University of California
Los Angeles

Simulation, Stitching, and Interaction Techniques for Large-Scale Ultrasound Datasets

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Matthew Wang

2016

Abstract of the Dissertation

Simulation, Stitching, and Interaction Techniques for Large-Scale Ultrasound Datasets

by

Matthew Wang

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2016

Professor Demetri Terzopoulos, Chair

Current medical ultrasound datasets used in training simulators lack adequate coverage due to imaging hardware limitations. We present software solutions both for generating and for interacting with large-scale ultrasound datasets. The generation process combines a physical simulation with a ray-tracing rendering technique to create synthetic ultrasound volumes for use as ground-truth test data. These datasets allow us both to train users and to evaluate an automatic registration solution used to align multiple real ultrasound volumes. We merge the aligned results with a multiresolution functional-based convex optimization technique to achieve seamless blends between adjacent volumes. A content-aware embedding algorithm places the merged data into a clean background template. We enable end-users to interact with the final results through a real-time mannequin-based translational tracking system.

The dissertation of Matthew Wang is approved.

D. Stott Parker

Eric Savitsky

Luminita Vese

Alan Yuille

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2016

To my parents, for always encouraging me to live up to my full potential,
and my siblings, for setting the bar so high.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The majority of the work presented in this dissertation was funded by SonoSim, Inc. I would like to thank Dr. Eric Savitsky, CEO, for providing the support and inspiration that formed the basis of this research. I would also like to acknowledge fellow SonoSim colleagues and former UCLA Computer Graphics & Vision Laboratory members Dr. Gabriele Nataneli, CTO, and Dr. Kresimir Petrinec, for all their help along the way. I also worked closely with labmate Eduardo Poyart during his time at SonoSim to develop many of the ideas presented here.

Speaking of the lab, I would like to extend a huge thanks to my adviser, Professor Demetri Terzopoulos, without whom this work would certainly not have been possible. He has spent countless hours guiding my research path, editing my papers, and writing letters of recommendation. I would also like to thank all my other labmates, past and present, who allowed me to bounce ideas off them over the years. I would particularly like to acknowledge the help and support I received from Sharath Kumar, Dr. Greg Klar, Masaki Nakada, and Tomer Weiss.

The simulation component of my work would not have been possible without the help of Dr. Chenfanfu Jiang, and the impressive work by the lab of Professor Joseph Teran. Several of his students, namely Dr. Daniel Ram, Dr. Ted Gast, Dr. Alexey Stomakhin, and Andre Pradhana, demonstrated significant patience over the years helping with this aspect of my research.

I would also like to acknowledge contributions from the lab of Professor Pirouz Kavehpour; in particular, Kelly Connelly for performing real-world experiments on ultrasound gel, as well as lab members Dr. Gaby Bran and Dr. Hamarz Aryafar for helping relieve my ignorance of fluid dynamics.

During my time at UCLA, I made many other friends who provided both academic and emotional support. Sarah Promnitz, Dr. Sai Deep Tetali, Jacob Mathew, Dr. Akshay Wadia, and all the others already mentioned could be counted on to provide enjoyable respites from the daily routine of PhD life. I relied heavily on close friend Dr. Becca Adams to work with

me during the final months of the preparation of this dissertation.

Finally I would like to thank my committee members, Professor Alan Yuille, Professor Luminita Vese, Professor Stott Parker, and of course Dr. Eric Savitsky and Professor Demetri Terzopoulos already mentioned, for their incredible support, feedback, and unimaginable collective intellect.

Vita

2005	B.S. (Computer Science), Stanford University, Stanford, CA.
2005	Technical Director Intern, Pixar Animation Studios, Emeryville, CA. Developed rendering optimization solutions for the feature film *Ratatouille*.
2006–2010	Technical Director, Dreamworks Animation, Glendale, CA. Developed and supported software for the feature films *Kung Fu Panda*, *Monsters Vs Aliens*, and *Megamind*.
2011	Teaching Assistant, UCLA, Los Angeles, CA. Led sections, held office hours, proctored exams, and graded assignments for CS143, Database Systems.
2011–2012	Software Engineer Internships, Google, Santa Monica / Mountain View, CA. Developed novel vision systems for object detection and autonomous driving.
2012	M.S. (Computer Science), UCLA, Los Angeles, CA.
2012–present	Software Engineer, SonoSim, Inc., Santa Monica, CA. Developed a suite of tools for volumetric data processing as it relates to ultrasound medical simulation and training.

CHAPTER 1

Introduction

1.1 Motivation

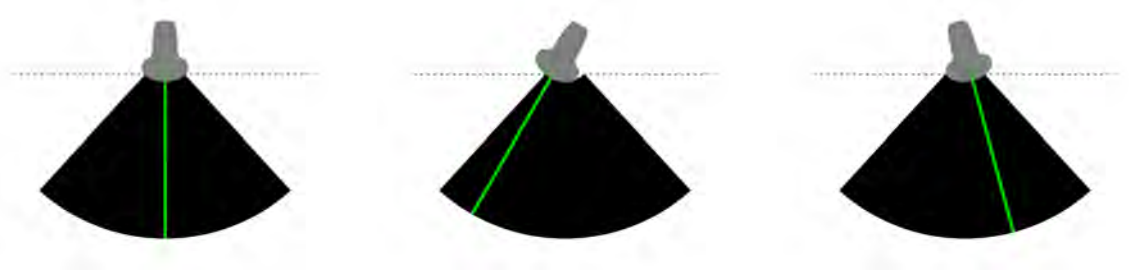
Over the past few years, ultrasound technology has become an increasingly popular tool in medical diagnosis. While commonly associated with imaging fetus development during pregnancy, this technology can be applied to any anatomical structure, including organs, bones, and even the eyes. As adoption of ultrasound diagnosis expands, training becomes an increasingly critical component. Familiarity with the appearance of different anatomical features and pathologies requires extensive experience, which has traditionally required hours of clinical exposure. Several companies, however, have sought to facilitate the training process through virtual simulators. Such solutions consist of presenting the user with a variety of datasets, either synthetic (computer generated), or acquired (real patient data). While synthetic datasets are not subject to the limitations of the image acquisition methods, they often lack the realism necessary for proper training. Acquired datasets offer true-to-life imagery, but can be limited in scope by the hardware involved.

The SonoSim®Ultrasound Training Solution (Figure 1.1) provides a prime example of using real patient data in a training simulator (Savitsky, 2013). 3D ultrasound datasets are acquired by expert sonographers and embedded within a virtual patient. To view this data, the user controls a hand-held mock probe containing an embedded inertial measurement unit (IMU). As the user rotates the probe, the simulator tracks the movement and displays the appropriate 2D slice through the 3D dataset. Acquiring, assembling, and interacting with large-scale datasets of this nature poses a number of challenges for the SonoSim product. Our research focuses on solving these issues.

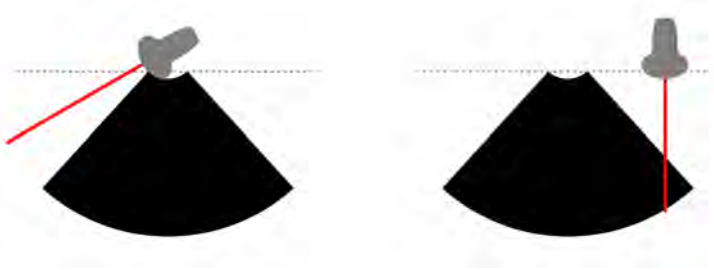
Figure 1.1: The SonoSim®Ultrasound Training Solution.

1.2 Problem Statement

One of the main drawbacks of acquired datasets is their limitation in physical size and angular coverage. In a clinical setting, where a physician or sonographer typically views a 2D slice of the body, the user has complete freedom over the placement of the probe, sliding it along the patient's body, or rotating it to obtain a better view. Some of this freedom can be captured in a dataset by preserving a full 3D volume of data, generated from a 3D probe, or reconstructed from a series of slices. The 3D data acquisition process, however, requires that the probe position and orientation remain static in order to produce an accurate dataset. Furthermore, the probe hardware is limited in the angular coverage it can acquire. Consequently, a single 3D volume cannot support translational movement of the simulated probe, and it limits the range of meaningful rotation (Figure 1.2). The necessary solution then is to acquire multiple datasets and merge them into a single dataset with greater coverage (Figure 1.3). However, several key problems arise:



(a) Original scan orientation (b) Simulated scan to the left (c) Simulated scan to the right



(d) Scanning outside the rotational extent of the data (e) Scanning outside the translational extent of the data

Figure 1.2: 2D illustration of the data coverage problem. The black region represents the acquired dataset. The green lines represent valid scan configurations during simulation. The red lines represent invalid scan configurations during simulation. In 3D, the black region would be a volume, and the colored lines would be 2D image slices.

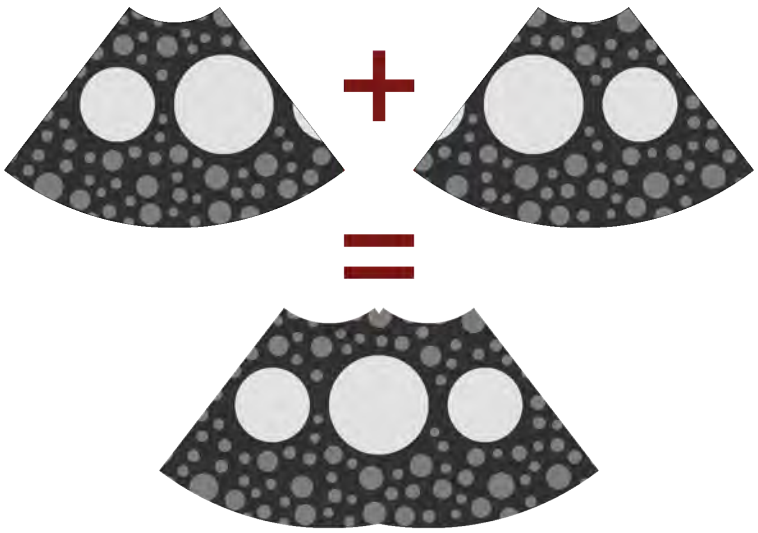


Figure 1.3: Illustration of combining multiple ultrasound datasets to achieve greater coverage.

1. The datasets are noisy, and therefore difficult to align.

2. When the probe movement involves translation, the data is distorted by non-uniform physical deformations as the probe pushes against the patient's skin, again hindering alignment.

3. Ultrasound artifacts such as shadows are position-dependent, and therefore differ between datasets, presenting both alignment challenges as well as visual inconsistencies.

The problem then is to develop an automatic or semi-automatic solution to align and stitch together adjacent datasets in light of these challenges. Furthermore, since the increased coverage of the resulting datasets affords greater scanning freedom within the simulator, we need to develop an improved interaction solution that supports translation.

1.3 Our Solution

To tackle this problem, we propose a semi-automatic alignment process using synthetic datasets with perfect ground truth for evaluation. The availability of synthetic datasets is critical, as it eliminates the need for manually aligned data, which is both unreliable and difficult to obtain. Synthetic datasets provide perfect alignment knowledge even in scenarios where the amount of noise or other image artifacts are too great to allow meaningful interpretation. Thus, we begin our solution by simulating the ultrasound acquisition and imaging process. Next, we consider the alignment problem. Using a combination of magnetic tracking, automatic alignment, and manual alignment, we are able to place adjacent volumes in their proper configuration. We train our medical team and evaluate alignment results with a scoring metric based on our ground truth synthetic data. Once volumes have been aligned, they are blended together with our novel region-based stitching algorithm. This fully automatic solution eliminates seams, and provides smooth transitions between volumes. Finally, we apply our volume embedding algorithm, which places the merged anatomical data into a background noise template to restore the signature ultrasound form factor. To enable translational interaction with the resulting large-scale datasets, we present an optical surface

tracking solution that enables users to move a probe and needle across a contoured shell, as if interacting with a real patient.

1.4 Contributions

Some of the novel contributions of our work are listed below:

1. **Ultrasound Data Synthesis:** We present a novel synthetic ultrasound data pipeline, including a novel ray-tracing solution to generate authentic-looking ultrasound datasets containing signature artifacts such as noise, shadows, and echoes.

2. **Registration and Evaluation:** We present a novel data-driven magnetic tracking calibration solution, a suite of semi-automatic alignment tools, and a training and evaluation scheme based on our synthetic data.

3. **Volume Stitching:** We present a novel n-dimensional image blending algorithm based on region identification, and introduce a distance map variant as well as a functional minimization variant.

4. **Volume Embedding:** We present a novel solution to an unexplored problem, in the form of our ultrasound volume embedding algorithm. This includes generating synthetic ultrasound noise, and applying a content-aware overlay process.

5. **Translational Tracking and Patient Simulation:** We present a novel human-computer interaction technique by introducing an optical surface tracking system, and apply this to patient simulation for ultrasound training. We also foresee a number of potential applications beyond this use case.

1.5 Overview of the Dissertation

We now present a chapter-by-chapter overview of the dissertation, describing each component of our solution in more detail.

Chapter 2: Related Work We review relevant prior work on ultrasound data synthesis and image alignment, volume stitching and embedding, as well as data interaction and patient simulation.

Chapter 3: Ultrasound Data Synthesis Generating our synthetic ultrasound datasets involves three key components. First, we generate procedural probe and tissue models. Our parameterized probe model supports a wide variety of probe types. Our tissue model provides a dense geometry representation based on randomized sphere packing. Next, a full physical simulation is run to mimic the interaction of the ultrasound probe, ultrasound gel, and tissue. This involves coupling rigid body dynamics (probe), fluid simulation (gel), and soft body dynamics (tissue) in a cohesive framework based on the Material Point Method. The final step is to generate image data from our simulation output using a physics-based rendering approach. This involves a ray-tracing solution to mimic ultrasound waves traveling through tissue, producing effects such as shadows and echoes, which play a critical role in the alignment problem.

Chapter 4: Registration and Evaluation We present three solutions to the volume registration problem. The first is a magnetic tracking approach that requires a 3D printed probe attachment to avoid interference, and a calibration procedure to determine the sensor/data relationship. Next, we present an automatic alignment algorithm based on an iterative refinement technique within a six dimensional search space. Finally, we describe our manual alignment system, which provides a variety of volume manipulation tools. We also present an evaluation metric based on our synthetic ultrasound datasets and their ground truth alignments.

Chapter 5: Volume Stitching Given an alignment, we merge the individual volumes into a single volume using a fully automatic blending algorithm. This algorithm takes into account the overlapping volumetric regions of the input datasets to produce a smooth transition optimizing the blend distance, resulting in a seamlessly stitched output volume. We

present two variations on this method: one based on distance maps, the other based on a multiresolution functional minimization. To make the latter approach computationally competitive, we apply a multilevel optimization scheme whereby we apply our functional minimization solution at increasingly higher resolutions.

Chapter 6: Volume Embedding To clean up the irregular boundaries of our stitched volumes, we employ an embedding algorithm to place the data into a background noise template. To produce a background template with adequate resolution, we augment real noise data with synthetic noise based on our ultrasound image formation process. To embed our anatomical data within the noise, we diffuse the low-frequency surface values of the anatomical dataset, and apply these results to the high-frequency information of our noise dataset. The two volumes are then blended into a single dataset, producing natural looking results conforming to the signature form factor associated with ultrasound imagery.

Chapter 7: Translational Data Interaction and Patient Simulation We present a new interaction technique to support the greater coverage afforded by our results. Since our data now allow translational interaction, not just rotational, we developed an optical surface tracking technique that allows users to interact freely with a 3D printed mannequin or body part. The system takes advantage of the light diffusion properties of translucent plastic to identify and track infrared LEDs when they come in contact with the surface. The user can place a mock probe and needle anywhere on this surface to view the corresponding ultrasound data and perform needle insertion procedures.

Chapter 8: Conclusion We conclude the dissertation by summarizing our work, discussing its limitations, and suggesting promising avenues for future work, including possible extensions and different applications.

CHAPTER 2

Related Work

Our work covers a wide range of topics, from rendering, to image stitching, to human-computer interaction. Many of these areas have been studied extensively, and therefore provide a point of comparison for our own work. We review this related work below, organized by the relevant chapters of this dissertation.

2.1 Ultrasound Data Synthesis

Existing work in producing synthetic ultrasound data generally fall into two main categories: acoustic wave simulations based on Field and Field II (Jensen, 1996), and systems designed for real-time interaction, such as those by Kutter et al. (2009) and Zhu et al. (2006).

Jensen (1996) laid the groundwork for a program now known as Field II, a Matlab-compatible acoustic wave simulator for high-quality ultrasound simulation. While the wave-based nature of this system allows for simulating accurate acoustic physics, it results in an inherently slow solution, requiring processing times of over an hour to render a single 2D image. As such, the system is best suited for experimenting with transducer array designs as opposed to generating synthetic data.

Both Kutter et al. (2009) and Zhu et al. (2006) combine volumetric computerized tomography (CT) data with a ray tracing technique to generate real-time simulated ultrasound slices. The work of Kutter et al. (2009) differs from ours in several key ways:

1. Their system operates on volumetric CT data, while ours operates on geometric data.

2. Their system does not take into account some of the more complex echo behavior as

presented in our solution; they only consider echoes along the initial transducer ray path, leaving out some of the key visual artifacts of ultrasound imagery.

3. Theirs is a 2D system, while ours is designed for 3D datasets (though we can also trivially generate 2D images).

4. Theirs is a real-time system, while ours is off-line. This is in part because they are only generating 2D datasets. In Chapter 7 we will also develop a real-time system for rendering 2D ultrasound imagery.

The work presented by Zhu et al. (2006) similarly uses segmented CT data to generate ultrasound-like 2D images in real time through ray tracing. The work of Zhu et al. (2006) differs from that of Kutter et al. (2009) in that they also generate volumetric textures using a Laplacian image pyramid technique, and blend this with their ray-traced CT data.

The rendering portion of our work falls somewhere between these two extremes. We aim to produce higher-fidelity results than the real-time solutions, without the extreme computational cost of a full wave-based simulation. We also focus on generating 3D datasets, while previous work typically focuses on 2D slices. One major difference between our approach and existing solutions is that before we begin our image rendering pass, we simulate the physical interactions that affect gel placement and tissue deformation. These interactions play a critical role in generating some of the artifacts that prove most challenging in the ultrasound data alignment problem.

2.2 Ultrasound Image Alignment

An ultrasound alignment solution is presented by Detmer et al. (1994), which also uses a magnetic tracker for determining the placement and orientation of ultrasound imagery. Their calibration solution involves acquiring around 40 2D images of a single point, as defined by the intersection of several strings. Pagoulatos et al. (2001) improve upon this technique by constructing a more complex calibration rig consisting of N-fiducials, a latticework of strings connecting known points. Their system requires only one scan once the transformation

between the magnetic base station and the calibration rig is known.

Our approach differs in several respects. First, we are using 3D ultrasound datasets, as opposed to 2D images used by the work above. Second, our calibration technique requires fewer scans than that of Detmer et al. (1994) and a simpler setup than that of Pagoulatos et al. (2001).

2.3 Volume Stitching

Image data stitching problems have been investigated for use in 2D imaging tasks such as the creation of mosaics and panoramas. One of the simplest and most common solutions is to weight each contribution by the distance of the current pixel to the edge of the image rectangle (Szeliski, 1996). Since such a solution can result in ghosting due to misalignment in the registration process, a multiband approach (Burt and Adelson, 1983) can extend this idea by blending low frequencies across a larger spatial region and higher frequencies across a smaller spatial region. A different approach to the seam-elimination problem is presented by Peleg (1981), which uses an iterative refinement method to find a smooth function that can be subtracted from each source image to eliminate gray-level offsets between images.

Our solution differs in that we identify different types of regions involved in the image blending process and the boundaries separating them, which allows us to create a more meaningful blend within regions of overlap. Furthermore, it is important to note that most photographic image blending problems deal with simple rectangles, while our solution supports arbitrary shapes, and even holes. This ability to support arbitrary shapes is necessary to account for the different shapes produced by different image acquisition devices and techniques. Finally, our focus is on 3D volumes, rather than 2D images, though the concepts apply to any dimension.

2.4 Volume Embedding

Our need to embed ultrasound data within a background template is rather unique, and therefore not a specific topic of past research, to our knowledge. However, techniques for similar problems are prevalent for general image processing. Poisson image editing (Pérez et al., 2003) provides a method for seamlessly compositing a portion of one image, A, into another, B. This solution computes a new image that attempts to preserve the gradient of A relative to the edge values of B, resulting in composites that often look very natural. The resulting overlap region, however, contains data that does not preserve the input values of either image. For this reason, we cannot apply such a technique, as we must preserve our anatomical data as much as possible. Indeed we see that our problem is somewhat the inverse of that in (Pérez et al., 2003), in that we wish to modify the non-overlap region of the image instead.

Inpainting (Criminisi et al., 2004) provides another option to our problem. This refers to the process of filling missing regions of data with synthetic information derived from surrounding data or another input source. While this sounds like a good fit for our problem, such content-aware solutions can be unpredictable, and we do not want to risk introducing any artifacts that may take on the appearance of anatomy. Furthermore, we would like to maintain strict visual behavior in our filled regions, such as the proper stretching of noise artifacts relative to the volume origin. Inpainting cannot fulfill this requirement. Instead we generate a background volume that does.

2.5 Data Interaction and Patient Simulation

Much work exists in the space of surface tracking for human computer interaction (Jacob, 1996). Technologies such as ball mice (English et al., 1967), optical mice (Lyon, 1981), resistive touch screens (Downs, 2005), capacitive touch screens (Kim et al., 2011), and surface acoustic wave touch screens (Adler and Desmares, 1987) all provide accurate and robust solutions to 2D translational tracking. Of particular interest is the type of tracking provided

by the light pen (Stotz, 1963). This is in some ways the most similar to our solution, in that it uses light to solve the tracking problem, albeit in the inverse direction. It is important to note that all systems mentioned thus far conform to a 2D plane, while we aim to develop a tracking solution for a 3D surface.

Some existing ultrasound simulators leverage full 3D tracking to solve this problem, similar to the probe tracking systems described in Chapter 4. Zhu et al. (2006) use calibrated magnetic trackers to perform needle insertion procedures on a mannequin. Such systems, however, are both expensive and obtrusive.

Several solutions to the translational problem have been investigated at SonoSim, Inc. Modifications to the standard translational input devices mentioned earlier were considered, extended in such a way as to allow for the coupling of the probe. The current solution is based on a proprietary pressure sensor. The 7cm x 7cm semi-flexible sensor has a resolution of 14×14 pressure-sensitive cells. The output of the sensor can be treated like an image, and a variety of algorithms have been applied to interpreting the pressure heat-maps. While this solution solves the translation problem, there are several limitations to this approach:

1. Size. The active area cannot be extended without production of a new sensor.

2. Contours. The pad is mostly limited to curvature in one dimension, and even this curvature is limited to a fairly large curvature radius. Therefore the pad cannot easily conform to arbitrary 3D contours.

3. Disambiguation. The pad cannot easily distinguish between pressure sources, be that the probe, the needle, or the user's hands. The identity of a source must be determined based on the shape signature, or by enforcing a temporal ordering of object placement and limiting what can come in contact with the surface. Shape signatures are unreliable and computationally expensive. Temporal ordering introduces a high burden on the user, and it requires continuous tracking, which can be unreliable.

4. Proximity. The limited resolution, combined with pressure diffusion caused by protective layers, makes it difficult to separate sources when their proximity is small.

5. Residual. The pressure pad exhibits a ghosting behavior wherein the sensor will continue to detect a residual signal for a short period after the stimulus has been removed.

6. Force. The pad requires a minimum pressure to robustly separate a stimulus signal from background noise. This is problematic for our needle, which has a retractable tip that is purposefully designed to require small force to mimic the behavior of a true needle insertion.

7. Fragility. Attempting to apply even a slight contour to one of our pressure pads caused irreparable damage.

8. Development. The pads and their development cycle are costly and cannot be prototyped with standard off-the-shelf components.

Our solution solves all of these problems by using optical tracking instead of a pressure pad.

CHAPTER 3

Ultrasound Data Synthesis

Synthetic data allows us to work within a controlled environment such that we can better understand the factors involved in aligning multiple ultrasound volumes. Most importantly, synthetic data provides us ground truth. That is, we know the exact relationship between the probe positions associated with each volume and, therefore, know the relationship between the volumes themselves, regardless of what we can identify within the data itself.

Generating useful synthetic ultrasound data requires us to simulate both the physical interactions involved, as well as the image formation process. We aim to develop a solution that captures the following key characteristics of ultrasound data that pose the greatest challenges during alignment:

1. Deformation. The interaction of the probe and tissue results in deformation of the anatomy. These deformations are dynamic and inconsistent between volumes.

2. Shadows. Since shadows originate from the probe head, their appearance and placement changes as the probe is moved.

3. Echoes. Like shadows, echoes evolve as the probe moves and result in inconsistencies between volumes.

4. Noise. A major challenge in interpreting ultrasound imagery is separating signal from noise.

To capture all of these features in our synthetic data, we begin with a physical simulation involving rigid bodies, soft bodies, and viscous fluids. We then feed the results into a ray tracing algorithm to generate the final volumes.

3.1 Models

Before we can begin the simulation itself, we need to generate geometric representations of the elements involved: probe, tissue, and gel. The first two elements must be procedural, in order to create a diverse set of synthetic data.

3.1.1 Probe Model

The probe geometry plays a critical role in the ultrasound image formation process. Unlike photography, where the form factor of the camera plays little to no role in the acquisition process, the exterior form of an ultrasound probe affects the imagery directly. This arises from the fact that the probe must be placed in physical contact with the structure we wish to image. As a result, the probe deforms the structure. Depending on the tissue being imaged, this deformation can cause notable artifacts within the first several centimeters below the skin. Because of this coupling, the geometry of the probe cannot be ignored.

The shape of the ultrasound probe head is a great example of function dictating form. In a standard 3D probe, the transducers are arranged in an arc (typically a segment of a circle) in order to capture a wide field of view. These transducers must be placed close against their enclosure in order to ensure proper transmission of the ultrasound waves, and the enclosure must be thin enough so as to not impede this transmission. Consequently, the exterior surface reflects the arc form of the transducer arrangement.

Furthermore, a 3D probe must mechanically sweep its linear array of transducers during volumetric scans, which dictates the form of the probe in the sweep direction. This rotational movement defines a circular path that the transducer array must follow. Thus, our probe head is defined by the circle segment dictating the transducer arrangement, and another circle segment dictating the sweep movement. Note that these two circle segments can be defined with different radii and different origins. As a result, the shape cannot be represented by a simple sphere or ellipsoid.

To accommodate this form, we developed a parametric description of the probe geometry

that accurately reflects the internal workings. These same parameters will be applied during the rendering phase to ensure the correct correspondence between the physical model and the image formation process. In describing these parameters we will use the following definitions:

Definition 3.1.1. Fan. Relating to the plane defined by the transducer array. Typically this will refer to the imaging plane used during a 2D scan.

Definition 3.1.2. Sweep. Relating to the plane defined by the rotational axis of the transducer array. This is orthogonal to the fan plane. This comes into play during a 3D scan.

With our terms defined, we can list the three scalar parameters we expose to define the probe head:

1. θ_f Fan angle
2. r_f Fan radius
3. r_s Sweep radius

Figure 3.1 shows the geometric meaning of these parameters. Figure 3.2 shows two probe models generated from these parameters.

3.1.2 Tissue Model

For our tissue model, we needed the ability to densely fill a volume with large and small scale features, in order to represent the variety seen in anatomical structures. This data would also need to be generated procedurally in order to produce multiple datasets quickly, ruling out hand-modeled geometry. For simplicity, we decided to use spheres.

Another requirement of our tissue geometry was that it deform well. Thus, we chose to avoid standard spherical coordinates in favor of icospheres (subdivided icosahedra) due to their uniform nature; the uniform tessellation of an icosphere allows it to deform consistently regardless of the type and direction of deformation. Figure 3.3 demonstrates the process of generating an icosphere from an icosahedron.

