Physically exploring volume data using hand-held mobile hardware

Chris Heinrich

a thesis submitted for the degree of

Master of Science

at the University of Otago, Dunedin,

New Zealand.

18 September 2015

Abstract

Volume data is widely used in many different domains including medical imaging. The primary platform to visualize volume data is usually desktop hardware, while the use of hand-held mobile devices to browse volume data is widely ignored. In this thesis, we present an approach for visualizing volume datasets using standard mobile hardware by implementing a novel interface that turns the mobile device into an exploration tool. The key idea of our approach is that we place a virtual representation of the volume data into the user's environment. The user can physically move the mobile device through their environment to interactively slice through the virtual volume. The slice of the virtual volume that is displayed to the user is based on the position of the mobile device inside the virtual volume. Our presented approach enables users to access volume data using mobile hardware as well as it introduces a novel interface to browse the three dimensional volume data. We conclude this work with an explorative study investigating the usability and utility of our interface.

Acknowledgements

I have many thanks to give to all those made this possible. I would first like to thank my supervisors. Dr.Richard O'Keefe for his ingenuity, patience, enthusiasm and positivity. Dr.Tobias Langlotz who provided guidance, push and always went the extra mile for me.

I would like to thank my family, who I would be lost without. My dad for inspiration, always being there to talk to and always believing in me. My mom for inspiration, having an unrelenting bright outlook and spoiling me with cooking. My sister who has always been there for me, has always outdone me and has kept me looking over my shoulder in a good way. My grandparents who have all given me the love and confidence to keep me going through hard times.

I would like to thank my colleagues. Jonny Collins for always playing devil's advocate and expanding my horizons. Jorg Muller for his friendship, machine-like expertise and humor. Elias Tappeiner for his friendship and belief in the American Dream. Dr.Holger Regenbrecht for his advice and German lessons throughout this process. Finally, the Human Computer Interaction (HCI) group which meets every Wednesday. They all provided feedback, encouragement and guidance throughout this whole process.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Volume Data

Imaging of 3D volume data is commonly used in many different domains such as medicine and geological surveying. Compared to traditional 3D data used in computer graphics, volume data represents a vast amount of information and, thus, is large in size. The sheer size of this type of data makes it computationally expensive to process. This has restricted its use to mainly desktop computers where volume data can be interactively visualized because of their hardware capabilities, in particular, the graphics card. Recently, mobile hardware became more powerful to the extent where it is theoretically possible to store and render volume data on off-the-shelf hand-held mobile devices. Because of their form factor, input capabilities, and integrated sensors mobile devices offer many unique interface possibilities compared to desktop computers. This raises the question if we can implement a real-time, interactive interface to view volume data on hand-held mobile devices and what effect would this have on the user's experience?

3D volume datasets are datasets that contain information representing every volumetric unit (voxel) inside a given volume. 3D volume data is unique and differs from conventional 3D data. Conventional 3D data represents some information in three dimensions, but it doesn't contain information for every volumetric unit. What conventional 3D data typically shows is surface data, with no information about what is beneath this surface. This leaves a wealth of information under the surface that is not represented. Volume data on the other hand, represents every unit in the volume which adds a multitude of information to the dataset.

The interface to interact with these volume datasets on the desktop computer consists of displaying the dataset on a 2D monitor and using a keyboard/mouse to interact with the data. We can usually rotate and move through the volume, but are limited by the 2D interaction to browse and view the 3D volume data. This presents a dilemma where one has three dimensional data, but one is viewing and interacting with it in a two dimensional way.

The area of Computer Graphics that deals with visualizing volume data is referred to as volume rendering. This area encompasses different techniques used to display these volume datasets. When one is trying to visualize a given volume, it is likely that one is only interested in visualizing a particular aspect of the given volume. Therefore, the Volume Renderer renders parts of the volume irrelevant (transparent) to gain depth through the volume and to visualize the portion of the volume you want to see. Algorithms to accomplish this are computationally expensive. Desktop computers contain dedicated graphic cards that feature graphic APIs which allow them to handle complex graphical computations. However, more and more mobile devices have powerful graphics cards which has opened new possibilities for what is possible to accomplish on mobile devices.

Mobile devices have become increasingly prevalent in peoples everyday lives. Zickuhr and Rainie (2014)[1] ran a survey that found in 2014, 61% of Americans own a smartphone and 42% of Americans own a tablet computer. With students, it is even more prevalent. Dahlstrom and Brooks (2014)[2], ran a study that found in 2014, 86% of students own a smartphone and 47% own a tablet computer. They estimate that by 2015, 91% will own a smartphone and 58% will own a tablet computer. As mobile hardware usage has increased their hardware has advanced as well. This advanced hardware has led to the improved graphical capabilities and processing power of these

[1] http://www.pewinternet.org/2014/01/16/e-reading-rises-as-device-ownership-jumps/

[2] http://net.educause.edu/ir/library/pdf/ers1407/ers1407.pdf

devices. It hasn't quite reached the performance capabilities of desktop computers, but we are now able to do many tasks on a mobile device that previously have been constrained to a desktop computer.

1.2 Motivation

As stated initially, volume data is commonly used in the medical domain. It is created from scanners which use techniques such as magnetic resonance imaging (MRI) or computerized axial tomography (CAT) scan. These imaging techniques produce volume datasets by combining 2D images taken at regular intervals throughout the volume. These combined images are collectively used to represent and visualize part of the human body or the human anatomy. They are used for many medical purposes such as diagnosis, surgery planning and education.

In education, a Harris Interactive (2013)[3] survey was conducted on students from the ages of 8-18 which reported their attitude towards the use of tablet computers in education. They found that 92% of those students agreed "tablets will change the way students will learn in the future", 90% agreed that "tablets make learning more fun" and 82% agreed that "tablets help students do better in classes". Tablets have advantages over smartphones because of their increased size and what that size entails. The increased size allows for increased hardware specifications and display capabilities over their smartphone counterparts. Studies, such as Desjardins, van Oostveen, Muirhead, and Goodman (2011), look into tablet computers and how they can be applied to education/learning. However, the results from that study were inconclusive. The question still remains, can tablets be useful in education/learning and for what purpose?

The human anatomy is something we know a lot about, however, we still struggle to teach and visualize it in a technological way. For medical students, visualizing and learning the human anatomy is a difficult task. Waterston and Stewart (2005) ran a survey of practicing clinicians and found that the majority of clinicians feel that current anatomical education given to medical students is inadequate and below the necessary safe levels for medical practice. This is backed up by Collins (2008) who quoted Andrew T Raftery as saying there is a "crisis" in the teaching of anatomy because of the increase in claims relating to lack of anatomical knowledge. Medical students typically attend lectures where they are presented anatomical structures and then dissect cadavers to

[3]https://www.pearsoned.com/wp-content/uploads/Pearson-Student-Mobile-Device-Survey-2013-National-Report-on-Grades-4-to-12-public-release.pdf

gain a real visualization of the structures. Collins (2008) says exploring cadavers is done to help gain an understanding of the three dimensional structure of the human body. He says it also helps develop spatial reasoning skills necessary to understand and interpret imaging [volumetric] data. McLachlan, Bligh, Bradley, and Searle (2004) explains, however, that some medical schools are getting rid of the use of cadavers entirely. This is relating to the emerging belief that cadavers do not provide the benefits previously thought. They argue that information gathered from cadavers do not translate to living anatomy or to the commonly viewed cross-sections of medical imaging (such as MRI/CAT scans). As a secondary motivation for their removal, they argue cadavers are expensive, cause a portion of students to experience mental and physical discomfort and present ethical issues. On the other hand, current available technology and resources available to students are typically limited to 2D diagrams, pictures, 2D image slices, anatomical slices, and mobile applications that show virtual 3D renderings of the human anatomy. With the students belief, from survey above, that tablets can provide enticing benefits to learning, we are curious to explore this further in anatomy education.

1.3 Research Question

Our initial research question is 'If we can implement a real-time, interactive interface to view volume data on mobile devices, what effect would this have on the user's experience?' Our research question is consequently split into two areas: Implementation of an prototype application including an interface for browsing volume data and analysis of the realised prototype and application using user feedback.

For the implementation, we wanted to explore 'Is it possible to store and interactively visualize volume data on a hand-held mobile device?' We explore if this is feasible with current mobile hardware specifications and what constraints they present. Mobile devices also offer unique opportunities as they require and support different interfaces compared to traditional desktop computer interfaces. While mobile devices mostly lack input devices such as a mouse and a physical keyboard, they integrate touch screen, cameras, orientation sensors, and sensors that can be used for implementing an interface in a portable hand-held design.

For analyzing the prototype and its application, we wanted to evaluate 'How does using mobile hardware affect the user's experience visualizing and interacting with volume data?' We wanted to explore 'In what domains would an interface to explore

volume data on a mobile device be useful?' Specifically, 'Could this interface be useful in an education context, particularly in the scope of anatomy education?'

1.4 Requirements and Constraints

With the frequent use of volume datasets in the medical domain, we decided to focus on visualizing the human anatomy using mobile hardware. We decided upon using a tablet device as our mobile hardware (over a smartphone) because of its storage capacity, processing power and display capabilities which still outperform smartphones. Additionally, we chose a specific dataset, the Visible Human Project dataset, as our human anatomy volume dataset. The Visible Human Project dataset is made up of a collection of images that were taken at regular intervals throughout a human cadaver. Now that we had our volumetric dataset, we needed to figure out how to use a tablet device to explore it.

We intend to adapt ideas from augmented reality (AR) and, in particular, magic lenses for our work. Augmented reality refers to capturing a user's environment through a mobile device's camera and augmenting that environment with virtual content. AR typically uses a device to capture its environment by using its camera and overlaying some part of the environment with digital content in real time. In order to correctly do this, we must know the position and orientation of the device and what part of the environment to overlay. Usually, a visual marker is placed in the user's environment and digital content is overlaid on top of this. This visual marker is analyzed by the device and allows the device to compute where it is in relation to the visual marker. One is usually able to get the position and orientation of the camera (based on the visual marker) in the user's environment. We wanted to implement a similar interface, but without showing the camera feed and instead focusing on the virtual volume data.

Our goal was to use the tablet device as an exploration tool to explore the human anatomy. As we will later demonstrate, we ran into hardware constraints that forced us to reduce the amount of the human anatomy that we could store and render. This led to us only rendering the neural anatomy portion of the Visible Human dataset (instead of the whole body). We wanted to explore the uniqueness of a tablet device (namely its size and hardware) and how this could be used to interact with volume data. We wanted the interaction to occur in real time and for all the processing to be handled on the tablet device. We wanted to use readily available, off-the-shelf hardware because that is what the everyday student would have access to.

On the software implementation side, we defined certain requirements. We decided upon 15 frames per second (fps) as the minimum frame rate that would be acceptable for real-time, user interaction as 15fps is widely considered as required for interactive feedback. We decided that compression of volume data is out of the scope of what we wanted to accomplish (namely the interaction of mobile hardware with volume data). With our goal to do all the processing on the tablet, we decided that pre-processing of the volume data into optimized data structures was acceptable. This wouldn't affect the integrity of the dataset, but improve the performance of the application.

1.5 Contribution

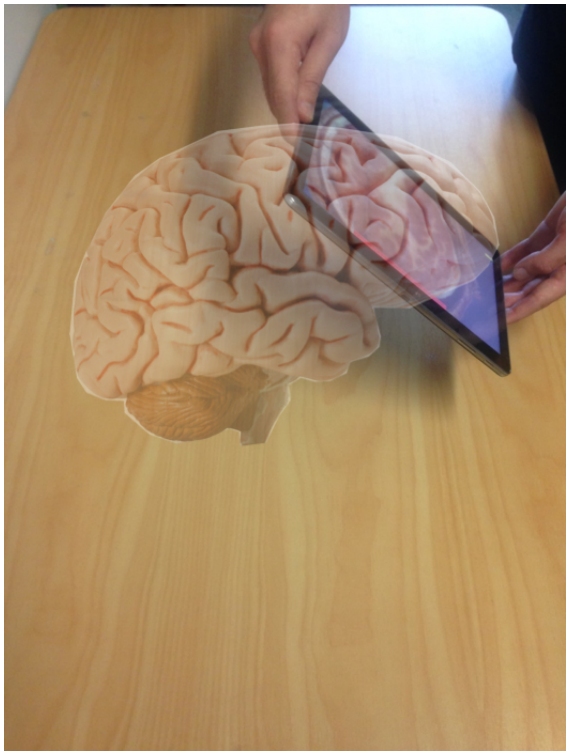


Figure 1.1: Our interface concept that uses a tablet device as an exploration tool to physically explore volume data

What is the point of three dimensional data if it doesn't feel 3D to the user? With the requirements from above in mind, we proposed an interface that allows the user

to explore a volume dataset (using a tablet device) by way of physically moving the device through the volume data. The volume dataset is stored on the tablet and is mapped into the user's environment based on the actual physical dimensions of the volume. We will use tracking technology also used for AR to get the real-time position of the tablet device and be able to tell where it is inside our virtual volume. Since the tablet screen is 2D, what is displayed to the user is a cross-section of the volume data based on where the tablet device is inside the virtual volume. The dimensions of the dataset and tablet device are taken into account which allows the user to view the dataset with its real scale. The end result being, the slice of the data that you see on the tablet screen is based on where the tablet is inside the virtual volume. This interface allows the user to physically interact with the data in 3D space and is to the best of our knowledge a novel contribution.

