The next section will further delve into the expectations of 3D medical imaging, the hardware requirements, and the programming strategies needed to achieve effective real-time renderings in the future.

2.3 Computational cost and feasibility of real-time 3D rendering

2.3.1 Hardware specifications: GPU versus CPU computational processing and rendering

The benefits of producing comprehensible 3D ultrasound images in real-time are immense in the field of medicine, and the computational strategies discussed in the last section about re-

construction and rendering are critical to achieving this goal. However, along with valuable and understandable images, the images must be produced at a frame rate fast enough for immediate decisions and responses to be made by the physician. This is especially crucial within the context of this thesis work, where the visualization will allow the operator to model and select the site to deliver HIFU. If the connectivity is not up to speed, both the practicality and effectiveness of the program will decrease. While most software processing is conducted by the CPU, many computational benefits have been achieved by utilizing the GPU to handle the processing of ultrasound rendering programs.

Wave and ultrasound simulation are closely related to the future goals of this thesis. Proper and fast production of ultrasound images for the performance of HIFU simulation would greatly enhance therapeutic accuracy and results. The implementation of OpenGL in this work stores all of the rendering data in buffers and draws the nearest object in view of the screen. This method is similar to ray casting for 3D ultrasound imaging, as ray casting stores all the data points along each B-mode image and requires low-level and fast storage utilization [36]. This approach differs from direct volume rendering (DVR) which only stores the information of the closest data point to the screen. In the case of this work, all RF data must be maintained and stored. Since a great amount of data must be remembered, the visualization algorithms could be greatly optimized by utilizing the capabilities of the GPU.

Kutter *et al.* recognized the potential of using standard APIs to achieve GPU access without using manufacturer-dependent software like NVIDIA's CUDA, which will be discussed in the next section [36]. Instead, Kutter *et al.* implemented their framework in C++, OpenGL 2.1, and GLSL [36]. Their research team managed to execute read and write functionality with the GPU by using 2D and 3D textures within the API of OpenGL, as these data objects are processed and stored by GPU hardware. Kutter *et al.* compared the throughput of their ultrasound imaging framework for multi-model registration of CT and ultrasound using Wein's model of a 2.2 GHz mobile Core 2 Duo processor to different GPUs [36]. For a single image with dimensions 128x96, the Core 2 Due processor took 3.5 ms to compute the simulation [36]. The same process conducted by the

GPUs was performed at a significantly faster magnitude, especially when increasing the number of simulated tiles [36]. This research demonstrates the advantage of using the GPU for ultrasound simulation tools. However, using a specific rendering toolkit like NVIDIA's CUDA to interface with the GPU would be more straightforward than the strategies discussed in this section.

2.3.2 Integration of CUDA and parallel processing

Utilizing the processing power of the GPU opens the door to a means of exploiting parallel processing. Parallel computing is especially useful for real-time ultrasound imaging since a large amount of data must be gathered and reconstructed quickly to accommodate fluent usability. While OpenGL was discussed as a method for GPU-based rendering and data storage, NVIDIA's CUDA toolkit was considered for extensive manipulation of the GPU's computational power [37]. Choe *et al.* investigated the use of GPU supported volumetric reconstruction for A-mode data collected by an ultrasound ring array [38]. Their algorithm was first attempted using a multi-core CPU-based imaging system to achieve real-time results, and the method displayed the captured data at 10 frames per second [38]. By implementing GPU-based software, the display rate was increased to 45 frames per second [38]. In order to display the captured RF data at the frequency needed to provide the physician with uninterrupted control, the volume rendered that is targeted for ablation must be rendered quickly. CUDA threads were assigned to decode each A-scan from the 64 transducer channels for the mentioned method [38]. One CUDA stream handled signal processing while the other managed data transfer [38]. This demonstrated a use case of CUDA in volumetric ultrasound reconstruction and imaging.

In the context of the work of this thesis, CUDA can be implemented to support the computational load associated with the rendering data and real-time processing. Where OpenGL is used to render vertex data onto the screen, CUDA is used for data generation and calculations [39]. OpenGL's buffers can be transferred back and forth with CUDA's memory space, allowing for fluid integration [39]. Although CUDA was not integrated into this project, it was considered, and

a framework will be discussed that describes how to inject this technology into the current code in the future.

Chapter 3

Design Strategies and 3D Rendering Methods

3.1 Problem and solution approach

The purpose of this research work is to develop a visualization program to effectively target regions within a body for HIFU therapy by using ultrasound imaging, and the tool will incorporate the necessary specifications to precisely target a region for tissue ablation. The properties of the transducer array will be entered into the system in order to accommodate a variety of equipment. The tool should be easily understood and operable by a physician. The captured radio frequency (RF) data should be displayed in real-time in the future to provide dynamic comprehension of the subject. When the RF data is rendered as a 3D volume, the desired region of interest can be found for a specific patient, and the volumetric scene of the tissue model and HIFU transducer array can be interacted with. This will prompt the physician to recognize the tissue to be ablated in real-time.

For this project, C++ was chosen for its fast compiler and its object-oriented capabilities. OpenGL was the selected application programming interface (API) due to its wide use in medical imaging practices, inherent C++ libraries, and GPU-based graphics rendering tools and texture manipulation. The algorithms developed for this work consisted of preprocessing data to be prepared as vertex information for the GPU. As for the interface that facilitates user interaction with the rendered scene, Qt was used. Qt has vast online documentation and is a flexible C++ graphical user interface (GUI) toolkit. Finally, a means of parallel processing using CUDA was considered, but not implemented, to achieve the 3D rendering in real-time as the B-mode images are acquired and will be discussed further in Chapter 5 since it was not successfully integrated as part of this work.

3.2 Cylindrical and spherical transducer array generation in C++

The first task of the work was to effectively model a cylindrical panel and spherical panel transducer array with C++ code. Effectively, the goal was to map the data to the coordinates of each

vertex point for each element of the array. This task was already accomplished in MATLAB by our research group, and functional code was generated in C++ that utilized the MATLAB framework. The original MATLAB version was slow, both computationally and at rendering the array model, and MATLAB does not facilitate the low-level rendering tools that would have better suited the purpose of this project. Figure 3.1 below depicts the cylindrical panel transducer as constructed in MATLAB. This transducer consists of 704 piezoelectric elements, a 40 mm radius of curvature, 1.3 mm element width, 30 mm element height, 1.5 mm element spacing, 11 elements to create the toroidal shape, and a 40 mm radius of curvature of the toroidal shape. These specifications were used as the primary template for the MATLAB to C++ conversion of the code. The spherical panel shown in Figure 3.2 has 137 piezoelectric elements, 1.5 on-center spacing of elements, 1.3 mm element width, 20 mm diameter, and a 40 mm radius of curvature.

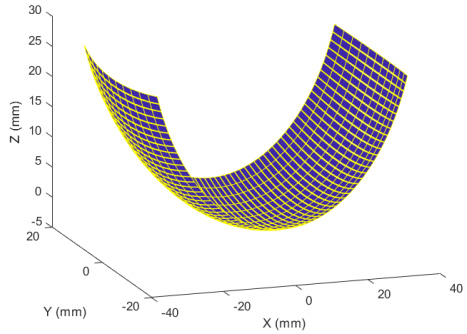


Figure 3.1: Cylindrical panel transducer array model visualized in MATLAB.

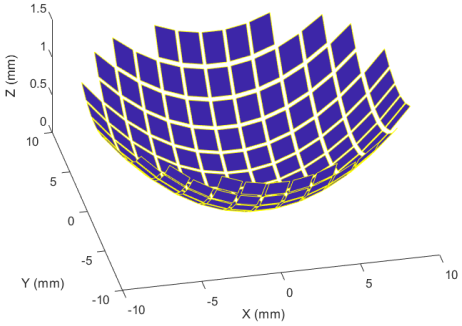


Figure 3.2: Spherical panel transducer array model visualized in MATLAB.

The construction of the transducer data relies principally on linear algebra operations and is conducted in multiple steps. Following the progression of the code's execution, the first step is to explicitly define the parameters for the model as indicated in the previous paragraph. The created cylindrical panel function designated orthonormal 3D basis vectors and then built a linear array in the x and y-direction based on a number of specified elements, and the function outputted a matrix where each column holds information regarding a position, size, and rotation of the corresponding indexed element. Once this function execution was complete, another function handled the array data and modeled it in 3D space. A challenge existed in converting the MATLAB code to C++ due to the simplicity of the MATLAB code regarding matrix transformations and arithmetic. Where these processes were natively understood by the MATLAB compiler, they had to be explicitly coded in the C++ version in a way that allowed for intuitive reuse and understanding. Members of our research lab converted the mathematical operations for the cylindrical array construction into C++, and the work of this thesis involved appropriately visualizing this data as well as performing the mathematics and visualization for a spherical array transducer. The mathematics involved for the cylindrical panel array and spherical array construction will be discussed in this section. The techniques used for visualization will be covered in the next section, along with other elements of the OpenGL rendered scene for tissue ablation.

A function was created for building the properties associated with every element of the transducer array. These properties later assisted in properly drawing the array in 3D space. The function for the cylindrical panel received the number of elements in the x and y-direction, spacing in the x and y-direction, width in the x and y-direction, radius of curvature, position of the center of the array, unit vector in the x-direction, and unit vector in the z-direction. The x-angle was updated every iteration for each element in the x-direction and was used to compute the position and rotation of each array element as the elements in the y-direction were looped through. The matrix containing all of the metadata regarding each element was returned from this constructed function. Once the metadata matrix was computed, the attributes were decoded and separated into three two-dimensional C++ vectors for the x, y, and z-coordinates of each vertex. One array element had four

corner vertices, and thus the two-dimensional vectors had four column items for each row. In this implementation, these positions were stored on the heap for direct memory access and were later used to construct the vertices using the OpenGL API.