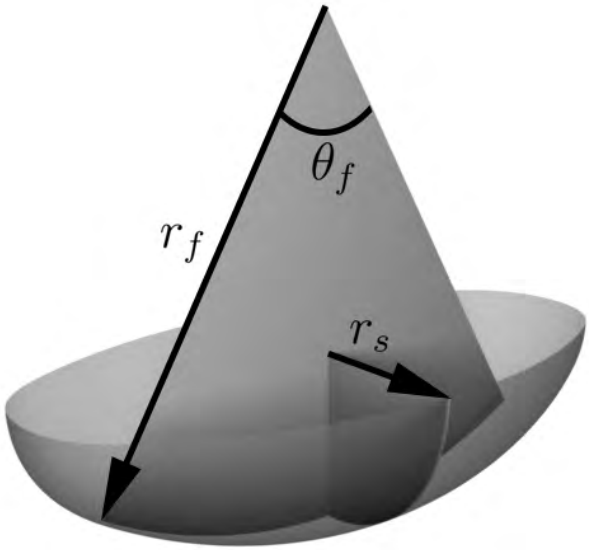


Figure 3.1: Diagram showing the geometric meaning of our probe head parameters.

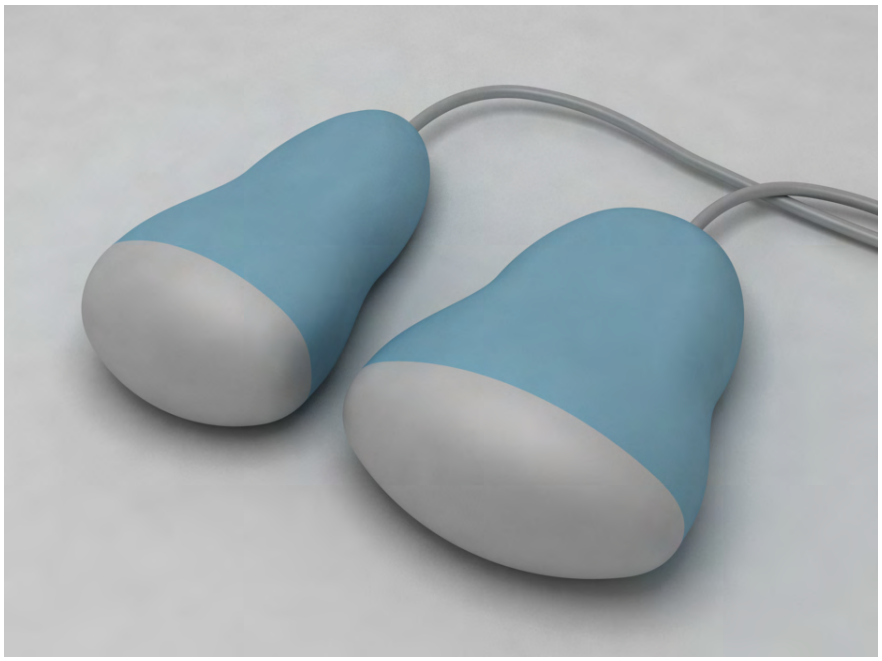
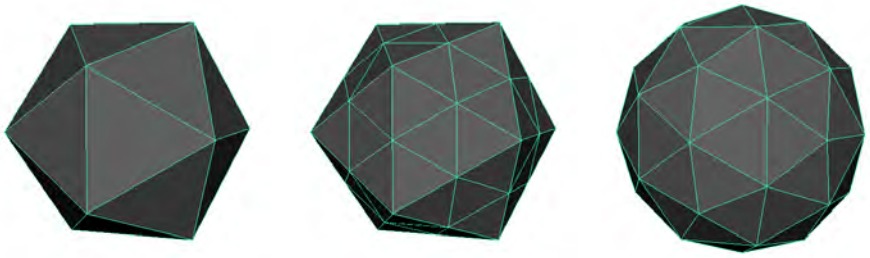
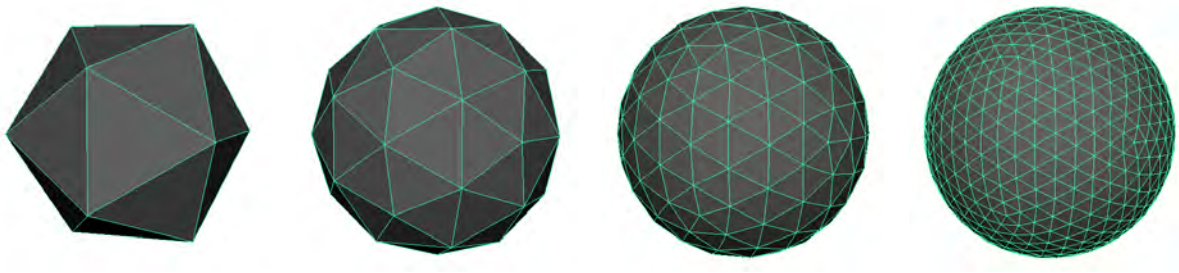


Figure 3.2: A rendering of two probes generated procedurally. Coloration and cables are added for illustration purposes.



(a) Icosahedron (b) Subdivided (c) Projected to sphere

Figure 3.3: Subdividing an icosahedron to create an icosphere. This process can be repeated indefinitely to generate higher resolution spheres.



(a) Level 0 (b) Level 1 (c) Level 2 (d) Level 3

Figure 3.4: Icospheres of different subdivision levels.

Since our goal was to fill the space densely, we employed a simple level-of-detail system to allow for higher resolution icospheres for larger features, and lower resolution icospheres for smaller features (Figure 3.4). This greatly reduced both the memory footprint and the computational footprint of our rendering pass.

To fill our volume with icospheres, we applied a variant of sphere packing. Typically this refers to the process of arranging spheres of identical size as compactly as possible within a given space without any overlap. Our needs differed in two ways, namely that we would be packing spheres of different sizes, and that we only needed to ensure the non-overlap criteria. Our approach uses a randomized algorithm, similar to that of Manna (1992), which randomly selects a point in empty space and inserts a sphere, with a radius defined by the distance to the closest neighboring sphere. Our approach involves generating spheres of random size and placement, checking for overlap with existing spheres, and discarding those that conflict. We ran this process until the desired number of spheres were added to the

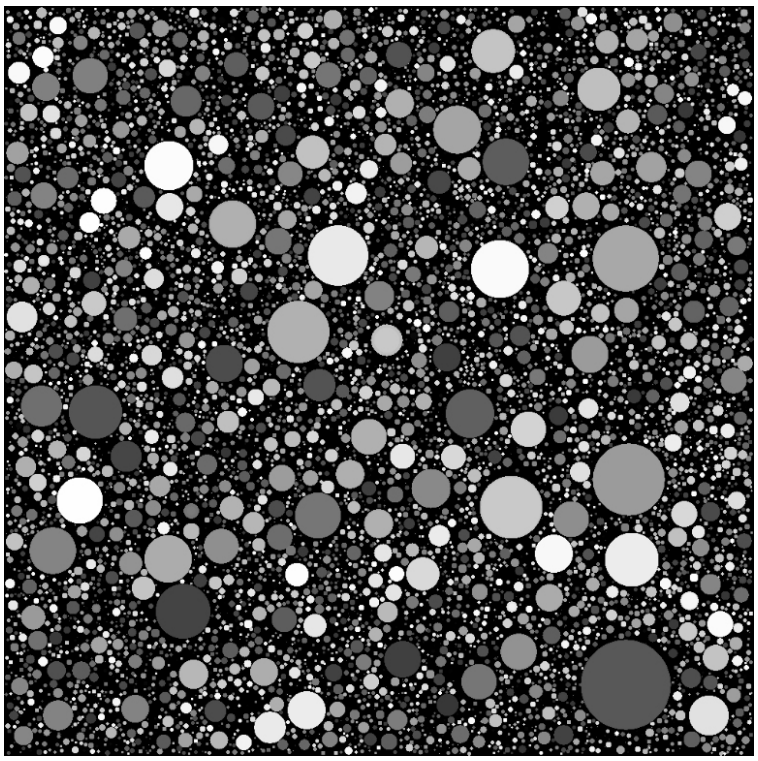


Figure 3.5: A cross section of a volume densely filled with icospheres. Different intensities represent different echogenic properties.

space. To speed up the overlap check, we added each sphere to a three dimensional hash table representing our volume, with each cell containing a list of spheres occupying some portion of that space (Gissler et al., 2011). In this way we could quickly check the local region of a new candidate for conflicts, as opposed to iterating across all other spheres. Each sphere was randomly assigned a set of parameters defining its material characteristics. Figure 3.5 shows an example of such a packing.

3.2 Motion Tracking

In addition to defining our models, one other component must be provided as an input to the physical simulation: probe motion. To acquire this motion, we used a six-degree-of-freedom magnetic tracker (see Chapter 4 for more on this). We simply held the tracker and made a typical probe movement, from approaching the anatomical region, to arranging the gel, to moving to an adjacent part of the body. This movement was then transferred to our probe

model for simulation.

3.3 Physical Simulation

The physical simulation provides two key contributions to our final synthetic data. The first contribution is the introduction of accurate probe deformation into our tissue geometry. The second, and more challenging contribution, is the integration of the ultrasound gel. Since gel plays a key role in the image formation process, it must be simulated accurately.

3.3.1 Smoothed Particle Hydrodynamics

We first explored the possibility of simulating the gel with Smoothed Particle Hydrodynamics (SPH). This model offers many benefits over grid-based Eulerian solutions in that it trivially supports conservation of mass, is relatively straight-forward to implement, and provides a natural extension for coupling with rigid and soft bodies.

The main idea behind SPH lies in its use of interpolation to provide a continuous field from a discrete set of particles. The influence of a given particle extends to its neighbors within a fixed smoothing radius, which parameterizes a symmetric smoothing kernel. This provides a reasonable approximation since the influence of nearby particles should approach zero quickly. In our implementation, we used smoothing kernels based on (Müller et al., 2003) and pressure computations based on (Batchelor, 2000), with some of our constants chosen from (Monaghan, 2005). This implementation produced promising results, as shown in Figure 3.6.

We further augmented our SPH solution in order to accommodate the probe and tissue, which would require complex boundary conditions and coupling. We based our boundary conditions on the work of Akinci et al. (2012) and were able to achieve good results, as shown in Figure 3.7.

We were able to apply this solution to our scanning scenario (Figure 3.8), but the high viscosity of gel proved difficult to model well with SPH, which required incredibly small

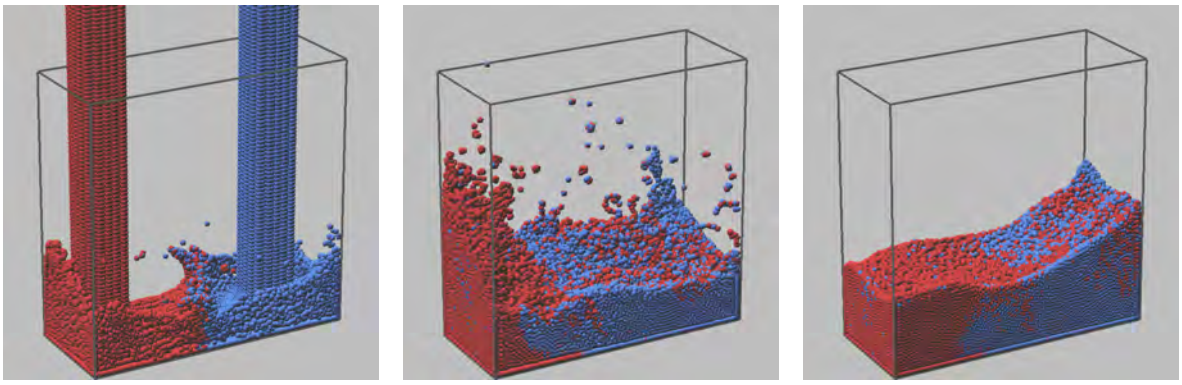


Figure 3.6: Three frames from a test run of our SPH implementation.

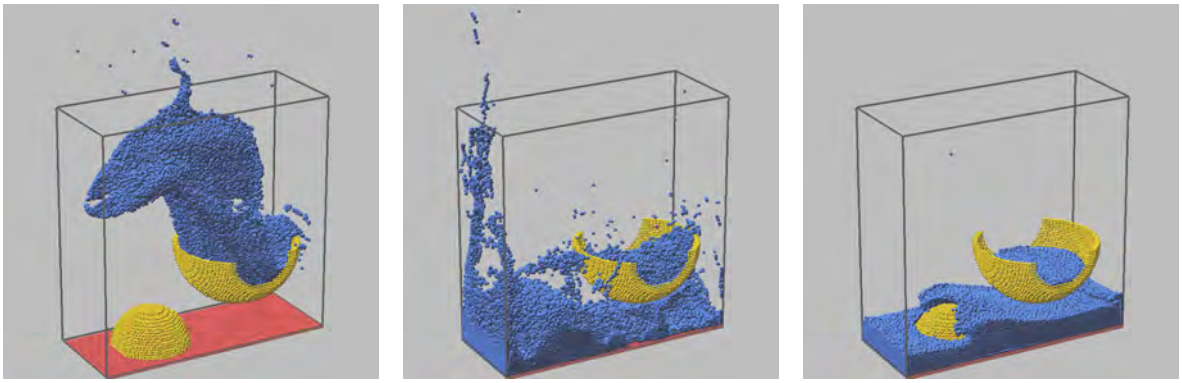


Figure 3.7: Three frames from a test run of our particle boundary implementation.

time-steps, and thus became impractical to simulate for our purposes.

3.3.2 Material Point Method

Given the challenges we faced with SPH, and because physical simulation is not the focus of our work, we chose instead to use an existing solution. We collaborated with the UCLA Mathematics Department to leverage the strength of their Material Point Method (MPM) techniques. Their robust code-base is geared towards simulating non-Newtonian fluids such as our ultrasound gel, making it an ideal solution for our work.

To simulate the ultrasound gel, they applied a viscoplastic constitutive model as described by Ram et al. (2015). To model the tissue as an elastic object and couple it with the gel, they applied the work of Jiang et al. (2015). Finally, to optimize the simulation, they applied the

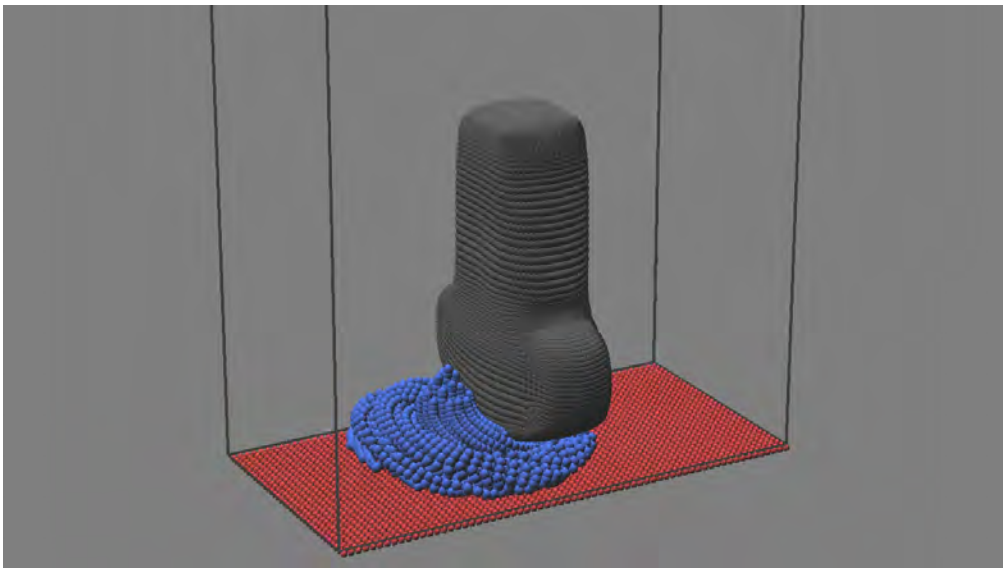


Figure 3.8: Test interaction between highly viscous gel and an ultrasound probe.

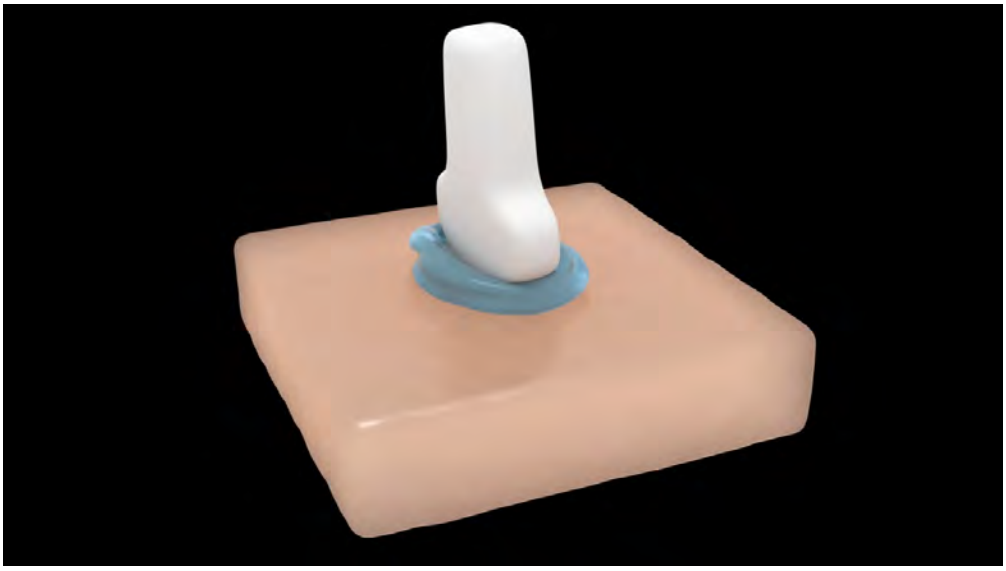
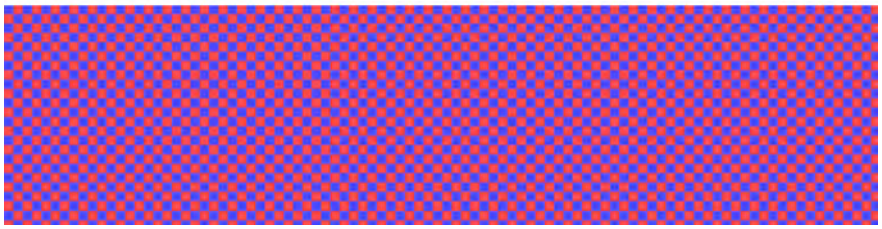
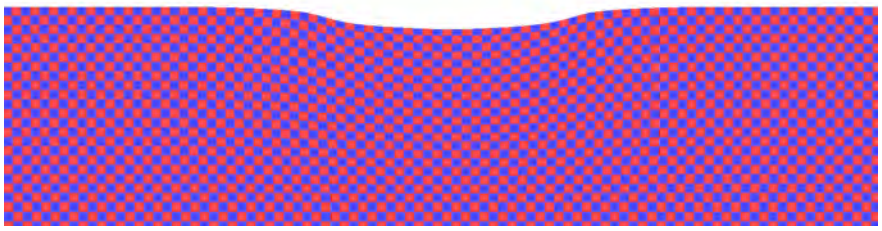


Figure 3.9: Rendering of the interaction between highly viscous gel and an ultrasound probe.



(a) Before deformation



(b) After deformation

Figure 3.10: Example of applying our simulation output to a checkerboard grid. This shows a slice through our tissue block, with deformation caused by the ultrasound probe and gel.

techniques of Gast et al. (2015). Using their system, we were able to obtain the results we were seeking without much modification of the existing code. Figure 3.9 shows an example of these results.

3.3.3 Applying Physical Simulation

We simulated the tissue with a uniform block of particles spaced evenly in a grid. In this way we could easily interpret this data as a collection of tetrahedra, within which we could embed our icosphere geometry. By determining which tetrahedron contains a given geometry vertex, we can extract its barycentric coordinates and use this to track its position after deformation; we simply apply the extracted coordinates to the deformed tetrahedron to arrive at the new vertex position. Through this process we can reapply the same deformation to any number of geometric models.

3.4 Rendering

Generating synthetic ultrasound imagery from our physical simulation is one of the key steps of our solution. To produce results that properly mimic real ultrasound data, our system had to model some of the core concepts of the ultrasound image formation process.

3.4.1 Probe Design

An ultrasound probe uses acoustic waves to measure the distance of hidden anatomical structures (Donald et al., 1958). The probe relies on piezoelectric transducers to both emit ultrasound waves and detect their response after they bounce off underlying anatomy and return to the probe. The waves are reflected in varying degrees when passing through areas of density variation; for example, between muscle tissue and veins. Both the time delay and the intensity of the echo can be used to locate the structure and generate a pixel value in the appropriate area of the resulting image. In a 2D probe, a single linear array is installed at the probe tip to generate a fan of data. A 3D probe does the same with a 2D array of transducers, or by mechanically rotating a linear array to capture a series of 2D slices, which are then assembled into a 3D volume.

3.4.2 Wave Physics

Ultrasound waves in the range of 2–20 MHz are typically used for medical imaging. Higher frequencies produce smaller wavelengths, which can resolve greater detail. However, these frequencies cannot penetrate as deeply and can only be used for smaller structures closer to the surface. Since acoustic waves cannot travel through a vacuum, regions of lower density (particular gaseous regions) impede the transmission of the waves, producing shadowed regions in the resulting image.

Echo intensity depends on the angle at which a wave strikes an acoustic interface. The closer the angle is to the normal of the surface, the greater the intensity.

One of the core concepts of ultrasonography is that of echogenicity. This refers to the

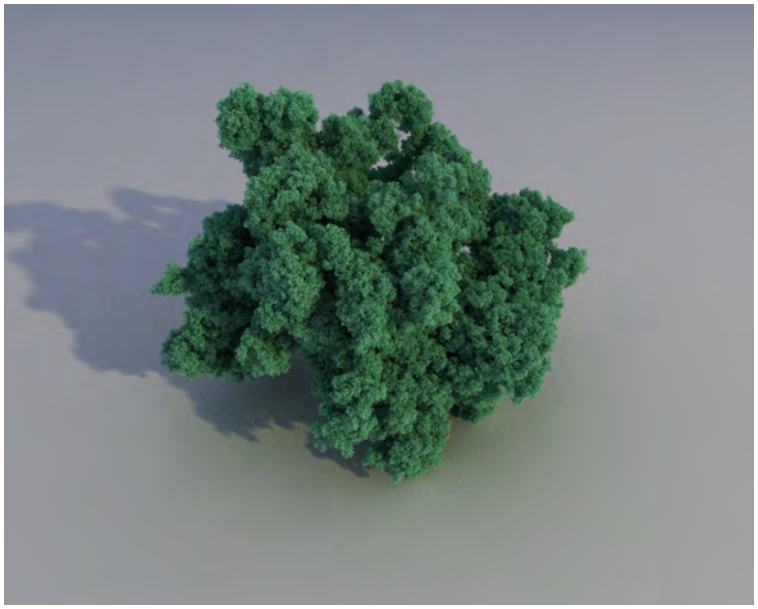


Figure 3.11: An example of the extreme realism possible with ray tracing.

extent to which a material returns an echo. A material with a higher echogenicity ("hyperechogenic") generally appears brighter, while a material with a lower echogenicity ("hypoechogenic") appears darker (Prince and Links, 2006).

3.4.3 Ray Tracing

Typically associated with rendering physically-based photo-realistic visual images (Figure 3.11), ray tracing (Glassner, 1989) presents an appealing option for creating synthetic ultrasound images. This technique provides a natural mechanism for producing features such as shadows and bounces (echoes). While such an approach does not traditionally accommodate wave-specific effects such as constructive and destructive interference, these are less crucial to the overall appearance of ultrasound images.

In many ray tracing systems, rays are traced backwards (Arvo and Chelmsford, 1986), starting from the image plane, passing through the lens (or pinhole), and out into the scene. Techniques such as photon mapping (Jensen, 2001) also trace rays in the forward direction, starting at the light source(s) instead of the camera. Our solution does not need to make this source/destination distinction, since the equivalent of a light source in ultrasonography is the

transducer, which is also the equivalent of the camera. Thus, our approach involves emitting rays from each transducer into the surrounding geometry. For 3D volume renderings, we also trace rays from each transducer position along the sweep trajectory. Each time a ray scatters or intersects an object, the location and travel distance is logged. After each hit, the ray is reflected based on the angle of incidence. This continues until the total travel distance exceeds the depth setting of our volume.

In addition to the duality of the transducer as both emitter and receiver, our ultrasound ray tracing differs from traditional ray tracing in the type of data captured. In a traditional ray tracer, the recording device (film) captures intensity and wavelength. In our ultrasound ray tracer, the recording device (transducer) captures intensity, response time, and bounce location.

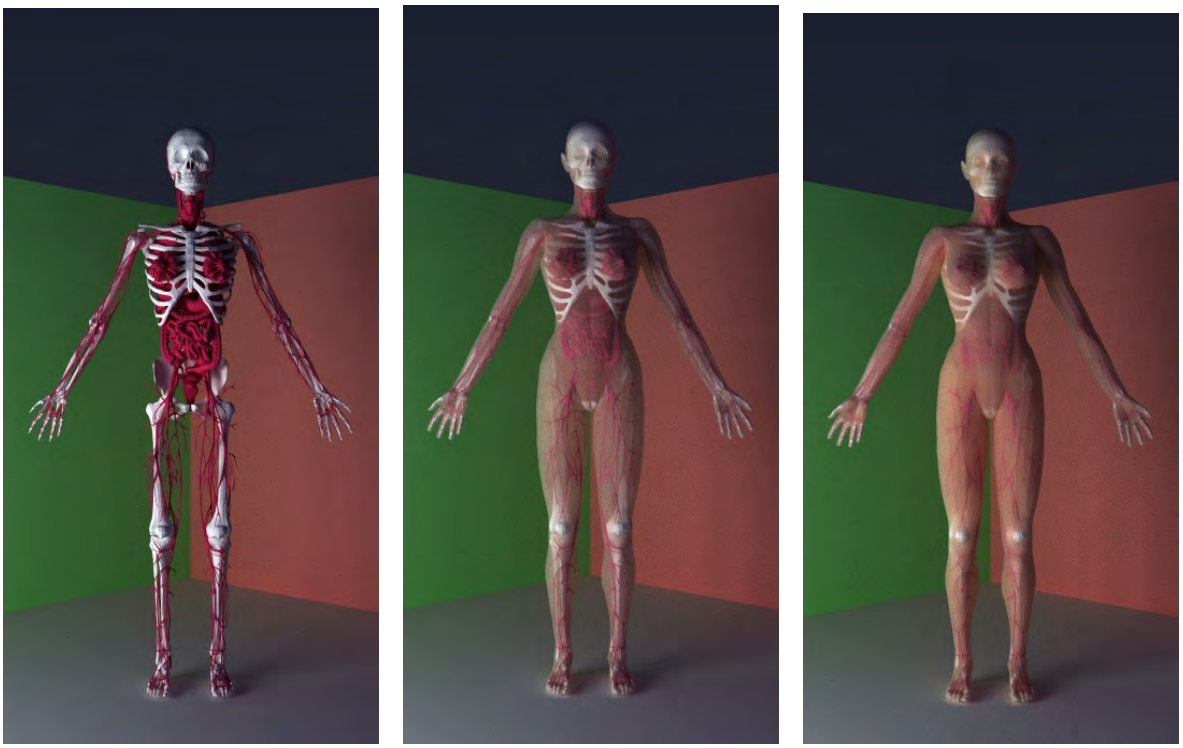
3.4.4 Volumetric Scattering

Another key difference between our ray tracer and a traditional ray tracer is the reliance on volumetric scattering. While most modern ray tracing software incorporate volumetric scattering (or subsurface scattering (Hanrahan and Krueger, 1993)) as a feature, it is at the forefront of ultrasound image formation. We take a probabilistic approach to this problem, and derive a distribution function to determine if and where a ray scatters as it passes through a fixed length of volume between geometry intersections. The probability $P(x)$ that a ray scatters while traveling distance x through some material with scattering coefficient $0 < p < 1$ is given by

$$P(x) = 1 - (1 - p)^x. \tag{3.1}$$

Here, p represents the probability of the material scattering a ray over some dx, and is one of the properties assigned to a piece of geometry during the procedural generation phase. $(1-p)^x$ provides the probability that the ray does not scatter within x. Note that a material that has no scattering properties (e.g., a vacuum) will have $p = 0$, and a material that immediately scatters a ray (e.g., a solid surface) will have $p = 1$.

Using this probability distribution, we first determine whether our ray scatters between



(a) Without skin model

(b) With skin model, some scattering

(c) With skin model, more scattering

Figure 3.12: Examples of scattering within a photo-real setting. Note how scattering causes deeper features to be obscured.

its last geometry intersection and the next (this distance is x). To do this, we simply compute $P(x)$ and then choose a random variable $0 \leq r \leq 1$. If $r \leq P(x)$ then we have a scatter bounce.