We also evaluate the usability and utility of our interface by collecting early feedback on our interface from domain experts. This was because they would provide high quality feedback and it would allow us to use a small number of participants to gain meaningful feedback. Since we decided to render the neural anatomy portion of our human anatomy volume dataset, we sought feedback from neural anatomy domain experts. Our evaluation consisted of six neural anatomy domain experts, including four neural anatomy professors. The feedback we got indicated unanimously that the interface was easy to learn and intuitive to use. We received strong feedback that the utility of our application could be centered around education and could help users understand the three dimensionality of neural anatomical structures. One of the main disadvantages of our interface that we found was the limited viewing angles of the volume data because the camera always needed to see the visual marker. We acquired feedback on future work that could be done with this type of interface and believe it should be explored further.

Overall, we argue that our work is a contribution to the field of mobile human-computer interaction as well as to e-learning using mobile devices.

1.6 Thesis Outline

This thesis aims to show the steps taken to explore our idea of physically moving through volume data using mobile hardware.

1. **Related Work (Chapter 2):** We look at published work in the related fields of Augmented Reality, magic lens interfaces, and volume rendering. We also present some works that has been done with our volume dataset, the Visible Human Project.

2. **Setting the Context of this Project (Chapter 3):** This chapter introduces our mobile device, volume dataset, programming environment and programming languages.

3. **Tracking the Device Position (Chapter 4):** We discuss different approaches for tracking our device using the integrated hardware.

4. **Rendering (Chapter 5):** We explain how we render the slice of our volume data based on the tablet's position.

5. **Evaluation (Chapter 6):** An evaluation of our application was conducted to get feedback on its usability and utility.

6. **Conclusion (Chapter 7):** We discuss future work that could be done on this project and give a summary of what was accomplished.

Chapter 2

Related Work

Contents

In this chapter, we present our literature review. This project combines different areas in computer science to implement our approach of physically slicing through volume data using mobile hardware. The two that have the most influence are augmented reality and volume rendering. We want to stress that this project doesn't fit squarely into one single area of computer science. It draws upon different aspects of both areas to create our interface.

2.1 Volume Rendering

Volume datasets are made up of a number of voxels (volume pixels/3D pixels). These voxels typically contain information such as colour and opacity (how transparent this voxel is). This can be represented as RGBA and stored using 4 bytes. Volume datasets in medicine can be created by using a scanner (such as MRI or CT scanner). These scanners scan through a volume and record the density throughout it. A transfer function can be defined that maps density to colour for every voxel in the dataset. This is used in a medical context to render, for example, the bones a certain colour that allows you to easily distinguish them from the rest of anatomy. Volume datasets

are typically large in size and this presents rendering constraints based on the hardware and software used to render it. They are typically large in size because every voxel can be represented by 4 bytes (RGBA). This means that for a relatively small volume dataset of size 1000px x 1000px x 1000px, it would contain 1,000,000,000 voxels. With each voxel being 4 bytes, this is 4,000,000,000 bytes which is approximately 4GB of data. As you increase the resolution along any of the axes, the size of the volume dataset greatly increases.

Volume rendering refers to the area of computer graphics that entails different techniques for displaying volumetric data. The concept of volume rendering was first presented in Drebin, Carpenter, and Hanrahan (1988). This paper presents a technique for rendering volume data by shading voxels based on the blending of surface data and the data underneath to provide a better representation of the volume being rendered. This is largely referred to as Direct Volume Rendering (DVR) which is defined by Ikits, Kniss, Lefohn, and Hansen (2004) as "methods [to] generate images of a 3D volumetric data set without explicitly extracting geometric surfaces from the data". This is put in more simple terms by Kindlmann (2002) who explains DVR has "direct mapping from volume data points to composited image elements". In our Volume Rendering portion of the literature review we mainly discuss two ways to render volume data: slice-based rendering and direct volume rendering. Slice-based rendering refers to displaying a slice or cross-section of the volume dataset. By creating a cross-section, or slice, of the volume data you are rendering along some defined plane through the volume. This is what we use in our approach, however, we surveyed both approaches for our literature review.

2.1.1 Slice-based Rendering

Our approach slices through the volume using the mobile hardware as our plane. We are, therefore, observing cross-sections of the volume dataset. Goble, Hinckley, Pausch, Snell, and Kassell (1995) introduced a two-handed interface tool to navigate 3D volume data. Their system was called "Netra" and was designed to be used for neural surgical training. Their two-handed interface consisted of a plate device that was held to a dolls head to determine the cross section of the brain to display. The second device was a stylus pen that determined where a virtual probe was located in the virtual head. Using this two-handed interface they were able to interactively view cross sections of the brain. This early paper shows relevance to our application because they use an input device to view different cross sections of volume data without introducing transparency

to gain depth through volume.

Fröhlich and Plate (2000) presented an approach that allows you to interactively browse volume data. They present the cubic mouse which is an input device that allows users to specify three-dimensional coordinates. The device "consists of a cube shaped box with three perpendicular rods passing through the center. The rods represent the the X,Y, and Z axes of a given coordinate system. Pushing or pulling of the rods specifies constrained motion along the corresponding axes". By moving the different rods, you are moving the 2D cross-section of that axis of the volume data. They provide an example with a neural anatomy volume dataset. They show that the three rods represent different anatomical planes (Figure 7.2 on page 59) of the neural anatomy. By pushing or pulling one rod, you are moving the 2D cross-section for one of these three anatomical planes. You are able move these planes interactively. Today in medicine, doctors still view volume data by moving 2D cross-sections along these three axes. They use a keyboard and mouse instead of the cubic mouse, but the idea is still relevant. This important paper allowed for interactive viewing of cross sections in medical volume datasets using a novel input device.

Qi and Martens (2005) explored the problem of visualizing (via volume rendering) 3D data on 2D interfaces. They focused on the position of the clipping plane within volume rendered data. They proposed three interfaces using a cube device with different secondary devices such as a mouse, stylus pen and plane (square) shaped device to interact with the data. Bornik, Beichel, Kruijff, Reitinger, and Schmalstieg (2006) presented a system for interactive visualization and manipulation of medical datasets for surgical planning. They used a tablet device with a stylus-style input device to manipulate data. This unique stylus-styled input is called "Eye of Ra" and has 2 buttons plus a scroll wheel to trigger 3D interaction tasks.

2.1.2 Direct Volume Rendering

The four most popular techniques according to Meißner, Pfister, Westermann, and Wittenbrink (2000) used for direct volume rendering are: Raycasting, Splatting, Shear-wrap and 3D texture-mapping hardware-based approach. We briefly explain one of these techniques, Shear-wrap. Shear-wrap volume rendering was popularized by the paper by Lacroute and Levoy (1994). This paper explains how they transform the volume by shearing the 2D slices so that rows of voxels can be aligned with a row of pixels in the output rendered image. This resulted in them being able to render a 256^3 voxel volume in 3 seconds. This technique allows for faster volume rendering at the

cost of quality. Meißner *et al.* (2000) explains that this "has been recognized as the fastest software renderer to date". These DVR techniques allow you to observe the volume data after parts of it have been rendered transparent (to gain depth through the volume).

These previous approaches have used desktop hardware, but a recent paper has looked into using mobile devices for volume rendering. Noguera, Jiménez, Ogáyar, and Segura (2012) compares several methods for interactive volume rendering on mobile devices. They use OpenGL ES 2.0 which is the previous version of the software we use (OpenGL ES 3.0). They don't explore the use of 3D textures in their approach because they explain that "most OpenGL ES 2.0 implementations available in todays mobile devices do not support 3D textures". They implemented an approach that aimed to limit complex computations on the shader and account for this lack of 3D textures. They compare their approach against raycasting and found that their approach nearly doubles it in performance (fps). This paper doesn't involve using the mobile device as a exploration tool and the interactive aspect comes in moving the viewing plane.

2.1.3 Distinctions from This Project

Our approach differs from direct volume rendering for a couple reasons. Firstly, the rendered volume isn't observed from afar. Typically in DVR you define a camera position in relation to the volume data. In our application, the slice of the rendered data you are viewing is based on where the tablet is our volume. Therefore, the camera is inside the volume instead of afar. We are using the tablet as a plane to view cross-sections of our volume data. Secondly, the rendered data hasn't been changed in any way. In DVR, typically the pixels are changed according to the transfer function. This allows for transparency to be introduced to some of the voxels so that you can see further through the volume. This transparency, however, means you have changed the surface data to gain depth through the volume. Our volume dataset (Visible Human) isn't transparent in any way, but you could use a transparent dataset and our interface would still perform in its intended form.

2.2 Visible Human

The Visible Human Project dataset has been used largely for two purposes: imaging and compression. Like us, many researchers have used this dataset to visualize the human anatomy. However, researchers have also used this dataset to develop new al-

gorithms for compressing and transferring such large volume datasets. As we discussed in the introduction, compression is out of the scope of this thesis. We will, however, discuss some work that has been done in this area for potential future work. Zamora, Wilson, Mitra, and Thoma (2000) implemented a web-based system to distribute the Visible Human dataset. They use a 2-step lossless compression algorithm to send the large volume dataset over the internet. They achieved an average data reduction of 7.8:1. Rodler (1999) used the Visible Human dataset to present an algorithm that uses wavelet based compression for large volumetric data. They reported a 50% reduction in size compared to previous best known results. Nguyen and Saupe (2001) used the Visible Human dataset to present an algorithm for "rapid high quality compression of volume data". They used a voxel (block) based approach and created a 512^3 volume which is approximately 256 MB They report that their approach compressed the data 6 times more than the previous state of the art approach. This wasn't our focus, but future work on this interface could look to implement compression to possibly store more of the volume data then we were able to do.

Rigamonti, Bryant, Bustos, Moore, and Hoffman (2000) presented a paper that is relevant to our focus because they explored using the Visible Human dataset to teach neural anatomy. They provided a virtual dissection room for students to interact with the Visible Human dataset. This paper explained why the traditional way to teach anatomy is complicated and the benefits of a virtual approach. Some of the reasons they used to argue traditional methods to teach anatomy are complicated include: limited number of dry skulls available to anatomy departments (lots of people using one skull), difficulties distinguishing normal details from student-made errors, fragility of many contributing bones and the impossibility to teach all the connected openings once the skull has been broken into. They explain how some medical schools have turned to plastic models to teach the neural anatomy. These have their drawbacks as well, which they argued are that "no plastic skull can faithfully reproduce the true three-dimensional feel, palpable, measurable anatomy of a real-life skull". Their interface uses volume rendering to display the dataset in this virtual dissection room. They show models of the neural anatomy which can include artifical colours to distinguish certain structures or regions. Transparency is used in these to gain depth through the volume rendered. This project also featured creating cross-sections based on moving the anatomical planes. This application was an early application that explored using volume data to teach the neural anatomy.

Our approach uses standard mobile hardware to display the Visible Human dataset,

while Guthe, Wand, Gonser, and Straßer (2002) used standard PC hardware to display the same dataset. They presented a "new algorithm for rendering very large datasets at interactive framerates on standard PC hardware". They used a 3D texture-mapping hardware-based approach as their volume rendering technique. Their algorithm uses wavelet based compression which is out of the scope of this thesis. They achieve a frame rate of 10 fps on low quality. The Visible Human dataset has been largely used for compression and visualization research, but there is a wide range in how these tasks have been implemented.

2.3 Augmented Reality

Azuma (1997) defines Augmented Reality (AR) "as any system that has the following characteristics: combines real and virtual, is interactive in real time, and is registered in three dimensions". To achieve this, the software must know where in the real environment to overlay some type of virtual content. This is commonly done by introducing a visual marker into the user's environment. The device uses a camera to analyze the marker and is able to compute its pose relative to the marker. Using this pose, you can correctly align the virtual content so that it looks correct on the screen given some user's perspective of the scene. Another technique to gain the user's location in the real environment is done by using GPS. However, according to Milette and Stroud (2012) this this isn't suitable for indoor environments and takes a substantial amount of time to acquire their position. This led us to not consider this as a possible option for acquiring the device's position. AR is used in many different domains and this has led to a wide range of implementations of AR software.