The computations for the spherical transducer array were very similar to the cylindrical panel. The function that constructed the element data for the spherical array accepted the following parameters: on-center spacing of elements, width of elements, diameter of array, radius of curvature, position of the center of the array, unit vector in the x-direction, unit vector in the z-direction. Similar to the cylindrical panel, metadata was determined iteratively for the x and y-direction of the elements in the spherical array. This data was decoded and stored within two-dimensional C++ vectors to be processed for rendering. The following section will delve into the techniques used for rendering the transducer array and the other objects in the OpenGL scene, contributing to the main area of work.

3.3 OpenGL scene construction

A critical part of this work was to visualize the HIFU transducer and its relative location to the tissue region of interest. The RF data was rendered in a cubic shape, displaying slices of B-mode images that composed a volume of tissue. Reconstruction techniques discussed in the background chapter that were applied will be delved into, along with ultrasound 3D imaging techniques. An approach was chosen to best suit a model for a physician to select an area for tissue ablation from pre-captured data. A framework for incorporating real-time 3D imaging in this model, which was considered but ultimately not integrated into the final product of the work, will be discussed later in this document.

The OpenGL libraries that were at first adapted for this project were GLFW version 3.2.1 and glad. The linear algebra library, GLM, was also imported and included to streamline matrix operations. Once the 3D rendering was integrated with a Qt interface, Qt's version of OpenGL, which is similar to freeglut, was utilized. For the setup with GLFW and glad, the toolkit as well

as the window context was initialized. Then, the window frame and mouse input buffers were constructed for real-time updating, and any user inputs were set to interact with the GLFW window specifically. Using the included glad C file and library, every OpenGL function per the API was loaded into the program. This allowed for OpenGL functions to be used and to appear within the GLFW window constructs. 3D graphics were enabled for the scene for the GPU. The shaders and textures were initialized, and their functionality will be discussed at a later point. Finally, the vertex array object (VAO) was bound to an OpenGL buffer, which was then processed by the GPU, and the render loop was created. Within the render loop, the 3D scene was updated iteratively until the program was terminated upon input. The camera, projection, and model matrices, which were responsible for proper viewing, were updated as the user interacted with the 3D environment. This high-level framework for the OpenGL rendering process should be kept-in-mind as the key components are refined and explained.

3.3.1 Transducer array rendering

Having constructed the vertex data for the transducer arrays, the methods for rendering the elements will be discussed in this section. The primary focus of this work was to visualize the cylindrical array. Thus, the rendering techniques will be evaluated under the context of the cylindrical panel. However, the same processes apply for rendering the spherical array or any future added array constraints as well, with minor updates or necessity for method overloading.

The transducer array being rendered in the OpenGL environment was intended to model the HIFU transducer delivering mechanical waves to the tissue region to be ablated. The 3D array was located in 3D Euclidian space. The image previously shown in Figure 3.1 represented a MATLAB visualization of the cylindrical panel array. The goal of this portion of the work was to render the same model with OpenGL. Once the x, y, and z two-dimensional C++ vector objects (where each object contains one of the three 3D coordinates that pertain to the four corners of each element of the phased array transducer) were developed from the parameters provided for the cylindrical panel

computations, they were designated to a vertices array that contained sequential data for drawing OpenGL triangles. This vertices array was comprised of vertex data and texture coordinates. The shader design for the transducer array is represented in the code in Figure 3.3 and Figure 3.4, using the GLSL syntax. For the vertex shader code shown in Figure 3.3, the final position of the on-screen vertex was presented relative to the world coordinates of the rendering space. The position in world space was calculated by multiplying the projection matrix, view matrix, model matrix, and vertex position, in their respective order. The texture coordinates were decided by relating the *x* and *y*-texture coordinates to the Barycentric coordinates of a triangle. Thus, (0,0) corresponded to one corner of a triangle and (1, 0) to another. The fragment shader importantly incorporated GPU bilinear interpolation as discussed in the background. The textures mapped to each element of the transducer model were interpolated between two triangles that formed a quadrilateral. Thus, the color mapping was computed on the GPU that executed the GLSL code.

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec2 aTexCoord;

out vec2 TexCoord;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main()
{
	gl_Position = projection * view * model * vec4(aPos, 1.0);
	TexCoord = vec2(aTexCoord.x, 1.0-aTexCoord.y);
}
```

Figure 3.3: The GLSL vertex shader for the OpenGL transducer array model.

```
#version 330 core
out vec4 FragColor;

in vec2 TexCoord;

// texture blend
uniform sampler2D texture1;
uniform sampler2D texture2;

void main()
{

       FragColor = mix(texture(texture1, TexCoord), texture(texture2, TexCoord),
       0.2);
}
```

Figure 3.4: The GLSL fragment shader for the OpenGL transducer array model.

3.3.2 HIFU focal point

Along with rendering the HIFU transducer array, the rays of the propagation direction and focal point were visualized. The intersection point of the rays represented the focal point and the region within the tissue that will be targeted for ablation. Based on the transducer array, a default focal point distance was specified; in the current implementation, 0.2 units of Euclidian distance corresponded to a millimeter in physical space for simplicity. However, this was estimated for developmental purposes and can be redefined later for increased accuracy. Four red colored rays were emitted from the four outer corners of the cylindrical panel array. Figure 3.5 and Figure 3.6 represent the GLSL shader files for the vertex and fragment shader for the focal point rendering respectively. Unlike the texture mapping that occurred for the transducer array model, the code in Figure 3.5 designated a constant RGB color for the fragment shader shown in Figure 3.6.

```
#version 330 core
layout (location = 0) in vec3 aPos;

out vec4 vertexColor;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main()
{
       gl_Position = projection * view * model * vec4(aPos, 1.0);
       vertexColor = vec4(1.0, 0.0, 0.0, 1.0); // set the output variable
       to a dark-red color
}
```

Figure 3.5: The GLSL vertex shader for the OpenGL focal point rays.

```
#version 330 core
out vec4 FragColor;

in vec4 vertexColor; // the input variable from the vertex shader (same name and
same type)