The next task is to determine, probabilistically, where in $(0, x)$ the bounce occurred. First, we choose a random variable r within the range

$$0 \leq r \leq \frac{(1-p)^x - 1}{\log(1-p)}. \tag{3.2}$$

We then compute the scatter distance

$$d = \frac{\log[r \log(1-p) + 1]}{\log(1-p)}. \tag{3.3}$$

Finally, we use d to update our ray origin and choose a new direction from a uniform

distribution around our original direction vector. We tested this scattering technique within a photo-real setting to see if the results appeared reasonable. Figure 3.12 shows some of these results.

We can see in Algorithm 1 how we incorporate this volumetric scattering model into our ray-tracing algorithm. The algorithm maintains a stack of material attributes to determine what medium (and therefore what scattering properties) must be considered.

3.4.5 Image Formation

Once we have traced rays for each transducer, taking into account geometry bounces as well as scatter bounces, we can begin the image formation process. This process differs significantly from standard photo-real image formation in two key ways. First, our data do not lie on the imaging sensor as they do in traditional rendering, where we can think of the image as existing on the virtual image plane. Instead, our data exist in the 2D/3D space in front of the transducers. Second, our data are not neatly packed into pixels or voxels as in traditional rendering. Instead, we must interpolate our data from discrete points and rasterize this into a standard image or volume grid representation.

Our image formation stage also involves post-processing our data. As mentioned above, we store positional information for each bounce during our ray tracing pass. To turn these data into final intensity values, we must take into account the relationship between the bounce location and the transducer. As was mentioned in Section 3.4.2, a wave's influence on a receiving transducer is determined by its propagation direction. A wave arriving head-on produces the strongest signal, while a perpendicular direction produces the weakest signal.

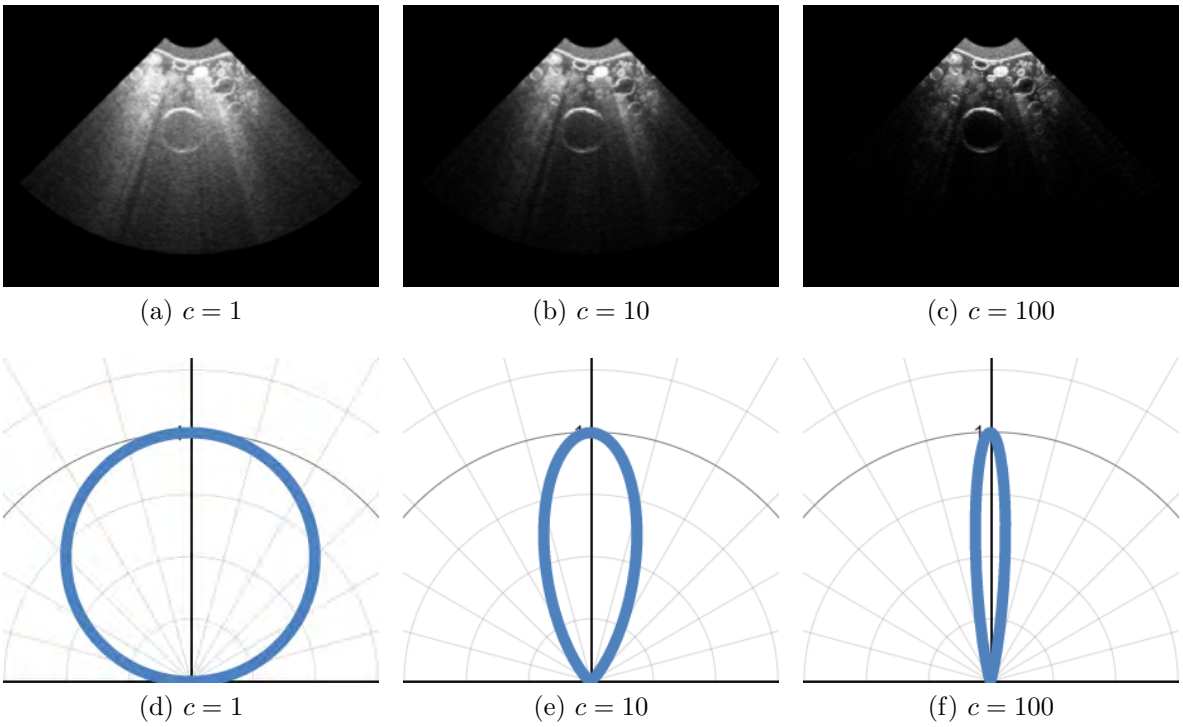
The exact behavior of the angular fall-off depends on the arrangement, design, and processing of the transducer signals, but it can generally be modeled as a sinusoid. For a transducer with position $\mathbf{p}$ and direction $\mathbf{v}$, and a ray bounce at position $\mathbf{b}$, we compute its attenuation as

$$\alpha = \left(\frac{\mathbf{n} \cdot (\mathbf{b} - \mathbf{p})}{||\mathbf{n}||\ ||\mathbf{b} - \mathbf{p}||} \right)^c, \tag{3.4}$$

where c is a clarity parameter that allows us to control how clear or fuzzy the image is due

```
initialize octree;
material attribute stack = [];
foreach sweep position do
    foreach transducer do
        for number of samples do
            generate a ray for the current transducer and sweep position;
            path length = 0;
            push back environment material attributes;
            while path length < max depth do
                if ray intersects geometry then
                    read top material attributes from attribute stack;
                    if scatter bounce then
                        compute scatter bounce position and direction;
                        path length += ||bounce position - ray origin||;
                        record bounce info and path length;
                        update ray origin and direction;
                    else
                        if surface bounce then
                            compute bounce direction;
                            path length += ||bounce position - ray origin||;
                            record bounce info and path length;
                            update ray origin and direction;
                        else
                            if exiting a structure then
                                pop material attribute;
                            else
                                push material attribute of intersected geometry;
                            end
                            path length += ||intersection position - ray origin||;
                            update ray origin;
                        end
                    end
                else
                    exit loop;
                end
            end
        end
    end
end
```

Algorithm 1: Our ultrasound ray-tracing algorithm.



(a) $c = 1$ (b) $c = 10$ (c) $c = 100$

(d) $c = 1$ (e) $c = 10$ (f) $c = 100$

Figure 3.13: Demonstration of the effect of changing the clarity parameter c. The top row shows the effects on the resulting rendering. The bottom row shows a 2D illustration of the attenuation function, where the vertical axis represents the transducer direction. Note how a low clarity value results in a hazier image.

to off-directional contributions. The greater the value of c, the clearer the image. We can see a visual representation of this parameter in Figure 3.13.

Next, we must determine where to place our intensity values in the 2D or 3D image. Since ultrasound images determine placement based on time delays, we ignore the accurate positional information we used to determine receiver attenuation, and instead simply place our intensity value at the proper distance along the transducer direction based on the time delay stored with the position information. This intentional inaccuracy contributes to the telltale artifacts of ultrasound imagery.

Finally, we rasterize our intensity values to a 2D image or 3D volume by interpolating neighboring values based on their depth and angular position. We apply trilinear interpolation (Levoy, 1988) to produce smooth results. Our results capture the signature look of ultrasound imagery, as show in Figure 3.14.

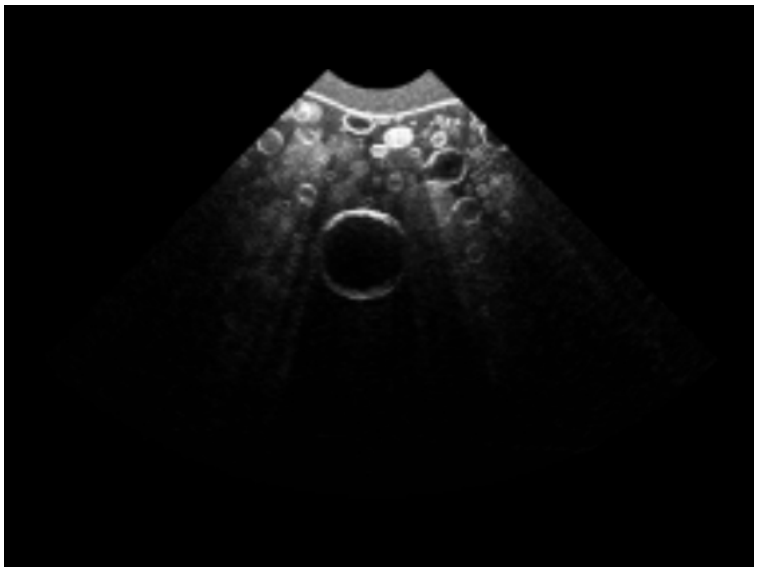


Figure 3.14: A sample rendering from our synthetic data pipeline. Note the signature ultrasound artifacts such as deformation, noise, shadows, and echoes.

3.4.6 Acceleration Structure

Because of the high polygon count of our dense tissue representation, we use an octree acceleration structure to make the geometry intersections more efficient. In a brute force approach, each ray would need to be tested against every object in the scene, resulting in a linear runtime $\mathcal{O}(n)$ on the number of objects in the scene. This can result in extremely poor performance when many rays and many objects are considered. An octree (Glassner, 1984) boosts this performance significantly by subdividing the space recursively. With this method, the scene space is first divided into 8 quadrants. If no object intersects one of these quadrants, we can identify that quadrant as empty. If an object does intersect a quadrant, it is once again divided into 8 subquadrants and the process repeated. This entire process can be computed as a preprocessing step. During the ray tracing phase, we simply check each quadrant that the ray intersects to see if we have any potential intersections. If none exist, we can stop immediately. If a potential intersection does exist, we recurse to the next level and repeat the process, ultimately reducing the run time to $\mathcal{O}(\log n)$.

3.5 Results

As shown in Figure 3.14, our synthetic data pipeline produces natural-looking results that capture the core ultrasound artifacts such as deformation, noise, shadows, and echoes. We will show in Chapter 4 how these datasets can be used effectively both for training and alignment evaluation.

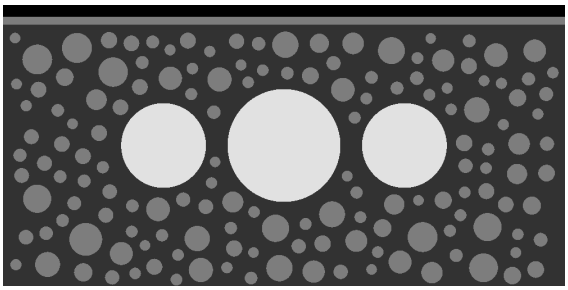
CHAPTER 4

Registration and Evaluation

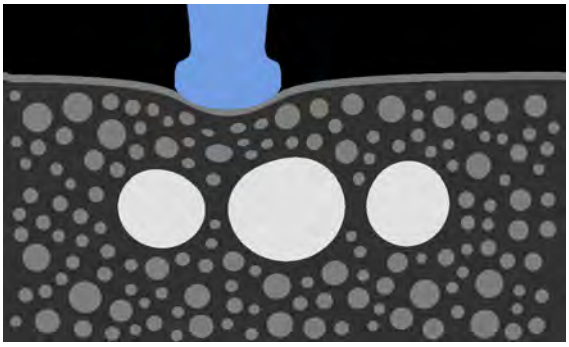
Registration — the process of aligning adjacent volumes — plays a pivotal role in generating large-scale ultrasound datasets. It also poses some of the greatest challenges. Due to the numerous artifacts present in ultrasound data, determining proper alignment is not only difficult, but in some ways impossible.

The problem is two-fold: First, we must recognize that a true alignment is a combination of complex anatomical deformations. Some of these deformations are caused by the pressure of the probe head against the anatomy (Figure 4.1) while others are caused by patient movement (Table 4). Second, image formation artifacts such as noise, blur, and shadowing lead to ambiguities in the correspondence problem. Often an entire visual structure, such as a ureteral jet, may be present in one volume but not another, due to its transient nature. Thus, a truly accurate alignment is impossible.

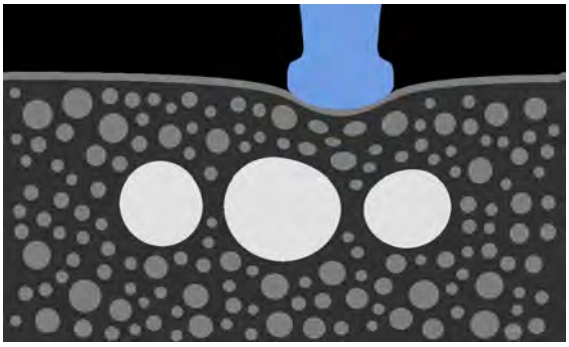
To overcome these challenges, we make the simplifying assumption that the alignment is a rigid transformation. Since the dominant differences between volumes are due to the rigid movement of the ultrasound probe, this is a reasonable simplification to make. We present three methods for determining these transformations, from recording them through magnetic tracking, to computing them through an automatic alignment algorithm, to defining them manually through a user interface. Depending on the circumstances and quality of a given scan session, some or all of these techniques may be applied.



(a) Non-deformed anatomy



(b) With probe deformation to the left



(c) With probe deformation to the right

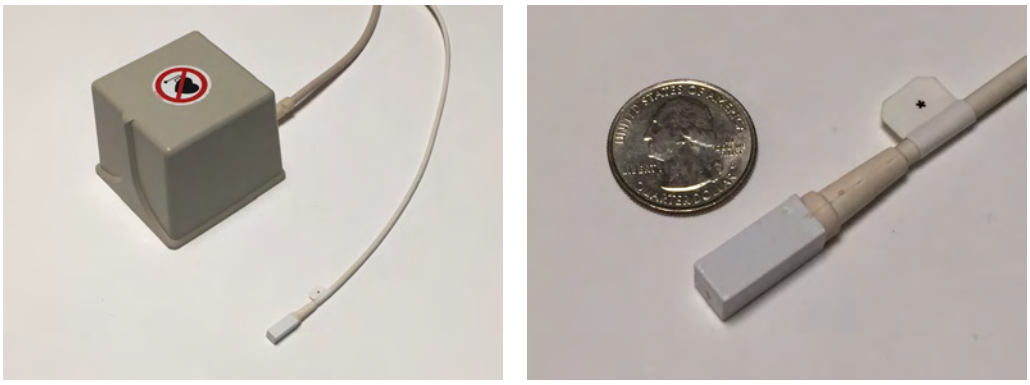
Figure 4.1: Illustration of deformation.

Voluntary	**Involuntary**
Squirming	Blood Flow
Talking	Bowel Peristalsis
Laughing	Bladder Size
Breathing	Ureteral Jets
	Fetal Movement

Table 4.1: Examples of different types of patient movement. Note that some of the voluntary movements may in fact be more accurately characterized as involuntary for some patients.

4.1 Magnetic Tracking

Ideally we would like to provide a good initial estimate to our algorithmic alignment solution. This can be achieved by monitoring the probe position during the scan session itself through motion tracking. The main challenge here is incorporating a system that is agile and unobtrusive, since we would like to acquire data during routine clinical exams. An optical tracking motion capture rig along the likes of a visual effects setup is not possible under these constraints, as such a solution typically requires an entire room to be outfitted with



(a) Ascension base station and sensor (b) Close up of the sensor

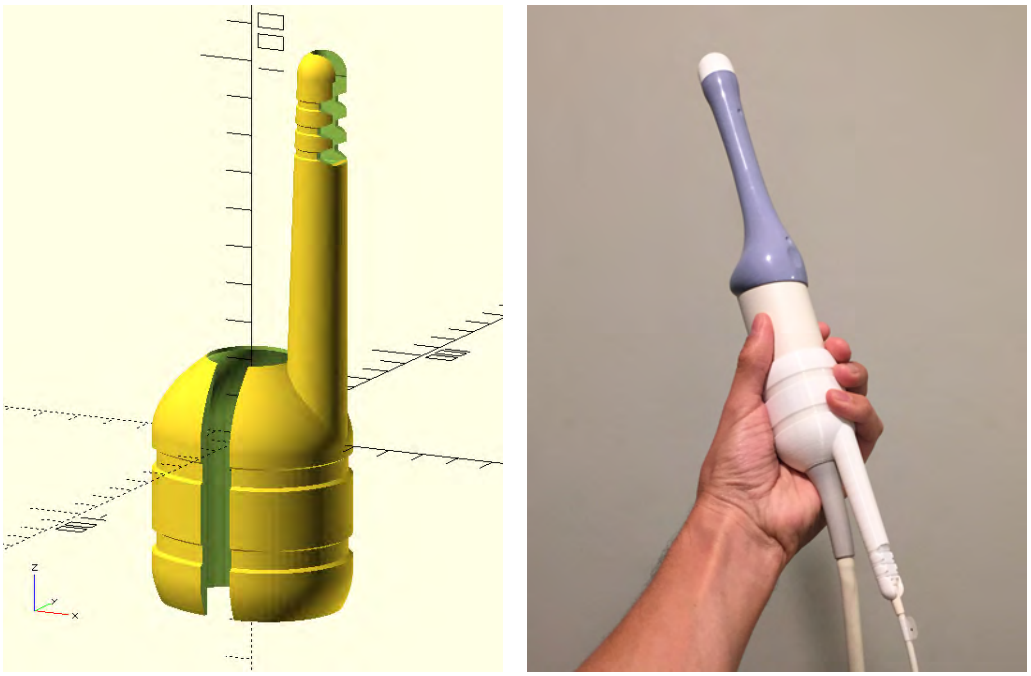
Figure 4.2: The Ascension magnetic tracking solution.

sensors (Meyer et al., 1992), and line-of-sight cannot be obstructed. In addition to being impractical, an optical tracking setup also raises privacy concerns due to the use of cameras. To get around these problems, we present a solution based on magnetic tracking.

4.1.1 Setup

Magnetic tracking provides an ideal motion tracking solution since it can be incorporated with a minimal physical footprint, generally just a base station, sensor, processing box, and laptop. To this end we employed an Ascension six-degree-of-freedom magnetic tracker (Figure 4.2). One of the challenges with this solution was overcoming the electro-magnetic interference generated by the ultrasound probe. Early tests with the magnetic tracker attached to the body of the probe resulted in highly unreliable and ultimately unusable results.

While the probe interference could theoretically be mapped and compensated for, we chose instead to build an attachment to move the sensor outside of the interference region. We used OpenSCAD to design an attachment that would precisely fit our transvaginal probe, allowing the magnetic sensor to be placed several inches behind the probe, where interference no longer poses a problem. Careful attention was taken to design the attachment in such a way as to fit snugly against the contours of the probe with very small tolerances to allow for consistent and rigid placement. One design constraint was that the attachment would need to be removed between scans to allow for cleaning of the probe. Thus, repeatability in



(a) OpenSCAD render

(b) Final 3D print attached to both the probe and sensor

Figure 4.3: Transvaginal probe attachment to add magnetic tracking.

placement was very important. Likewise, the slot for the magnetic sensor had to be designed with similar constraints in mind, since the sensor would have to be removed for use with difference probes. Grooves were built into the design to allow for zip ties and rubber bands to help secure the assembly. A rendering of the final design is shown in Figure 4.3a.

The attachment was 3D printed, and the interior coated with rubber to prevent slippage. As per the design, zip ties were used to secure the attachment to the probe, and strong orthodontic rubber bands were placed around the magnetic sensor to hold it in place. The finished rig is shown in Figure 4.3b.

This setup allows the sonograph to scan as usual, with the base station discretely hidden under the patient table to avoid interfering with the procedure. It is important to note that a non-metallic (e.g., wooden) table is ideal, as it will not introduce disturbances into the magnetic field.

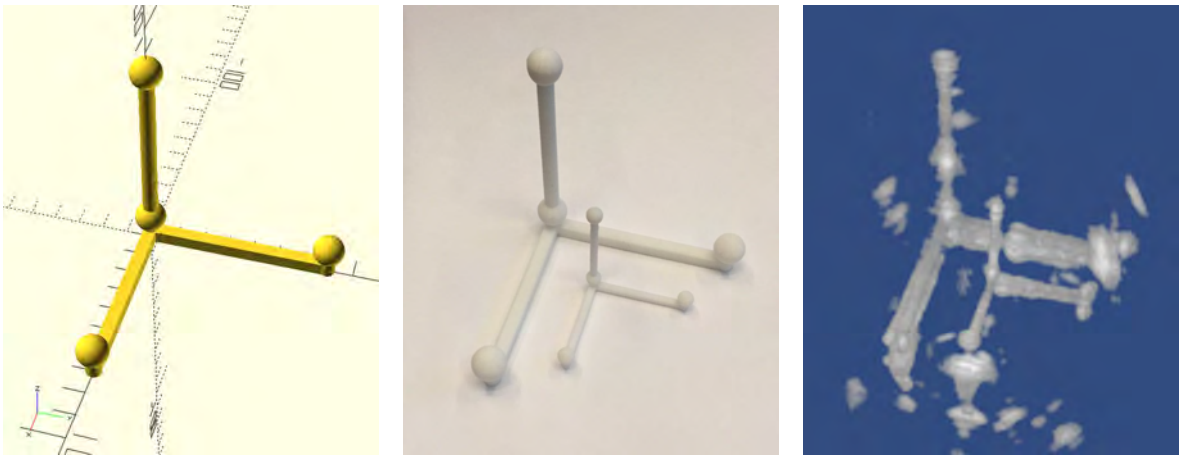
4.1.2 Calibration

To leverage the magnetic sensor rig for volume alignment, we must first identify the relationship between the sensor transform and the resulting volume. Since both the sensor space and the volume space use the same units of measurement, this relationship is a rigid transformation with six degrees of freedom. Given that we had to extend the sensor several inches beyond the back of the probe, the distance between the probe head and the sensor is large enough to result in substantial error if our calibration is not accurate.

One solution to this calibration problem is to simply measure the translational offset with a ruler or caliper and assume negligible rotational error (i.e., assume the sensor has been attached carefully enough that it is well-aligned with the shaft of the probe). However, this is a very imprecise calibration solution, particularly given the lack of clear landmarks on the probe body from which to measure. Furthermore, we would still need to determine the relationship between the probe head and the data itself, as well as the relationship between the magnetic sensor and its point of origin. Thus, we would need to rely on three separate offsets, each of which contains potential error.

A more robust solution is to rotate the probe/sensor assembly about a fixed point, for instance by placing the probe head into a small divit to prevent sliding. Using the knowledge that the probe transforms only contain rotational information, we can use this to compute the translational offset. Again, we must assume we can handle the rotational offset through careful alignment. This solution presents advantages over the first solution in that it reduces human error, and arrives at a best-fit solution from many data points (we can take continuous samples during the rotation process). It also eliminates the need to know the relationship between the magnetic sensor body and its origin. However this still requires that we know the offset from the probe head to the data itself. Thus, we would need to rely on two separate offsets, which is better than the first solution, but still not ideal.

Our approach involves calibrating directly to our data. In this way, we eliminate any in-between steps and remove any assumptions or ambiguous measurements that may result in compounded error. For this approach to work, we must perform three scans and manually



(a) OpenSCAD render (b) 3D prints (1:1 and 1:2) (c) 3D visualization of our scan

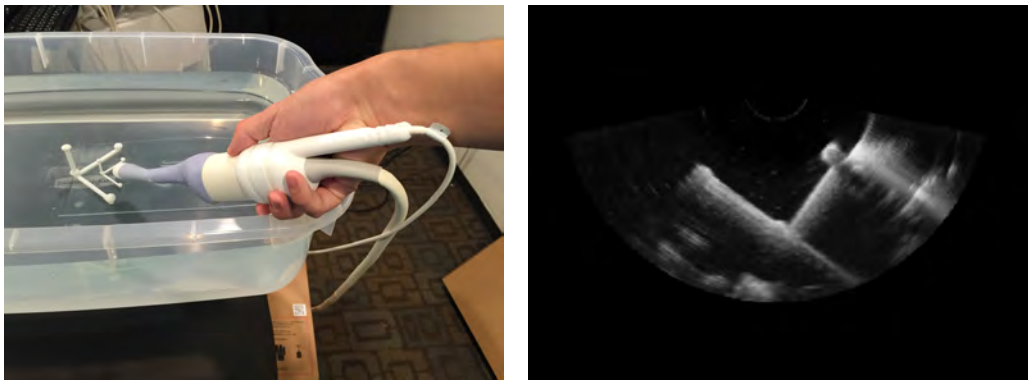
Figure 4.4: Our calibration rig. The balls serve as strong visual features for alignment.

align the resulting volumes. Combining this information with the magnetic sensor data for each scan, we can compute the best fit rigid transformation offset.

To facilitate our calibration process, we designed a special calibration shape with strong visual landmarks. As with our probe attachment, this was constructed in OpenSCAD (Figure 4.4a) and 3D printed (Figure 4.4b). We printed two versions of our calibration rig at different scales to ensure that enough landmarks would be visible in our scans.

To scan the rig properly, the prints were glued to a glass plate and submerged in a water bath, which provides a medium through which to propagate the ultrasound waves. We then acquired five volumes, three to align manually as input to our calibration computation, and two more for testing the resulting solution. We made sure to move the probe substantially, both translationally and rotationally, between scans to ensure a broad baseline for calibration. Figure 4.5a shows our setup during one of these scans. Figure 4.5b shows how the calibration rig appears in a slice through one of our acquired volumes. Figure 4.4c shows a 3D visualization of the volume with our calibration rig embedded within.

Once we acquired the volumes, we loaded them into our manual volume alignment tool (Section 4.3) and easily aligned them based on the strong visual landmarks in the data. It is important to note that anatomical datasets are generally far more difficult to align due to the subtlety of features, and ambiguities introduced by deformation. It is also important



(a) Calibration scanning setup (b) 2D slice of our scan data

Figure 4.5: Calibration setup and corresponding data slice.

to note that this manual step is generally a one-time-only process, as once calibration is determined it should not need to be changed.

To leverage our two datasets (sensor transformations and manual alignment transformations) to arrive at a calibration solution, we must formalize their relationship. Let us treat S_i as the known matrix representing the rigid transformation of the sensor and M_i as the known matrix representing the rigid transformation of our manual alignment for scan i. Let O be the unknown matrix representing the rigid transformation offset between our sensor space and our volume alignment space. Finally, let C be the unknown calibration matrix representing the rigid transformation between the sensor and our volume data. Our goal is to identify C. The relationship between these matrices is as follows:

$$S_i C = O M_i. \tag{4.1}$$

Two scans gives us

$$S_1 C = O M_1 \tag{4.2}$$

and

$$S_2 C = O M_2. \tag{4.3}$$

Taking the inverse of (4.2) and multiplying with (4.3) yields

$$\begin{aligned}(S_1C)^{-1}(S_2C) &= (OM_1)^{-1}(OM_2)\\ C^{-1}S_1^{-1}S_2C &= M_1^{-1}O^{-1}OM_2\\ C^{-1}S_1^{-1}S_2C &= M_1^{-1}M_2\\ S_1^{-1}S_2C &= CM_1^{-1}M_2.\end{aligned} \tag{4.4}$$

If we define $A = S_1^{-1}S_2$ and $B = M_1^{-1}M_2$, we have

$$AC = CB, \tag{4.5}$$

which is a special case of the Sylvester Equation (Sylvester, 1884) $AX + XB = C$, where $C = 0$ (note that this is not the same C as our calibration matrix). There are many known solutions to this equation, including the commonly referenced Bartels-Stewart algorithm (Bartels and Stewart, 1972), as well as others, such as that by Golub et al. (1979). However, our scenario is special in that our A, B, and X (i.e., C) are known to be rigid transformations. In addition to putting constraints on the solution, this also requires us to take special care to enforce that our result is indeed a rigid transformation. Fortunately, Shiu and Ahmad (1989) provide a robust solution to this problem for a similar calibration technique.

The solution presented by Shiu and Ahmad (1989) requires that we set up two of our special case Sylvester Equations in order to avoid an under-constrained system:

$$A_1C = CB_1 \tag{4.6}$$

and

$$A_2C = CB_2. \tag{4.7}$$

To achieve this, we see that we simply need one more aligned volume. From our data,

comprised of S_1, S_2, S_3, M_1, M_2, and M_3, we can obtain

$$A_1 = S_1^{-1} S_2 \tag{4.8}$$

$$B_1 = M_1^{-1} M_2 \tag{4.9}$$

$$A_2 = S_1^{-1} S_3 \tag{4.10}$$

$$B_2 = M_1^{-1} M_3. \tag{4.11}$$

Providing two equations moves us from an under-constrained to an over-constrained problem. To identify a unique solution, Shiu and Ahmad (1989) have us perform a least squares fit, first on the rotational component of the solution, and then on the translational. Applying their solution allows us to arrive at the rigid transformation defining our calibration matrix C.