To understand the current state in AR, survey papers were used to to provide broad context in the available frameworks and how AR applications are currently being used in different fields. Van Krevelen and Poelman (2010) and Carmigniani, Furht, Anisetti, Ceravolo, Damiani, and Ivkovic (2011) provided a solid overview of available AR toolkits and different applications (notably medical applications). Craig, McGrath, and Gutierrez (2011) offered not only available AR toolkits, but also reasons for why they prefer Android over iOS for AR applications. Gotow, Zienkiewicz, White, and Schmidt (2010) explored three key challenges for AR developers on smartphone devices. Two of the challenges relate to work done in this thesis: filtering raw sensor data and implementing a magic lens. They explained how hard it is to implement a magic lens interface on "resource-constrained smartphones". This is because of the

image taken from the mobile camera must be "transformed and rendered in real-time according to information about the user's position, orientation, and heading within the environment". They went on to explain how geomagnetic sensors on mobile devices (which can provide user heading information) can be combined with GPS to estimate field of view (which is used in some AR applications). They cited a problem that the accuracy of these sensors is not adequate. They explained that a method to filter sensor data, the Savitzky-Golay smoothing filter, is not usable in mobile AR because the algorithm is computationally expensive and infeasible on mobile hardware. Rose, Potter, and Newcombe (2010) focus on AR's use strictly in an educational context, including medical imaging. They showed how physicians place markers on a patient body and these are used to show the patient superimposed with "augmentations of human body components" which is used to improve understanding between the two. They also explored rising smartphone/tablet ownership rates and interface designs for such devices. Kalkofen, Reitinger, Risholm, Bornik, Beichel, Schmalstieg, and Samset (2006) discussed how to expand Augmented and Virtual Realities' use in the medical context. They wanted to expand the use of these technologies to "cover the whole clinical workflow from pre-operative to intra-operative and post-operative support". They explored using these technologies for surgical planning, surgical simulation (with haptic feedback) and displaying information to the patient. AR has been used in the medical domain to provide understanding to patients and physicians. These technologies, however, use largely 3D reconstructions of anatomical structures. These 3D reconstructions have a drawback of not resembling real anatomical structures. AR presents applications that can be used, especially in the medical domain, to provide new understandings to the patient and physician.

2.3.1 Distinctions from This Project

Our approach differs from Azuma (1997)'s definition of augmented reality for one main reason, the user's environment isn't shown back to the user. The marker based tracking serves only to give us the device's position so that we can calculate where in our virtual volume (which is placed in the user's environment) the device is. It isn't used to augment the user's real environment with virtual objects. The camera is never used to capture the user's environment for the purposes of relying it back to the user. Therefore, this project occurs in virtual reality, not augmented reality and does not combine the two.

2.4 Magic and Volume Lens

In this section, we discuss research about interfaces and how they are used to interact with volume data. We mainly focus on the magic lens, but briefly touch on others. The magic lens interface was first introduced by Bier, Stone, Pier, Buxton, and DeRose (1993). It was proposed in design only, but introduced the concept of "incorportating visual filters that modify the presentation of application objects to reveal hidden information and enhance data of interest". This interface provides "context dependent" feedback to the user and serves as the basis for modern augmented reality. Magic lens, as used in today's augmented reality, provides a handheld lens into the real environment. This magic lens determines how you interact with the virtual world.

Viega, Conway, Williams, and Pausch (1996) introduced the term volumetric lens which is a magic lens in a volume context. Looser, Billinghurst, and Cockburn (2004) used this volume lens in augmented reality interfaces. One example they use is of a rendered house and the volume lens reveals details underneath such as wall beams. This type of interface allowed you to see underneath the surface information, but you couldn't gain further depth. You were basically shown 3D surface data and provided a volume lens that changed it to other "depth" 3D surface data depending on where the volume lens was positioned. This implementation of volume lens isn't similar to what we want to in our approach. We want to move the volume lens deeper to gain depth through the volume instead of keeping the volume lens outside the surface of the volume. This means we use the volume lens to move through the volume, while they use the volume lens to reveal certain, predetermined depth elements of their 3D scene.

Mendez, Kalkofen, and Schmalstieg (2006) present an interaction tool based on magic lens that allows for context sensitive augmentations to occur. What this means is that the context sensitive magic lens affect objects in a scene differently depending on their context. They present different applications including a medical one that depicts a tumor and displays relevant information of surrounding structures. They expand upon this in Kalkofen, Mendez, and Schmalstieg (2007) by adding "focus objects" into the scene. These objects are both overlaid and partially occluded by the context objects. They now call it Focus and Context (F+C) visualizations for augmented reality. Finally, the trio published Kalkofen, Mendez, and Schmalstieg (2009) which adds to this F+C visualizations by adding "support for comprehension of spatial relationships between virtual and real-world objects".

Gallo, Placitelli, and Ciampi (2011) present a system that explores medical images using a Microsoft Xbox Kinect as the input device. This allows the user to interact with the data using hand and arm gestures. They also detailed interaction techniques specifically designed for deviceless exploration of medical image data. Baricevic, Lee, Turk, Hollerer, Bowman, *et al.* (2012) presented the results of their user study that used geometrically correct user-perspective (as opposed to device-perspective) rendering using magic lens. This study used both tablet and phone sized displays to complete a selection task. They found that the tablet sized displays were significantly faster at completing their selection task.

2.5 Discussion

Our approach to physically slice volume data using a mobile device as an exploration tool mainly encompasses two areas of computer science, volume rendering and augmented reality. Volume rendering is a broad area that utilizes many different techniques to display volume data. Our approach is only interested in creating/viewing cross-sections through the volume data. We are not interested in using our mobile hardware to view a fully sampled rendering of the whole volume. However, it would be possible to change the opacity of our volume data to be able to more clearly identify certain regions/structures. Displaying our volume data is, merely, one aspect of our interface.

The second aspect of our interface is knowing the position of the mobile hardware in the user's environment and, therefore, where it is inside the virtual volume we place in that environment. In augmented reality, knowing the device's position and orientation is vital to correctly augmenting the user's environment (as captured by a device's camera). Magic lens is an interface that is commonly used in Augmented Reality that represents a lens into the real environment to display context-driven digital content. Our approach is similar to a volume lens because the tablet screen can be seen as a lens into our volume data. By moving this volume lens through the volume data, the user is shown context (position) dependent content (cross-sections).

Chapter 3

Setting the context of this project

Contents

In this chapter, we present context information about our project. We introduce the volume dataset that we chose. We discuss selecting our tablet device and the decisions that went into that. We conclude the chapter by talking about some programming choices we made with programming languages.

We set out at the beginning of our project with a straight forward idea, physically explore volume data using hand-held mobile hardware. We then had to start answering some of the "how" and "what" questions. How do you plan on implementing this? What device are you going to use? What is your volume data? These decisions required careful consideration because a poor decision could hinder the whole interface. We started with choosing our volume dataset as this decision would impact the others.

3.1 Introducing the Visible Human dataset

We chose the Visible Human Project[1] as our volume dataset. As shown in the literature review, this dataset has been used for visualization purposes (displaying the human

[1] www.nlm.nih.gov/research/visible/visible_human.html

Table 3.1: Visible Human Dataset

	Male	Female
# Images	1,871	5,189
Resolution(Pixels)	2048 x 1216	2048 x 1216
Cut Interval	1mm	0.33mm
Size	15GB	40GB

anatomy) and for algorithmic purposes (how to process/transfer such large datasets). We chose this dataset because of its price (free), fame (as a volume dataset in published works) and attainability. This dataset consists of two datasets: a male and a female. The male was taken first and was completed in 1994. The female was completed in the following year, 1995. These datasets are free, however, you must contact the Visible Human Project and state what you want to use it for before being given an FTP address to download the datasets. These datasets were created by surrounding a cadaver with liquid and then freezing it. Slices were made at regular intervals through this frozen cadaver. These slices were then photographed and digitalized. This resulted in a volume dataset of the human anatomy. The male dataset had cuts made every 1mm (meaning a slice was 1mm thick which corresponds to the depth defined as Zdepth), while the female every 0.33mm. The female cut interval was significant because the researchers had created these datasets for the purpose of visualizing the human anatomy. For both these datasets, the image resolution for each slice was set so that 1 pixel in the dataset slice corresponds to 0.33mm in width and height of the actual volume. This means that the actual (physical) size of each volume can be computed as follows:

Visible Human Male dataset

$$Xwidth = 2048\text{px} \times 0.33\text{mm} \tag{3.1}$$
$$Xwidth = 675.84\text{mm}$$

$$Yheight = 1216\text{px} \times 0.33\text{mm} \tag{3.2}$$
$$Yheight = 401.28\text{mm}$$

$$Zdepth = 1871\text{slices} \times 1\text{mm} \tag{3.3}$$
$$Zdepth = 1871\text{mm}$$

Visible Human Female dataset

$$Xwidth = 2048\text{px} \times 0.33\text{mm} \tag{3.4}$$

$$Xwidth = 675.84\text{mm}$$

$$Yheight = 1216\text{px} \times 0.33\text{mm} \tag{3.5}$$

$$Yheight = 401.28\text{mm}$$

$$Zdepth = 5189\text{slices} \times 0.33\text{mm} \tag{3.6}$$

$$Zdepth = 1712.37\text{mm}$$

The Visible Human male volume ends up being 675.84mm x 401.28 x 1871mm (by using equations 3.1, 3.2 and 3.3). The Visible Human female volume ends up being 675.84mm x 401.28 x 1712.37mm (by using equations 3.4, 3.5 and 3.6). The X and Y resolution are the same for both subjects. The male was taller than the female, hence, a longer volume. We will use this later in the Rendering Chapter for real-scale rendering.

Researchers found after the male dataset, taken first, that researchers preferred cubic voxels. The male dataset could be represented as 0.33mm x 0.33mm x 1mm voxels. However, these aren't cubes. When the female dataset was taken, they made the cuts every 0.33mm to create perfect cubic voxels of 0.33mm x 0.33mm x 0.33mm. We chose to use the Visible female dataset as our volume dataset because it had cuts at shorter intervals and, therefore, a higher resolution and quality. We didn't know if this extra resolution would be used for at the end, but we figured it could be nice to have. With our volume dataset chosen, we had to decide what mobile hardware we wanted to use to display it.

3.2 Choosing the tablet device

When considering what tablet device to buy, the first decision we came across was whether to choose an Apple (iOS) or Android tablet. We decided on Android for a couple reasons. We referred to the paper by Craig *et al.* (2011) from the literature review about their recommendation. This paper provided three reasons for using Android over iOS. Firstly, they pointed out how Android is open platform and has many

different vendors and devices available. This was important because our requirements require a mobile device with high-end hardware specifications and Apple offers only a limited number of models to choose from. Apple also requires a license fee to develop on the iOS SDK, while Android is free. Secondly, they explain that there a good number of SDKs available for Augmented Reality on Android. This was important for us because we didn't know what software we were going to use at the beginning of the project and having more options was advantageous. Lastly, the authors believed that Android is well engineered/documented and is easier to program than iOS. We came into this project with limited experience programming for iOS, so the option to work with a more familiar language (Java) was enticing.

Apart from the reasons provided by Craig *et al.* (2011), Android has in addition the largest market share and thus the larger user base which is a very important point when deciding to prioritize platforms. Ong (2014) reports that in 2014, Android's share of the smartphone global market was 84.6%. We decided on Android for the reasons provided for our prototype implementation.

After we chose our operating system, we had to decide on what Android mobile device to buy. We narrowed our choices by looking at their hardware and made sure they satisfied our requirements. The Visible Human female dataset is 40 GB, so we needed a device that could store the entire dataset. This was because our initial goal was to render the whole dataset. We were also looking for a device with the latest generation GPU supporting the recent OpenGL ES 3.0 because OpenGL ES 3.0 introduced support for 3D textures, something that could be useful when implementing volume rendering. We also wanted a display that could display the dataset in high or even it's original resolution of 2048 x 1216 pixels. After considering different brands and models, we decided upon on the 2013 Samsung Galaxy TabPRO 10.1 as our hand-held mobile hardware.