void main()
{
    FragColor = vertexColor;
}
```

Figure 3.6: The GLSL fragment shader for the OpenGL focal point rays.

3.3.3 RF data interpolation and rendering

One of the most challenging pieces of this thesis was the final rendering model. This involved intuitively and quickly representing B-mode data as a 3D graphic that was to be displayed in the OpenGL scene. Sample RF data were provided from our research lab and were collected using a linear phased-array transducer and a tissue phantom. Obstructions within the captured data mimicked objects, such as bone or lesion tissue, that reflect a higher percentage of ultrasound waves. In order to avoid the computational expense of scan conversion and the irregularity of a

3D sector graphic, a one-to-one mapping between the B-mode RF data and Cartesian coordinates of the OpenGL scene were assumed; bilinear interpolation via GPU processing was implemented as the vertex color interpolation method. The ability to include scan conversion in the future is available, and the data would assume a 3D sector shape instead of a rectangular prism. Looking at the methods of visualization discussed in the background chapter, a new method of visualization was attempted that merged the ideas described by VBMs and PBMs. Much of the research and developmental work was gauged towards this technique. The concept of voxel filling utilized the field's current understanding of a 3D grid composed of cubic units but adapted the interpolation concept to only finding the colors along two axes—the y and z-axes in the instance of this work. By casting the RF data along parallel planes in Euclidean space and using the assumption previously mentioned, these colors can be directly gathered from each slice of data. A PBM approach was also adapted by bilinearly interpolating the matrix of the B-mode capture to a scale large enough for visualization. Thus, the pixels in between the expanded data points were defined by interpolating the four nearest RF data points using the OpenGL shader. The first attempt at achieving this goal implemented voxel-by-voxel plane shading but was a bad use of memory since color data had to be stored for each vertex on the stack of the program. The method led to poor usage of CPU memory and slow rendering and processing. In the second and current method, each B-mode image was saved as a one-to-one mapped JPEG file using the stb image processing library for C++. For developmental purposes, the matrix of RF data was transcribed to a text file format to be easily parsed and read into the program.

Then, once the B-mode slices were saved as images, the images were mapped as textures to the cubic region in 3D space and visualized using OpenGL's GPU-based bilinear interpolation. The slices were placed at a closeness to each other to simulate consistency in physical imaging space and to resemble a complete volume. To increase accuracy at a later time, the distances of each plane could be determined relative to the captured data and ultrasound probe being used to gather the images, pre-composed or in real-time. The focal point rendering lies within the volumetric region, indicating the site for ablation.

3.4 Graphical user interface design and OpenGL integration

In order for the interaction with the 3D model environment to be comprehensive, an effective graphical user interface (GUI) had to be designed and implemented per the thesis work. Although rendering the OpenGL scene in a C++ console application sufficed for achieving the renderings, challenges existed in integrating interactive capabilities for the operator. After investigating different toolkits for producing a user-friendly GUI for the scene, Qt was selected due to its extensive online documentation, platform independent installation, and integration of OpenGL for graphics developers.

3.4.1 Qt and OpenGL integration

The most important aspect that had to function within the Qt interface was the rendering of the already-designed and computed models. The Qt toolkit revolves around the existence and manipulation of widgets, or components that appear on the program window that the user can directly affect. Examples of widgets are text boxes, buttons, check boxes, dialogs, and so on. An advantage of utilizing Qt, as mentioned earlier, is that the toolkit contains an OpenGL widget option and a plethora of tutorial and example applications. Qt version 5.12.0 was used along with its native integrated development environment (IDE) Qt Creator version 4.8.0. Qt's native OpenGL library is a platform-independent wrapper for GLX, WGL, or AGL C language APIs [40]. The functionality of the module is similar to that of Mark Kilgard's GLUT library or the more modern and open source freeglut [40].

Since the Qt OpenGL module was comprised of different libraries then the ones used for the original model renderings, GLFW and glad, the initial code had to be adapted to the new configuration. Although the implementation of rendering was comparable, difficulties existed with configuring the existing code with the object-oriented structure of Qt's widgets and windows. A GUI could be created using Qt's form design IDE, which provides the programmer with drag-and-drop abilities to customize the user interface for the application and add code to the layout, but

the OpenGL widget had to be custom designed in order to support the necessary functionality of this work. The following classes were created for each of the models that were designed: cylindrical panel transducer array, spherical panel transducer array, focal point visualization, and RF data volume composition. The fields and vertex array objects were initialized upon the creation of the Qt window. The shaders and OpenGL shader program objects were instantiated and compiled as well. Then, instead of implementing a render loop to iteratively process window updates like before, a Qt OpenGL draw function was defined. This function was executed by the OpenGL widget whenever a visual update occurred with the rendering. In this draw function, the rendering code was maintained from the original implementation with adjustments to fit the different configuration. Challenges were encountered when maintaining the viewing pipeline for the projection, view, model, and world coordinates, especially when achieving the necessary rotation and movement around the scene. The same implementation was transcribed from the original code and the final outcome will be further discussed in the results chapter of this thesis.

3.4.2 Transducer array customizability

A purpose for the GUI was adjusting the parameters of the cylindrical panel array, spherical panel array, and any other HIFU transducer models that could be added to the framework in the future. There exist many kinds of HIFU transducer arrays that a physician could be using for ultrasound therapy. Thus, the modeling tool should comply with the capability of adjusting the specifications of the array elements being visualized. For this thesis work, two array models were studied: cylindrical panel and spherical shell. This section will discuss the GUI design for each and the pipeline for changing the model being visualized.

The cylindrical panel has the following parameters as previously discussed: number of elements in the x and y-direction, spacing in the x and y-direction, width in the x and y-direction, radius of curvature, position of the center of the array, unit vector in the x-direction, and unit vector in the z-direction. Each of these fields were presented to the operator in an understandable

way that allowed for simple adjustment and updating. After considering different approaches for presenting a GUI with these fields, a Qt dialog window was decided upon. A Qt dialog consists of a window form that is separate from the main application and only contains the parameter fields for the cylindrical panel. If the user intends to update the specs of their modeled cylindrical panel array, then the user could open the dialog window from a menu bar, change the parameter metrics, and submit the changes to the OpenGL widget. This pipeline does not disrupt the view and space of the form containing the OpenGL widget with the rendered models. Figure 4.5a in the results chapter displays an image of the described dialog and GUI and what the physician could expect to see when updating their HIFU transducer in the scene.