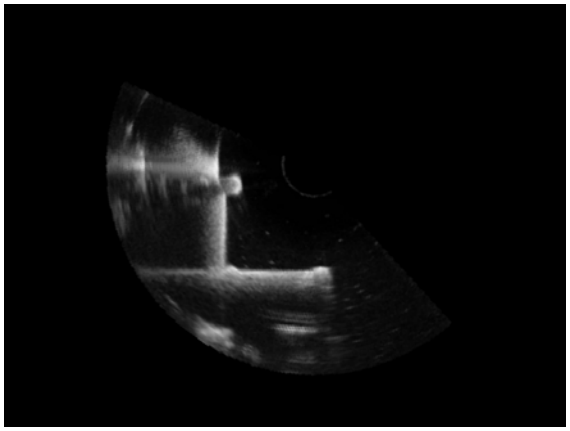
4.1.3 Results

As a quick verification of our results to (4.5), we compared the main translational offset to a ruler measurement and saw that the results were close. Furthermore, a quick inspection of the rotational components of our matrix showed column vectors prominently aligned to standard unit vectors, suggesting that our sensor was closely aligned to the probe axis, as expected (we know we aligned some axis of the sensor with some axis of our volume based on the design of our probe attachment).

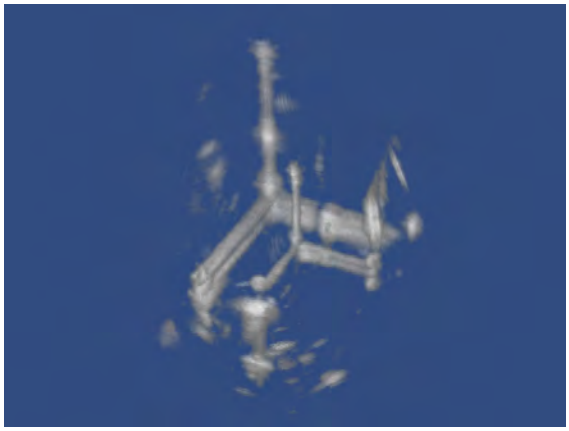
As a more robust test of our calibration matrix, we applied the results to our remaining two unaligned volumes. To place these volumes in the same space as our manually aligned volumes, we also computed O as follows:

$$O = S_1 C M_1^{-1}. \tag{4.12}$$

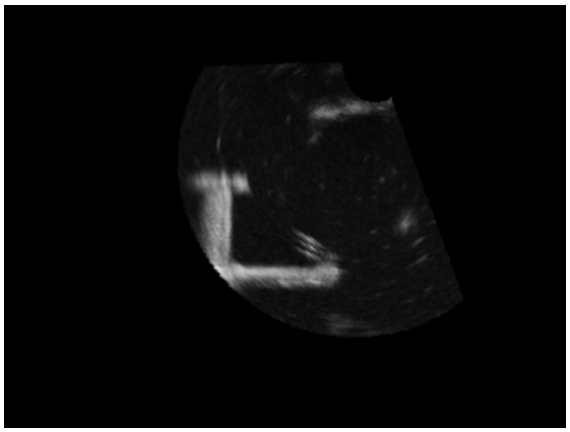
We could then visually inspect the results, where we observed that the automatically aligned volumes were within only a few millimeters of their expected position (Figure 4.6). Based on extensive experience performing manual volume alignments, we can confidently say that



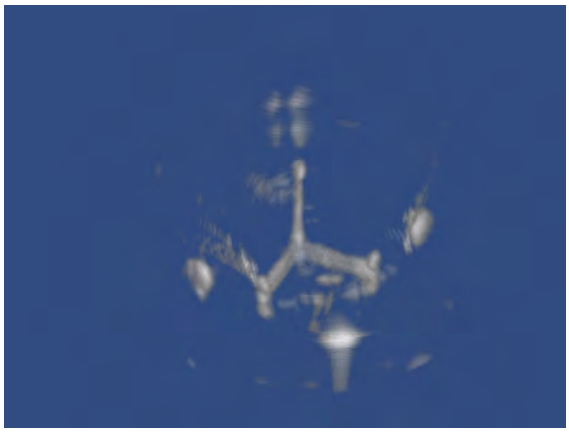
(a) Slice through volume 1



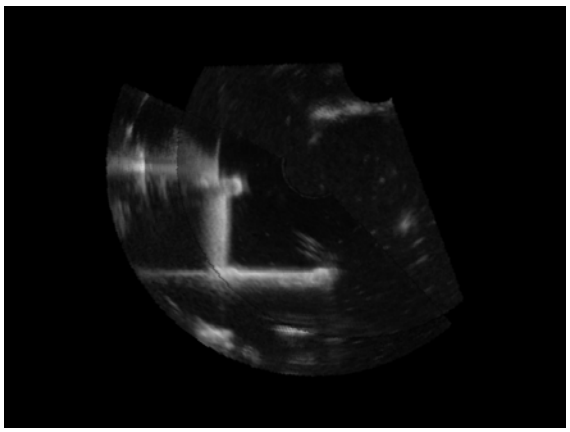
(b) 3D visualization of volume 1



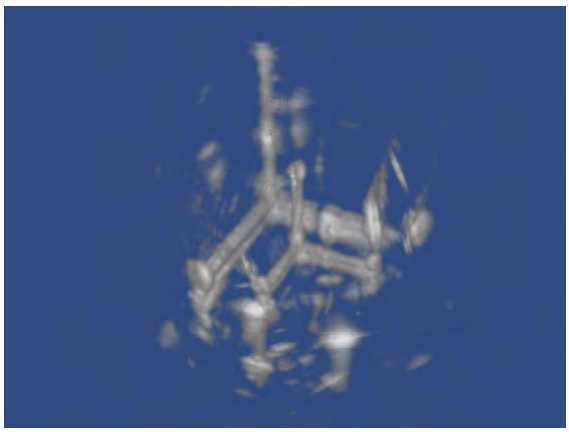
(c) Slice through volume 2



(d) 3D visualization of volume 2



(e) Slice through both volumes



(f) 3D visualization of both volumes

Figure 4.6: A visual analysis of two volumes aligned with our calibrated magnetic-tracking solution. The first two rows show each volume in isolation. The bottom row shows the two overlayed. We can see slight evidence of the imperfect alignment, but overall the results are good.

this tracking-based alignment provides a very helpful starting point. Achieving this kind of rough placement manually is generally the most time-consuming part of the alignment process. Thus, we anticipate significant time savings with our magnetic tracking solution. As mentioned previously, our solution requires far fewer scans than that of Detmer et al. (1994) and a much simpler calibration rig than that of Pagoulatos et al. (2001).

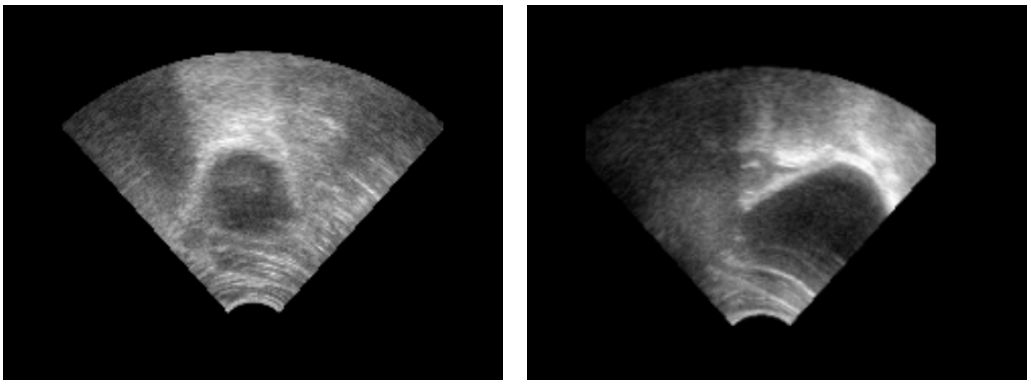
4.2 Automatic Alignment

While motion tracking provides a reliable starting point to the alignment process, additional work is required to arrive at a final solution. Furthermore, some scenarios prevent the use of motion tracking, in which case another automated solution is desirable. We present an automatic alignment algorithm to aid the volume registration process. Our method is similar to that described by Collignon et al. (1995), and involves refining a six-degree-of-freedom search, beginning at a course level, identifying the best fit, and refining this solution. Ultimately this results in a search tree where the branching factor is n^6, where n is the number of samples taken in each dimension. For $n = 3$, this produces a branching factor of $3^6 = 729$.

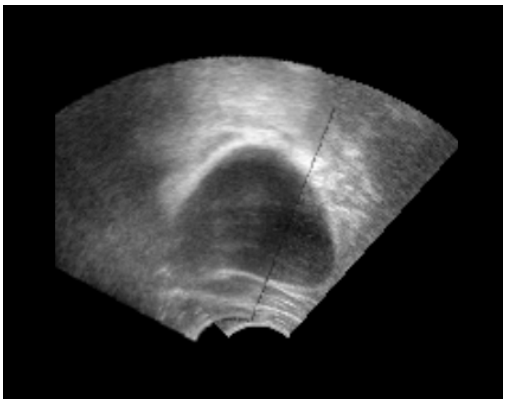
4.2.1 Base Error Metric

To establish which branch to traverse, we must establish a metric for rating each sample. Our solution uses intensity differences as the main error metric. Given an alignment, we iterate across the voxels of one volume and find the differences between the values in the corresponding location of the second volume. If the values are identical, this means there is 0 error and we have a perfect match. As the difference increases, our confidence in the alignment decreases.

One problem with this approach is that alignments with very little overlap have a high probability of matching well by chance. For example, if we consider an alignment where the two volumes only overlap by one voxel, and both happen to have the same value, this would appear to be a perfect alignment. To get around this problem, we enforce a minimum



(a) Volume A (b) Volume B



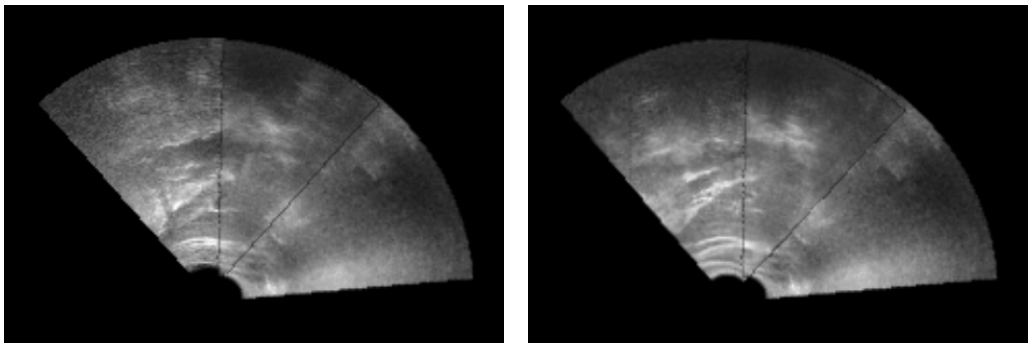
(c) Automatic alignment

Figure 4.7: A demonstration showing fully automatic alignment of two volumes.

number of overlapping voxels as a percentage of total voxels. Thus, our error metric is

$$error = \begin{cases} \frac{1}{n} \sum_i^n |v_{ai} - v_{bi}|, & \text{if } n \geq mt; \\ \infty & \text{otherwise,} \end{cases} \tag{4.13}$$

where v_a and v_b are two volumes, n is the number of overlapping voxels, m is the total number of voxels in a single volume, and $0 \leq t \leq 1$ is the threshold value. Figure 4.7 shows how this method can be used to achieve a fully automatic alignment of two volumes. Figure 4.8 shows how this method can be used effectively to improve upon an expert's ability to manually perform the alignment task.



(a) Manual alignment (b) Automatic refinement

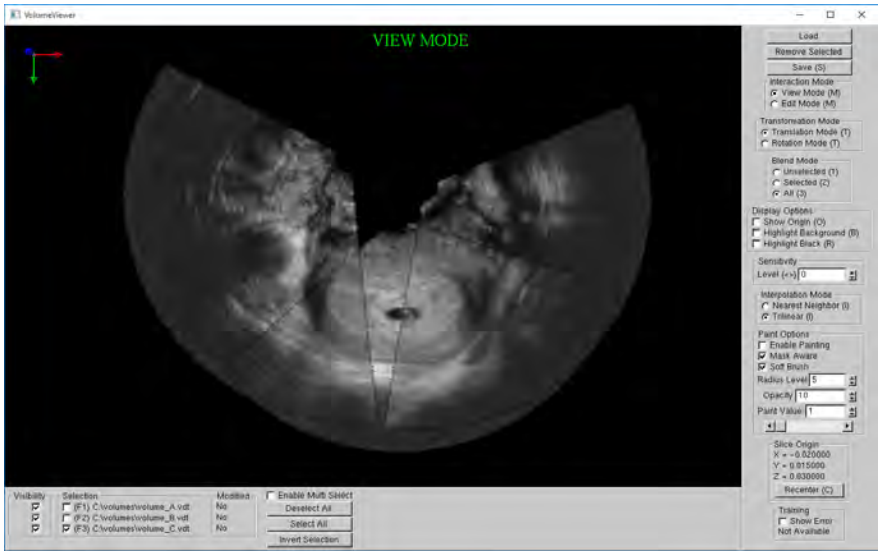
Figure 4.8: A comparison of manual alignment (left) and automatic refinement (right). A medical expert in the field of ultrasound proclaimed that the automatic alignment produced a far superior result.

4.3 Manual Alignment

Even with our automatic alignment process, there are many situations where user interaction is still necessary. For example, scan sessions in which our magnetic tracking system is not available will often require medical technicians to perform manual alignments. While our alignment algorithm performs well in many of these cases, excessive artifacts can interfere with good alignment, and we must rely on expert knowledge of the known anatomy in order to perform a proper alignment.

To this end, we developed an interactive alignment tool (Figure 4.9) that enables users to work within 2D slicing planes to perform the alignment process. Our tool provides a number of options to facilitate the alignment process. Its core features include the following:

- Volume translation and rotation. These are the main controls for moving a volume or volumes to produce an alignment.
- Slice view axis and position control. With these controls the user can quickly switch between slice views to see how their alignment appears in each dimension.
- Selection controls. The user can choose to select a single volume at a time, or multiple at once if a group must be adjusted together. Options are available to invert the selection, deselect all volumes, or select all volumes.



(a) Main window showing 2D slices

(b) 3D view

Figure 4.9: Screen shots of our volume alignment tool. Here, we see three separate volumes of an early-stage pregnancy aligned together.

- Blending options. Sometimes a user needs to view just the selected volume, while other times it is beneficial to view all volumes blended together. Hot-keys allow for quick toggling between the selected volume(s) and the non-selected volume(s).

- Interpolation options. The user can choose between nearest-neighbor interpolation and trilinear interpolation.

- Visibility controls. Often it is desirable to hide some volumes (i.e., disable their visibility) when many volumes have been loaded.

- Translation and rotation sensitivity. High sensitivity allows for quick, broad adjustments, while low sensitivity allows for finer control.

- Automatic backups. Due to the trial-and-error nature of volume alignment, providing automatic backups makes it easy to revisit an earlier solution.

In addition to alignment tools, we also provide basic volumetric painting tools, which allow the user to remove any artifacts that might show up as a result of the stitching process (Chapter 5). Users can control their brush size, color, and opacity. We also provide a feature that prevents painting outside the valid data-regions so the user can paint confidently right up to the edges of the volume. Furthermore, users can carve away portions of a volume if they are undesirable; for example, when multiple volumes contain redundant data. Figure 4.10 illustrates the effects possible with painting.

Since our datasets are volumetric, we also provide a 3D view. This provides a more intuitive perspective to the user, as they can immediately get a sense for the scope of data with which they are working. Our data is visualized with a custom depth-sorted volumetric particle engine that allows for a number of useful features:

- Show slice plane. One of the most useful features of the 3D view is the ability to visualize the placement and orientation of the current slice in the context of the complete volume.

(a) View of the slicing plane in which the painting was performed

(b) Slicing plane moved back to reveal the volumetric nature of our brush strokes

Figure 4.10: Experimenting with volumetric painting. We can see the effects of different brush strokes, from the very thin squiggle, to the prominant brightened region near the peak of the left-most volume. We have also performed carving of the two side volumes to better view the uterus data of the central volume.

(a) Min threshold = 1 (b) Min threshold = 75 (c) Min threshold = 150

Figure 4.11: Changing the 3D view minimum value threshold.

- Particle downsampling. Since rendering large numbers of particles can be computationally demanding, we allow users to downsample the datasets to suit their needs. This allows users to leverage the power of the 3D view regardless of their computational resources.

- Particle Size. This allows users to control the particle size to their liking, based on the anatomy being viewed.

- Opacity. This controls the opacity of each particle. Lowering the opacity allows users to easily see inside a volume.

- Value-based opacity. With this enabled, the opacity of a given particle is determined by multiplying the overall opacity with that particle's value. This has the effect of making darker regions more transparent than brighter regions, which is useful in ultrasound data where darker values typically represent voids or shadows.

- Minimum and maximum thresholds. These values allow the user to interactively hide data below or above these thresholds. Thus, if there is a bright object users would like to visualize, they can remove everything below that value. This is how we arrived at the visualizations in Figure 4.6. Figure 4.11 demonstrates the effects of increasing the minimum threshold.

In addition to the features mentioned, we also experimented with providing a full six-degree-of-freedom interaction modality within the volume alignment tool. To achieve this,

Figure 4.12: Our Polhemus magnetic tracker. A smaller, cheaper alternative to the Ascension tracker (Figure 4.2).

we employed a magnetic tracking system similar to the one we used for our tracking solution. We incorporated the Polhemus system (Figure 4.12) to allow users to move volumes freely as if they were holding them in their hands. While this provided a very tactile approach, we discovered that small incremental adjustments through keyboard inputs provided much more refined and consistent control.

Our volume alignment tool also provides options to help with training, by providing a numeric score for feedback, as well as error vector visualiation in our 3D view. Section 4.4 describes these evaluation features in more detail.

4.4 Evaluation

To evaluate the performance of both our algorithm and users alike, we use our synthetic volumes as test data and apply a quantitative scoring metric.

4.4.1 Ground Truth

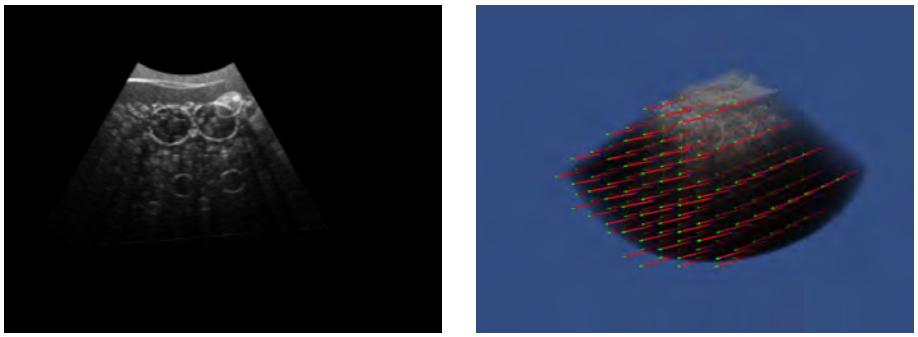
One of the key motivations for generating synthetic data is the availability of ground truth. Since we know exactly how the probe is positioned for a given dataset, we can use this as a definitively "correct" alignment solution. Figure 4.13d shows the results of aligning two volumes based on this ground truth.

4.4.2 Scoring

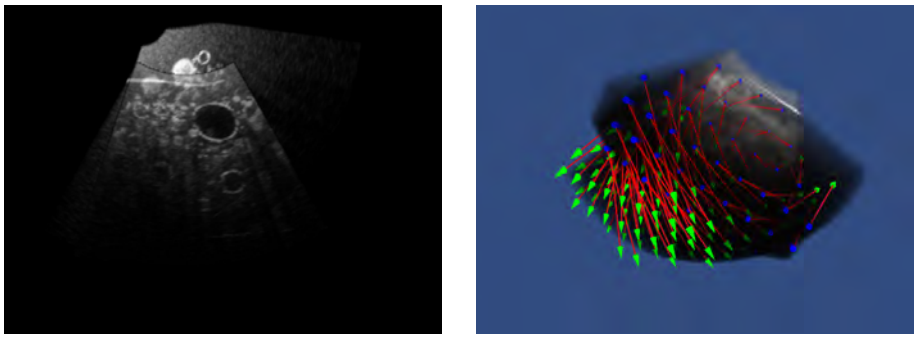
Given the ground truth solution, we must determine a measure that reasonably evaluates how close an alignment is to this correct solution, taking into account both translational and rotational error. We introduce a score as a distance measure. A value of 0 indicates a perfect alignment. Greater values denote poorer solutions.

To weight translational and rotational error equivalently, we base our distance measure on the Euclidean distance between corresponding points. It is important to note that for this to work as intended, the points we consider must be representative of the size and shape of the volume we are aligning. Thus, we sample only within the valid data region of our volume and compute the average Euclidean distance.

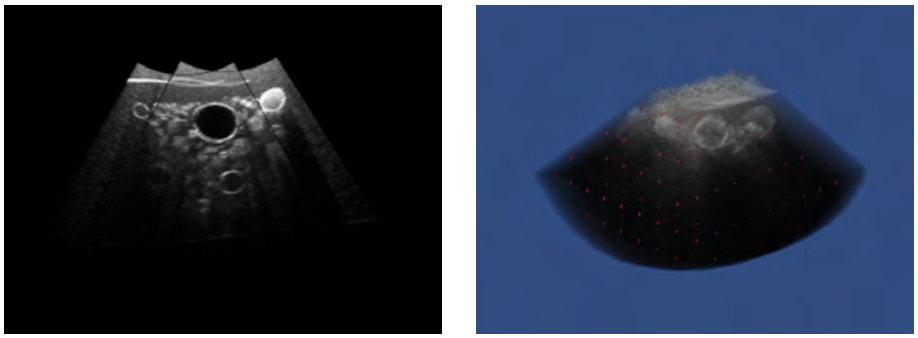
Scoring can be used interactively within our volume alignment tool (Section 4.3) to provide guidance during user training. If users choose to, they can view their score in real time as they make adjustments. If their score increases, they know they have made an error. If their score decreases, they know they are on the right track. Error vectors are available in the 3D view to help users better understand how their alignment can be improved. These features can be disabled such that a training session can be performed without guidance, and then displayed at the end to assess the quality of the final alignment. Figure 4.13 provides examples of what a user might see with different alignments, as well as the corresponding scores.



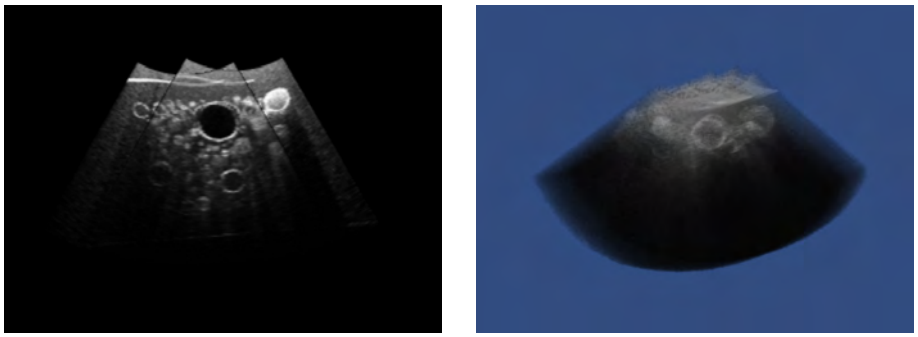
(a) Unaligned volumes (Error = 0.017157)



(b) Extreme misalignment (Error = 0.034249)

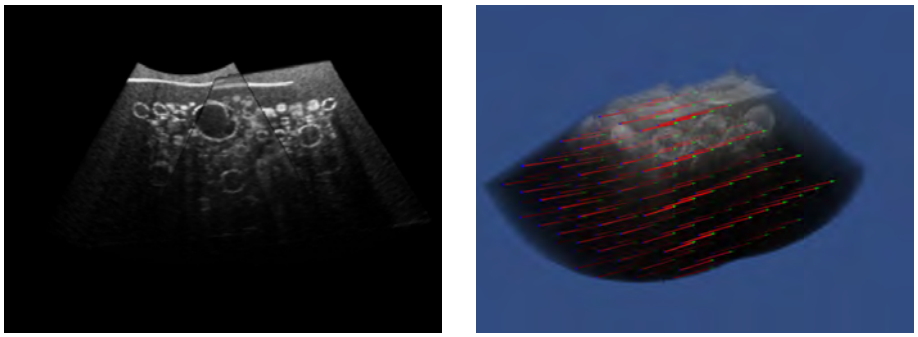


(c) Expert manual alignment (Error = 0.001079)

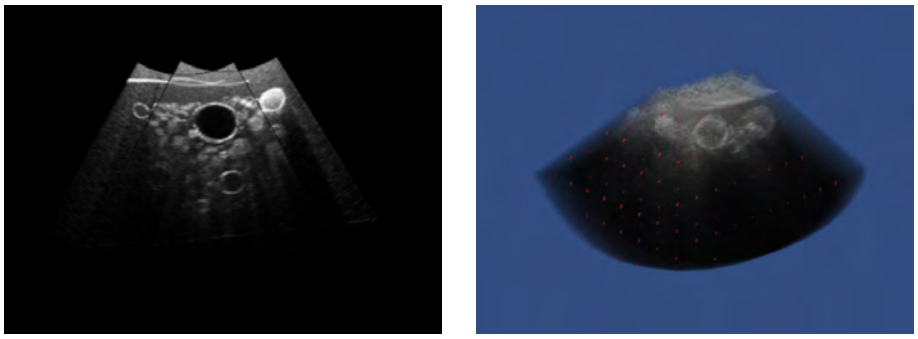


(d) Ground truth (Error = 0)

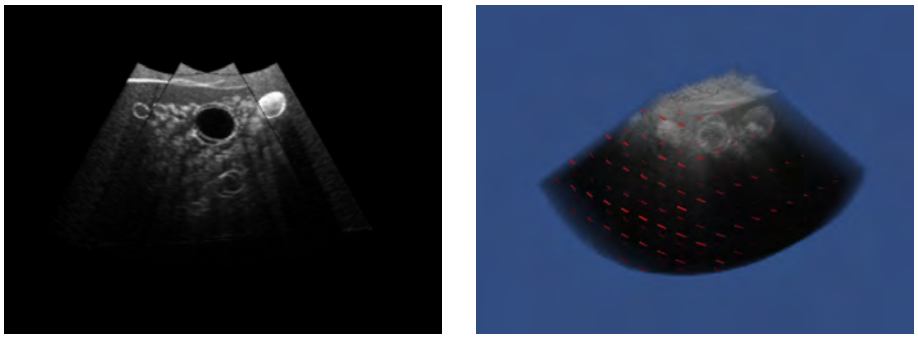
Figure 4.13: Examples of error vectors and scores for different alignments.



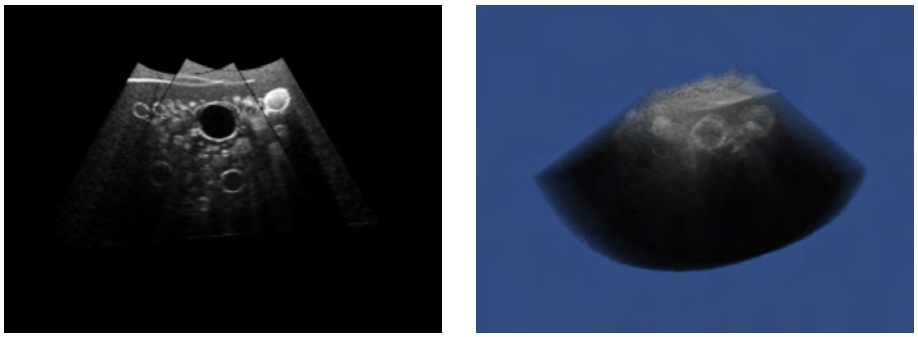
(a) Novice manual alignment (Error = 0.021358)



(b) Expert manual alignment (Error = 0.001079)



(c) Algorithm alignment (Error = 0.002815)



(d) Ground truth (Error = 0)

Figure 4.14: Visual comparison of alignment solutions.