3.3 Introducing the tablet device

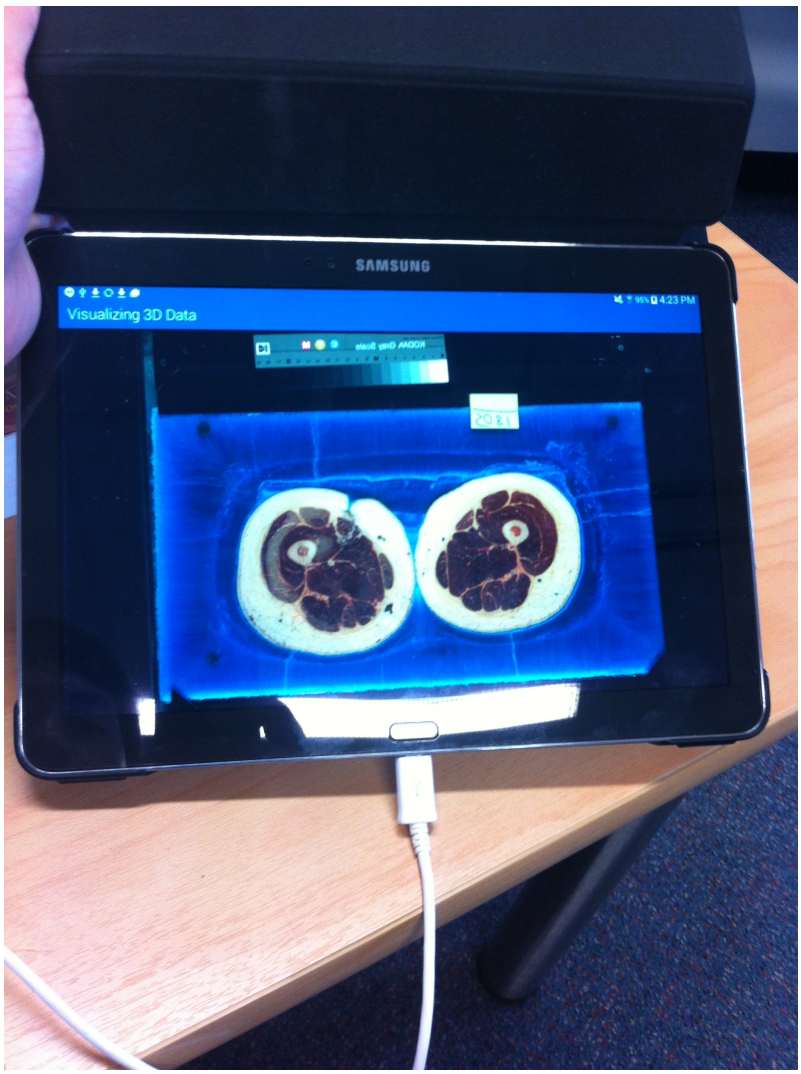


Figure 3.1: 2013 Samsung Galaxy Tab Pro 10.1 displaying a slice of the Visible Human dataset

The 2013 Samsung Galaxy TabPRO 10.1 tablet computer was chosen as it satisfied our requirements and was readily available in local hardware vendors. Figure 3.1 shows our tablet displaying a slice image from the Visible Human female dataset. The physical dimensions for this device are shown in Appendix A.3. Overall, we were pleasured with this mobile hardware because it was able to satisfy our hardware and software requirements.

3.3.1 Hardware

Table 3.2: Samsung Galaxy TabPRO Hardware Specifications

	2012	**2013**	2014
Processor	Dual Core	2 Quad Core (ARM)	2 Quad Core (ARM)
Memory	1GB	2 GB	3GB
Storage(Internal SSD)	16 GB,	16 GB	16GB
Storage(External)	32 GB	64 GB	128GB
Camera (Rear facing)	3.2MP	8MP	8MP
Camera (Front facing)	0.3 MP	2MP	2MP
Display Resolution(px)	1200 x 800	2560 x 1600	2560 x 1600
Graphics GPU Cores	2	6	6

The mobile hardware for our device was top-of-line at the time we purchased it. Figure 3.2 shows the hardware for our tablet (2013) and a brief overview of how mobile hardware has advanced before and after our chosen model. The 2013 model was able to store our 40 GB of Visible Human female volume data entirely on the device. This would mean that we would not have to explore compression of our volume data or data transmission to our device. In the end, the hard drive storage didn't turn out to be as constraining as we thought. The memory turned out to be a bigger constraint on how much volume data we were able to interactively browse. This is because the amount of memory allocated to the GPU that were were able to access was limited (300MB). We made the decision to focus this thesis on physically exploring volume data instead of trying to render the whole volume dataset. It would be interesting to see our same implementation on the 2014 model and see if they allocate more memory for developers to use with the GPU. If more memory is allowed for us to use on the GPU, we could theoretically browse a larger sized volume using our approach. Already in the planing stage we considered also using the camera and marker tracking for getting pose information of our device. The rear-facing camera was the one that we could use to capture our environment and visual markers. This camera was 8MP which we thought would be adequate in case we decided to use the camera for vision-based pose tracking. The display resolution was 2560px by 1600px which was more than the Visible Human female dataset (2048px by 1216px). This didn't end up mattering though as we had to reduce the resolution of our image slices to create more depth (Z-axis) for our virtual volume. The hardware for the 2012 vs our 2013 model appears to have advanced

substantially. The memory was doubled, External Storage was doubled, Rear facing camera MP resolution was doubled, GPU cores was tripled and the resolution of the screen was doubled. This same jump did not occur from 2013 to 2014. The only notable differences are External storage capacity (which wouldn't of impacted us in any way) and the memory was increased by half (which potentially could have affected us).

3.3.2 Sensors

Mobile devices come with a number of different sensors that are used for many purposes. Our device contained an accelerometer and gyroscope sensor which will be explained in the next chapter. It also contained a light sensor. This measures light on the front of the device and can be useful for automatically dimming the screen in bright environments. It also can be uesd as a proximity sensor that measures light to tell when something is close to it. This is useful on mobile phones when a person moves the phone next to their ear. The last sensor of note is the magnetometer sensor which will report the magnetic field in X, Y and Z. Milette and Stroud (2012) cite these sensors as jumpy and less accurate than other sensors. Overall, we were pleased with the hardware of the device we chose and it performed as expected.

3.3.3 Software

Our chosen tablet device runs on the Android KitKat 4.4[2] operating system. As a Samsung product, it comes with some Samsung bloatware (Touchwiz interface, Samsung applications). We didn't find any of these programs/features hindered our use of the device. The Android storage system can be frustrating to program with since it is different depending on the manufacturer of the device and their internal and external storage filesystem naming standards. Manufacturers refer to internal storage as the Solid State Drive (SSD) and external storage as the MicroSD storage card.

The most important part of software of our tablet device is that it included support for OpenGL ES 3.0. This is would be important for the rendering of of dataset because it allowed for 3D textures. This will be explained further in Chapter 5 - Rendering. The previous versions of this line of tablets only supported OpenGL ES 2.0, which only supported 2D textures. There were implementations of OpenGL ES 2.0 that supported

[2]https://www.android.com/versions/kit-kat-4-4/

3D textures, but it was not supported on all devices with OpenGL ES 2.0. With our device now chosen, we had to make some decisions of how to program the device.

3.4 Android NDK and SDK

An important decision came up once we selected Android as our tablet operating system, should we use the Android Native Development Kit (NDK) or the Android Software Development Kit (SDK)? The Android SDK[3] is a set of tools and libraries that Android provides to develop applications for their operating system. The Android SDK is written in Java[4] and supports many development environments. The official development environment is Android Studio[5], however, alternatives exist such as Eclipse[6] and NetBeans[7]. These environments allow you to create applications which are executed on Android Runtime (Dalvik Virtual Machine)[8].

The Android NDK[9] is a toolset that allows developers to write portions of the application in native code, which is typically programmed in C or C++. This toolset is typically used to optimize CPU intensive portions of an applications code. Code written for the NDK is compiled by the Dalvik Virtual Machine and can be called from Java code using the Java Native Interface (JNI). Android points out that using the NDK will "not benefit" most applications and that the added complexity of using the NDK should only be used if the optimized code is "essential" to your application.

To help decide if we should use the NDK, we referred to research that has been done on this subject. Lin, Lin, Dow, and Wen (2011) designed an experiment that ran different 12 test programs in both native code and Java. They found that native code was faster by 34.2% (in overall performance over the 12 programs). Lee and Jeon (2010) ran a 5 part experiment with both native C and Java code that evaluated JNI communication delay, integer calculation, floating-point calculation, memory access algorithm and heap memory allocation algorithm. They found that integer calculation, floating-point calculation, memory access algorithm and heap memory allocation algorithm performed faster using the native C than using the same algorithms on the Dalvik Virtual Machine only. In our approach, floating point and integer calculation

[3]https://developer.android.com/sdk/index.html
[4]https://www.oracle.com/java/index.html
[5]http://developer.android.com/tools/studio/index.html
[6]https://eclipse.org/
[7]https://netbeans.org/
[8]http://source.android.com/devices/tech/dalvik/
[9]https://developer.android.com/tools/sdk/ndk/index.html

will be useful as we are constantly calculating the device's position and orientation. Son and Lee (2011) provides the most relevant results to our application because they measure an AR engine that is of similar nature as our envisioned approach and is written in Java and native C code. The Augmented Reality engine they used is called NyARToolKit[10]. This is a port of ARToolkit and was written entirely in Java. They identified the portions of the application that were taking the most operation time and rewrote that portion in native C code. They found that they could increase the speed of NyARToolKit by 1.869x when using the NDK.

Despite the programming overheads when using the Android NDK, we found these papers results significant enough to decide to use the Android NDK in our application for what we thought would be intensive portions of the application: Tracking and Rendering.

3.5 Summary

We have presented some context information about our project. We first had to chose our volume dataset to use in our interface prototype. We selected the Visible Human Project dataset for our volume dataset. We chose this dataset for its fame in volume rendering, availability and amount of volume data.

Now that we had our volume data, had to chose our mobile hardware and this process consisted of many decisions. We chose the Android operating system for our mobile hardware. We chose this because of reasons found in Craig *et al.* (2011) and its large marketshare. We selected the 2013 Samsung Galaxy TabPro 10.1 as our mobile hardware because of its advanced mobile hardware. With our mobile hardware selected, we had to make a decision whether to use the NDK or SDK to program our interface on the device. We chose to program using the Android NDK because of literature done on the subject. Literature cited a performance boost by using the NDK, which we believed was necessary for the intensive portions of our application.

[10]http://nyatla.jp/nyartoolkit/wp/

Chapter 4

Tracking the Device Position

Contents

In this chapter, we discuss tracking of the mobile device. We need to track the mobile device to get its position. We wanted to use this position to tell where the mobile device is inside our virtual volume and later display the correct view based on the device position. We began by looking at the sensors available on mobile devices that could be used for getting positioning information. We ran a brief experiment featuring these to see how they would serve our purpose. We then looked at using the mobile device's camera in combination with vision-based pose tracking approaches to get its position. We ran another brief experiment with this type of tracking. We found that using the mobile device's camera for positioning was more suitable than using the on board sensors for our application.

4.1 Sensor-based Tracking

We first looked into using the mobile device's on board sensors to get the device's position. The two sensors that looked promising were the accelerometer and the gyro-

scope. These sensors come standard on most mobile devices and are used for a number of different purposes.

4.1.1 Accelerometer

The accelerometer sensor measures the acceleration (g-force) of a device. This is can be imagined as "attaching a mass to springs and seeing how far the mass deviates from its equilibrium position" Milette and Stroud (2012). However, an accelerometer integrated in a phone is a miniaturisation of the idea using MEMS (Microelectromechanical systems). The key idea is that related to gravity. Gravity is a force that is felt everywhere on Earth. Its acceleration can be measured at 9.8 m/s^2. This force also applies to the mass which is part of the accelerometer which means that when the device is stationary, it still reports the rate of gravity. Likewise, when the device is in a free-fall, it reports a acceleration of 0 because the same gravitational forces are acting equally on the mass in the accelerometer. Accelerometers are usually measuring the acceleration for each axis and are typically used to detect screen orientation, detect if the device is being shook, or if the device is being moved. Android reports acceleration in m/s^2 and reports 3 values for acceleration along the X, Y, and Z axis. Our initial idea was that given an initial position 0 we could use the measure acceleration for each axis to estimate the current position by integrating over time.

4.1.2 Gyroscope

The gyroscope sensor is used to measure the "Coriolis force" caused by rotation. The Coriolis force refers to "the tendency for a free object to veer off course when viewed from a rotation reference frame"Milette and Stroud (2012). It's initially inspired by spinner tops which try to keep to axis of rotation. When one tries to change the axis of rotation for an rotating object (e.g. by rotating the device) there occurs a force (Coriolis force) that tries to work against it and tries to keep the original rotation axis caused by the conservation of angular momentum. If one could measure the force that tries to conserve the angular momentum one could infer the external force that tries t change the angular momentum. However, in phones there is no rotating mass but a vibrating mass also realised using MEMS. The key idea is that similar to spinner tops vibrating objects also tend to keep on vibrating in the same plane as its support rotates. On a mobile device, this sensor's design is that you have a tiny mass that is being pushed back and forth along one axis. When the gyroscope is rotated, the

Coriolis force is applied and makes the mass move away from the direction it was vibrating originally. This movement is what is measured in all 3 axes (X, Y, Z) by the gyroscope. Therefore, this sensor is only measuring when the device is rotating. When it is stationary, it measures 0 in all 3 axes. In laymen terms, it measures the velocity or speed at which the device is rotating. You cannot directly measure angles, however, over time you can combine the values to calculate an angle. However, when tracking the gyroscope over time, noise is introduced and will produce a larger error over time. This is known as drift and requires combining different sensors to reduce this error. Android reports the gyroscope in radians per second around the X, Y, and Z axes. The gyroscope is typically used to measure angular velocity and measures angles of a mobile device. Our initial idea was that combined with the accelerometer measuring the position we could use the gyroscope to measure the orientation of the device. However, before realising this we were interested in the error of these integrated sensors to see if our initial idea seams feasible.