The spherical transducer array follows the same procedural pipeline as the cylindrical panel. To update the array's parameters, a dialog window can be opened from the main application, and the adjustments can be submitted to the OpenGL widget. A review of the spherical array parameters is as follows: on-center spacing of elements, width of elements, diameter of array, radius of curvature, position of the center of the array, unit vector in the x-direction, and unit vector in the z-direction. Figure 4.5b in the results chapter depicts the form that will prompt the user to change their model specifications.

3.4.3 Model interaction and ablation region selection

Another important feature of including a GUI for the program was achieving operator interactivity with the rendered OpenGL scene. This was especially important when selecting the ablation region in the RF volumetric model. The physician will desire to steer the beam emitted by their HIFU transducer mechanically or manually to properly target the tumor or diseased area. This region will be visualized by the composition of ultrasound images rendered in the 3D scene, and the focal point can be steered during the surgical procedure to locate and treat the desired tissue area. For this capability, spin box widgets were added to the GUI that are located next to the OpenGL widget. The focal point was set upon initialization of the environment as default, but changing the

values for the x, y, and z-positions of the focal point updated the model. The y and z-direction corresponded to cross-section and depth within the subject whereas the x-direction changes affected the focus in the plane perpendicular to the rendered B-mode slices. The display of this form can be seen in Figure 4.6 in the results chapter. The precision of the focal point adjustments can be refined as necessary, along with the accuracy. For development and visualization purposes the movement according to physical space was arbitrary.

Chapter 4

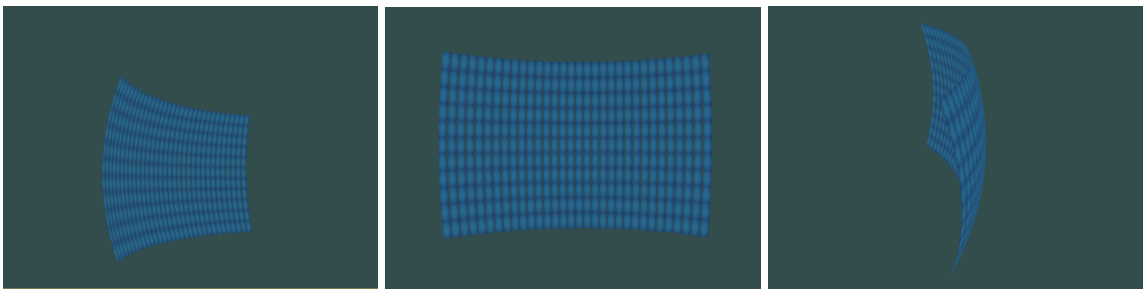
Results

4.1 Visualization results

Much of the programming work done for this project consisted of achieving appropriate 3D renderings for HIFU tissue ablation scenes. Usability for the physician as well as computational cost and speed were considered. As the application was being developed, the most pivotal aspect was to produce an interface to assist with the procedure of HIFU tissue ablation while integrating the noninvasive technology of ultrasound B-mode imaging.

The results of the renderings yielded a 3D transducer model for the HIFU transducer, from which the focused ultrasound beam is emitted. The transducer rendering was successfully converted into the C++ and OpenGL framework. The focal point was visualized with approximated physical positions in the code for developmental purposes. Finally, the B-mode ultrasound radio frequency (RF) data was imaged as 2D planar slices and visualized in 3D space according to the linear reconstruction of a phased array ultrasound capture. The volume was visualized as a rectangular prism to depict the 3D space of the tissue imaged by the ultrasound data and subjected to targeting by the HIFU beam. Thus, the physician can see the volumetric region that is available for ablation targeting. Each of the achieved OpenGL renderings will be shown and discussed in the following sections.

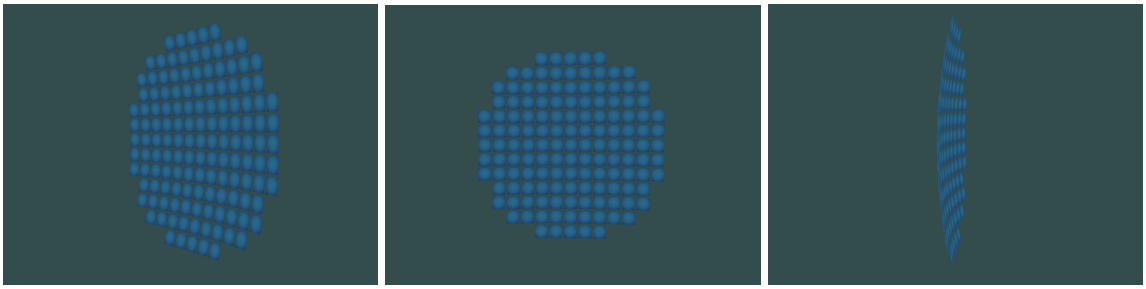
4.1.1 3D transducer rendering results



(a) (b) (c)

Figure 4.1: Image (a) shows a side angle of the rendered cylindrical panel in an OpenGL scene. (b) shows a frontal view, and (c) shows a view from the upper right corner of the transducer array.

The images above in Figure 4.1 depict the renderings of a HIFU transducer that was drawn using OpenGL. There are 11 rows of 32 piezoelectric elements for this model, as easily seen in Figure 4.1b. This demonstrates the conversion from MATLAB to C++ for the transducer model and provides the operator with a 3D view of the transducer. The transducer can be moved around in the 3D scene to attain different viewpoints.

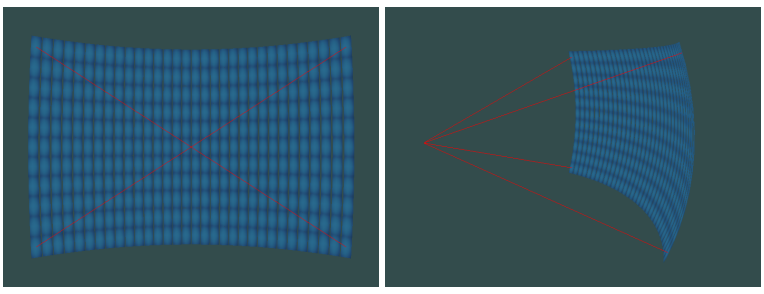


(a) (b) (c)

Figure 4.2: Image (a) shows a side angle of the rendered spherical panel in an OpenGL scene. (b) shows a frontal view, and (c) shows a view from the far left side of the transducer array.

The images in Figure 4.2 show the rendering of the spherical panel array. The array models a total of 137 piezoelectric elements. Movement around the scene can be achieved in the same fashion as the cylindrical panel. This shows how other transducer models could be added in the future to this framework.

4.1.2 3D focal point rendering results



(a) (b)

Figure 4.3: Image (a) depicts a frontal view of the focal point rendering. In (b), a view from the right side of the transducer shows the focal point projection mark a location in 3D space.

The purpose of creating the rendered focal point lines was to indicate the ablation site of the HIFU transducer for the physician. The focal point can be adjusted and can later be adapted to relate to the physical constraints of the actual HIFU transducer being used for an ultrasound procedure, and this will help model methods for beam steering and accurate tissue heating. This model represents HIFU targeting, and so one is able to aim the ultrasound waves at a region of diseased tissue in the 3D scene.

4.1.3 3D RF data rendering results

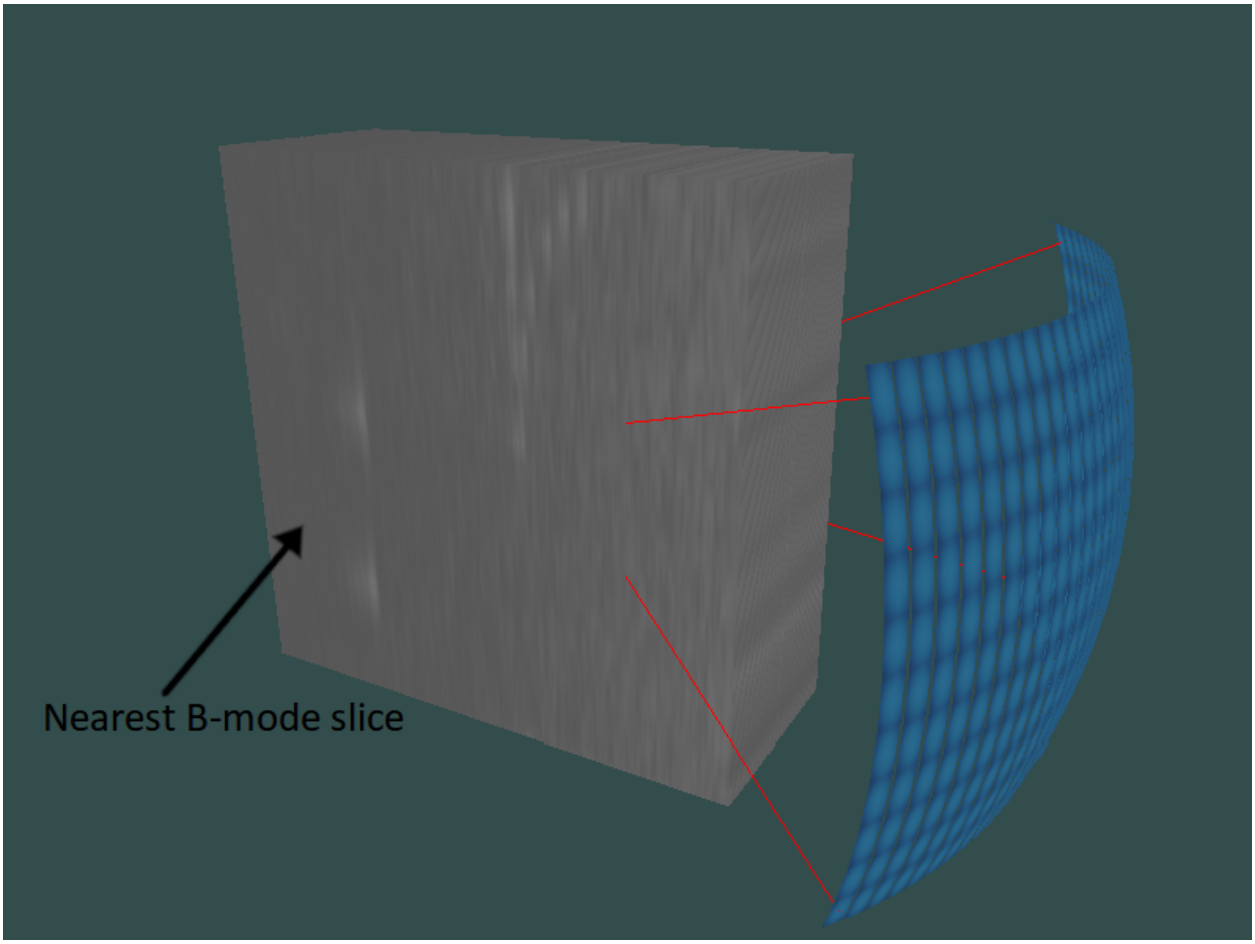


Figure 4.4: 3D model of 80 RF slices rendered in the HIFU ablation scene. The slices are repeated B-mode images in this case, for developmental simplicity, but can be adapted to fit the model.

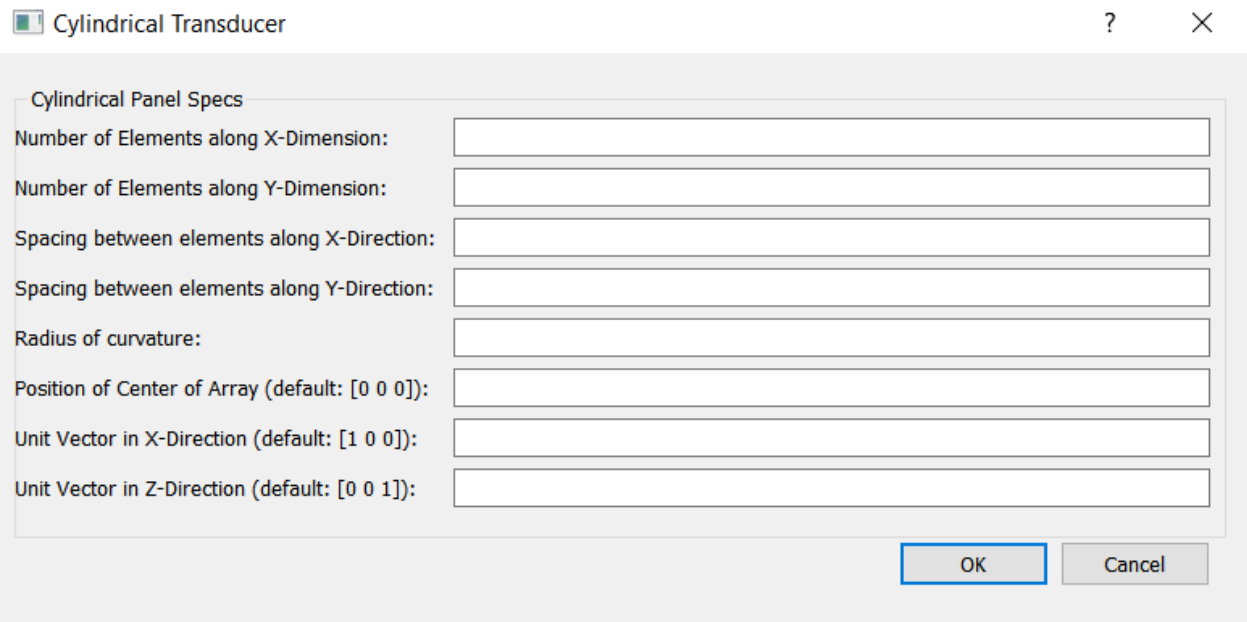
As described in the methods chapter, slices of 2D B-mode ultrasound images were amalgamated to create an implicit 3D ultrasound imaged volume. The nearest image plane in Figure 4.4 depicts the one-to-one mapping of ultrasound data to a rectangular plane. As a transducer is moved linearly in the normal direction of this plane, more ultrasound images will be captured to construct this volume. If all of these planes are captured in one instance by a 3D probe, then real-time imaging could be implemented using this model by updating every slice simultaneously. The focal point in Figure 4.4 is located within the volume of rendered tissue, and the tissue imaged at that position is to be ablated. Although the exact focal point is not seen in Figure 4.4 inside the volume, the volume can be internally traversed by the OpenGL camera, and that point can be recognized. However, implications of the 3D volume are difficult to understand once the camera of the scene enters the volume, which is a limitation of this model and could be improved upon in the future. The view could dynamically prompt updates for the visibility of slices to only display the ones afflicted by the heating process of the HIFU ablation. The texture mapping result that implemented bilinear interpolation can be observed in Figure 4.4 as well. The RF data was saved as an image and then mapped onto two triangles that formed a rectangular slice. Figure 4.4 and a description of the results presents the outcome of the work discussed in the methods chapter.

This representation of slice projection differs from the methods described in the background research. Slice projection often shows separate images of different perpendicular orientations to give the physician multiple angles and allow them to interpret its 3D meaning [30][34]. The RF data in Figure 4.4 reveals anatomy information perpendicular to, in this case, the x-axis according to the set world coordinates. The other two axes represent implicit data that is understandable from the closeness of each slice. However, aliasing, visual artifacts due to the lack of spatial resolution, occurs along these axes, as seen on the top and right of the rectangular prism volume shown in Figure 4.4. Thus, understanding details in this dimension of information is difficult, but such a design strategy was pursued in order to accelerate the time required to render the data and to image the depth of each 2D plane in 3D space. A solution to the aliasing would be to interpolate the color data for the gaps along each edge of every slice. This would necessitate a higher computational

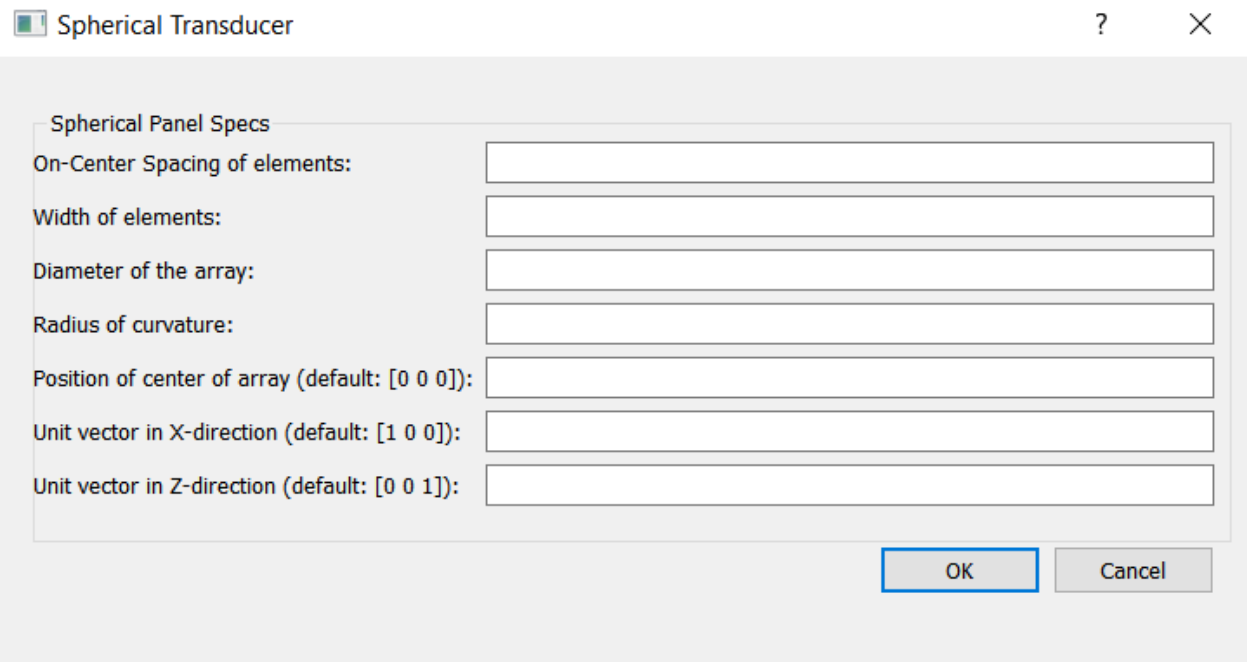
cost and would eliminate the details from the depth of each image, such as those seen in the nearest slice in Figure 4.4. A gap-filling technique, as is used for PBMs [30][34], could be adapted for the rendering algorithms created for this thesis work. The VBM technique in this work was extended by coloring two faces of every voxel and implying the color of the entire voxel from the close proximity of each slice.

4.2 Usability of graphical user interface