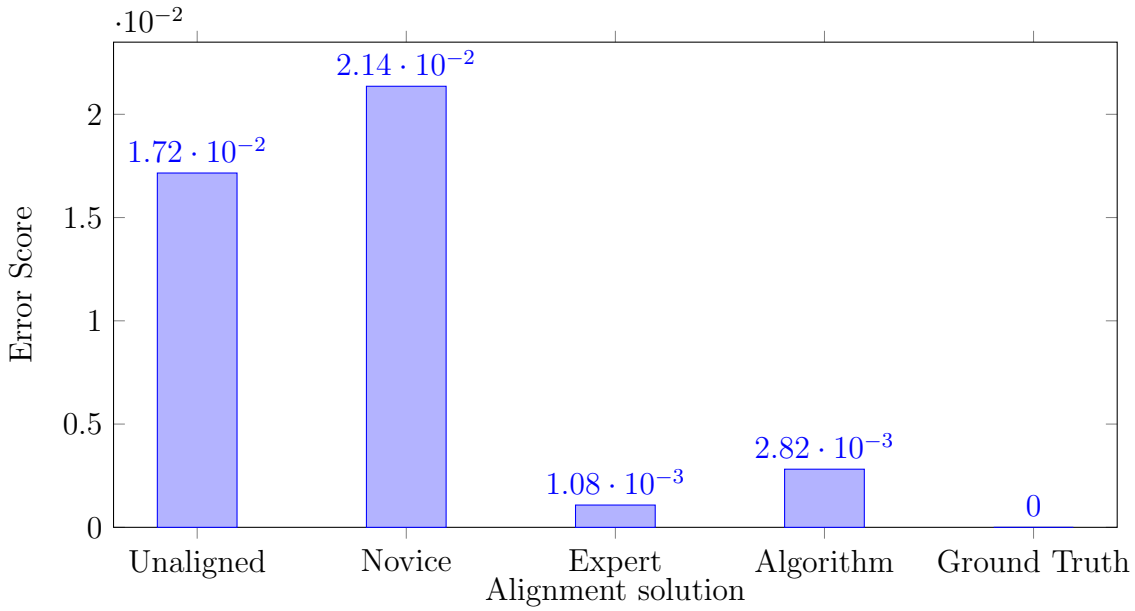


Figure 4.15: Quantitative comparison of alignment solutions. While our algorithm alignment is not perfect, it can perform far better than a novice user.

4.5 Results

We tested both our automatic solution, as well as several manual alignments using our synthetic data and evaluation metric. Our results (Figure 4.14 and Figure 4.15) show that our scoring system accurately reflects the difference between a novice and expert user, and that our automatic solution provides results on par with the expert.

Based on these results, we feel that our evaluation metric provides a reasonable method to compare our automatic solution to manual alignments, and also serves as a useful training tool. Prior to the introduction of this solution, alignments had to be gauged subjectively, and thus there was no solid feedback to enable a user to perform better. With our solution, users can experiment with different alignments and see how their score changes. This real-time feedback allows the user to quickly analyze the data to see what makes for a good alignment. These skills can then be applied to real world data where known alignments are not available.

Combined with our magnetic tracking solution, which provides a reliable initial alignment, our automatic and manual alignment tools provide a robust solution to the volume

registration problem.

CHAPTER 5

Volume Stitching

Once an alignment has been determined with confidence, the next step to producing a new dataset is merging the volumes together. This is achieved through a novel image stitching algorithm that we have developed. Our solution presents a new approach to this problem, by applying Boolean operations to identify regions of overlap, and then computing smooth transition functions within these regions. We developed two techniques for defining these functions: one based on distance maps, the other based on a functional minimization. While our solution is fully automatic, we expose an optional blending parameter to enable advanced control over the transition appearance. Figure 5.1 provides a simple 2D example that we will consider as we discuss the different aspects of our algorithm.

5.1 Region Identification

The main idea behind our algorithm is the notion that there are three important regions to consider for a given volume:

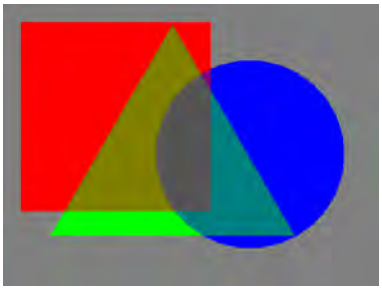
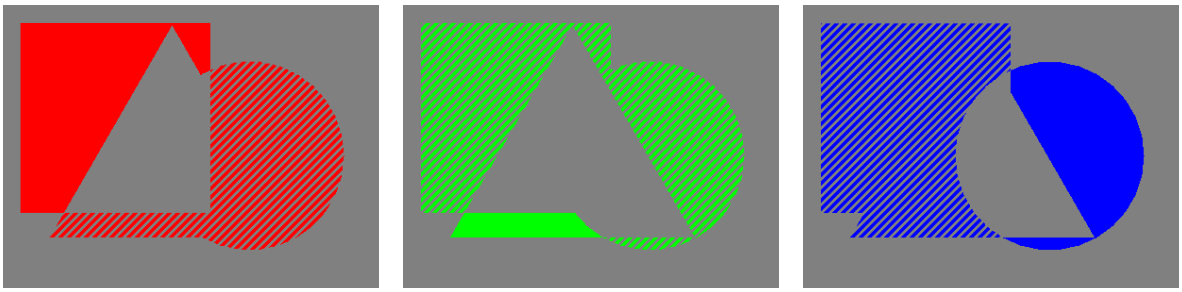


Figure 5.1: Simple 2D example illustrating three objects of different shapes and color. Here, they have been naively blended through a simple average.



(a) Square regions (b) Triangle regions (c) Circle regions

Figure 5.2: Pure regions (solid) and empty regions (striped) for the square (a), triangle (b), and circle (c).

1. *Pure Regions.* Regions that contain data only from the current volume.

2. *Overlap Regions.* Regions that contain data from the current volume as well as others.

3. *Empty Regions.* Regions that contain no data from the current volume, only data from other volumes.

To generate each of these regions, we use Boolean operations on volumetric masks in the output space. We generate for each volume a binary mask where all voxels inside the volume have value 1 and all values outside the volume have value 0. We can then apply the Boolean operations of subtraction ($1-1=0$, $1-0=1$, $0-1=0$, $0-0=0$) and union ($1+1=1$, $1+0=1$, $0+1=1$, $0+0=0$). A pure region P_i for a given volume with mask m_i is obtained by subtracting the union of all other volume masks:

$$P_i = m_i - \left(\bigcup_{j=1, j\neq i}^{n} m_j \right). \tag{5.1}$$

An empty region E_i for a given volume is obtained by subtracting the current volume mask from the union of all other volume masks:

$$E_i = \left(\bigcup_{j=1, j\neq i}^{n} m_j \right) - m_i. \tag{5.2}$$

Overlap regions do not need to be computed explicitly as the boundaries of these regions are already contained in the boundaries of the pure and empty regions (Figure 5.2).

We can regard the three regions like the three main areas of a shadow. The pure regions are like the umbra of a shadow (pure blackness), the overlap regions are like the penumbra of a shadow (partial occlusion), and the empty regions are like the unshadowed regions. Hence we have named our solution "Penumbra".

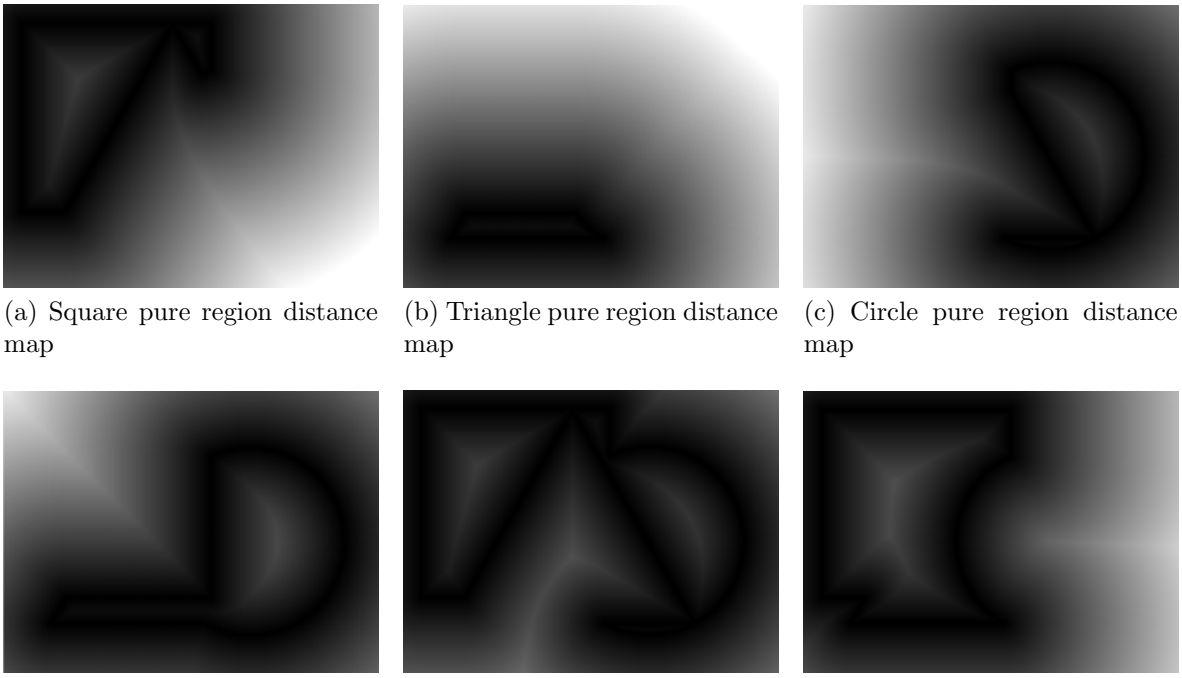
The challenge is to determine the proper contribution from each volume at each output voxel. The pure and empty regions are trivial to solve as they contain 100% and 0% contributions, respectively. The remainder of this chapter focuses on how to assign contribution weights within the overlap regions, once they are identified, using knowledge of the pure and empty regions to inform this decision. Like a penumbra, we would like a smooth gradation from pure to empty.

5.2 Distance Map Approach

One approach to defining a smooth transition from pure to empty regions is to consider the distances to each region. We would like to approach 100% as the distance to a pure region approaches 0, and 0% as the distance to an empty region approaches 0. We can precompute these distance maps for the pure region and empty region of each volume, producing a total of $2n$ distance maps. We use the algorithm of Saito and Toriwaki (1994) to efficiently compute these distance maps across the entire output space. The input is a region mask generated from the first step of our algorithm and the output is a monochrome image with each voxel value representing the distance to the nearest non-zero mask value. Figure 5.3 provides a visualization of these maps.

5.2.1 Weight Determination

Once we have precomputed the distance maps, we begin the traversal stage of the algorithm, where we visit each output voxel and determine the contribution (weight) from each volume. At each voxel, we first determine which volumes overlap this region by checking the associated full-volume masks. Next, we compute the individual scale factor s_i for each contributing volume. That is, if we consider a single volume in isolation, what is the interpolation value



(a) Square pure region distance map

(b) Triangle pure region distance map

(c) Circle pure region distance map

(d) Square empty region distance map

(e) Triangle empty region distance map

(f) Circle empty region distance map

Figure 5.3: Each column corresponds to the respective shape in Figure 5.2 (square, triangle, and circle). The top row shows the distance maps for pure regions. The bottom row shows the distance maps for empty regions.

between its pure region and its empty region? With p_i representing the nearest distance to the pure region of volume i, and e_i representing the nearest distance to the empty region of volume i, we have

$$s_i = \frac{e_i}{p_i + e_i}. \tag{5.3}$$

Equation (5.3) produces a linear ramp between the pure and empty regions. Notice that as e_i approaches 0, s_i approaches 0, and as p_i approaches 0, s_i approaches 1, as we would like.

Once we have computed the individual scale factors for each of the k contributing volumes, we can arrive at the final weight value w_i for each volume:

$$w_i = \frac{s_i}{\sum_{j=1}^{k} s_j}. \tag{5.4}$$

This ensures that the total weighted sum $\Sigma_{i=1}^{k} w_i = 1$. To arrive at the final voxel value v_f,

we use the weighted average of all contributing voxels:

$$v_f = \sum_{i=1}^{k} w_i v_i. \tag{5.5}$$

The results of this solution can be seen in Figure 5.13c, Figure 5.13g, and Figure 5.13k. While better than existing techniques (see Figure 5.13), the presence of crease-like artifacts leaves room for improvement.

5.3 Functional Approach

While our distance map approach works well, we can produce even smoother transitions by solving a functional minimization problem. Specifically, we apply the discrete membrane spline regularization functional shown below, as described by Terzopoulos (1986), with boundary conditions set to 1 within pure regions and 0 within empty regions. For 2D images this takes the form

$$F(u) = \sum_{1 \le i,j \le n} \left[(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 \right], \tag{5.6}$$

and for 3D volumes it takes on the form

$$F(u) = \sum_{1 \le i,j,k \le n} \left[(u_{i+1,j,k} - u_{i,j,k})^2 + (u_{i,j+1,k} - u_{i,j,k})^2 + (u_{i,j,k+1} - u_{i,j,k})^2 \right]. \tag{5.7}$$

We compute the gradient of these equations in order to apply an iterative steepest-descent minimization technique. The result is a smooth membrane minimizing the squared gradient magnitude. We perform this minimization step independently for each volume to be stitched. Figure 5.4 shows the inputs and outputs of the functional minimization for each object. We optimize our functional minimization process by processing only the bounding box containing our region of interest.

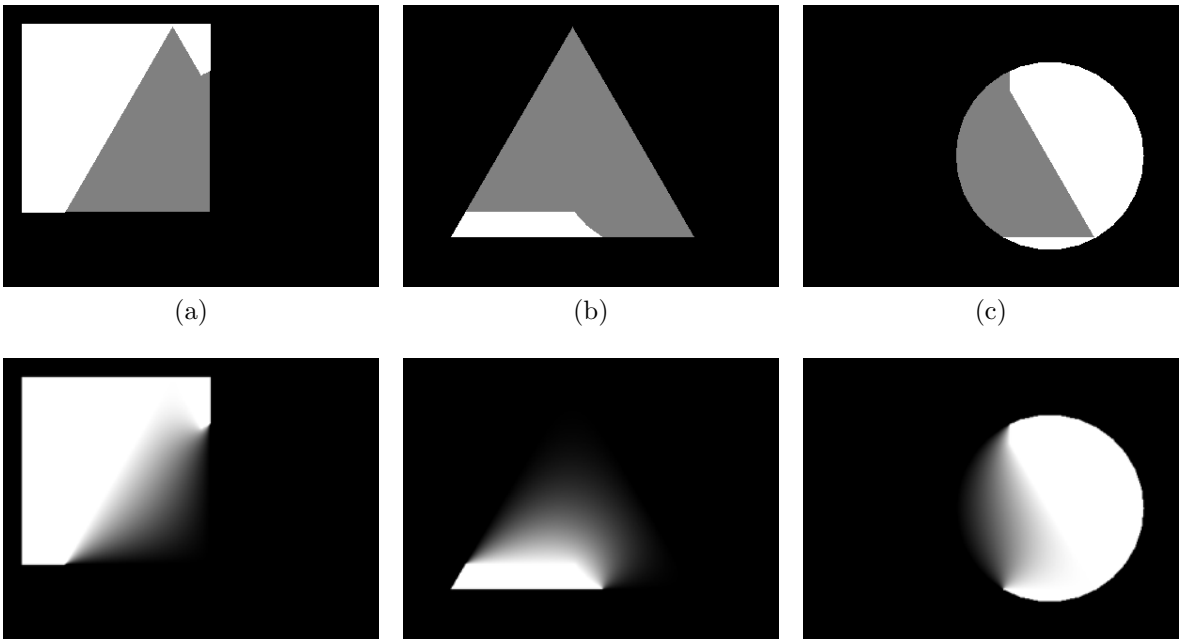


Figure 5.4: Functional inputs (top row) and outputs (bottom row) for each shape (square, triangle, circle, respectively). White represents pure regions, while black represents empty regions.

5.3.1 Weight Determination

As with our distance map solution, we must determine the contribution from each volume. At each voxel, we look up the the scale value s_i of our functional output for each of the k contributing volumes. This value s_i is the only difference between our functional approach and our distance map approach. Once determined, we arrive at the final weight (5.4) and voxel values (5.5) as before.

5.3.2 Optimization

While our functional-based approach produces smoother results than our distance-based approach, it exhibits poor efficiency, particularly on 3D data (Figure 5.5). To improve this aspect of the solution, we implemented a multiresolution optimization (Terzopoulos, 1988), whereby we solve the functional at a low resolution first, then scale up these results to initialize a higher resolution minimization. This can be repeated until the final resolution is

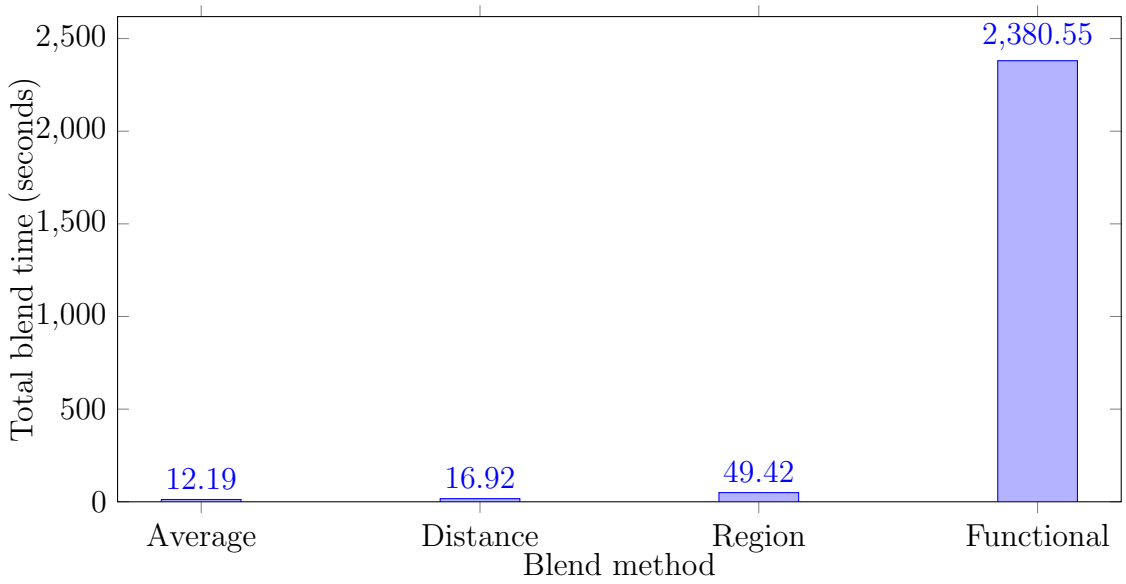


Figure 5.5: Total blend time vs the blend method. The total time includes all pre-processing steps and the final voxel traversal phase for a merge of three volumes. The functional method stands out significantly.

achieved. We focus now on 3D datasets, where this optimization is most relevant.

Several important considerations must be taken into account during this process. First, we must properly combine the output of our low resolution results with our high resolution region information during each level transition. Specifically, we must scale up our low resolution solution to the resolution of the current level, and scale down our high resolution region data to the resolution of the current level, and then merge the two.

For scaling up, we use trilinear interpolation to maintain smoothness. For scaling down, we use a simple nearest neighbor solution, since we want to avoid interpolation altogether to preserve the binary significance of our region data.

To merge the two, we first clamp the results of our up-ressed volume to (0, 1) since 0, and 1 are reserved for empty and pure regions, which we do not want to inherit from the low resolution representation. Instead, we extract such voxels from our down-sampled volume and insert them into the up-ressed volume. In other words, we are reintroducing pertinent region information that was lost during down-sampling at the lower level.

To process each level in an equivalent manner, we must take care to normalize our

functional output value for the given resolution. This is important for determining our termination criteria, which should not be biased by the number of voxels processed at a given level. For a resolution scale factor r in each dimension, and d dimensions, we can modify (5.7) as follows:

$$F(u) = \sum_{1 \le i,j \le n} r^d \left[\left(\frac{u_{i+1,j} - u_{i,j}}{r} \right)^2 + \left(\frac{u_{i,j+1} - u_{i,j}}{r} \right)^2 + ... \right]. \tag{5.8}$$

We multiply each summation by r^d since we have this many more voxels to consider. We divide each component by r since everything is scaled by r. Simplifying, we have

$$\begin{aligned} F(u) &= \sum_{1 \le i,j \le n} r^d \left[\frac{(u_{i+1,j} - u_{i,j})^2}{r^2} + \frac{(u_{i,j+1} - u_{i,j})^2}{r^2} + ... \right] \\ &= \sum_{1 \le i,j \le n} r^d \left[\frac{(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + ...}{r^2} \right] \\ &= \sum_{1 \le i,j \le n} \frac{r^d}{r^2} \left[(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + ... \right] \\ &= \frac{r^d}{r^2} \sum_{1 \le i,j \le n} \left[(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + ... \right] \\ &= r^{d-2} \sum_{1 \le i,j \le n} \left[(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + ... \right]. \end{aligned} \tag{5.9}$$

Thus, if we would like our functional value to provide the same value at any resolution, we must divide the result by r^{d-2}, or equivalently multiply our termination criteria by r^{d-2}. For 2D data this means multiplying our termination criteria by $r^{2-2} = r^0 = 1$, and for 3D data this means multiplying our termination criteria by $r^{3-2} = r^1 = r$.

Figure 5.6 shows the convergence of the functional minimization using different numbers of resolution levels. Note that when multiple levels are used, it appears that the functional value increases at each level, despite normalization. This is because we introduce more high-resolution detail at each level, adding complexity to the data, and therefore increasing the functional value. If we were to simply increase the resolution without reintroducing the lost high-resolution data, we would see something closer to the monotonically decreasing function

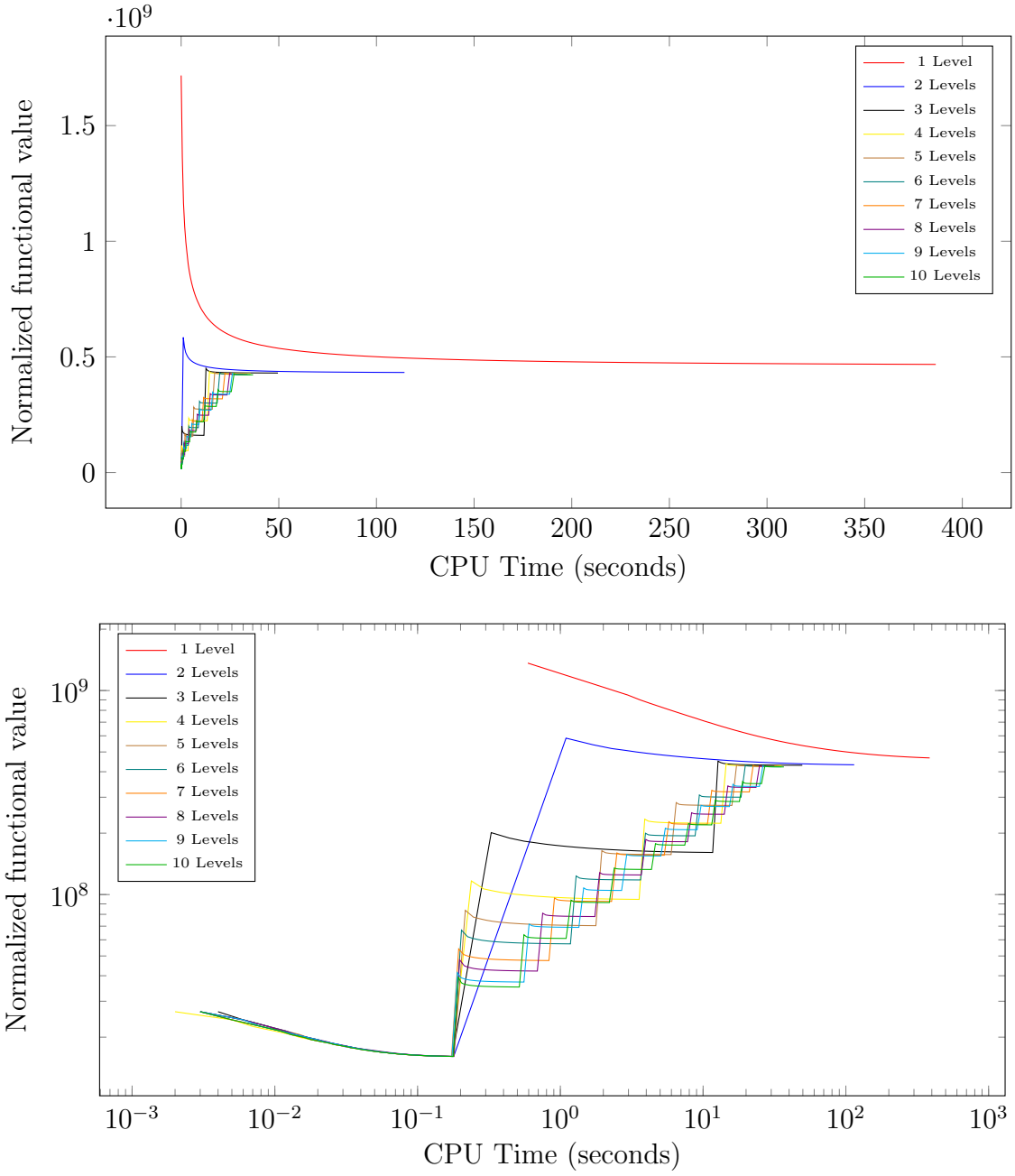


Figure 5.6: Normalized functional value vs CPU time for different choices of the number of levels. Each plot starts after the first iteration (hence the noticeable difference when the number of levels is one — the first iteration takes much longer to complete at high resolution). Each plot ends at the point where the termination criteria was met. The top graph uses a standard linear scale. The bottom graph presents the same data with a log scale for closer inspection.

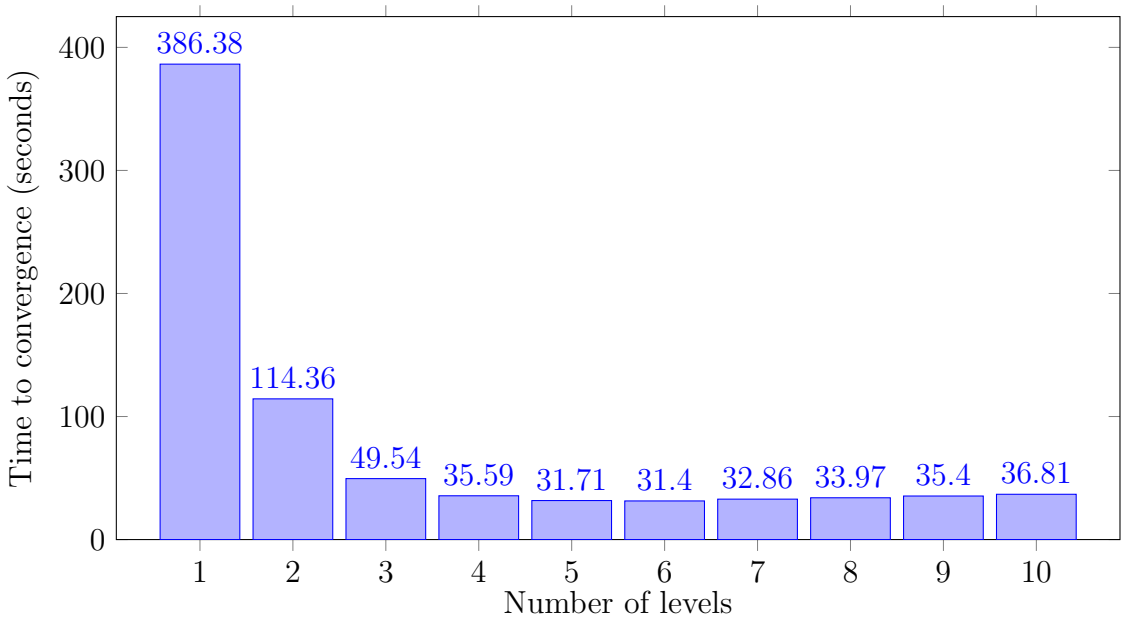
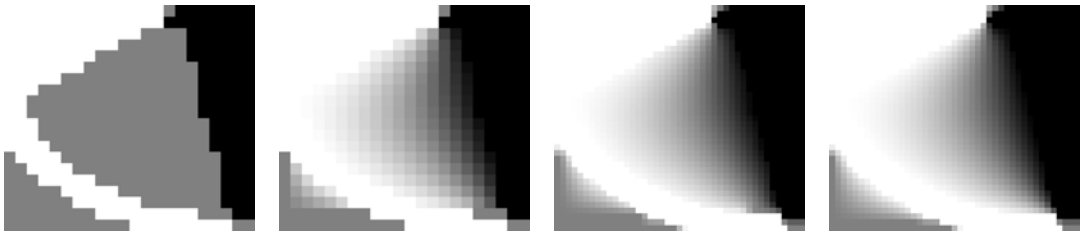


Figure 5.7: Convergence time vs the number of levels. We can see from this that six levels produces the optimal performance.