4.2 Sensor-based Tracking Experiment

The purpose of this experiment was to measure the error of the gyroscope and accelerometer sensors. To accomplish this the tablet device will remain stationary for a certain amount of time. The measure error (deviation from the given position) will let us know how reliable these sensors are to use and whether we could trust their sensor readings.

The experiment design was to place the tablet device on a flat surface for a certain amount of time and record the sensor readings throughout the duration of the experiment. This would allow us to see how much the sensor readings fluctuate and change even though the device is remaining still.

4.2.1 Results

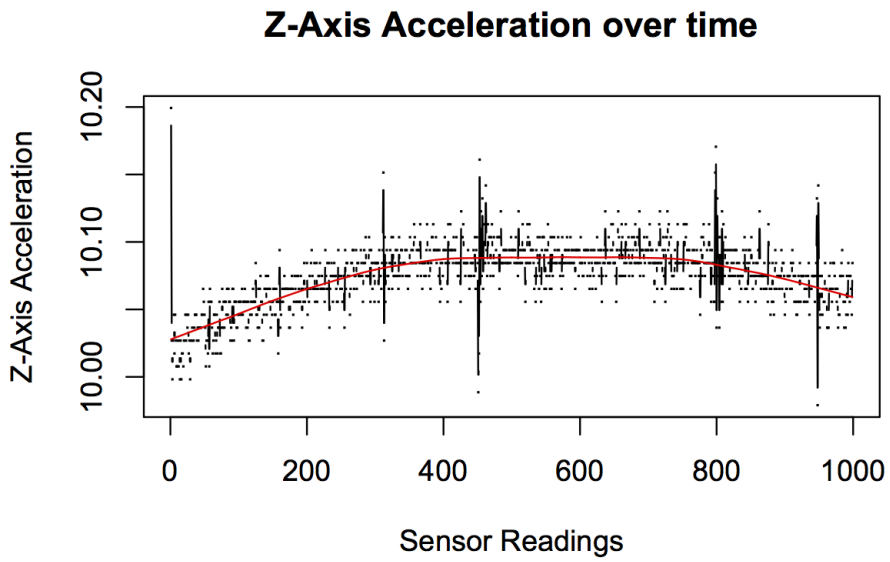


Figure 4.1: Result from accelerometer experiment showing noise and spikes in the data over time

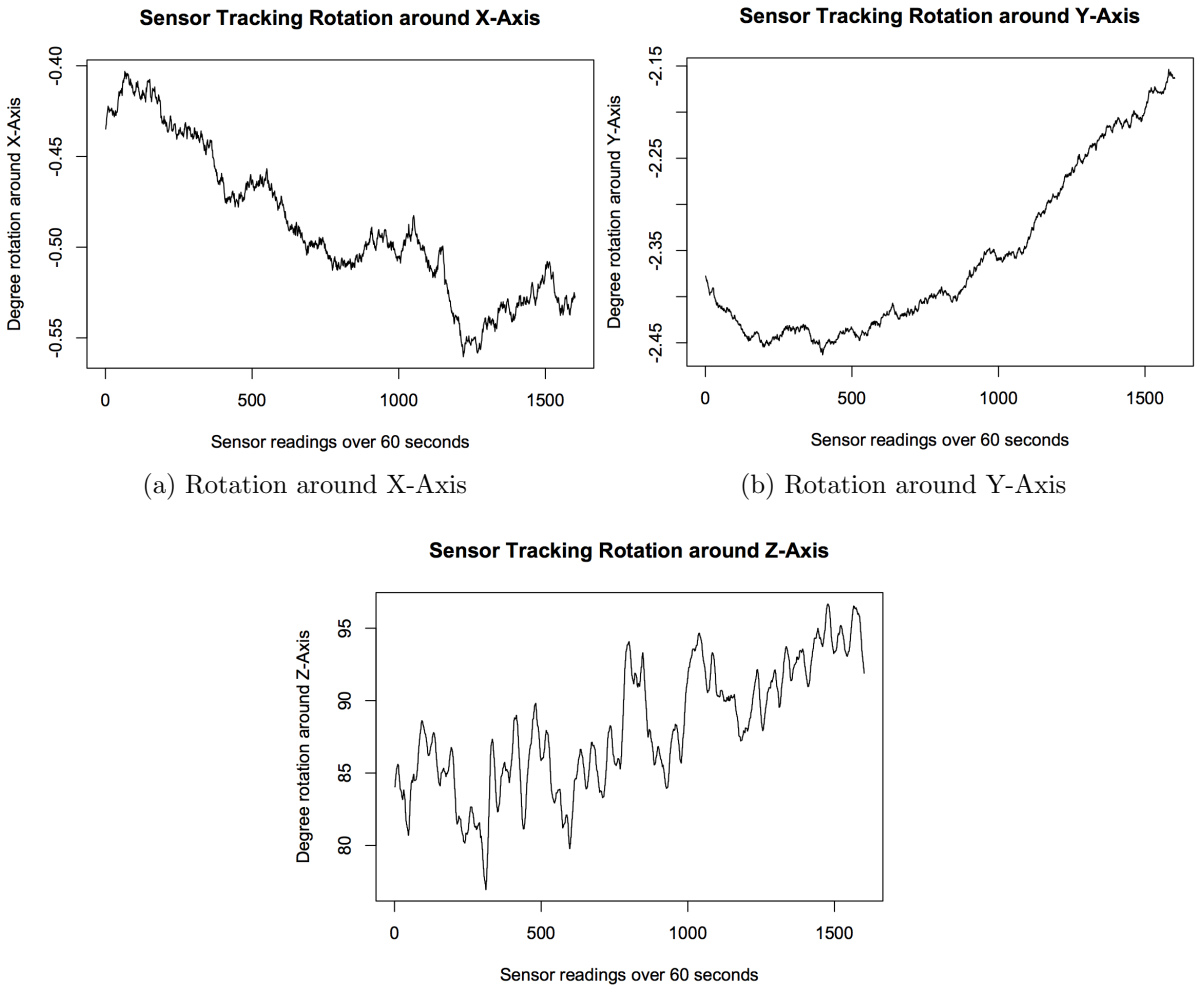
The results of this experiment showed a number of troubling indicators which meant that using the accelerometer and gyroscope to track the position of the device wouldn't be feasible. The red line in Figure 4.1 represents a smoothed line through the data using Friedman's "super smoother"[1] algorithm Friedman and Silverman (1989). This smoothing algorithm is a trend estimator the data attempts to filter out the noise and spikes in the data to provide a smoothed line through the data that allows you to see trends/patterns in the data.

Figure 4.1 shows the spikes that were recorded along the Z axis. The shape of the red (smoothed) line from Figure 4.1 shows the average of this sensor reading. The accelerometer reading should be 9.8 m/s^2 as that is the acceleration due to gravity. The results show that the sensor readings are spiking away from gravity before starting to move back towards it. Similar results were found along the X axis, but not the Y axis.

The spikes in Figure 4.1 show that the sensor will randomly take a reading that is very different than the readings taken just before or after it. This would be dangerous for our application because when someone is moving the tablet device slowly, it could all of a sudden jump in an unpredictable nature which would affect the depicted slice of the volume data the user is currently viewing. Spikes weren't as prevalent along

[1]https://stat.ethz.ch/R-manual/R-devel/library/stats/html/supsmu.html

the X and Z axis's, however, they did still occur. Figure 4.2 shows the gyroscope



(a) Rotation around X-Axis

(b) Rotation around Y-Axis

(c) Rotation around Z-Axis

Figure 4.2: Sensor Tracking experiment results showing gyroscope angles over 60 seconds

sensor readings for rotations around the X, Y and Z axes. Graphs (a) and (b) show drift over time. Drift is when a value slowly moves away from its intended value over time. However, the range for this drift is very small. This drift appears to be clearly happening around the X and Y axes rotations(Graphs (a) and (b)). The rotation around the Z-axis (graph (c)) is more troubling. It shows spikes in the data ranging around 15 degrees. This range of spikes along this Z-axis would influence the slice of the volume that the user is trying to view. Drift is a common issue and one of the ways to address is by filtering the sensor data. This is out of the scope of this thesis, however, Sabatini (2006) provides a start for future work with Kalman filtering. Following these

results we decided to reject our initial idea of using the integrated sensors to determine the devices position and orientation as they are to prone to errors that could negatively affect out overall system.

4.3 Vision-based Tracking

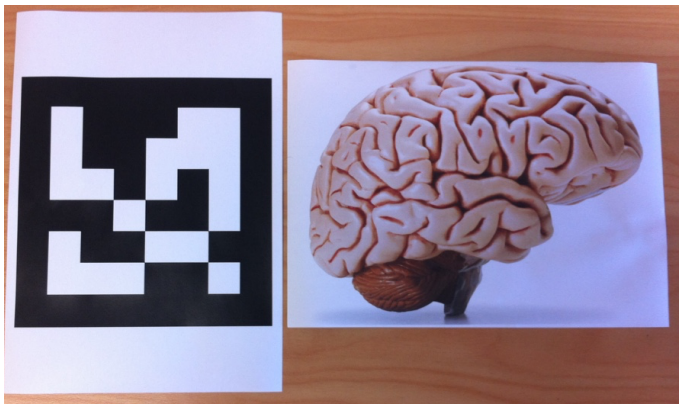


Figure 4.3: Image on left shows marker-based tracking which tracks by identifying a barcode-like image, while the image on right shows a image that allows for its tracking by identifying the natural features in the image

So far we have looked into sensor based tracking, which we ruled out because of the errors associated with these types of sensors. We then turned to Vision-based Tracking. Vision-based Tracking refers to using a camera to analyze its environment for information about its position. There are many techniques, for example, marker-based tracking and natural feature tracking. Marker-based tracking refers to when a camera identifies a fiducial (barcode-like) marker and gets its pose based on that. Natural feature tracking refers to when a camera identifies natural features in an image and get its pose based off those natural features.

We will focus on using natural feature tracking because it allows us to use an image that is related to our volume dataset. We refer to the image that is analyzed for natural features (Marker on right in Figure 4.3) as a visual marker.

4.3.1 Pose Tracking from Natural Features

We referred to Wagner, Reitmayr, Mulloni, Drummond, and Schmalstieg (2008) which discusses pose tracking based on natural features on mobile devices. This paper explains how they implemented a modified approach based on the Scale-invariant feature transform (SIFT) algorithm. SIFT is an algorithm used to detect and describe local features in an image. This algorithm is used to compare features detected in a new image (from a camera) from those in a reference image. You can compare these features from the reference and new image to find matches using the euclidean distance between features.

To help explain this, we will describe an environment that has a visual marker (Figure 4.3 right image) in it. We have stored this image on our device and have also specified its size. When the camera identifies this image based on features that have been found in both images (camera feed image and reference image), the camera is able to tell where it is in relation to that image.

Gordon and Lowe (2006) explains how this is done in AR to get 2D to 3D correspondence from features extracted from the camera video to those previously computed (using calibration parameters) of the view scene. This is then used to get the current camera pose and correctly augment the user's scene with digital content.

4.3.2 Vuforia Marker Tracking SDK

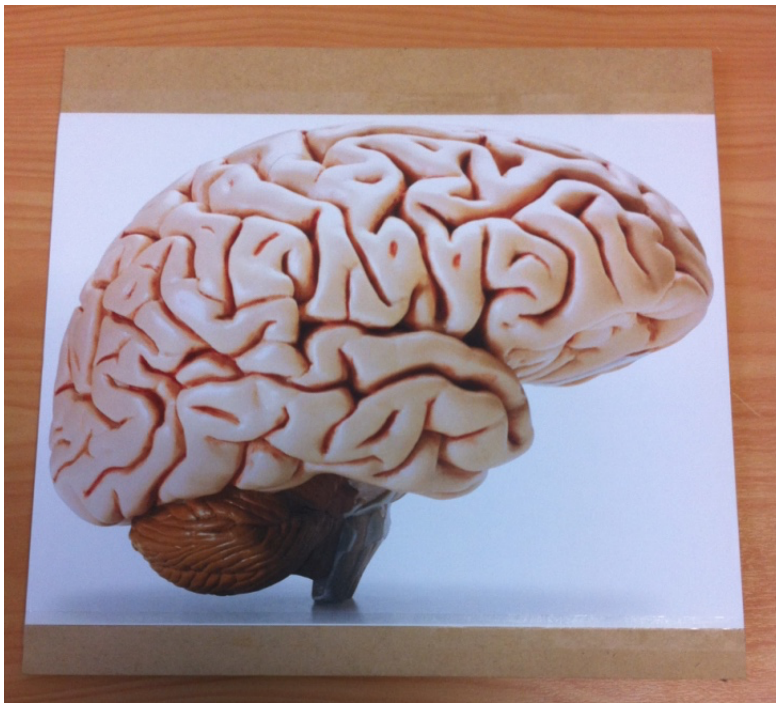


Figure 4.4: Our Visual Marker that we created which Vuforia analyzes to get the pose of mobile device camera

Vuforia[2] is an Augmented Reality (AR) Software Development Kit (SDK) designed for mobile devices only. Vuforia AR is implemented using natural feature tracking and tracking a pose based on these features. Vuforia works by identifying and tracking a visual marker (Image Target) and overlaying that target with virtual information.