Qt was integrated as the toolkit to update the 3D HIFU ablation scene. Specifically, a graphical user interface (GUI) was designed to update the parameters of the transducer array models, and widgets were added that allow for the adjustment of the focal point of the HIFU beam. This GUI is important since it gives any physician using this tool the capabilities to easily interact with their model and provides them with the ability to customize their equipment constraints for the HIFU tissue ablation procedure.



(a)



(b)

Figure 4.5: Dialog form (a) is for updating the specifications of the cylindrical transducer model, where (b) is for the spherical transducer.

The dialog windows prompt the operator for specifications regarding their HIFU transducer. Changing the parameters of the model will update the 3D transducer rendering from its default settings and change the OpenGL model. This procedure is the same for cylindrical (Figure 4.5a)

and spherical (Figure 4.5b) transducer arrays per this project work.

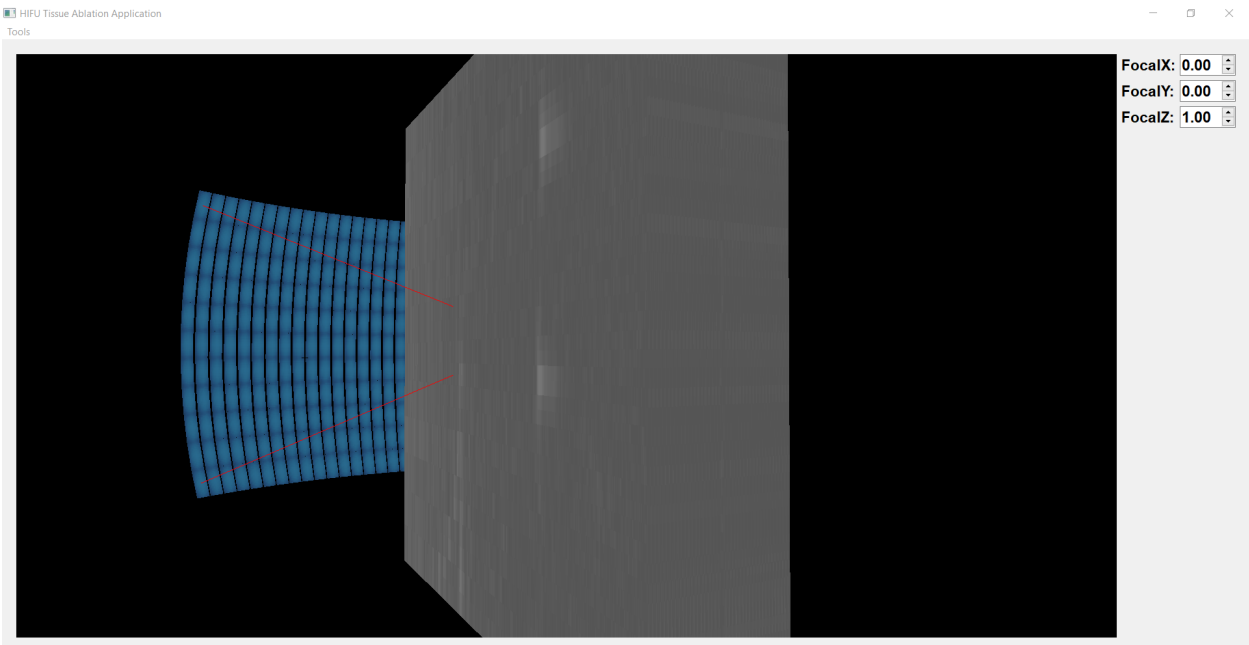


Figure 4.6: HIFU tissue ablation application.

Figure 4.6 shows the GUI in its current state for the application developed for interaction with the HIFU tissue ablation model. The number fields on the right side of the window allow for the adjustment of the focal point or beam steering. Although this Qt GUI shows minimal input fields, adding new widgets would not be challenging, and the version could be adapted for the future development of this project. The final rendering example (Figure 4.7) shows an object imaged by the B-mode scans that is targeted by the HIFU beam. This was done by adjusting the focal point parameters and specifying which slice to image in the code for increased visualization. In the future, the slices could be extracted by the physician as part of the GUI's capabilities.

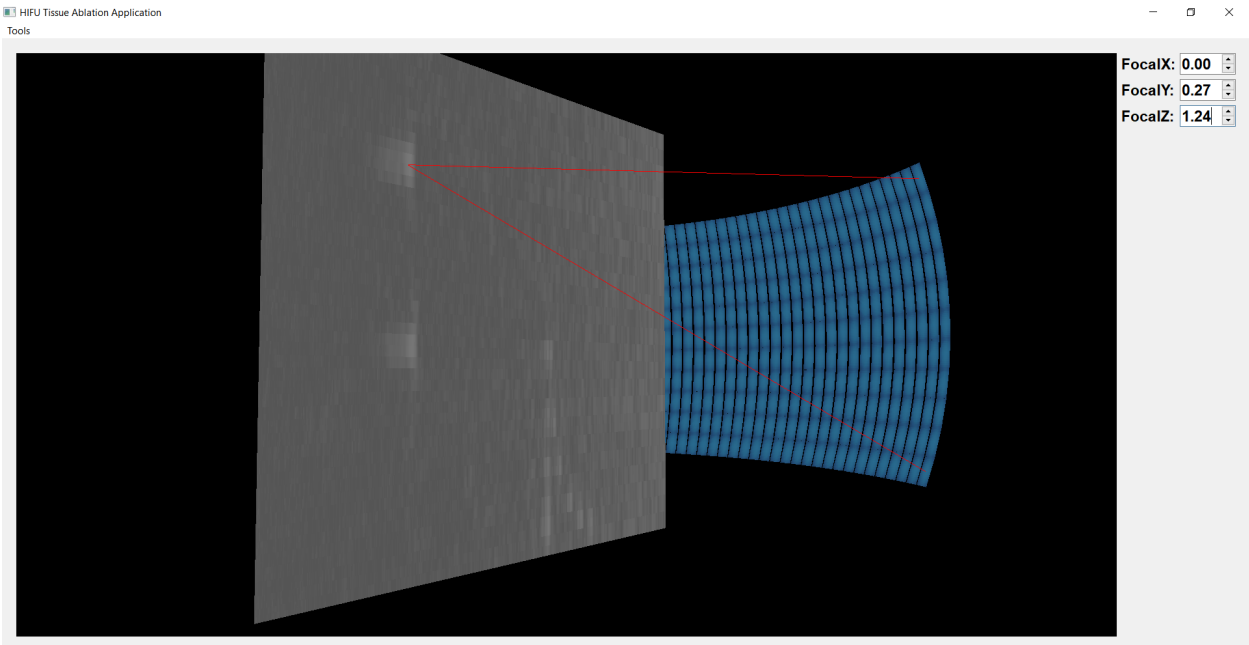


Figure 4.7: A specific B-mode slice was extracted, and an object in the image was targeted by adjusting the HIFU focal point.

4.3 Memory and processing performance

A major challenge of the work was executing the computations and performing the renderings at a sufficient speed for visualization. All the optimizations for this thesis were performed without the inclusion of parallel processing due to time restraints, but this technique is discussed as an item to be achieved in the future work of this project. Thus, design strategies were changed in order to reach the best processing time. A major area of improvement that will be discussed in this section is the change of the texture mapping method that was used for the 3D rendering program. The first attempt at rendering the RF data involved storing each vertex's color information in an array and bilinearly interpolating the color of each voxel side using OpenGL's smoothing feature when indicating the color mapping with GLSL. Although this technique worked to image the desired result, storing the vertex color information for the thousands of data points in a B-mode image used a large amount of processor memory. When more image slices were added to construct the 3D volume, as mentioned in the methods chapter, the rendering algorithm used too much memory to image the number of image slices that would be needed in the future. The real-time manipulation

of the 3D model scene was very slow in this instance. The current version of the program involves saving the RF data as an image, using the stb image processing library, and then mapping the image to two triangles per slice in the OpenGL environment. This change greatly saved buffer and computational memory for the vertex creation algorithm. Thus, the speed of interacting with the 3D models was improved. Table 4.1 below displays the use of processor memory as the number of image slices increased for the individual vertex color mapping method (Version 1) and the updated image saving algorithm (Version 2).

Table 4.1: Memory usage of two B-mode OpenGL imaging approaches (including the rendered cylindrical transducer and focal point).

	Version 1: Storing color value for each vertex	**Version 2: Saving RF data as an image, then mapping to plane**
# of Slices:	Processing Memory (MB):	Processing Memory (MB):
5	255.2	33.3
10	481.2	39.0
20	934.7	46.5
40	$1.8GB$	64.2

As seen in Table 4.1 above, as more B-mode images were rendered and reconstructed into the 3D volume, the amount of processing memory increased for each version of the algorithm. However, for Version 1 of the program, a substantial amount of memory was used. Consequently, the update to Version 2 of the algorithm was kept as the current method due to its improved memory usage and faster interaction frame-rate.

As for the computational speed of the program, a large amount of time was spent allocating memory for and processing the RF slice data. This brings up the necessity for GPU parallel processing as a means of achieving real-time imaging on standard PC hardware in the future. The means by which this goal could be accomplished is discussed in the future work chapter.

Chapter 5

Future Work

5.1 Overview

To conclude, the work of this thesis has effectively demonstrated the ability to model ultrasound images as a 3D volume using accelerated processing and visualizing an ablation point of a HIFU beam for a physician. Providing such visualization and a user-interface gives the physician the exactness needed to target a diseased region inside the body from B-mode ultrasound images. However, a primary functionality that is necessary for the work to enter real practice is real-time imaging and simulation of the HIFU beam being transmitted through the medium. From the research conducted in this thesis and the framework that has been built for this work, the current model would serve well for the implementation of GPU parallel processing; this would make the computational weight feasible for achieving the goals mentioned. The sections in this chapter will not only outline the future goals of this work but also suggest guidelines for incorporating them into the developed model.

5.2 Implementing parallel processing via CUDA

The benefits of using NVIDIA's CUDA for GPU computational support have already been realized and discussed in the background chapter of this document. The gathered research has supported the notion that GPU-powered ultrasound imaging produces faster results and uses memory and processing for standard personal computers in a better manner than CPU core parallelization. This section will delve into methods for incorporating CUDA with the designed framework and in what ways the development would make the achieved 3D imaging update in real-time. As conveyed in the results section, the computational cost of construction (or reconstruction) of the transducer array model takes valuable time, and rendering more slices of B-mode images to assemble the cubic volume of the patient's tissue is very computationally expensive. Although storing the RF data as an image and mapping it to 2D planes conserved processing memory, this technique did not increase the runtime of the software. Thus, CUDA's parallel processing could support

the computational load associated with the graphics rendering that relies on a great deal of data handling.

CUDA code can be hosted on a CPU run program but executed by the GPU. The keyword _global_ informs NVIDIA's compiler, nvcc, that the method defined after the keyword is to be executed on the GPU [41]. Thus, any local- or programmer-allocated pointers that occur in a CUDA defined function exist on the GPU's memory or on the device [41]. This memory is separate from CPU memory which provides greater flexibility by making more memory available for the software that is being run. CUDA's API is adapted from C's API, and pointer memory can be allocated and freed using the following functions: cudaMalloc(), cudaFree(), and cudaMemcpy() [41]. This type of memory management could be utilized in the context of storing the computed transducer vertex data and the RF data that is read and converted into image format. Currently, this memory is stored as an object on the program's stack, which consumes a great deal of the CPU's memory and results in significantly slower processing when many B-mode images are reconstructed. Parallelism can be prompted upon calling a device method for execution. For example, consider a function that normalizes the RF data, normalize_RFdata. Using the mentioned constructs, the code in Figure 5.1 demonstrates a simple operation done within a GPU executed function. Figure 5.1 is for example purposes only and does not contain the actual code used to normalize the RF data.