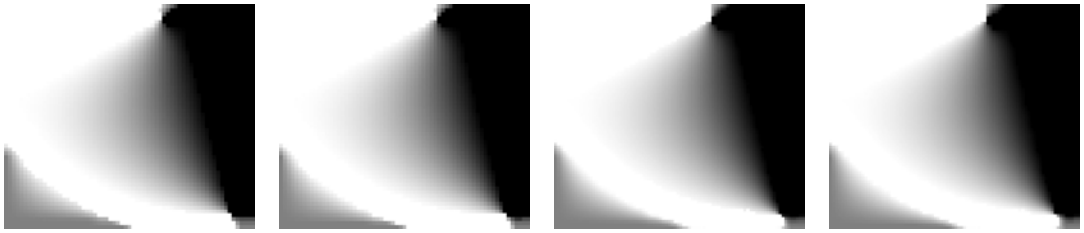
one might expect.

Figure 5.7 shows more concisely the performance difference for different numbers of levels. Based upon these results, we see that six levels provides the optimal performance. Figure 5.8 shows visually what this process looks like at each level of a six-level optimization for a given blend region. Figure 5.9 provides a comparison, demonstrating the progress made by a one-level solution in the amount of time it took for the six-level solution to complete.

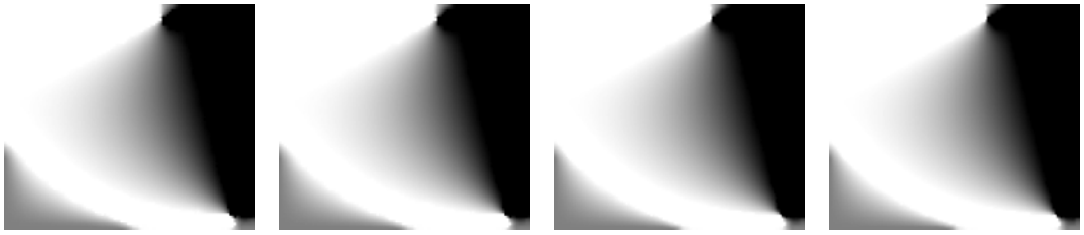
Figure 5.10 adds our optimized solution to the comparison graph. While our multiresolution approach is still slower than our distance-based region approach, it falls within the realm of being practical, providing substantial improvement over the non-optimized functional approach. Note that increasing the termination criteria can further speed up performance (to the point of being even faster than the distance-based region approach), though this will produce poorer results. The termination criteria used in our graphs was chosen to offer a good compromise between speed and quality.



(a) Level 1 input (b) Level 1 output (c) Level 2 input (d) Level 2 output

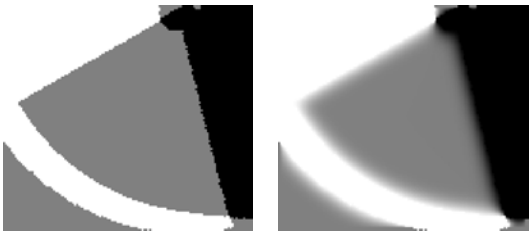


(e) Level 3 input (f) Level 3 output (g) Level 4 input (h) Level 4 output



(i) Level 5 input (j) Level 5 output (k) Level 6 input (l) Level 6 output

Figure 5.8: Demonstration of multiple resolution level optimization. Each input is derived by up-ressing the previous level's output and adding the high resolution region information.



(a) Input (b) Progress

Figure 5.9: Progress made by a one-level solution in the amount of time it took for the six-level solution to complete.

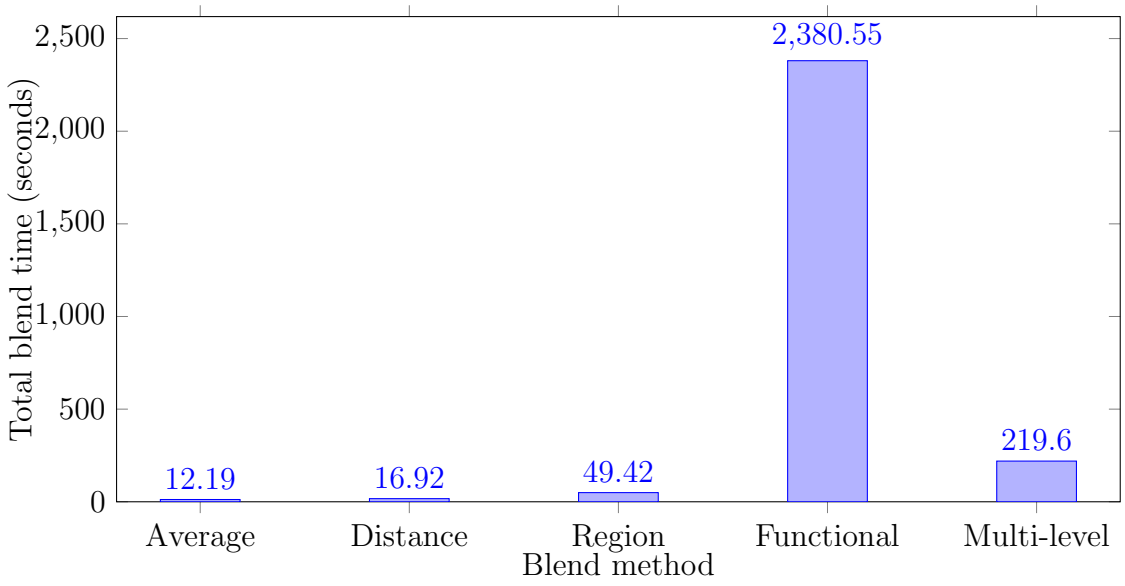


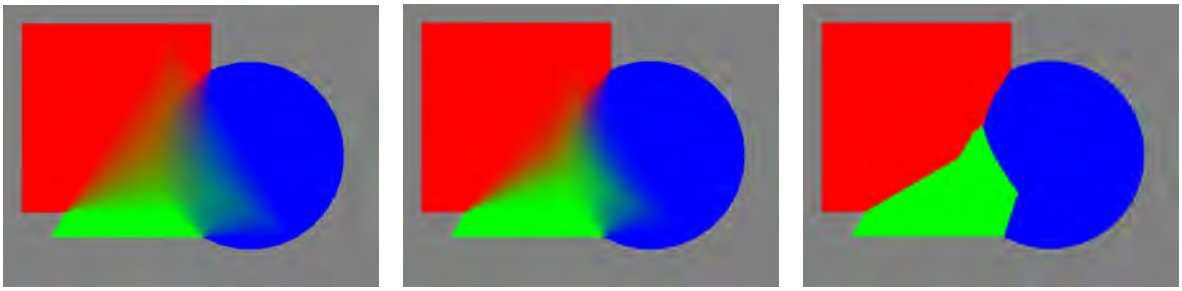
Figure 5.10: Total blend time vs the blend method, including our multiresolution approach (far right). Our optimized solution provides significant gains over the unoptimized version (second from the right).

5.4 Parameterization

We can expose some user control over the blend appearance by introducing a blending parameter b. Our goal with this parameterization is to provide control over the relative width of the blending region. At one extreme, we would have a maximal blend across the entire overlap region, while at the other extreme we would have hard edges reminiscent of Voronoi diagrams. Ideally, somewhere in between, we would achieve a transition that sacrifices blend distance for smoothness. We arrived at such a parameterization by using b as an exponent on s. This modifies (5.4) as follows:

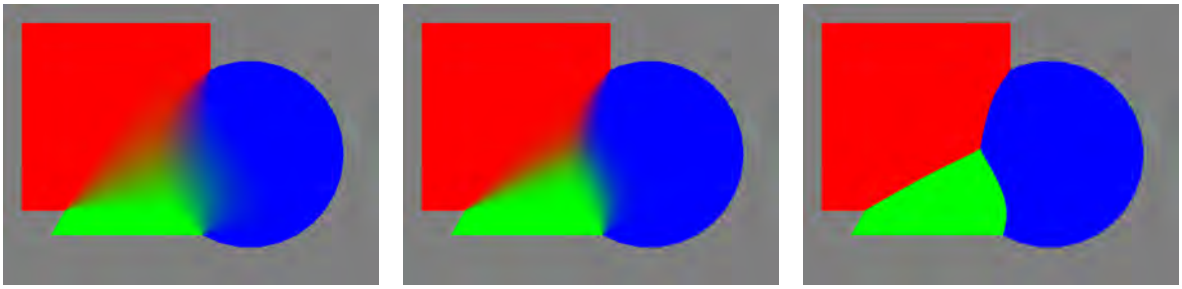
$$w_i = \frac{s_i^b}{\sum_{j=1}^{k} s_j^b}. \tag{5.10}$$

This is akin to the notion of a soft maximum. As b increases, the maximal scale takes greater precedence, increasing the associated weight relative to the others, in effect expanding the pure region of the associated volume. Figure 5.11 demonstrates the effect of modifying b for our distance-based region solution, while Figure 5.12 does the same for our functional-based



(a) $b = 1$ (b) $b = 2$ (c) $b = 300$

Figure 5.11: Effects of increasing the blending parameter for our distance-based solution.



(a) $b = 1$ (b) $b = 2$ (c) $b = 300$

Figure 5.12: Effects of increasing the blending parameter for our functional-based solution.

region solution. We found experimentally that $b = 1.5$ provides a reasonable blend, though this parameter can be controlled by the user for case-specific tweaking.

5.5 Results

Based on visual comparisons between existing blending solutions and our algorithm (Figure 5.13), we conclude that we have produced a quality solution that effectively generates natural transitions in our volumetric datasets. Qualitative assessments from sonographers and physicians alike indicate that the seams are well-hidden and that stitched volumes cannot be easily distinguished from unstitched volumes.

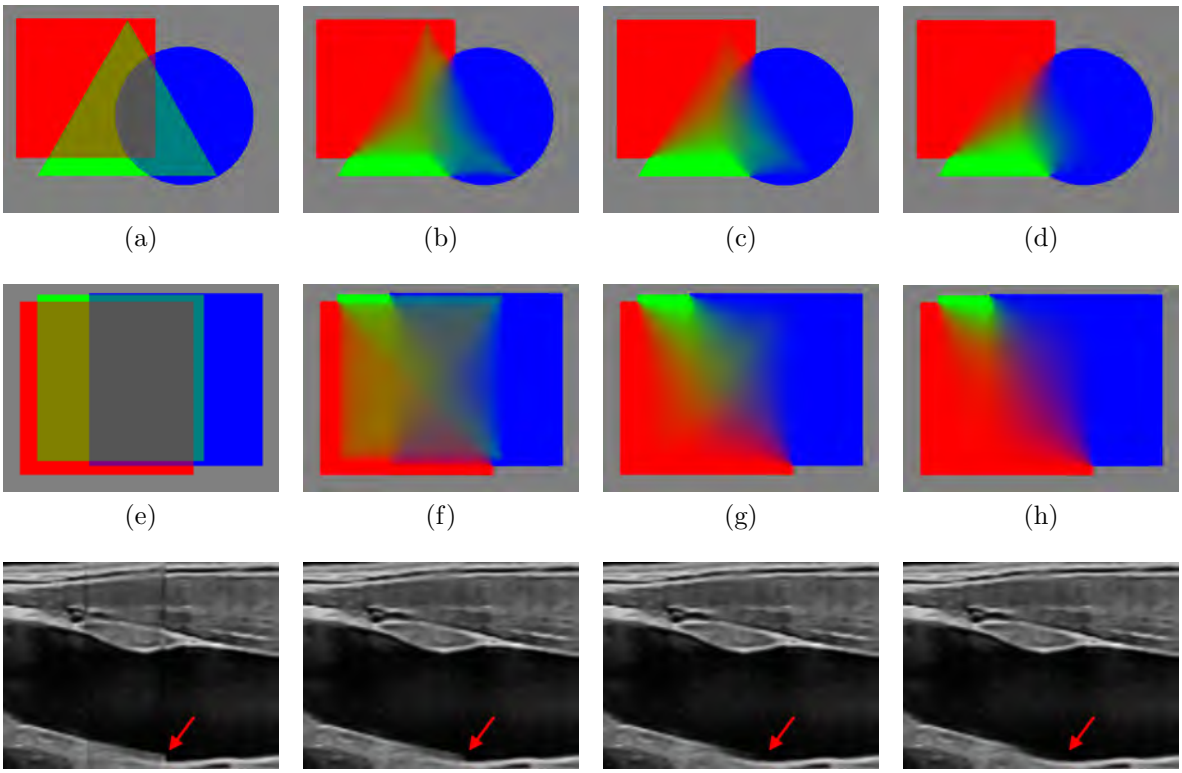


Figure 5.13: Comparison of results. Left to right: The first column show the results of a naive average. The second column shows an existing solution (Szeliski, 1996) generalized to arbitrary shapes. The third column shows the results of our distance-based region solution. The fourth column shows the results of our functional-based region solution. Each row represents a different example. The bottom row shows 2D cross sections demonstrating the results on real volumetric ultrasound data. Note the hard seam edges in (i). While (j) hides the main seams well, a significant artifact remains (red arrow). These artifacts have been greatly reduced in (k) and even more so in (l).

CHAPTER 6

Volume Embedding

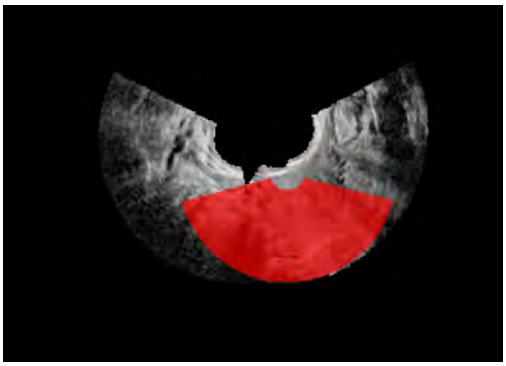
While seams between adjacent volumes are effectively eliminated by our volume stitching process, one telltale sign that the resulting volume has been merged remains: the boundary. The peculiar shape of a stitched volume (Figure 6.1) ruins the impression of working with a real-world ultrasound volume. A simple solution to this problem is to crop out the boundary. However this results in the loss of useful information (Figure 6.2a). Ideally we would like to retain as much valid data as possible, and instead incorporate additional data to fill in the missing regions (Figure 6.2b). We achieve this through our volume embedding process.

This process takes advantage of the prevalence of noise in ultrasound volumes by filling in the voids with both real and synthetic noise. Simply overlaying our stitched volume onto a noise volume, however, is not sufficient, as we see in Figure 6.3. Even with blending along the edges of our anatomical data, the background noise data is just too inconsistent to look natural. To appear authentic, the noise must match the variance and local appearance of the surrounding real data. We achieve this goal by intelligently diffusing intensity values both inwards and outwards from the stitched surface. We then apply these values to a tone-mapped noise volume. Finally, we blend the resulting noise volume with our stitched volume.

6.1 Real Noise

The first step of our embedding process is generating the background noise volume in which to embed our stitched volume. To provide authenticity to our final result, we start with noise volumes acquired from the same ultrasound hardware as our anatomy volumes. Since

Figure 6.1: Cross section of a stitched ultrasound dataset. Note the irregular boundary.



(a) Coverage possible by cropping

(b) Coverage possible by embedding

Figure 6.2: Comparison of coverage possible by resolving irregular boundaries through cropping versus embedding. Note that cropping eliminates almost all benefits gained from stitching multiple volumes.

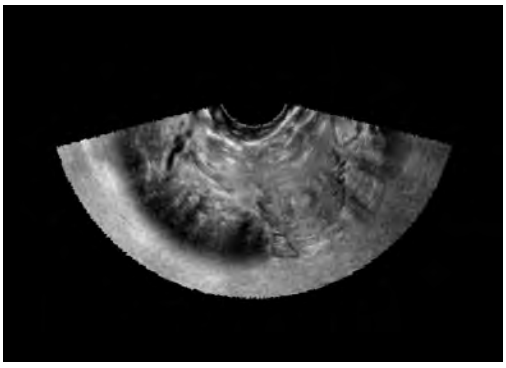


Figure 6.3: Cross section of our anatomical data naively overlayed on top of noise. Note the unnatural transition into noise.

(a) Probe without gel. Note the lack of data in the ultrasound image

(b) Probe with gel. Note the appearance of noise in the ultrasound image

Figure 6.4: Comparison of the data produced with and without gel.

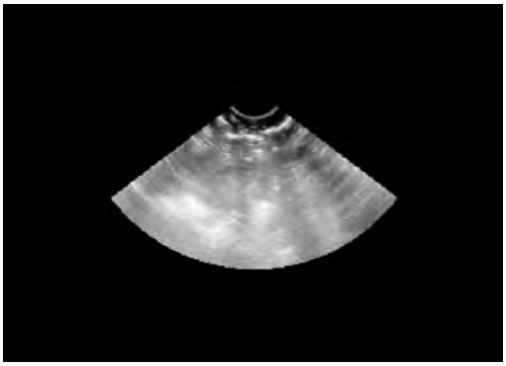
an ultrasound probe cannot capture data without ultrasound gel (Figure 6.4a), we acquire our noise volumes by coating the surface of the probe in gel (Figure 6.4b) and performing a standard 3D scan while holding the probe in the air. This fills the volume with the noise characteristically associated with ultrasound images.

To maintain a level of detail that matches our anatomical scans, we acquire the noise volumes with the same angular coverage (Figure 6.5). This means we must also stitch our noise volumes. Fortunately we can do this in a clean and consistent manner, since we do not need to worry about aligning structures within the volumes. We developed an automatic process to identify the necessary shape information about the noise volumes and align them accordingly (Figure 6.6). A user-specified overlap parameter determines the angular extent of the resulting merged noise volume.

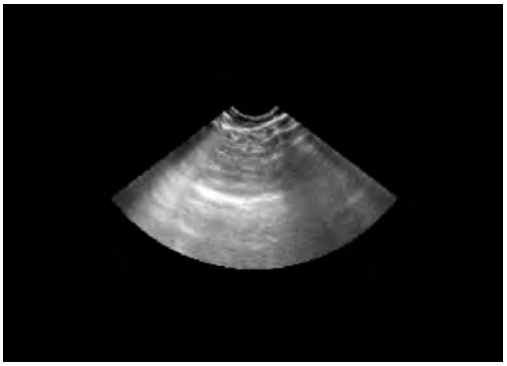
Once these noise volumes are aligned, our tool blends them together in the same manner as our anatomical data (Figure 6.7).

6.2 Synthetic Noise

The main challenge with using purely real noise volumes is the lack of adequate resolution. We must scale up our noise volumes in order to accommodate the greater coverage of our stitched data. Since the imaging hardware limits the resolution of our data, the noise volumes



(a) First noise volume



(b) Second noise volume

Figure 6.5: Cross sections of raw noise volumes. Note the inadequate angular size.

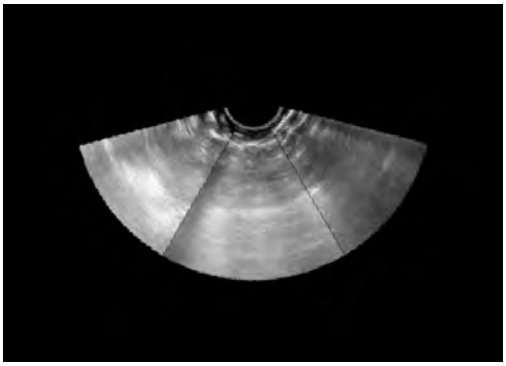


Figure 6.6: Cross section of automatically aligned noise volumes.

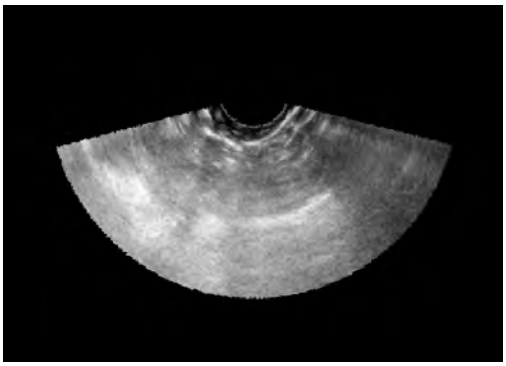


Figure 6.7: Cross section of blended noise volumes.

have a lower resolution once they are enlarged relative to our anatomical data. To work around this limitation, we introduce synthetic noise.

One way to generate this synthetic noise would be to simply apply standard noise algorithms such as white noise or Perlin (1985) Noise to our 3D volume. However, such noise does not look natural in ultrasound images due to its uniform appearance. To create synthetic noise that appears authentic, we must consider the ultrasound image formation process.

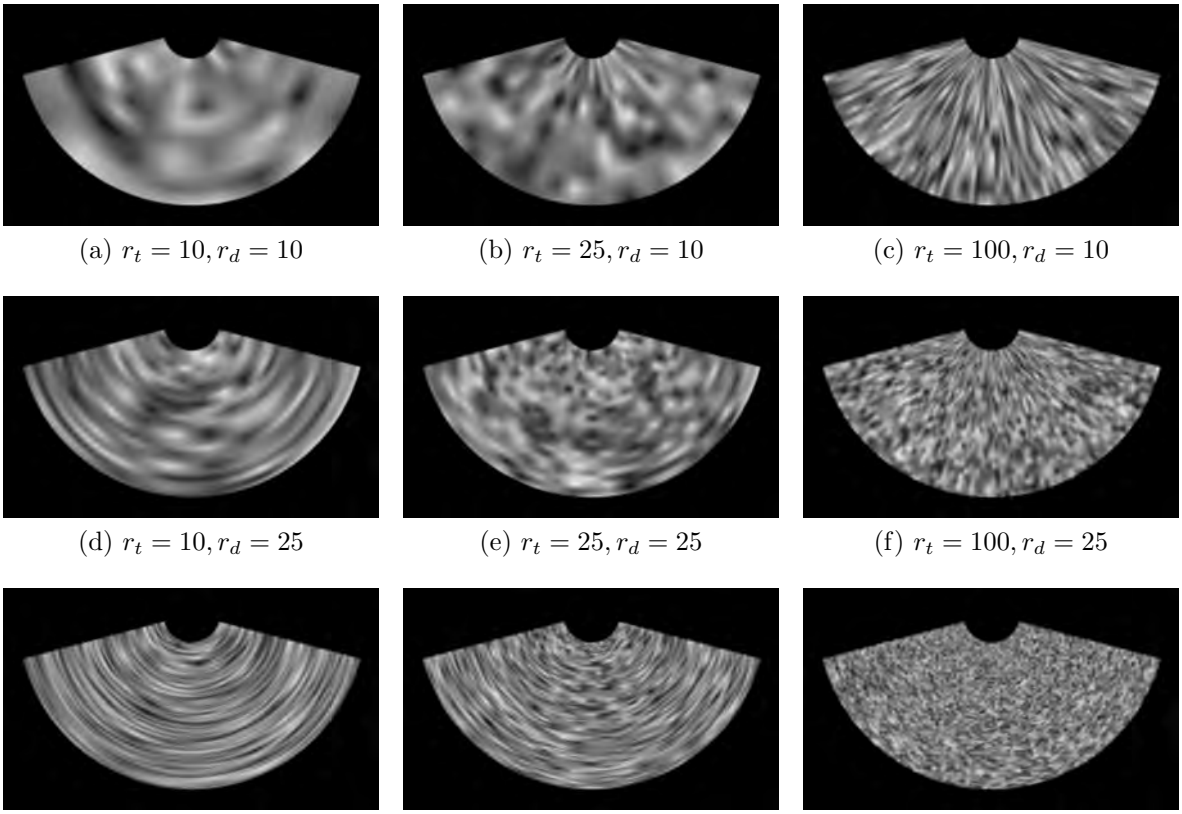
As explained in Section 3.4.1, ultrasound images are formed by computing time delays of acoustic echoes along a curved configuration of piezoelectric transducers. The data along these line segments are interpolated in a non-linear fashion to produce the final 2D or 3D image. This non-linear interpolation produces the curved noise artifacts characteristic of ultrasound datasets.

We borrow the same principles from our synthetic data pipeline (Section 3.4.5) to generate our synthetic noise, only instead of data gathered through ray tracing, we assign each transducer a set of points of random intensity, evenly spaced in depth. Parameters such as the number of simulated transducers, the number of sweeps, and the depth resolution allow fine control over the synthetic noise appearance. These parameters are exposed to the user, allowing them to choose values that best match the noise of the anatomical data. Figure 6.8 shows the effects of changing the noise resolution parameters.

6.3 Diffusion

We begin the diffusion process by down-sampling and blurring our input volume (Figure 6.9). We down-sample for computational efficiency, and blur to eliminate the high-frequency information, which we do not wish to preserve. Ultimately we will be incorporating high-frequency noise instead.

Since we wish to diffuse boundary values, we next extract all surface voxels from our volume (Figure 6.10). This surface is based on a simple neighbor check for null values. For



(a) $r_t = 10, r_d = 10$ (b) $r_t = 25, r_d = 10$ (c) $r_t = 100, r_d = 10$

(d) $r_t = 10, r_d = 25$ (e) $r_t = 25, r_d = 25$ (f) $r_t = 100, r_d = 25$

(g) $r_t = 10, r_d = 100$ (h) $r_t = 25, r_d = 100$ (i) $r_t = 100, r_d = 100$

Figure 6.8: Adjusting noise parameters. The transducer resolution, r_t, increases to the right, while the depth resolution, r_d, increases downwards. The effect of adjusting the sweep resolution, r_s, is not shown, but has the same effect as r_t, only in the sweep direction.

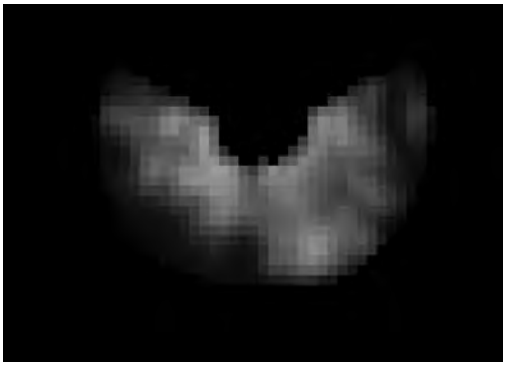


Figure 6.9: Cross section of our stitched volume down-sampled and blurred.

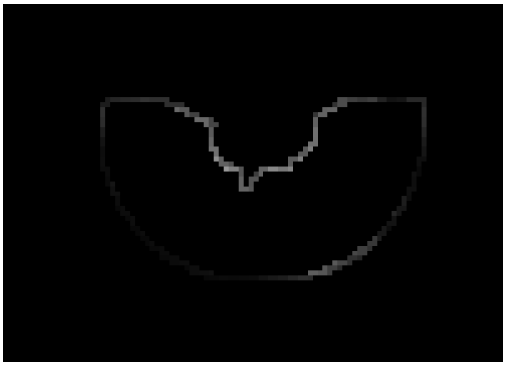


Figure 6.10: Cross section of our boundary surface.