This information can be a 2D/3D virtual object, video or image. The tracking is done in real-time to provide real-time correspondence to the user between the visual marker and the virtual information that is overlaid. The Vuforia Application Programming Interface (API) supports both Android and iOS devices. This API supports Java, C++, Objective-C and .Net (Unity) languages. It supports native development programming through the use of NDK in Java and C++/Objective-C in iOS.

Vuforia allows for numerous different types of visual markers that can be created by the developer. This is done by using their "TargetManager"[3] suite. Developers can create visual markers out of 2D image targets, 3D cuboid objects, 3D cyclinder objects and 3D custom objects. The developer can upload a image that they want to use as

[2]https://developer.vuforia.com/

[3]https://developer.vuforia.com/target-manager

their visual marker on the afore mentioned marker designs and the "TargetManager" will give that image a 1-5 star rating. This rating corresponds to the quality of tracking it will provide.

Vuforia was chosen as our marker tracking SDK after examining the other available frameworks based on survey papers from our literature review. ARToolKit / ARToolKitPlus[4] was not feasible because it does not work with Android devices and will not receive any further updates. Studierstube Tracker[5] was also not feasible because it is not publicly available and also does not work with Android devices.

4.3.3 How marker tracking works in Vuforia

Vuforia SDK works by using the mobile device's camera to identify a visual marker. It then analyzes this marker and returns a pose matrix (Figure 4.5). This pose matrix represents the pose of the visual marker in relation to the camera.

$$\begin{bmatrix} R_x & 0 & 0 & T_x \\ 0 & R_y & 0 & T_y \\ 0 & 0 & R_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.5: Pose Matrix that Vuforia uses to identify the position of the camera (tablet)

Figure 4.5 shows the Pose Matrix that is returned by the Vuforia SDK. The pose matrix contains the rotation matrix (left 3 columns of Figure 4.5). The "T" in the matrix represents the translation along the X, Y and Z axes in millimeters. The "R" refers to rotation around the X, Y and Z axes. Essentially, this matrix is used to describe how to move the visual marker to the camera. Since we want the opposite, we take the inverse of this matrix. Now we know where the tablet is in our real space, next we need to figure out where our "imaginary" Visible Human dataset is located in this real space.

[4]https://handheldar.icg.tugraz.at/artoolkitplus.php
[5]https://handheldar.icg.tugraz.at/stbtracker.php

4.4 Vision-based Accuracy Experiment

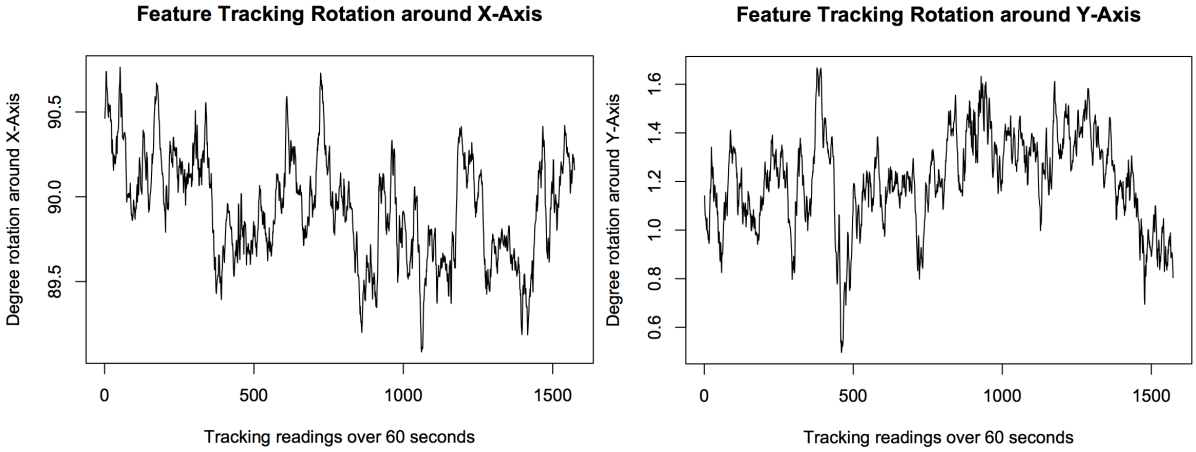
Figure 4.6: Images showing the experiment used to determine the feature tracking error

The purpose of this experiment was to measure the error of the feature tracking tracking SDK, Vuforia 3.0, on our tablet device. To measure this we decided to create an experiment that put the tablet at a stationary position for a set amount of time and record the camera pose and rotation matrix used by the marker tracking SDK. This would allow us to see how each of the parameters deviated over the course of the experiment to measure their possible error.

To create this experiment, we cut into a cardboard box to create a stationary platform for the tablet to sit on with a visual marker tapped to the bottom of the box. This would allow for the feature tracking tracking to occur and for the tablet to remain in the same spot for the duration of the experiment. A program was written to record the rotation matrix, camera pose, and distance to the marker over 60 seconds.

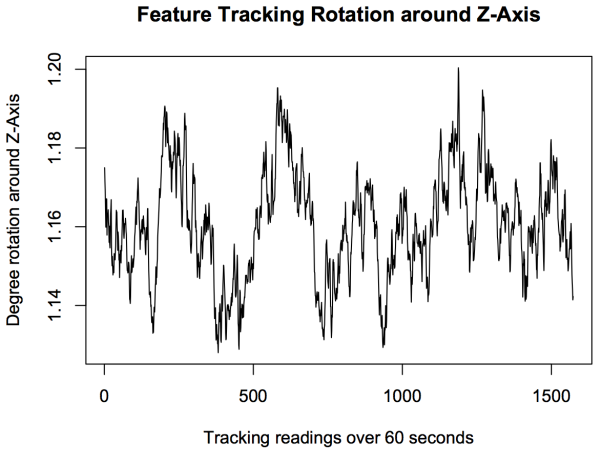
Once the results were gained, the rotation matrices were used to compute the euler angles. This was done with guidance from Slabaugh (1999). Euler angles are used to describe the orientation of an object (in our case the tablet). According to Euler's rotation theorem, any rotation can be described using 3 angles . These three angles represent the rotation around the X, Y and Z axes.

4.4.1 Results



(a) Rotation around X-Axis

(b) Rotation around Y-Axis



(c) Rotation around Z-Axis

Figure 4.7: Feature Tracking experiment results showing Euler angles computed over 60 seconds

The results of the feature tracking experiment once converted to Euler angles are shown in Figure 4.7. These graphs show spikes in the tracking data, however, they don't show drift. The range for these readings are largely within 1 degree. This small range is important because it means that is more consistent way to track . The range for rotations around the X and Y axes are larger than the sensor tracking, but not alarming high.

4.5 Experiment Conclusions

Based on the results of these two experiments, we concluded that feature tracking is the better choice to deal with tablet position. Drift in the sensor data could be detrimental to the user experience of our application over time. If the user is viewing, for example, the liver and is moving through it slowly and then all of a sudden the application jumps away it would lead to an unpredictable and unpleasant user experience. The large range for the Z-axis rotation using the gyroscope was also alarming and indicted it could not be trusted.

The small error range of the feature tracking algorithm show that it will be able to provide the necessary accuracy for our device to move through real space. It has spikes, but doesn't have the large range of spikes that would be detrimental to the user experience. It is able to do this reliably which we believed provided a stable enough tracking method to implement our approach.

Sensor Fusion

Sensor fusion is the combination of different sensors which reduces the error or uncertainty associated with individual sensor data. In our case, this would be the combination of both vision-based tracking and sensor-based tracking. This could lead to an approach that reduces the unreliability of sensor based tracking around the z-axis. We decided, however, that vision-based tracking would be acceptable for our application and didn't explore sensor fusion. Lang, Kusej, Pinz, and Brasseur (2002) and Ayub, Bahraminisaab, and Honary (2012) provide insight into how this could be implemented for future work.

4.6 Summary

To track the position of our mobile hardware, we surveyed both Sensor-based tracking and Vision-based tracking. For Sensor-based tracking, we looked at the gyroscope and accelerometer sensors. We ran an experiment to measure the error of these sensors and found that they tend to drift over time. The spikes around the Z-axis rotation were especially troubling because of the large range in the values. We also looked at Vision-based tracking using natural features. This type of tracking used a reference image to compare against a camera feed to find matches in natural features. It is then able to compute the pose from these two images by comparing there features (with help from

size information supplied when storing reference image). We ran an experiment using Vision-based tracking that found spikes in the tracking readings, but the range was minimal. We concluded that Vision based tracking was better for our interface. This is because Sensor-based tracking tends to drift and the troubling large range on the Z-axis. We decided upon using Vision-based tracking using natural features (Vuforia SDK) for determining the position of our mobile device inside our virtual volume.

Chapter 5

Rendering

Contents

In this chapter, we present the rendering approach of our application. We left off from the previous chapter with explaining and evaluating position tracking approaches leading us to choose the Vuforia tracking SDK as it gives the best position estimate. We now present how we will use the positional data from the natural feature tracking approach to render the visual representation of our volume data. Our approach is based on the idea of mapping our volume data into the user's environment. We call this mapped volume data our virtual volume. This virtual volume is then browsed using the tablet device by showing the slice of the volume corresponding to the current position of the tablet device.

We decided to map the virtual volume into the user's environment based on the volume's physical dimensions which means that the virtual volume has the same dimensions as the original dataset. We use the Vuforia tracking SDK from the previous chapter to get the tablet's position relative to a placed marker which defines the position of the virtual volume. Taking into account the pose information of the device, we use the OpenGL ES API to render the slice of our volume data that is shown to the user on the tablet.

5.1 Concept

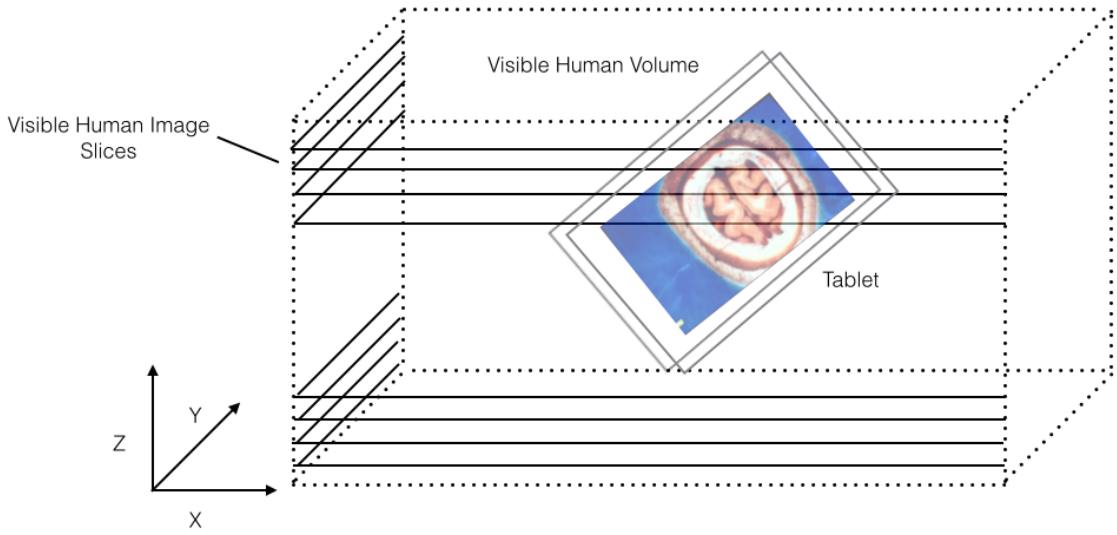


Figure 5.1: Our rendering approach that uses the tablet to "cut" through the Visible Human volume data which is assembled as shown

Our rendering concept refers to using the tablet device to physically slice through the volume data and render that slice to the user. This is shown in Figure 5.1 which depicts our volume data that is made up of a stack of Visible Human slice images. We are going to call our Visible Human volume data our virtual volume. This is because we want to place it in the users environment, but it is obviously not actually there. This virtual volume is represented by a set number of image slices from the Visible Human dataset. This will represent the Z-axis of the virtual volume. The X-axis of the virtual volume will be the X-axis of the slice image and the Y-axis of the virtual volume will be the Y-axis of the slice image. The resolution of these axes of the virtual volume will correspond to the resolution of our Visible Human data. The X and Y resolution of the virtual volume will be the same as the X and Y resolution of the Visible Human slice images. The actual Z resolution will be the number of image slices represented as pixels (100 slice images means 100 pixels for the Z-axis) but as we will later explain the GPU interpolates between slices to fill gaps between the slices. In

terms of physical dimensions, the virtual volume will be the same size as the actual Visible Human volume. This is to give a real feel for the data as the user moves through it. The part of the Visible Human dataset that we are showing are the slices that are relating to the brain. This data volume is 675.84mm x 406.28mm x 200mm.