```
_global_ void normalize_RFdata(float *rfData, float mean, float std) {
      rfData[blockIdx.x] = (rfData[blockIdx.x] – mean) / std;
}
```

Figure 5.1: Example CUDA C code that would normalize RF data with parallel blocks.

There are two main parallelization strategies that can be implemented via CUDA's API. Figure 5.1 is an example of using blocks to execute the normalize_RFdata function in parallel. The block's index is referenced by the variable blockIdx.x, and each invocation of normalize_RFdata is conducted by a single block [41]. If normalize_RFdata is invoked with the following command:

normalize_RFdata<<< bN, 1>>>(rfData, mean, std), then bN blocks execute the function on the GPU in parallel and index the rfData point with their corresponding index. The other means of parallelization utilizes threads. A single block can execute parallel threads [41]. The following command: normalize_RFdata<<< 1, tN>>>(rfData, mean, std) executed for Figure 5.2's code will have the same effect as the other function call on Figure 5.1's code. In this case, a single block will execute the function in Figure 5.2 using tN parallel threads.

```
_global_ void normalize_RFdata(float *rfData, float mean, float std) {
       rfData[threadIdx.x] = (rfData[threadIdx.x] - mean) / std;
}
```

Figure 5.2: Example CUDA C code that would normalize RF data with parallel threads.

5.3 Ultrasound wave propagation and visualization

Another important element of the future work goals that could be incorporated with the developed program is visualizing the HIFU beam through the medium and recognizing the effect of heating the tissue. This process would ultimately demonstrate the effect of HIFU therapy on the patient in real-time and represent the volume of the region being heated. In order to understand the propagation of the ultrasound wave through the tissue, the wave equation must be computationally approximated, and in this document, the problem is approached using the finite difference method. Once the wave propagation is realized, the beam could be visualized as being emitted from the transducer model and targeting the area of interest. This work should be achieved alongside parallelization with the GPU in order to efficiently distribute the computational load.

5.3.1 Solving the wave equation with the finite difference method

In the background chapter, the reasons for the occurrence of nonlinear propagation of an acoustic wave and why these would be the challenges associated with simulation were discussed. A

method for effectively approximating a function for wave propagation through changing and consistent mediums will be proposed, thus furthering the extent to which the 3D model produced could benefit therapeutic research and practice.

Linear model

Before delving into an approach for solving the nonlinear wave propagation model, the finite difference method for approximating a linear HIFU model will be reviewed. The acoustic wave equation for 1D propagation is shown in equation 5.1 below [16]. The formula represents the change of pressure in terms of its change with distance [16]. Equation 5.2 is the wave equation in 3D [16]. In these equations, ΔP is the change in local pressure in the medium and the speed of the sound wave is c [16]. The value of c can be computed by equation 5.3, where k is the bulk modulus of the tissue, describing a substance's behavior under uniform compression, and ρ is the density [16].

$$\frac{\partial^2 \Delta P}{\partial t^2} = c^2 \frac{\partial^2 \Delta P}{\partial x^2} \tag{5.1}$$

$$\frac{\partial^2 \Delta P}{\partial t^2} = c^2 (\frac{\partial^2 \Delta P}{\partial x^2} + \frac{\partial^2 \Delta P}{\partial y^2} + \frac{\partial^2 \Delta P}{\partial z^2}) \tag{5.2}$$

$$c = \sqrt{\frac{k}{\rho}} \tag{5.3}$$

Almekkawy et al. used the wave equation to model the acoustic propagation from a dual-mode ultrasound array to effectively simulate HIFU ablation of atherosclerotic plaque [42]. Their model utilized the Euler and continuity equations, an adaptation of the wave equation, shown in equation 5.4 [42]. In the equations, $\hat{u}$ is the particle velocity, and p is the pressure [42]. The equation 5.5 for the complex wave number k_n shows the relationship between the absorption coefficient γ and the wave attenuation coefficient σ, where c' is the dispersive wave velocity.

$$\begin{aligned} \frac{\partial \vec{u}}{\partial t} &= \frac{-1}{\rho}\nabla p \\ \frac{\partial p}{\partial t} &= -\rho c^2 \nabla . \vec{u} - c^2 p \end{aligned} \tag{5.4}$$

$$k_n = \sqrt{\frac{\omega^2}{c^2} + i\omega\gamma} = \frac{\omega}{c'} + i\sigma \tag{5.5}$$

Almekkawy et al. used a stretched coordinate to modify 5.4. This is necessary to make the function able to solve the partial differential equation shown in equation 5.6, which is the coupling of 5.4 [42]. The stretching parameter g is depicted in 5.7 and the gradient of g is represented in equation 5.8 [42].

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial^2 t} = \rho\nabla.\rho^{-1}\nabla p - \gamma\frac{\partial p}{\partial t} \tag{5.6}$$

$$g_\xi = s_\xi + i\frac{(h\sigma)_\xi}{\omega}(\xi = x, y, z) \tag{5.7}$$

$$\nabla g = \sum_{(\xi=x,y,z)} \hat{\xi}\frac{1}{g\xi}\frac{\partial}{\partial\xi} \tag{5.8}$$

In 5.7, ω is the angular frequency, s_ξ is a scaling factor, and $h = pc^2$ where $(h\sigma)_\xi$ is the loss of the perfect match layer, which allows for the extension of the solution of the finite difference model past its natural boundaries to enhance the recognition of the boundary conditions for acoustic wave propagation [42]. Equation 5.4 is modified again by replacing ∇ with ∇g and applying the stretched coordinates [42]:

$$\begin{aligned} s_\xi\frac{\partial}{\partial t}\hat{u}_\xi + (h\sigma)_\xi\hat{u}_\xi &= -\frac{1}{\rho}\nabla p \\ s_\xi\frac{\partial}{\partial t}p &= -h\nabla\cdot\hat{u} - (s_\xi\gamma c^2 + (h\sigma)_\xi)p \end{aligned} \tag{5.9}$$

When $s_\xi = 1$ and $(h\sigma)_\xi = 0$, for the interior region of the partial derivative boundaries, equation 5.9 is used for the entire computational domain [42]. After using the finite difference time domain method, the linear acoustic model can be solved using the following approximation [42]:

$$
\begin{aligned}
&s_\xi[\hat{u}_x(j_x, j_y, j_z, n) - \hat{u}_x(j_x, j_y, j_z, n-1)]/\Delta t \\
&+ (h\sigma)_\xi[\hat{u}_x(j_x, j_y, j_z, n) + \hat{u}_x(j_x, j_y, j_z, n-1)]/2 \\
&= -\frac{1}{\rho \Delta \xi}[p(j_x, j_y, j_z, n) - p(j_x - 1, j_y, j_z, n)]
\end{aligned}
\tag{5.10}
$$

The stepping equation for $\hat{u}$ is

$$
\begin{aligned}
&\hat{u}_x(j_x, j_y, j_z, n) \\
&= A\hat{u}_x(j_x, j_y, j_z, n-1) \\
&+ B[p(j_x, j_y, j_z, n) - p(j_x - 1, j_y, j_z, n)]
\end{aligned}
\tag{5.11}
$$

and the stepping equation for p is

$$
\begin{aligned}
&p(j_x, j_y, j_z, n+1) = Cp(j_x, j_y, j_z, n) \\
&+ D[\hat{u}_x(j_x + 1, j_y, j_z, n) - \hat{u}_x(j_x, j_y, j_z, n)]
\end{aligned}
\tag{5.12}
$$

For these equations, A, B, C, and D are defined below.

$$
\begin{aligned}
A &= \frac{s_\xi/\Delta t - (h\sigma)_\xi/2}{s_\xi/\Delta t + (h\sigma)_\xi/2} \\
B &= \frac{1}{\Delta\xi(\frac{s_\xi \rho}{\Delta t} + \frac{(h\sigma)_\xi \rho}{2})} \\
C &= \frac{s_\xi/\Delta t - (s_\xi \gamma c^2 + (h\sigma)_\xi)/2}{s_\xi/\Delta t + (s_\xi \gamma c^2 + (h\sigma)_\xi)/2} \\
D &= -\frac{h}{\Delta\xi[s_\xi/\Delta t + (s_\xi \gamma c^2 + (h\sigma)_\xi)/2]}
\end{aligned}
\tag{5.13}
$$

Thus, the wave velocity and pressure can be approximated throughout the acoustic wave propagation under the assumption of a linear model.

Nonlinear model

As shown by the wave equations, 5.1 and 5.2, and equation 5.3, the speed of sound will not be constant throughout the propagation of the acoustic wave through a body. The bulk modulus is a function of Δp, and Δp is a function of x, y, z, and t [16]. When the speed of sound depends on

frequency or pressure, nonlinearity is introduced, and the speed of sound is replaced by the phase velocity shown below.

$$v_p = \frac{\lambda}{T} \tag{5.14}$$

Extended from the 3D wave equation from equation 5.2, the nonlinear wave equation is described in equation 5.15, where speed depends on the amplitude, or intensity, of the wave, as covered in the background section about nonlinearity [16].

$$\frac{\partial^2 \Delta P}{\partial t^2} = c(P)^2 (\frac{\partial^2 \Delta P}{\partial x^2} + \frac{\partial^2 \Delta P}{\partial y^2} + \frac{\partial^2 \Delta P}{\partial z^2}) \tag{5.15}$$