Figure 6.11: Cross section of diffused surface values.

a volume $I : U \mapsto V$, where $U \subset \mathbb{N}^3$ and $V \subset \mathbb{R}^1$, we define the boundary Ω as follows:

$$\Omega = \left\{ u_{ijk} \in U, I(u_{ijk}) \neq 0 \;\middle|\; \exists u_{pqr} \in U, \text{ where } \begin{array}{l} i-1 \leq p \leq i+1, \\ j-1 \leq q \leq j+1, \\ k-1 \leq r \leq k+1, \\ I(u_{pqr}) = 0. \end{array} \right\}. \tag{6.1}$$

We then iterate across all output voxels and determine their distance from each surface voxel. We use these distances to weight the contribution of the corresponding voxel intensity. For all n surface voxels s_p, we compute their contribution weight at output voxel u_{ijk} as follows:

$$w_{ijkp} = \frac{(||u_{ijk} - s_p||)^{-1}}{\sum_{q=1}^{n} (||u_{ijk} - s_q||)^{-1}}. \tag{6.2}$$

This function weights the nearest surface voxel the most, and the farthest voxel the least, as desired. Figure 6.11 shows the results of this process.

While we can see the effect of the diffusion in these results, they do not provide the type of local surface diffusion we require. To provide more control over the diffusion properties, we introduce a diffusion parameter, $d \geq 1$, as an exponent on the voxel distance. This has the effect of exaggerating the contributions of closer voxels, thereby providing tighter, more localized diffusion. We update our weighting formula accordingly:

$$w_{ijkp} = \frac{(||u_{ijk} - s_p||)^{-d}}{\sum_{q=1}^{n} (||u_{ijk} - s_q||)^{-d}}. \tag{6.3}$$

Figure 6.12 demonstrates the effects of changing d.

We can choose different values of d depending on whether we are inside the stitched volume ($I(u_{ijk}) \neq 0$) or outside ($I(u_{ijk}) = 0$). We determined experimentally the following values of d to produce ideal diffusion:

$$d = \begin{cases} 4, & \text{if } I(u_{ijk}) = 0; \\ 6, & \text{if } I(u_{ijk}) \neq 0. \end{cases} \tag{6.4}$$

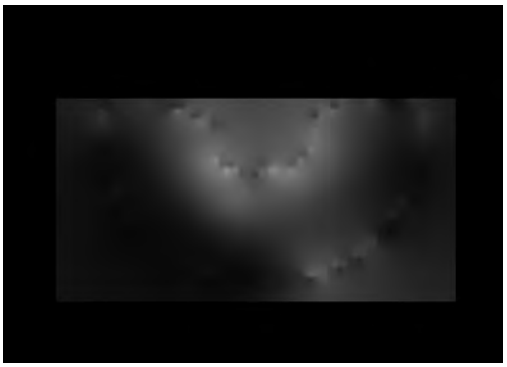
Figure 6.13 shows the results.

6.4 Tone Mapping

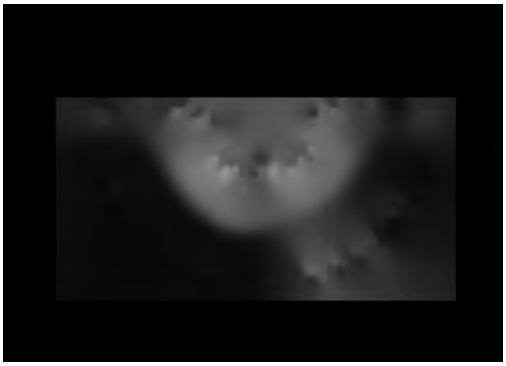
To apply the intensity of our diffused low-frequency data to our high-frequency noise data, we apply local tone mapping. Common in photography, particularly high dynamic range (HDR) photography, tone mapping is used to map colors from one range to another (Krawczyk et al., 2007). Often this is used to flatten or compress one range to a smaller range as required by a viewing device or data representation.

Local tone mapping refers to adjustments made based on a local neighborhood of information. This can take the form of removing or reducing low-frequency information, which is our goal here. To achieve this goal, we simply blur our noise volume to remove the high-frequency data, then divide our initial volume by the results. For a volume $B : U \mapsto V$

(a) $d = 1$



(b) $d = 2$



(c) $d = 4$

(d) $d = 8$

(e) $d = 16$

(f) $d = 32$

Figure 6.12: Different values of d.

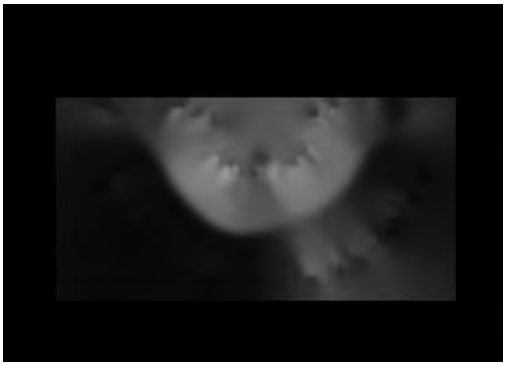
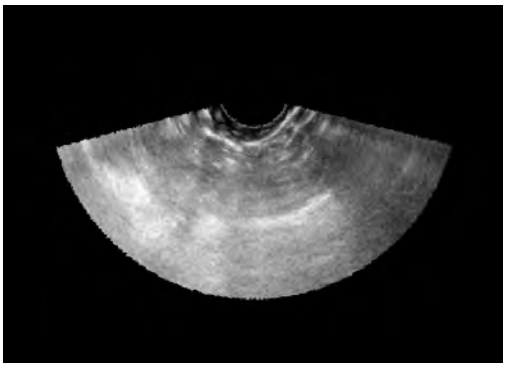
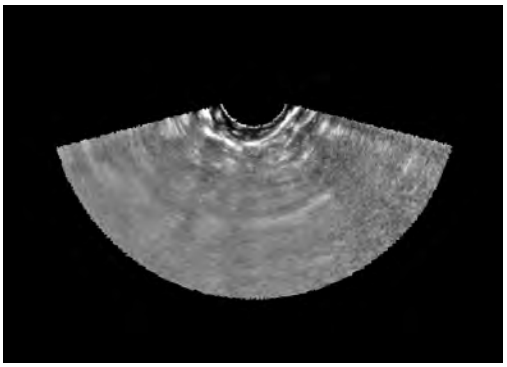


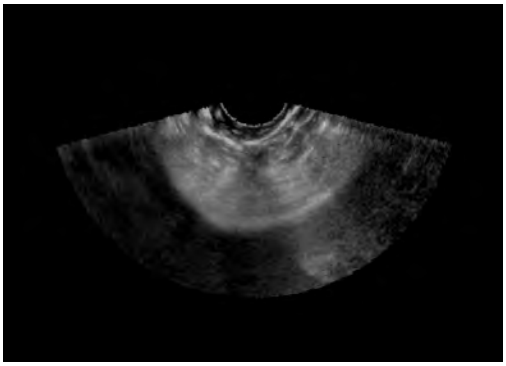
Figure 6.13: Cross section of diffused surface values with experimentally chosen values of d.



(a) Noise volume



(b) Noise volume tone mapped



(c) Noise volume with diffusion information

Figure 6.14: Removing and reintroducing low-frequency information.

representing the blurred version of volume I, our tone-mapped image T is defined as

$$T = \frac{I}{B}. \tag{6.5}$$

Figures 6.14a and 6.14b demonstrate the effects of this process. The result is a volume containing mostly high-frequency information.

Once we have removed the original low-frequency information, we can reintroduce our own low-frequency information from our diffusion volume. We simply multiply our tone-mapped image T by our diffusion image D to arrive at a new noise image

$$I' = TD \tag{6.6}$$

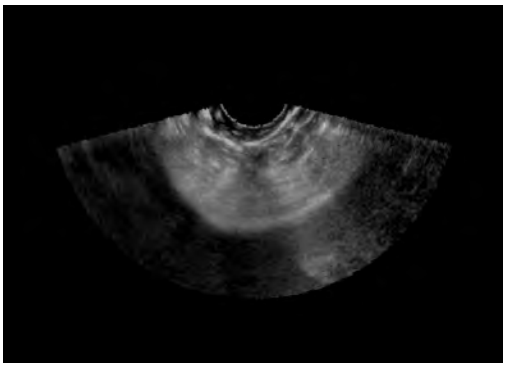
(Figure 6.14c). The resulting volume properly captures the local intensity of our stitched volume.

6.5 Blending

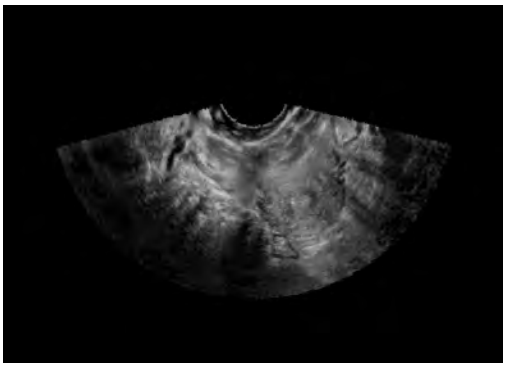
With our background image properly augmented to match our stitched volume, we can combine the two. To do this we overlay our stitched volume on top of the background volume, using a user-defined feathering radius to blend the boundaries of the anatomical data. To perform this feathering, we pre-compute a distance map (Saito and Toriwaki, 1994) based on the mask of our anatomical data. For a given voxel, we can then determine the distance to the boundary, and blend based on a linear ramp defined by the feathering radius.

6.6 Results

As we see in Figure 6.16, our embedding solution provides substantial improvements over a naive approach. There is no obvious transition from anatomical data to the background noise data. Thus, our stitched volume now takes on the distinctive ultrasound form factor

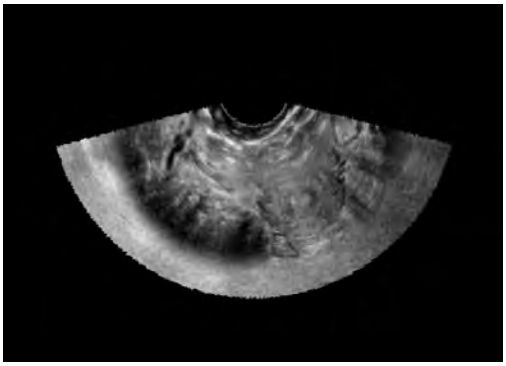


(a) Background volume

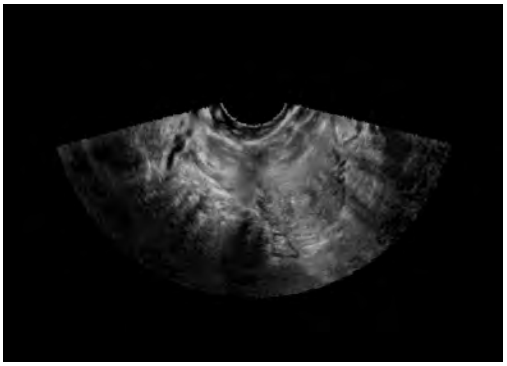


(b) Background volume + stitched volume

Figure 6.15: Overlaying our stitched volume on top of our augmented noise volume.



(a) Naive overlay



(b) Background volume + stitched volume

Figure 6.16: Comparison of methods.

without sacrificing believability. We have applied this volume embedding technique to a large set of case studies, producing convincing results that have been directly incorporated into the SonoSim product.

CHAPTER 7

Translational Data Interaction and Patient Simulation

One of the overarching goals of ultrasound volume stitching is to provide adequate data coverage to support translational interaction. While such interaction can be supported easily through a standard keyboard and mouse solution, providing a tangible mapping to real-world movement allows a much stronger form of interaction.

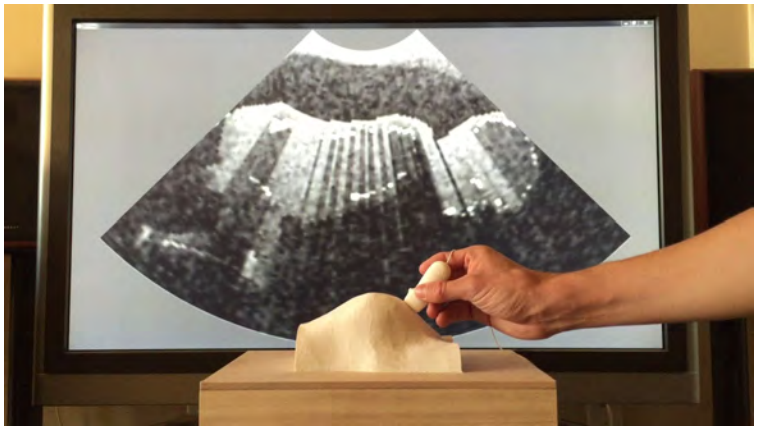
The SonoSim solution currently employs a three-degree-of-freedom rotational probe. Our goal was to build two additional degrees of freedom on top of this. Note that we do not need a full six-degree-of-freedom solution, as probe interaction is only meaningful when constrained to a surface. When a real ultrasound probe is lifted from a patient's skin, the surrounding air prevents the ultrasound waves from propagating, resulting in a dark image. Thus, there is no strong need for the additional degree of freedom (although we will discuss below how compression can be included in the proposed solution).

Another significant requirement of the translation solution is that it support contoured surfaces. That is, while the translation itself is limited to two degrees of freedom, the coordinates may lie on a surface in three dimensions. In other words, each two dimensional coordinate should map directly to a three dimensional position. This requirement is based on the desire to allow for interaction with a surface conforming to the human form.

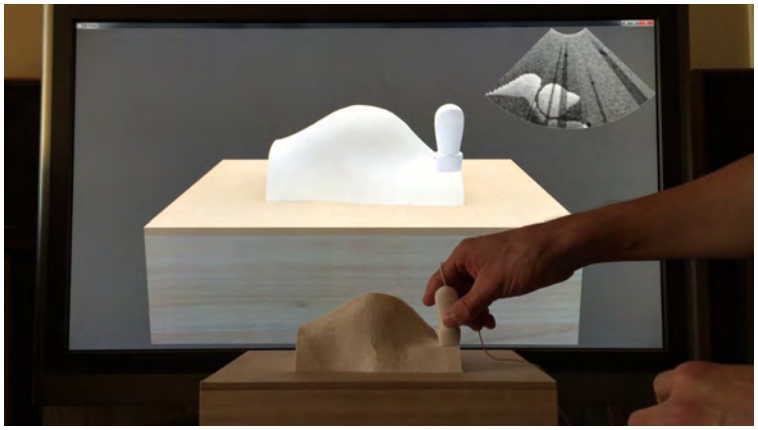
Finally, the solution should support the tracking of not only a probe, but a needle as well. This allows for the simulation of needle guided procedures, where the probe is used as the imaging device to help guide the insertion of a needle into the correct anatomical region.

Our solution to this problem is a 3D-printed anatomical model that uses optical tracking of infrared LEDs to locate and identify the probe and needle. An infrared camera is located inside/beneath the model and fitted with a wide angle lens for full coverage of the internal

(a) Final prototype



(b) Scanning the mannequin. The ultrasound window reveals the fetus inside.



(c) Visualization of the tracking applied to the digital model. Note the accurate positioning relative to the edge of the model.

Figure 7.1: This prototype models the abdomen of a pregnant woman.

surface. The model itself is printed as a thin translucent shell to allow the LED light to shine through while diffusing ambient lighting. Tracking of the probe and needle is accomplished by a fast and simple vision algorithm, and a mapping is performed to locate the three dimensional position of the LED. Synchronized flashing between the LEDs and the camera allow for distinguishing between the the probe and needle. Figure 7.1 shows our final prototype in action.

7.1 Overview of Our Approach

While the core of our approach is optical tracking, a blend of other technologies is required to make the entire solution complete, transparent, and robust.

The process works by embedding infrared LEDs within the tip of each device (probe and needle). The plastic housing is semi-translucent, allowing the light to escape without impacting the exterior form factor of the device. Since the LEDs are internal, and produce invisible infrared light, the system is hidden from view.

The 3D surface upon which the user is applying the probe and needle is similarly made from a semi-translucent material. This allows the infrared light to pass into the interior of the shape, where it can be detected by a camera. However, this only works when the LED is in close contact with the surface; otherwise the light is diffused by the translucency of the material. This is critical to the functionality of the device, as it allows the probe and needle to be isolated from background light sources. The surface is 3D printed from a digital model, ensuring exact correspondence between the physical world and the virtual representation.

The camera is fitted with a wide-angle lens, allowing visibility of the entire interior surface of the model. This camera/lens combination is held securely in place to ensure that the relationship between the imaging sensor and the model is consistent.

The infrared camera is able to see the light sources emitted by the infrared LEDs. The brightness and diffusion of these hot-spots is determined by the proximity of the LED to the surface. A simple vision algorithm run on a Raspberry Pi thresholds these values to

determine when an LED comes in contact with the surface, and identifies the centroid for tracking.

The system flashes the LEDs in sync with the image capture rate, such that only one device will be on for a given frame. This allows the algorithm to know exactly which hot-spot corresponds to which device.

The location and identification information is then transmitted wirelessly to the simulation computer through a simple UDP protocol. The simulation computer takes the 2D location and maps this to a 3D vector, which is then intersected with the digital model to locate the proper surface position. From there it is simply a matter of running the appropriate visualization.

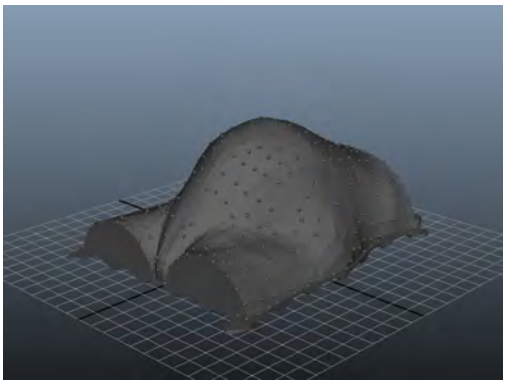
In practice this translational information would be combined with rotational information for a complete tracking solution.

7.2 3D Printing

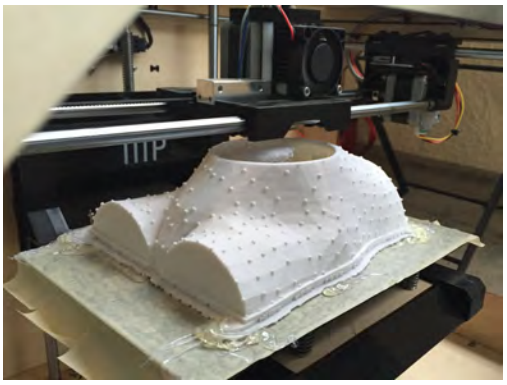
One of the core technologies that makes our approach possible is 3D printing. While other manufacturing techniques could be used instead, such as a vacuum formed plastic shell derived from a metal mold milled by a CNC machine, 3D printing offers quick experimentation, and the possibility of sharing models easily with other users. The exact correspondence between the physical model and the virtual model afforded by 3D printing is at the heart of our approach (Figure 7.2).

A key consideration is the thickness of the 3D printed shell. The shell must be thick enough to provide structural integrity and diffuse ambient light sources, but thin enough to allow our LED light to pass through. The shell thickness must also be consistent such that the brightness threshold in the vision algorithm can be applied robustly.

We considered multiple thickness consistency metrics during our design process. One approach would be to define the thickness relative to the camera position. That is, for a given point on the exterior surface of the model, follow a vector to the camera's optical



(a) Digital model



(b) Physical model during the printing process

Figure 7.2: Demonstration of the precise correspondence between the digital model and the 3D print.

center. The corresponding interior surface point should be a fixed distance along that vector.

Another thickness consistency metric is to maintain a fixed minimal distance between the exterior surface and the interior surface. This is the common definition we think of when we imagine a shell of constant thickness. This can be thought of as placing a hemisphere on the interior side of the exterior surface at every point. The resulting surface becomes our interior surface. Indeed, this is the result we obtain by applying the Minkowski algorithm to our exterior surface along with a sphere and retaining everything interior to our original surface. This is the solution we incorporated, taking advantage of the OpenSCAD implementation to derive our result. We chose to use a 2mm thickness, which provided a good trade-off between translucency and physical strength.

7.3 Calibration

Since maintaining an exact correspondence between the physical model and the virtual model is pivotal to our approach, calibration is paramount. The relationship between the image sensor and a given point on the model must be known precisely. Thus, we must take into account not only the standard intrinsic camera parameters (α_x, α_y, γ, u_0, v_0), but also the lens distortion and the physical relationship between the camera and the model.

One solution to this problem would be to calibrate the camera/lens combination with a standard camera calibration technique such as that found in OpenCV. However, this relies on the quality of their calibration model and does not solve the physical offset problem. Our approach is to eliminate any in-between steps such as these and directly map known surface points to their location in the image. We achieve this by including calibration points in our 3D model.

We can take two approaches to this calibration method: One option is to print two physical models, one with calibration points, one without. The other option is to print just one physical model, with calibration points that can be removed. In either case, the calibration model is identical to the final anatomical surface model, but with additional features added at calibration points. If going with the first option, these points could be small LED-sized holes that allow easy placement of the light source at the correct location for a given calibration point (Figure 7.3). For the second approach, these points must be raised bumps on the surface, which can be removed once calibration is complete (Figure 7.4).

The two-model approach has the benefit of providing an easier calibration experience. Since this approach can use indentations at calibration points, aligning the LED over the feature is very tactile, making it easy to achieve and maintain the correct position. The downside is that printing two models is time consuming, wasteful, and more expensive. Furthermore, the two models must be aligned precisely, introducing room for error.

The one-model approach has the benefit of requiring only a single print, and there is no danger of misalignment once calibration is complete. However the calibration process itself is far more difficult and tedious, since the LED must be held precisely at the top of each bump. Furthermore, the calibration points must be removed after calibration is complete, which can be a time consuming process, and this also eliminates the opportunity to recalibrate once complete.

For our early hemispherical test (Figure 7.3), we chose the locations of the calibration points such that there was greater density closer to the horizon, as these areas tend to get compressed in the image due to the nature of the lens distortion.

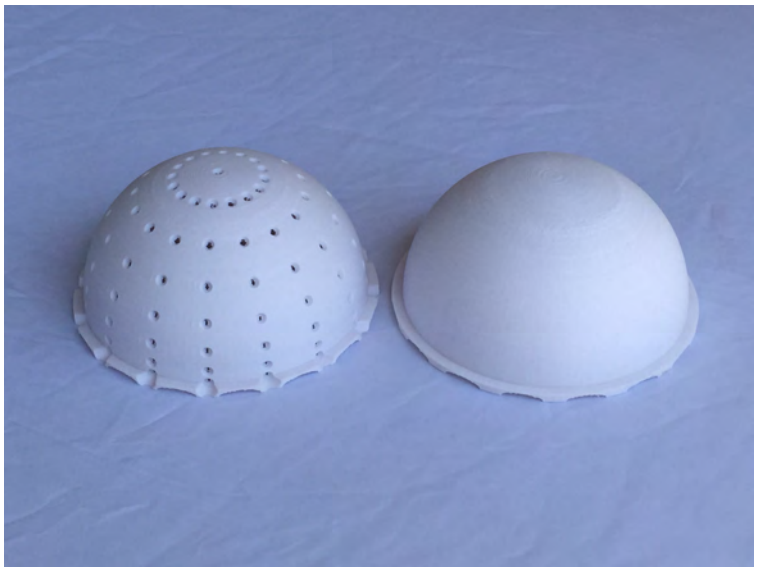


Figure 7.3: Example of using two separate prints for calibration. The dome on the left contains small holes designed to fit the LED perfectly for easy alignment.

(a) Before calibration points have been removed

(b) After calibration points have been removed

Figure 7.4: Example of using a single print for calibration.

For more complex geometry, we can take advantage of the polygonal nature of the model to choose our calibration points. This is the approach we took for our functional prototype. We employed a mesh simplification algorithm based on quadric error (Garland and Heckbert, 1997) to reduce the number of vertices of a polygonal abdomen model to a reasonable number (around 350) and used the vertex positions as calibration points. We refer to this new mesh as our low-res proxy.

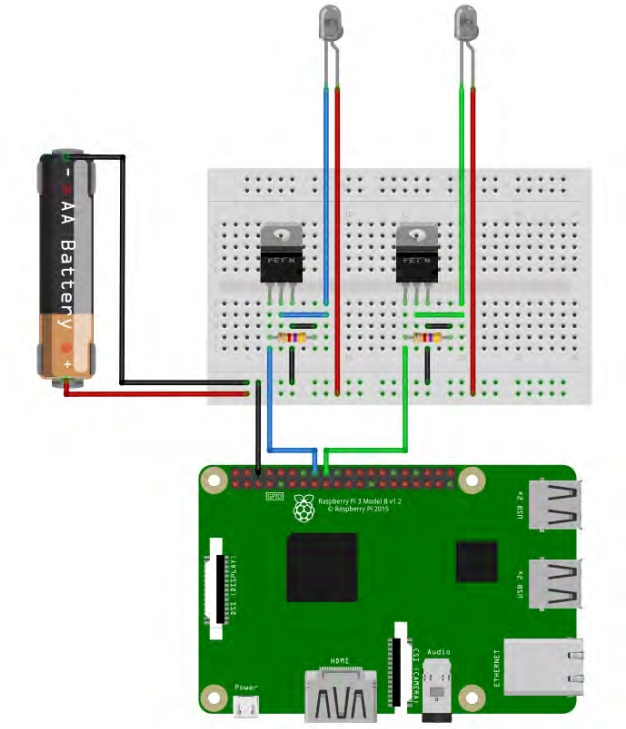
The calibration process simply involves iterating through each point, shining the LED at the corresponding location, and logging the position in the image. In Section 7.6 we will see how we use this information to map any arbitrary point using an interpolation scheme based on our low-res proxy.

This solution provides a simple and robust means of calibrating the physical model to the digital model. It eliminates the need to know any of the precise measurements involved, since the algorithm finds the direct mapping between points on the model and their corresponding points in the image plane. Thus, the model can be attached to the camera without any careful alignment process, as long as this placement is consistent. This method is even robust to deformations in the physical model that may result from the 3D printing process.

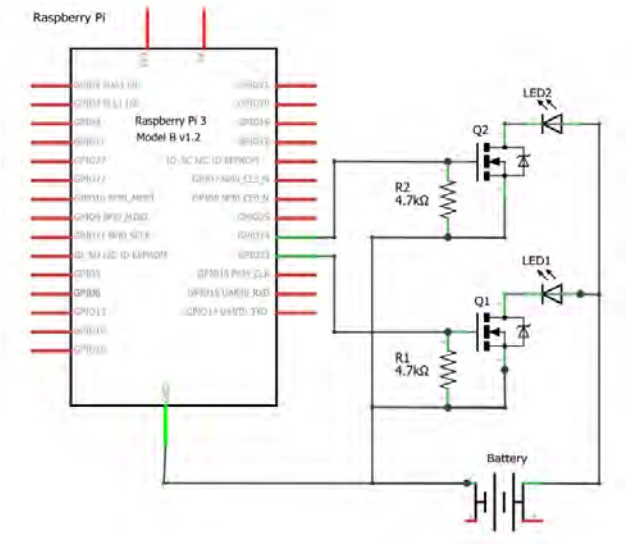
One drawback to using calibration points is that it requires calibration for each model. A partial workaround to this would be to create two hemispherical calibration rigs, one smaller than the other, with the same calibration points. From this we can extract consistent 3D vectors for each position on the image plane, allowing us to intersect with arbitrary geometry once calibration is complete. However, this still requires precise positioning of each new model.

Another drawback of our approach is that a large number of calibration points may be necessary to accurately account for the curvature associated with lens distortion. Since our interpolation scheme uses triangles (Section 7.6), these cannot easily represent such curvature. A solution to this would be to use a different geometric representation, such as NURBS.

We can also calibrate brightness with our approach. This involves simply recording



(a)



(b)

Figure 7.5: Illustration and circuit diagram of our LED set up.

the LED brightness in addition to its location information. We can then normalize this information to a constant intensity.

7.4 LEDs

Our solution uses 920nm LEDs. These emit light completely invisible to the human eye, yet show up well in an infrared camera. We use LEDS of standard 5mm diameter, which can be easily contained within a standard probe or needle form factor. The LEDs are rated at 1.5v and 40mA, which in our implementation requires an external battery source. To control the LEDs, we use two MOSFETs (one for each LED) to switch the LEDs on with a low-voltage signal from a Raspberry Pi. Figure 7.5 illustrates this set up.