Now that our virtual volume is set, we will introduce how we want to explore it. We will use our tablet device to move through this virtual volume. What is displayed to the user is based on where the tablet is inside the virtual volume. Since the virtual volume represents the actual physical measurements of the Visible Human volume, we are essentially cutting through the Visible Human volume dataset with our tablet screen. We want this to be done interactively and in real time. To accomplish this, we have broken this down into three stages: pre-process of the volume data, tracking of the tablet and displaying the relevant slice. Figure 5.2 briefly explains the 3 main

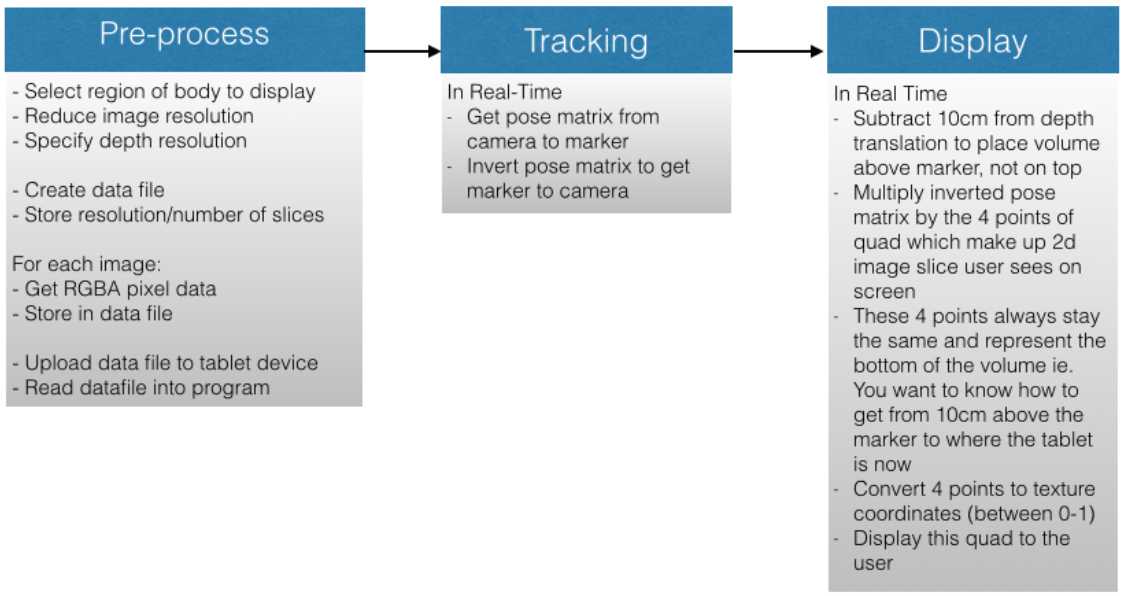


Figure 5.2: Complete process taken to display slice of volume

stages required to display this slice of the volume data.

5.1.1 Pre-process

We needed to pre-process our volume data because of hardware constraints. Through trial and error we ran into a limit with GPU memory that constrained us to only storing approximately 320 MB of volume data in GPU memory. We believe this has to do with how much memory the GPU is allocated. Each voxel of volume data is 4 bytes (RGBA) which led us to reduce the resolution of volume to 799(px) x 508(px) x 200(px) which is approximately 310 MB.

Pre-processing was done on a separate computer that is storing the entire Visible Human dataset. The pre-processing program takes a specified area of the body that the user specifies (Z-axis) and takes the relevant images' pixel data. Resolution of the X and Y axes can be changed here. This pixel data is what will be used to create our 3D virtual volume. The end result of this program is that it outputs a data file that stores all the image slice's pixel data which is in RGBA format. This file is then transferred to the tablet and will be read by our application. Our application reads in this file and creates the OpenGL ES 3D volume. Creating this file is a time intensive process and would add considerable time to our application if this process was done on the tablet at the start of every run.

5.1.2 Tracking

The tracking aspect was explained in the previous chapter. The Vuforia tracking SDK will give us a pose matrix that describes the position of a visual marker in relation to the tablet camera. If we take the inverse of this, it describes the position of the tablet camera from the visual marker. This will be used to tell us where the tablet is inside the virtual volume.

5.1.3 Rendering

Rendering is the process of creating a visual from a model. This process is described in the rest of the chapter. We explain OpenGL ES and how it creates our virtual volume from the pixel data from the pre-processing stage. We then describe how our slice of the virtual volume is rendered to the tablet screen. In order to render to our screen, we have to introduce the programming interface that will be responsible for implementing this.

OpenGL ES

OpenGL ES[1] is a cross-platform programming interface for rendering 2D and 3D graphics on mobile systems. OpenGL ES provides a low-level interface for the software and graphics hardware to interact. OpenGL ES 3.0 was released in August 2012 and is backwards compatible with OpenGL 2.0. OpenGL ES 3.0 is the version that was chosen for this project because it has support for 3D textures. This feature is vital for our project because our dataset is 3D and this feature allows for its rendering.

We chose to write the rendering in the C programming language using the Android Native Development Kit (Android NDK). We chose this because of the described benefits of using the Android NDK over Android SDK (see Chapter 3.4). We needed to achieve our requirement for real-time, user interface which was 15 fps. We believed using the Android NDK would help make the application reach this goal. Since we were already using the C/C++ implementation of Vuforia (from the previous chapter) it made sense to use the same language because it would make connecting the two easier and, more importantly, faster.

[1] www.khronos.org/opengles/

5.2 The Implementation

We will explain how we implemented our rendering concept by explaining it step-by-step. We will start by explaining how we defined our scene and then the mathematics behind displaying the slice of the virtual volume to the user.

5.2.1 Our Scene

Figure 5.3: Our starting scene that consists of our tablet camera and visual marker which defines our real-world origin at the center of the visual marker

Figure 5.3 shows our starting scene. We have our tablet device and our visual marker. The camera on the tablet is used to get the pose matrix as described in the previous chapter. This pose matrix describes the position of the visual marker in relation to the tablet. We invert the pose matrix to get the position of the tablet in relation to the visual marker. The center of the visual marker is going to be the origin for our scene. This position is 0,0,0 along the X,Y and Z axes. Our initial idea was to place the virtual volume right on-top of the marker to also make it easy for the user to understand the physical location of the virtual volume. However, we ran

into a problem there. When we placed the virtual volume directly on top of the visual marker, we found out that the Vuforia tracking SDK would loose tracking if the camera got too close to the visual marker. E.g., When the virtual volume would extend down to the virtual marker, we would need to position our tablet flat on the marker to be able to browser this end of the virtual volume. However, when doing this we would cast a shadow on the marker making it harder to track but even worst we would be so close to the marker that we are not any longer able to track it. The camera would lose tracking because it could not correctly focus and identify the natural features in the visual marker at such close range.

Adjustment to keep tracking

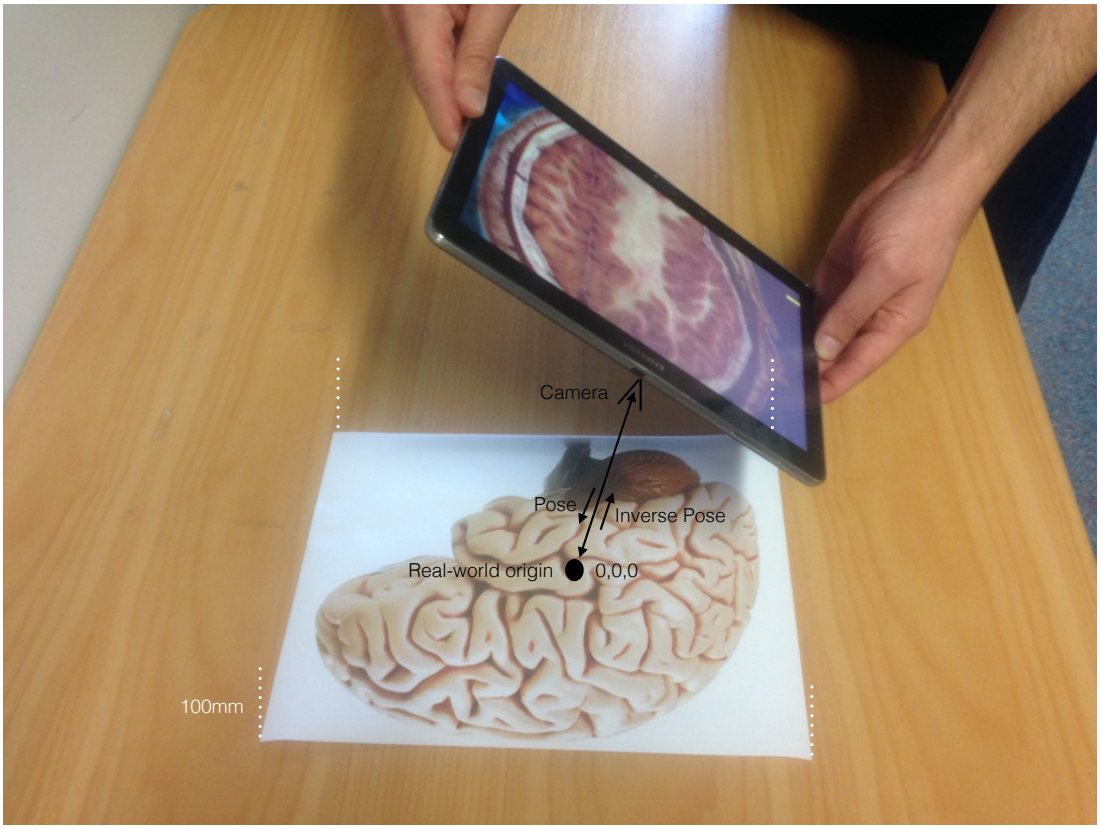


Figure 5.4: We have to move 100mm above the marker to avoid losing tracking

To prevent the camera from losing tracking, we decided to place the virtual volume not directly on top of the marker but place it so that there is a gap between the marker and the virtual volume. This would allow the user to view the bottom end of the virtual volume while we would still be able to focus the camera on the marker and consequently keep tracking. After empirical tests, we decided to place our virtual volume 100mm

above the visual marker. We chose this height because we found that the camera would not lose tracking of the marker at this height even with small rotations of the tablet. This would allow us to move the camera to the bottom of our virtual volume without losing tracking. Now our scene was ready for our virtual volume to be placed into it.