Gianmarco Pinton, whose dissertation and research has explored numerical tactics for simulating ultrasound waves for nonlinear propagation, provides methods for solving the nonlinear wave equation in order to understand the speed and behavior of wave phases in a medium [43]. The finite difference method of solving partial differential equations will be reviewed in this section as a means of approximating wave propagation. Where equation 5.15 generalizes the wave equation for diagnostic ultrasound, Pinton models therapeutic ultrasound wave propagation and the effects of quadratic nonlinearity with the Khoklov-Zabolotskaya-Kuznetsov (KZK) equation [43]. Such a model can be fitted to the simulation of HIFU waves. The nonlinear parabolic KZK equation presents a function for the effects of diffraction, absorption, and nonlinearity and is represented below in equation 5.16 [43]. The variable $t' = t - z/c_0$ and represents delayed time, and z is the propagation direction [43]. The first term on the right of the equal sign is diffraction, c_0 being the small signal speed of sound [43]. The second term represents thermoviscous attenuation, and δ is diffusivity [43]. The third term represents nonlinearity, where $\beta = 1 + B/2A$ is the coefficient of nonlinearity, and ρ_0 is the ambient fluid density [43]. B/A corresponds to the magnitude of the greatest order correction to the small signal speed of sound [43]. Depicting the last portion of equation 5.16, equation 5.17 presents the transverse Laplacian in cartesian coordinates [43].

$$\frac{\partial^2 p}{\partial z \partial t'} = \frac{c_0}{2} \nabla_\perp^2 p + \frac{\delta}{2c_0^3} \frac{\partial^3 p}{\partial t'^3} + \frac{\beta}{2\rho_0 c_0^3} \frac{\partial^2 p^2}{\partial t'^2} \tag{5.16}$$

$$\nabla_{\perp}^{2} p = (\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}) \tag{5.17}$$

Having presented a model, the KZK wave equation, and a method for understanding nonlinear therapeutic ultrasound propagation through real tissue, the finite difference method will be described. This can be applied to the above model in the time-domain. The crux of the finite difference method is to replace any derivatives in the equation with the corresponding numerical differentiation formula [44]. If a solution is not found via this method, convergence algorithms must be implemented [44]. This same work applies to the linear model discussed as well. Looking back at the simplified 1D wave equation 5.1, the following steps will demonstrate the process of the finite difference method. A uniform 1D space mesh is obeyed for this example as shown below, where l is a homogeneous bar length.

$$t_i = i\Delta t, x_j = j\Delta x, where \Delta x = \frac{l}{n} \tag{5.18}$$

The finite difference approximations for the second order derivatives in the 1D wave equation 5.1 is shown below in formula 5.19 [44]. The final big-O notation term represents an error of the function that is proportional with $(\Delta t)^2$ and $(\Delta x)^2$ respectively.

$$\begin{aligned} \frac{\partial^2 \Delta P}{\partial t^2}(t_i, x_j) &\approx \frac{\Delta P(t_{i+1}, x_j) - 2\Delta P(t_i, x_j) + \Delta P(t_{i-1}, x_j)}{(\Delta t)^2} + O((\Delta t)^2) \\ \frac{\partial^2 \Delta P}{\partial x^2}(t_i, x_j) &\approx \frac{\Delta P(t_i, x_{j+1}) - 2\Delta P(t_i, x_j) + \Delta P(t_i, x_{j-1})}{(\Delta x)^2} + O((\Delta x)^2) \end{aligned} \tag{5.19}$$

Finally, substituting the finite difference formula 5.19 into the partial differential 1D wave equation 5.1 constructs an iterative equation 5.20 that can be computationally solved with an algorithm and provides an approximation of local pressure changes ΔP in a medium for this example for $i = 1, 2, \ldots,$ and $j = 1, \ldots, n-1$ [44].

$$\Delta P_{i+1,j} = (\frac{c\Delta t}{\Delta x})^2 \Delta P_{i,j+1} + 2(1 - (\frac{c\Delta t}{\Delta x})^2)\Delta P_{i,j} + (\frac{c\Delta t}{\Delta x})^2 \Delta P_{i,j-1} - \Delta P_{i-1,j} \tag{5.20}$$

Solving the KZK equation for 3D HIFU propagation will require more sophisticated methods than this generalized example, and is out of the realm of this thesis work. However, the benefits

of efficiently integrating this task with the ablation model designed for this thesis will result in the 3D simulation of HIFU waves.

5.3.2 Discussion of 3D simulation of the HIFU wave

If linear and nonlinear HIFU wave propagation is modeled in the designed system, the result of heating the targeted region of tissue could be displayed using the OpenGL API as well as the appropriate physical responses of the beam of sound. Such a level of simulated responsiveness would greatly enhance the understanding and accuracy of tissue ablation procedures, which has not yet been achieved in this 3D fashion by using ultrasound images for real-time display. The computational models can be supported using parallel GPU algorithms with the CUDA toolkit. Thus, the real-time HIFU model can be visualized on a typical PC with NVIDIA graphics hardware, and simulating the wave by solving for speed and pressure by using the finite-difference method with respect to time for the 3D wave equation could accurately depict the behavior of HIFU wave propagation through the ultrasound rendered images.

Chapter 6

Conclusion

HIFU therapy offers substantial benefits to the medical field, and the research currently being conducted may allow it to become a primary method for cancer treatment. The procedure relies significantly on software engineering and the integration of programming technologies in order to successfully prepare for and complete the HIFU ablation procedure. The work in this thesis focused on creating a 3D visualization of a HIFU ablation procedure using only non-invasive technology. Instead of using magnetic resonance-guided HIFU (MR-HIFU), this thesis depended on ultrasound-guided HIFU (US-HIFU) to reduce any risk of side effects while working to maintain a high level of volume understanding. By reconstructing B-mode images as a 3D volume, standard ultrasound technology can be used along with the model developed in this thesis for simulating HIFU tissue ablation. Once real-time imaging and ultrasound wave modeling is integrated with this application, the program will allow physicians to simulate their HIFU procedure on a patient in real-time using only ultrasound technology. This technique is a unique approach for HIFU ablation simulation, especially being a method that can operate on a personal computer. As the program is further enhanced, dimensionality can be improved to represent measurements associated with physical space, thus increasing the accuracy and usability of the program. This thesis provides insight into the graphics tools and software being applied to improve 3D ultrasound imaging. The work of this undergraduate research, along with work in pursuit of combining software design strategies with ultrasound therapy, brings medicine closer to realizing an entirely non-invasive, safe, and increasingly reliable method for treating cancer and other diseases.

Bibliography

[1] Yu-Feng Zhou. High intensity focused ultrasound in clinical tumor ablation. *World journal of clinical oncology*, 2(1):8, 2011.

[2] Alexander Copelan, Jason Hartman, Monzer Chehab, and Aradhana M Venkatesan. High-intensity focused ultrasound: current status for image-guided therapy. In *Seminars in interventional radiology*, volume 32, pages 398–415. Thieme Medical Publishers, 2015.

[3] John G Lynn, Raymund L Zwemer, Arthur J Chick, and August E Miller. A new method for the generation and use of focused ultrasound in experimental biology. *The Journal of general physiology*, 26(2):179, 1942.

[4] Li Xiaoping and Zheng Leizhen. Advances of high intensity focused ultrasound (hifu) for pancreatic cancer. *International Journal of Hyperthermia*, 29(7):678–682, 2013.

[5] William J Fry, John W Barnard, Frank J Fry, and James F Brennan. Ultrasonically produced localized selective lesions in the central nervous system. *American Journal of Physical Medicine & Rehabilitation*, 34(3):413–423, 1955.

[6] Osama Al-Bataineh, Jürgen Jenne, and Peter Huber. Clinical and future applications of high intensity focused ultrasound in cancer. *Cancer Treatment Reviews*, 38(5):346–353, 2012.

[7] Feng Wu, Zhi-Biao Wang, Hui Zhu, Wen-Zhi Chen, Jian-Zhong Zou, Jin Bai, Ke-Quan Li, Cheng-Bing Jin, Fang-Lin Xie, and Hai-Bing Su. Extracorporeal high intensity focused ultrasound treatment for patients with breast cancer. *Breast cancer research and treatment*, 92(1):51–60, 2005.

[8] Ezekiel Maloney and Joo Ha Hwang. Emerging hifu applications in cancer therapy. *International Journal of Hyperthermia*, 31(3):302–309, 2015.

[9] Hyun Joo Jang, Jae-Young Lee, Don-Haeng Lee, Won-Hong Kim, and Joo Ha Hwang. Current and future clinical applications of high-intensity focused ultrasound (hifu) for pancreatic cancer. *Gut and Liver*, 4:S57–S61, Sep 2010.