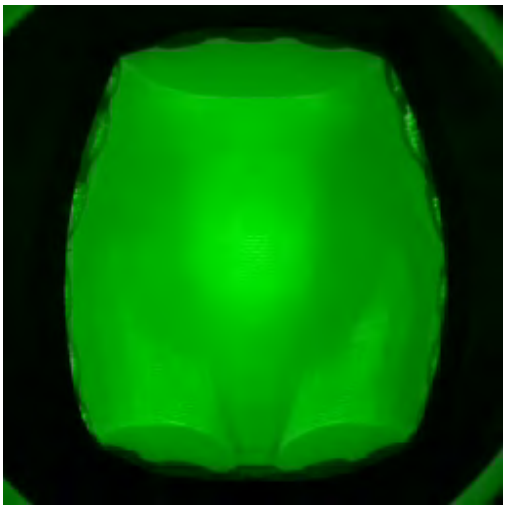
7.5 Vision

Our tracking camera is placed under the shell, with a wide angle lens that allows the system to see the entire interior surface (Figure 7.6). One of the benefits of our solution is the simplicity of the vision algorithm. The system simply thresholds the input image (Figure 7.7, magnified in Figure 7.8a) by removing all values below T (Figure 7.8b), then normalizes the remaining pixels to T (Figure 7.8c), and computes the centroid as the center of mass, where the density is the value of each pixel (Figure 7.8d). The normalization procedure transforms pixel values v from [0, 255] to [0, 1] based on T to provide better robustness:

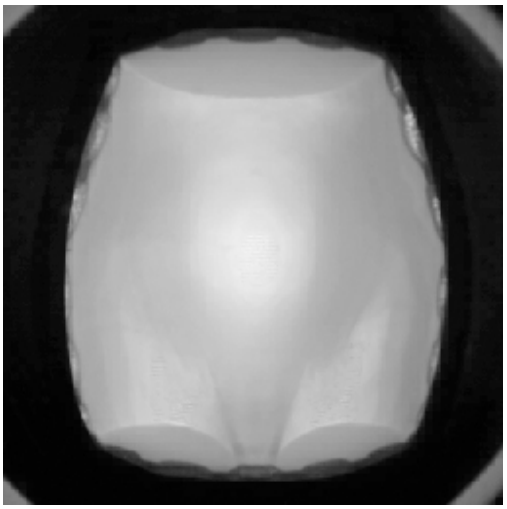
$$v' = (v - T)/(255 - T). \tag{7.1}$$

The benefit of this transformation is that pixels with values close to T may exhibit erratic behavior due to noise (a small dip below T will remove that pixel from consideration). Thus, moving these values close to 0 minimizes the impact of such fluctuations.

Identifying the centroid — as opposed to simply identifying the brightest pixel — serves two purposes:



(a) RGB data. The green hue comes from the fixed white-balance setting.



(b) Isolating the green channel. We only process this channel for efficiency.

Figure 7.6: View from our internal tracking camera looking up at the inside/underside of our abdomen shell (under strong lighting conditions for illustration purposes).

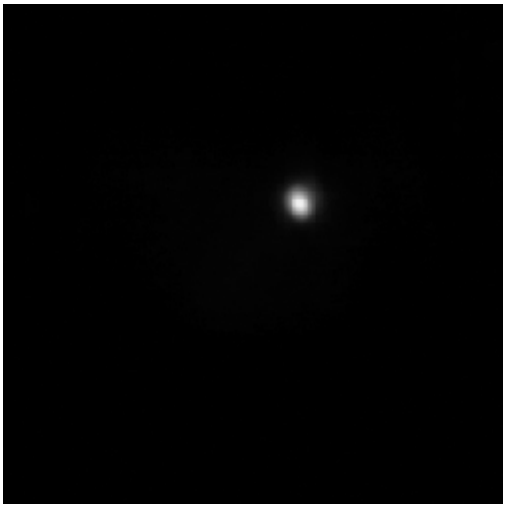
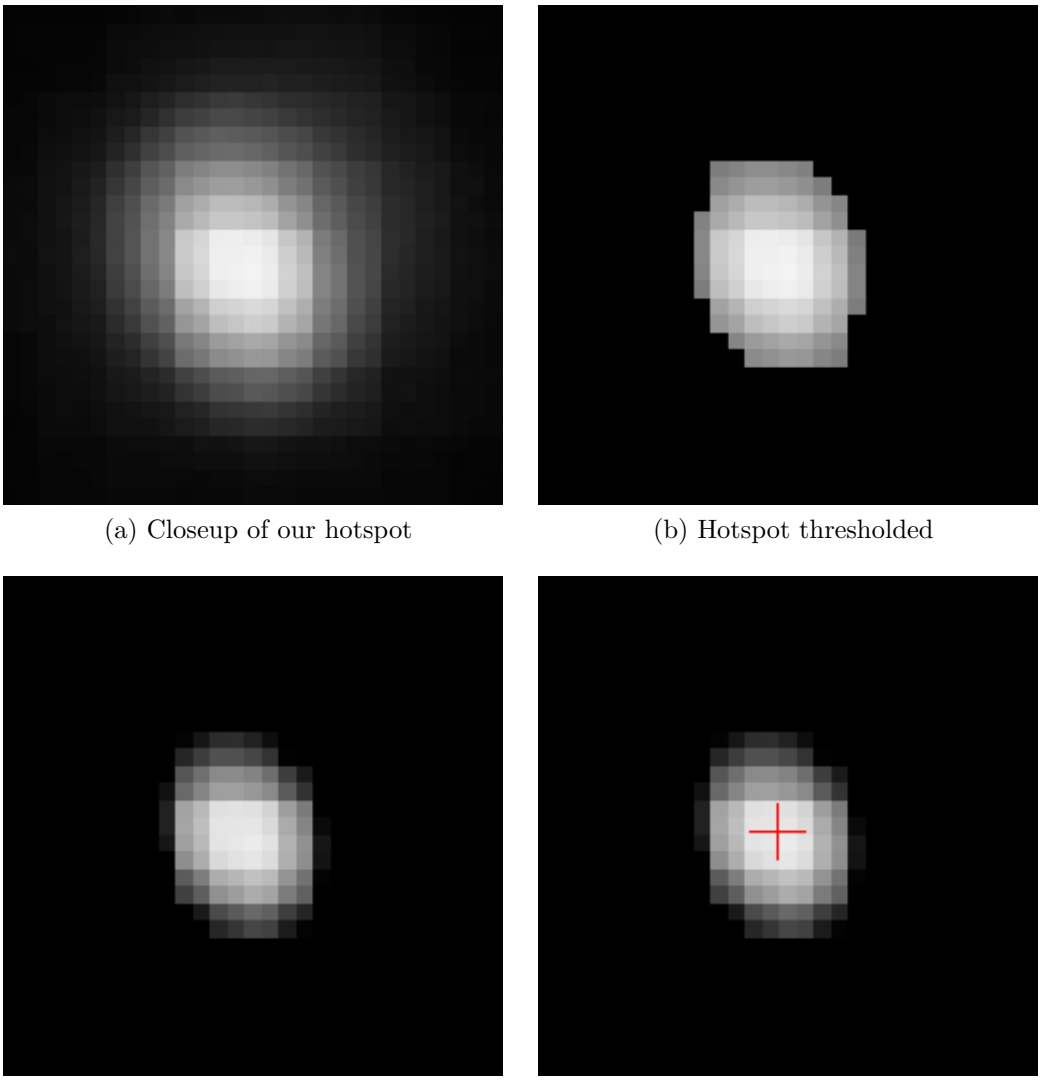


Figure 7.7: View from our tracking camera showing a hotspot from the probe LED.



(a) Closeup of our hotspot

(b) Hotspot thresholded

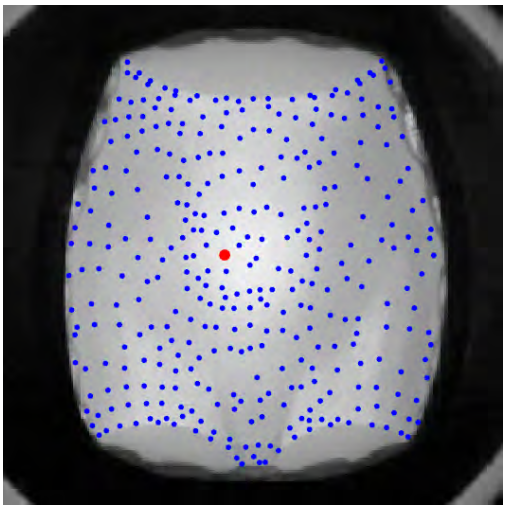
(c) Normalized to threshold value

(d) Centroid identified

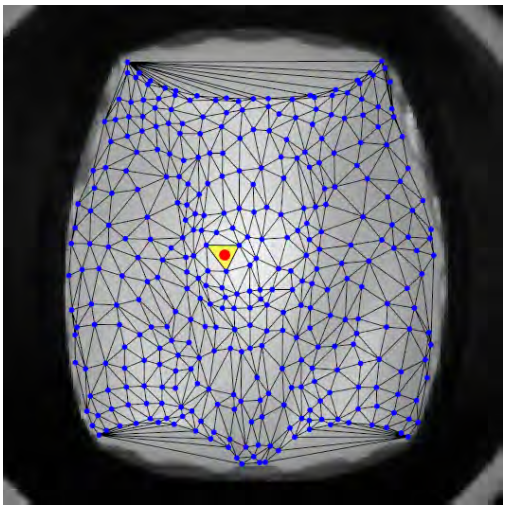
Figure 7.8: Our centroid identification process for the image in Figure 7.7.

1. First, this method is much more stable, as noise fluctuations get averaged out across the patch.

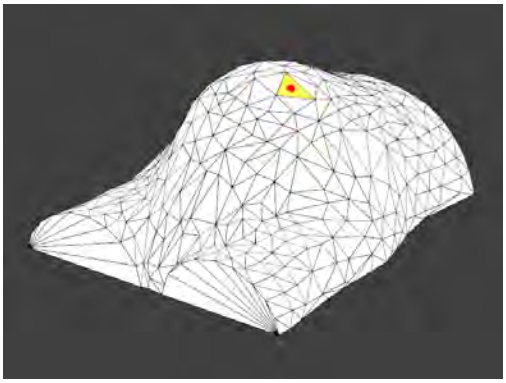
2. Second, this method leverages the intensity values to provide super-resolution positioning. This enables us to get away with lower resolution images while maintaining smooth tracking, allowing for faster processing times. It also allows more precise positioning, which is particularly important when operating within the physical world. This also means that tracking is more consistent even as the mapping from pixel to physical world may not be (e.g., pixels that map to faces at a shallow angle).



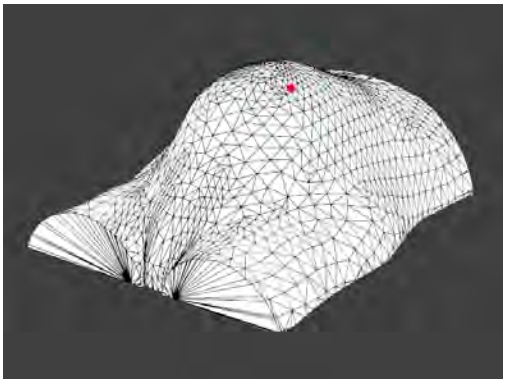
(a) 2D calibration points (blue) and hotspot centroid (red) in the camera image plane



(b) 2D calibration points connected by low-res proxy topology; active triangle and barycentric coordinates identified



(c) Hotspot centroid mapped to its corresponding 3D position in the low-res proxy mesh



(d) Hotspot centroid projected to the high-res mesh

Figure 7.9: Mapping from the 2D camera image space to the final 3D position.

7.6 Mapping

Since we have defined our calibration points based on the vertices of a low-res proxy of our model, we can easily map this mesh to the camera image plane by assigning each vertex its corresponding 2D position recorded during our calibration phase. Then, for a given arbitrary point in the 2D image (Figure 7.9a), our algorithm identifies which triangle in our flattened low-res mesh contains that point (Figure 7.9b). Given the low number of triangles in our proxy, this can be computed with a brute-force search. Next, the barycentric coordinates

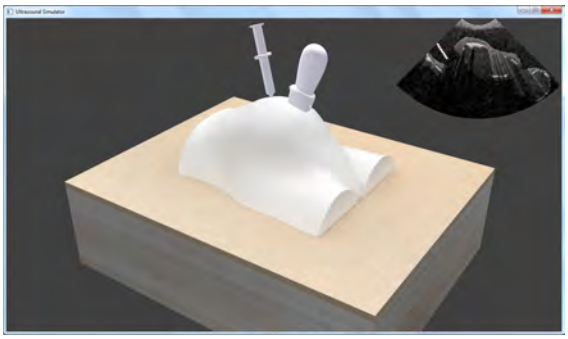
of this point are computed within the triangle, and applied to the corresponding triangle in our 3D proxy (Figure 7.9c). Finally this vector is projected to our full-res model to identify the final 3D surface location (Figure 7.9d). This projection is performed along the vector defined by the origin and the 3D position in our low-res proxy mesh. We use an octree to compute this intersection efficiently.

7.7 Visualization

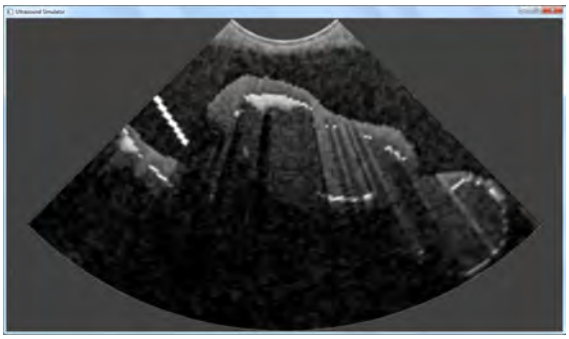
To demonstrate the full capabilities of our tracking system, we developed a simple application to provide a scan view as well as a 3D visualization of the tracking results. The scan view allows users of the prototype to experience a scan session as they might in a true clinical setting, where the only feedback is the ultrasound image itself. Ultimately this data will be derived from real patient volumes, as described in the previous chapters. However, since this work is still experimental, such data does not yet exist. Instead, we showcase the potential of the system with simulated data, using a real-time renderer we developed. As this is not the focus of our work, we will provide only a cursory explanation of this system.

Our renderer uses a ray-tracing technique similar to that described in Section 3.4.3, but simplified for real-time performance. As input data, we used polygon models of the internal organs, vasculature, and skeleton to fill our abdomen shell appropriately. We also articulated and positioned a separate anatomical model of a baby to serve as the fetus, which was placed into the womb. Each type of anatomy was assigned different intensity values, darker for vasculature, brighter for skeletal geometry.

For each simulated transducer element, our system emits a single ray, computes the intersection with the geometry using our octree acceleration structure, and records the value of the corresponding anatomy. This continues along the length of the ray until we have traversed the maximal depth of our scan. We rasterize the intensity information to a texture, simulating shadows by darkening each pixel according to the number of geometry intersections encountered. We render the needle separately as it is not included in our octree due to its dynamic nature.



(a) 3D rendering of the tracked probe and needle relative to the abdomen, with scan inset



(b) Scan view; The needle is visible as the white line on the left

Figure 7.10: Screen-shots from our translational tracking demonstration simulator.

7.8 Results

To showcase the robust nature of our tracking solution, we also provide a visualization of our prototype system itself. Since our setup is derived from digital 3D models, we were able to easily add these to a 3D viewer. The models were textured and subsequently rendered off-line to achieve photorealistic results that closely match the physical prototype, allowing the user to fully appreciate the one-to-one correspondence between the real setup and the virtual representation. Figure 7.10a shows a screen-shot of the results.

To achieve the noise artifacts in our scan visualization, we precompute a volume of noise and rasterize the corresponding slice based on the probe position. This is then blended with our geometry rendering to produce a reasonable illustration of an ultrasound scan. As a final step, we perform a simple warp in the image space to simulate probe/tissue deformation. Figure 7.10b shows a screen-shot of the results.

Our final prototype works very well in practice. The tracking is robust and consistent. Despite concerns to the contrary, interpolation is completely transparent to the user, producing no visible motion artifacts. The resulting experience is intuitive, and so convincing it has led users to believe they are actually scanning hidden internal geometry. The ability to differentiate between the probe and needle is an added bonus.

We now revisit the problems enumerated in Section 2.5, and see how our solution addresses each issue:

1. Size. Scaling up our solution can be achieved easily and cheaply to accommodate large anatomical models. This simply requires producing a larger plastic shell, while the sensing technology stays the same. Ultimately size is limited by camera resolution and/or LED brightness, but reasonable sizes should be achievable with limited hardware changes.

2. Contours. Our solution can accommodate complex shapes as long as all surface points are visible to the camera. This limitation can be overcome with a multi-camera extension.

3. Disambiguation. We can robustly disambiguate between multiple devices based on our coordinated flashing technique.

4. Proximity. This problem is non-existent in our solution, since each device is sensed at a different time interval.

5. Residual. Since there are no mechanical processes involved in our solution, we do not experience any residual artifacts.

6. Force. Since our solution does not detect pressure, this problem does not exist.

7. Fragility. Our solution does not require a delicate piece of hardware. All active components can be easily protected without interfering with the system.

8. Development. We were able to produce our solution using standard off-the-shelf components for under $100.

Thus, all problems with a pressure pad solution have been successfully solved or eliminated with our optical tracking solution.

CHAPTER 8

Conclusion

This dissertation covered a wide range of work geared towards solving the problem of generating and interacting with large-scale ultrasound datasets. These techniques have all been designed with real and pressing applications in mind, and have been influential in pushing the boundaries of ultrasound medical training. Notably, our alignment process, volume stitching solution, and volume embedding technique have all been directly applied to real product deliverables that were not possible before. Our synthetic data have helped improve these processes, and have also been used to train internal team members. Our translational interaction solution shows promise as a future product. All our methods work together towards the single goal of incorporating large-scale ultrasound datasets into medical training simulators.

8.1 Limitations

While our work provides robust results with tangible applications, we can identify several areas for improvement. Our synthetic data pipeline, for example, currently only produces datasets based on sphere packing geometry (Section 3.1.2). While this has proven sufficient for training and evaluation, it does limit the type of synthetic data that can be created.

Our automatic alignment algorithm, as presented in Section 4.2 could also use improvement. While our medical team has relied on its abilities extensively, the algorithm's value-driven nature imposes some limitations on its ability to provide correct alignments when shadows or echoes significantly alter the appearance between volumes.

Our translational data interaction and patient simulation system, which was presented

in Chapter 7, offers many benefits over existing technologies, but is not without its shortcomings. We examine these limitations next:

1. Ambient light. Our solution will not work under bright lighting conditions, such as outdoors. Large amounts of infrared light will overwhelm the light from our probe and needle, inhibiting our tracking solution. Brighter LEDs could be used, but at the cost of higher power consumption, increased cost, and greater risk of eye damage. In practice, we do not envision the ambient light issue to pose a significant problem, as most use-cases will be indoors under reasonable lighting conditions.

2. Modification. Existing probe and needle devices must be modified to include an infrared LED, which places additional burdens on the designs of the devices, and introduces power consumption concerns.

3. Pressure sensing. While augmentations can be made to support pressure sensing based on light intensity and diffusion, it will not be as reliable as a dedicated pressure pad. A more robust solution would be to add pressure sensing to the probe itself. This would provide the added benefit of allowing the user to perform pressure-related tasks (such as compression) independent of a tracking solution. While such a hybrid approach would be more costly than either solution by itself, it would provide the best user experience by combining the advantages of both systems.

8.2 Future Work

8.2.1 Extensions

Based on the sphere packing limitation mentioned above, a useful extension to our synthetic data pipeline would be the creation of a procedural anatomical model, simulating more accurately the geometry of tissue and bone, perhaps with user-specified parameters to describe the type of anatomy generated. One promising solution to this challenge would be to incorporate fractal geometry into the model, thereby fulfilling the dense space-filling requirement.

We can improve our automatic alignment solution by incorporating three dimensional extensions of common 2D feature detectors, such as histograms of oriented gradients (Dalal and Triggs, 2005), maximally stable extremal regions (Matas et al., 2004), and scale invariant feature transforms (Lowe, 1999). An even more ambitious approach would be to train an artificial neural network, using our synthetic datasets to provide the large quantity of data needed to train such a solution.

Our optical surface tracking system can be modified and improved upon by incorporating a number of possible extensions. As mentioned in the introduction to this project, we envision combining our translational system with existing IMU-based rotational systems to achieve a complete tracking solution. We have also alluded to several other extensions already, such as a multi-camera approach to support more complex geometries, and a hybrid pressure pad solution to achieve robust pressure sensing.

To take this system even further, we can imagine adding internal projectors to provide visual feedback on the physical model itself. In this way cutting, painting, or drawing on the surface would feel very real. High frequency flashing of the projector in sync with the LEDs would allow this to work without interfering with the tracking system.

Another extension that would improve the user experience would be the addition of an automatic identification capability supporting multiple 3D shells. This can be achieved cheaply by leveraging the existing tracking camera to process QR codes, or any other visual encoding scheme, placed on the inside of the shell. Another solution would be to embed RFID tags within the shells.

Finally, we can imagine improving the flexibility of our system by incorporating an internal depth sensor, which would allow for physical deformations of the model surface and perhaps eliminate the calibration process as well. Such a system may even allow support of arbitrary shapes without a preexisting digital model, as they could be inferred from the depth data.

8.2.2 Applications

As the most versatile component of our work, our optical surface tracking system lends itself to a number of potential applications beyond ultrasound simulation. For example, our system could be used to facilitate training of other medical procedures. We can imagine placing an LED at the tip of a mock scalpel to allow surgical students to practice incisions. With the internal projector modification mentioned above, the system could simulate life-like cut marks as the scalpel moves across the surface. One could even envision a scheme to allow interaction with the resulting skin flaps, all the way up to stitching the patient back up.

We can imagine applications beyond the medical realm as well. For example, an artist might print a mask, then experiment with different painting techniques. A tattoo artist might practice their craft on our system to get a feel for how to apply their designs to the contoured surfaces of the human body. Another application might involve generating terrain maps for strategic planning purposes, with the ability to quickly outline regions of interest and identify optimal transportation routes.

Perhaps the best application would be one that is universal. We can imagine creating a version of our system that is flexible enough to support any number of use cases. With the rise in accessibility of 3D printing, an online marketplace could be set up to allow sharing of special-purpose applications bundled with the appropriate 3D models.

Bibliography

Adler, R. and Desmares, P. J. (1987). An economical touch panel using SAW absorption. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 34(2):195–201. 11

Akinci, N., Ihmsen, M., Akinci, G., Solenthaler, B., and Teschner, M. (2012). Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics*, 31(4):62. 20

Arvo, J. and Chelmsford, M. (1986). Backward ray tracing. In *Developments in Ray Tracing, Computer Graphics, Proc. of ACM SIGGRAPH 86 Course Notes*, pages 259–263. 25

Bartels, R. H. and Stewart, G. (1972). Solution of the matrix equation $AX + XB = C$. *Communications of the ACM*, 15(9):820–826. 40

Batchelor, G. K. (2000). *An Introduction to Fluid Dynamics*. Cambridge University Press. 20

Burt, P. J. and Adelson, E. H. (1983). A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236. 10

Collignon, A., Maes, F., Delaere, D., Vandermeulen, D., Suetens, P., and Marchal, G. (1995). Automated multi-modality image registration based on information theory. *Information Processing in Medical Imaging*, 3(6):263–274. 43

Criminisi, A., Pérez, P., and Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212. 11

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. 100

Detmer, P. R., Bashein, G., Hodges, T., Beach, K. W., Filer, E. P., Burns, D. H., and Strandness, D. E. (1994). 3D ultrasonic image feature localization based on magnetic

scanhead tracking: In vitro calibration and validation. *Ultrasound in Medicine & Biology*, 20(9):923–936. 9, 10, 43

Donald, I., Macvicar, J., and Brown, T. (1958). Investigation of abdominal masses by pulsed ultrasound. *The Lancet*, 271(7032):1188–1195. 24

Downs, R. (2005). Using resistive touch screens for human/machine interface. *Texas Instruments Analog Applications Journal*. 11

English, W. K., Engelbart, D. C., and Berman, M. L. (1967). Display-selection techniques for text manipulation. *IEEE Transactions on Human Factors in Electronics*, 8(1):5–15. 11

Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH*, pages 209–216. 89

Gast, T., Schroeder, C., Stomakhin, A., Jiang, C., and Teran, J. (2015). Optimization integrator for large time steps. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1103–1115. 23

Gissler, M., Ihmsen, M., and Teschner, M. (2011). Efficient uniform grids for collision handling in medical simulators. *GRAPP*, 11:79–84. 19

Glassner, A. S. (1984). Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, 4(10):15–24. 31

Glassner, A. S. (1989). *An Introduction to Ray Tracing*. Elsevier. 25

Golub, G., Nash, S., and Van Loan, C. (1979). A hessenberg-schur method for the problem $AX + XB = C$. *IEEE Transactions on Automatic Control*, 24(6):909–913. 40

Hanrahan, P. and Krueger, W. (1993). Reflection from layered surfaces due to subsurface scattering. In *Proceedings of ACM SIGGRAPH*, pages 165–174. 26

Jacob, R. J. (1996). Human-computer interaction: Input devices. *ACM Computing Surveys*, 28(1):177–179. 11

Jensen, H. W. (2001). *Realistic Image Synthesis Using Photon Mapping*. AK Peters, Natick, MA. 25

Jensen, J. A. (1996). Field: A program for simulating ultrasound systems. In *10th Nordic-Baltic Conference on Biomedical Imaging, Vol. 4, Supplement 1, Part 1*, pages 351–353. 8

Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. (2015). The affine particle-in-cell method. *ACM Transactions on Graphics*, 34(4):51. 21

Kim, H.-K., Lee, S., and Yun, K.-S. (2011). Capacitive tactile sensor array for touch screen application. *Sensors and Actuators A: Physical*, 165(1):2–7. 11

Krawczyk, G., Brosch, D., et al. (2007). HDR tone mapping. In *High-Dynamic-Range (HDR) Vision*, pages 147–178. Springer. 77

Kutter, O., Shams, R., and Navab, N. (2009). Visualization and GPU-accelerated simulation of medical ultrasound from CT images. *Computer Methods and Programs in Biomedicine*, 94(3):250–266. 8, 9

Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37. 30

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157. 100

Lyon, R. F. (1981). The optical mouse, and an architectural methodology for smart digital sensors. In *VLSI Systems and Computations*, pages 1–19. Springer. 11

Manna, S. (1992). Space filling tiling by random packing of discs. *Physica A: Statistical Mechanics and its Applications*, 187(3):373–377. 18

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767. 100

Meyer, K., Applewhite, H. L., and Biocca, F. A. (1992). A survey of position trackers. *Presence: Teleoperators & Virtual Environments*, 1(2):173–200. 35

Monaghan, J. J. (2005). Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703. 20

Müller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159. 20

Pagoulatos, N., Haynor, D. R., and Kim, Y. (2001). A fast calibration method for 3-D tracking of ultrasound images using a spatial localizer. *Ultrasound in Medicine & Biology*, 27(9):1219–1229. 9, 10, 43

Peleg, S. (1981). Elimination of seams from photomosaics. *Computer Graphics and Image Processing*, 16(1):90–94. 10

Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318. 11

Perlin, K. (1985). An image synthesizer. *Computer Graphics*, 19(3):287–296. 74

Prince, J. L. and Links, J. M. (2006). *Medical Imaging Signals and Systems*. Pearson Prentice Hall, Upper Saddle River, NJ. 25

Ram, D., Gast, T., Jiang, C., Schroeder, C., Stomakhin, A., Teran, J., and Kavehpour, P. (2015). A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '15, pages 157–163. 21

Saito, T. and Toriwaki, J.-I. (1994). New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition*, 27(11):1551–1565. 58, 80

Savitsky, E. A. (2013). Multimodal ultrasound training system. US Patent 8,480,404. 1

Shiu, Y. C. and Ahmad, S. (1989). Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$. *IEEE Transactions on Robotics and Automation*, 5(1):16–29. 40, 41

Stotz, R. (1963). Man-machine console facilities for computer-aided design. In *Proceedings of the ACM Spring Joint Computer Conference*, pages 323–328. 12

Sylvester, J. J. (1884). Sur l'équation en matrices $px = xq$. *CR Acad. Sci. Paris*, 99:67–71. 40

Szeliski, R. (1996). Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30. 10, 69

Terzopoulos, D. (1986). Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):413–424. 60

Terzopoulos, D. (1988). The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438. 61

Zhu, Y., Magee, D., Ratnalingam, R., and Kessel, D. (2006). A virtual ultrasound imaging system for the simulation of ultrasound-guided needle insertion procedures. In *Proceedings of Medical Image Understanding and Analysis*, pages 61–65. 8, 9, 12