[10] J.Y. Chapelon, J. Margonari, F. Vernier, F. Gorry, R Ecochard, and A. Gelet. In vivo effects of high-intensity ultrasound on prostatic adenocarcinoma dunning r3327. *American Association for Cancer Research*, 52:6353–6357, Nov 1992.

[11] Rémi Souchon, Olivier Rouvière, Albert Gelet, Valérie Detti, Seshadri Srinivasan, Jonathan Ophir, and Jean-Yves Chapelon. Visualisation of hifu lesions using elastography of the human prostate in vivo: preliminary results. *Ultrasound in Medicine & Biology*, 29(7):1007–1015, 2003.

[12] Ernst Martin, Beat Werner, Ronald Bauer, Karin van Leyen, Daniel Coluccia, and Javier Fandino. Clinical neurological hifu applications: the zurich experience. *Translational Cancer Research*, 3(5):449–458, 2014.

[13] Andrew Casper, Dalong Liu, John Ballard, and Emad Ebbini. Real-time implementation of a dual-mode ultrasound array system: In vivo results. *2012 IEEE International Ultrasonics Symposium*, 60(10):2751–2759, 2013.

[14] Maryam Mohammadi Monfared, Hamid Behnam, Parisa Rangraz, and Jahan Tavakkoli. High-intensity focused ultrasound thermal lesion detection using entropy imaging of ultrasound radio frequency signal time series. *Journal of medical ultrasound*, 26(1):24, 2018.

[15] John E. Aldrich. Basic physics of ultrasound imaging. *Critical Care Medicine*, 35(5):S131–S137, 2007.

[16] M Halliwell. A tutorial on ultrasonic physics and imaging techniques. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 224(2):127–142, 2009.

[17] Effects of nonlinear ultrasound propagation on high intensity brain therapy. *Medical Physics*, 38(3).

[18] Aladin Carovac, Fahrudin Smajlovic, and Dzelaludin Junuzovic. Application of ultrasound in medicine. *Acta informatica medica*, 19(3):168, 2011.

[19] Siddhartha Sikdar, Ravi Managuli, Tsuyoshi Mitake, Tetsuya Hayashi, and Yongmin Kim. Programmable ultrasound scan conversion on a media-processor-based system. In *Medical Imaging 2001: Visualization, Display, and Image-Guided Procedures*, volume 4319, pages 699–712. International Society for Optics and Photonics, 2001.

[20] W Steelman and R Jain. Comparison of real-time scan conversion methods with an opengl assisted method, 2010.

[21] A Berkhoff. Fast scan conversion algorithms for displaying ultrasound sector images. *Ultrasonic Imaging*, 16(2):87–108, 1994.

[22] William A. Steelman and William D. Richard. Performance of new gpu-based scan-conversion algorithm implemented using opengl. *Ultrasonic Imaging*, 33(2):143–150, 2011.

[23] Qi Tian and Michael N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35(2):220–233, 1986.

[24] The industry's foundation for high performance graphics. https://opengl.org/. OpenGL News.

[25] An opengl library. https://www.glfw.org/.

[26] The freeglut project. http://freeglut.sourceforge.net/.

[27] John M. Kessenich, Graham Sellers, and Dave Shreiner. *OpenGL programming guide the official guide to learning OpenGl, version 4.5 with SPIR-V*. Addision-Wesley, 2017.

[28] Ultrasound. https://www.nibib.nih.gov/science-education/science-topics/ultrasound. U.S. Department of Health and Human Services.

[29] Shyam Natarajan, Leonard S. Marks, Daniel J.a. Margolis, Jiaoti Huang, Maria Luz Macairan, Patricia Lieu, and Aaron Fenster. Clinical application of a 3d ultrasound-guided prostate biopsy system. *Urologic Oncology: Seminars and Original Investigations*, 29(3):334–342, 2011.

[30] Qinghua Huang and Zhaozheng Zeng. A review on real-time 3d ultrasound imaging technology. *BioMed research international*, 2017, 2017.

[31] Philippe Arbeille, V Eder, D Casset, L Quillet, C Hudelo, and S Herault. Real-time 3-d ultrasound acquisition and display for cardiac volume and ejection fraction evaluation. *Ultrasound in Medicine & Biology*, 26(2):201–208, 2000.

[32] C.d. Barry, C.p. Allott, N.w. John, P.m. Mellor, P.a. Arundel, D.s. Thomson, and J.c. Waterton. Three-dimensional freehand ultrasound: Image reconstruction and volume analysis. *Ultrasound in Medicine & Biology*, 23(8):1209–1224, 1997.

[33] Ole Vegard Solberg, Frank Lindseth, Hans Torp, Richard E. Blake, and Toril A. Nagelhus Hernes. Freehand 3d ultrasound reconstruction algorithms—a review. *Ultrasound in Medicine & Biology*, 33(7):991–1009, Jul 2007.

[34] Thomas R. Nelson and Dolores H. Pretorius. Three-dimensional ultrasound imaging. *Ultrasound in Medicine & Biology*, 24(9):1243–1270, 1998.

[35] Ryutarou Ohbuchi, David Chen, and Henry Fuchs. Incremental volume reconstruction and rendering for 3d ultrasound imaging. *Visualization in Biomedical Computing 92*, 1808:312–323, Sep 1992.

[36] Oliver Kutter, Athanasios Karamalis, Wolfgang Wein, and Nassir Navab. A gpu-based framework for simulation of medical ultrasound. In *Medical Imaging 2009: Visualization, Image-Guided Procedures, and Modeling*, volume 7261, page 726117. International Society for Optics and Photonics, 2009.

[37] Cuda zone. https://developer.nvidia.com/cuda-zone, Dec 2018.

[38] Jung Woo Choe, Amin Nikoozadeh, Ömer Oralkan, and Butrus T Khuri-Yakub. Gpu-based real-time volumetric ultrasound image reconstruction for a ring array. *IEEE transactions on medical imaging*, 32(7):1258–1264, 2013.

[39] Joe Stam. What every cuda programmer should know about opengl. https://www.nvidia.com/content/GTC/documents/1055_GTC09.pdf, Oct 2009. The Fairmont.

[40] Qt opengl. https://doc.qt.io/qt-5/qtopengl-index.html.

[41] Geoff Gerfin. Parallel programming and debugging with cuda c. https://www.nvidia.com/content/PDF/sc_2010/theater/Gerfin_SC10.pdf, 2010.

[42] Mohamed K Almekkaway, Islam A Shehata, and Emad S Ebbini. Anatomical-based model for simulation of hifu-induced lesions in atherosclerotic plaques. *International Journal of Hyperthermia*, 31(4):433–442, 2015.

[43] Gianmarco F Pinton. *Numerical methods for nonlinear wave propagation in ultrasound*. Duke University, 2007.

[44] Peter J Olver. Numerical analysis lecture notes. 18:2017, 2008.

Academic Vita

Harrison Fesel

Education

The Pennsylvania State University — **Graduation: May, 2019**
Bachelor of Science - Computer Science | Mathematics Minor — ***University Park, PA***
Schreyer Honors College

Professional Experience

Software Engineering Intern — **June 2018 – August 2018**
JPMorgan Chase & Co. — ***Newark, DE***

- Worked on back-end and front-end development for a web application
- Implemented a web framework to make the application design scalable and easy to update
- Cooperated with other developers in an Agile environment
- Acquired knowledge in corporate finance and operations

Infrared Engineering Intern — **June 2014 – August 2017**
II-VI Incorporated — ***Saxonburg, PA***

- Designed and built applications using C# and SQL server databases that archived production data and generated formal reports
- Created macros in VBA to prevent operator error
- Guided a production chain of CdTe modulators
- Performed laboratory experiments on optical units for development and production
- Strengthened the infrared engineering team's progress by accelerating prototyping and writing data reports

Projects

PSU Learning Factory Capstone Redesign (Senior Project) — **Fall 2018**

- Updated the code-base and functionality for the PSU Capstone judging system web app
- Worked with a business unit of other students to meet requirements and target dates of sponsor

OpenGL Ultrasound Image Rendering (Honors Thesis) — **Spring 2018 – Spring 2019**

- Developed a framework in C++ with OpenGL to render 3D ultrasound images for HIFU tissue ablation
- Conducted research in ultrasound imaging, computer graphics, and computer vision

Dynamic Memory Allocation (Operating Systems) — **Spring 2018**

- Built a memory allocator in C that utilized a segregated free list for finding memory
- Managed memory on the heap while optimizing availability with coalescing to avoid fragmentation

Java Artistic Design Tool (Object Oriented Programming) — **Fall 2016**

- Created an application individually in Java that provided a functionality similar to that of Microsoft Paint

Computer/Technical Skills

- Fluent in Java, C++, C#, and C
- Experienced in HTML, JS, VB, and Python
- AngularJS development
- SQL database programming
- Product and data management
- Algorithm development

Involvement

National Society of Leadership and Success — **September 2016 – April 2019**

- Received the National Engaged Leadership Award for contributing to campus events and volunteer services
- Volunteered with Habitat for Humanity and participated in food and home supplies drive in Erie, PA

Association of Computer Machinery (ACM) — **Fall 2017 – Spring 2018**

- Advanced software skills as a member of the Competitive Programming and DevPSU sectors of ACM

Hack PSU Club — **Fall 2017**

- Coordinated the Special Events Team by helping organize PSU's hackathon and community events