Placing the volume into our scene

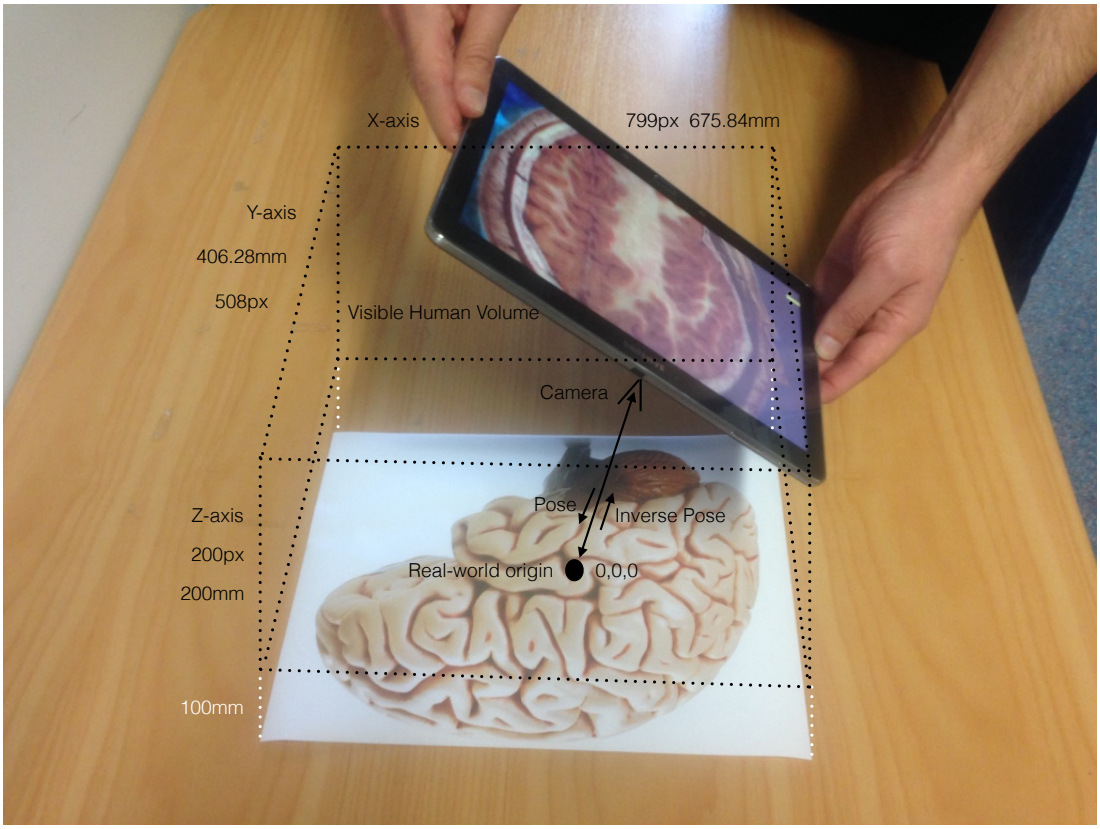


Figure 5.5: Our scene with the volume placed 100mm above marker

Figure 5.5 shows the virtual volume placed into our scene. We defined the dimensions of our virtual volume by the actual physical dimensions of the volume. The resolution of the virtual volume along the X, Y, and Z axes were 799 pixels (X), 508 pixels (Y), and 200 pixels (Z). In terms of our Visible Human volume dataset (2D Images), we reduced the X and Y resolution to 799px by 508px from 2048 by 1216. We reduced the resolution to allow for an increase in the Z resolution (number of slices). Our Z resolution is 200 slices (images) that represent 200mm of Visible Human data. We had image slices representing every 0.33mm of the human anatomy, but found that showing every 1mm was adequate for using our interface and allowed us to show more (depth) of the anatomy. The holes within the volume along the Z-axis will be closed when rendering by interpolating between two neighbouring slices. We have defined the physical dimensional aspects of our scene, but we now needed to define how OpenGL ES will access and display our virtual volume.

5.2.2 Rendering Scene

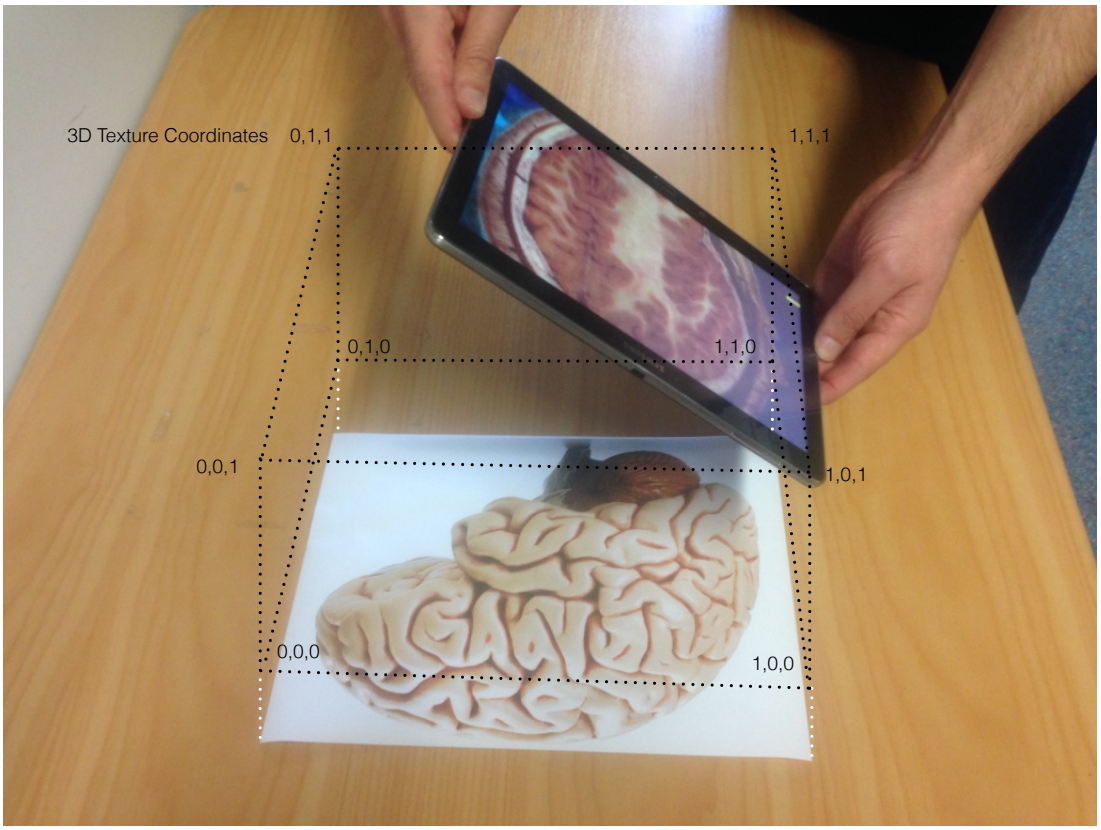


Figure 5.6: The 3D texture coordinates of our volume which uses a right-handed coordinate system

OpenGL ES 3.0 added the support for 3D Textures. These are created in OpenGL ES 3.0 by the creating an glTexImage3D[2] object. Figure 5.6 shows the texture coordinates of our 3D texture using OpenGL ES.

3D Texture

To create an glTexImage3D object you have to supply the necessary parameters as shown in Appendix A.4. We will cover the unique parameters to a 3D texture object. The "height" and "width" parameters represent the height and width of our slice images from the Visible Human dataset. The "depth" parameter is the number of image slices. The "format" parameter refers to how the pixel data is given, in our case we used RGBA. The "type" parameter specifies how we are going to supply our pixel data, in our case we used unsigned bytes. The "data" parameter is our byte stream of

[2]https://www.opengl.org/sdk/docs/man3/xhtml/glTexImage3D.xml

pixels we created during pre-processing. This creates our 3D texture object in OpenGL ES 3.0.

OpenGL ES uses texture coordinates to access the glTexImage3d object. These texture coordinates are normalized between 0-1 along the X,Y and Z axes. To access information in this glTexImage3d object, you have to provide all 3 (X,Y,Z) coordinates. OpenGL ES uses a right handed coordinate system. This means that if you use your right hand, you can show how the coordinate system is defined by using three of your fingers. To do this, make a closed fist (representing your plane) and stick out your thumb (X), index finger(Y) and middle finger(Z). In Figure 5.6, our origin is 1,1,1. With our texture coordinates defined, our scene was completed.

Shader

OpenGL defines a shader[3] as a "user-defined program designed to run some stage on a graphics processor". This program is written in OpenGL Shading Language (GLSL) and executes directly on the graphics processor. In our program we use two types of shaders: Vertex Shader and Fragment Shader.

Vertex Shader

A Vertex shader[4] is ran once for each vertex given to the graphics processor from the drawing command. For every single vertex given to the Vertex shader it outputs a single vertex. This is useful in 3D graphics when needing to transform a point in virtual 3D space. We use it to process our texture coordinates for our 3D texture. Our vertex shader code is attached in Appendix A.1. This code passes the texture coordinates to the Fragment shader. It also uses gl˙Position to calculate the vertex position in screen space.

Fragment Shader

A Fragment shader[5] calculates the colour of a pixel on the screen based on what the vertex shader has passed it. We use this to colour our pixels and create black pixels outside of our virtual volume. Our Fragment shader code is attached in Appendix A.2. This code sets the fragment color to black if the texture coordinates are outside our

[3]https://www.opengl.org/wiki/Shader
[4]https://www.opengl.org/wiki/Vertex˙Shader
[5]https://www.opengl.org/wiki/Fragment˙Shader

3D texture. If the texture coordinates are inside the 3D texture, it sets the fragment color from the 3D texture data.

Completed scene

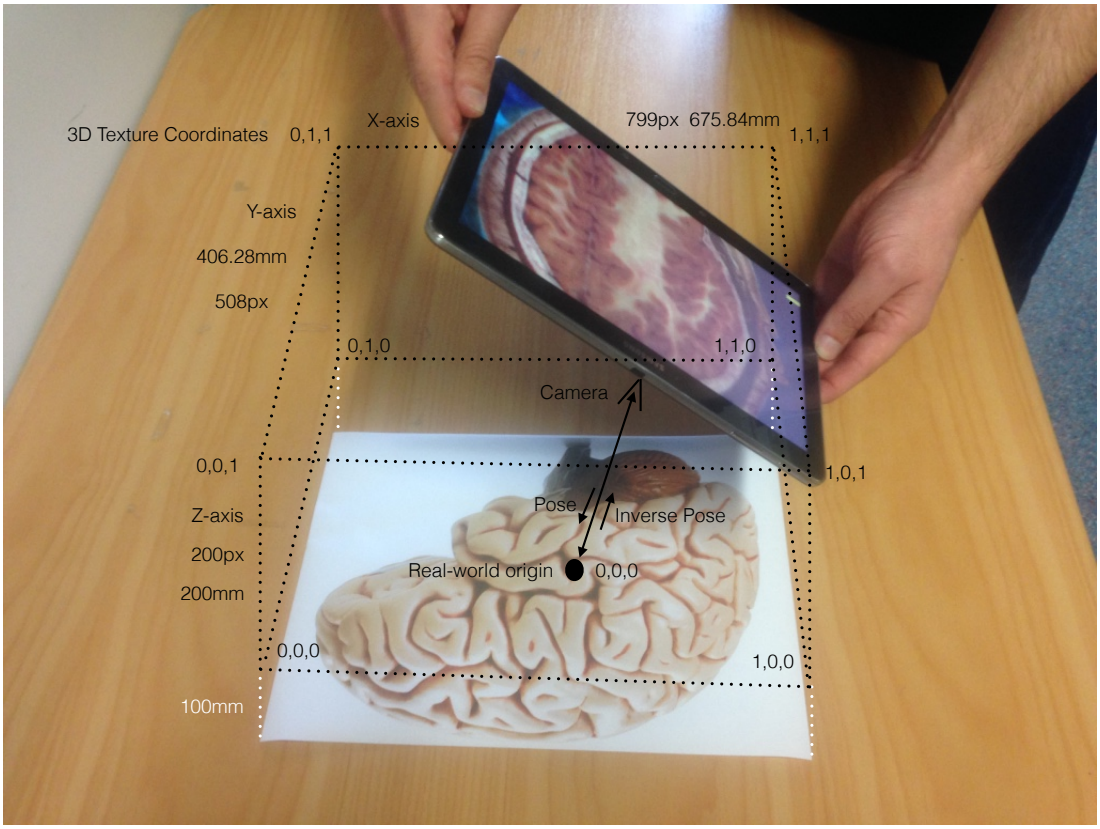


Figure 5.7: Our completed scene that shows how we will compute what slice to display

Figure 5.7 shows our completed scene. We have the physical dimensions of the actual visual human volume defined as well as the texture coordinates that OpenGL will use to access our virtual volume data. With our scene complete, we now needed to determine how to use show the slice relating to where the tablet is inside the virtual volume.

5.2.3 Tablet Coordinates

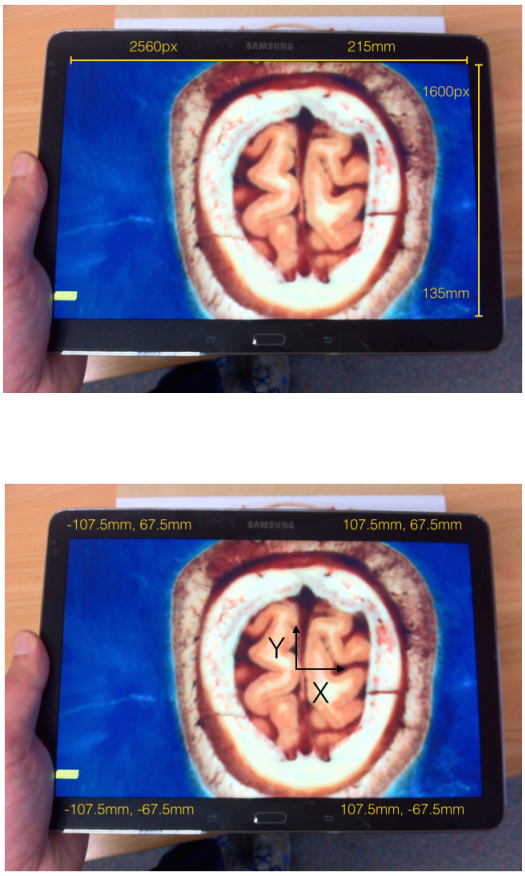


Figure 5.8: Top image shows the physical dimensions of our tablet screen and the bottom image shows how we used that to get the 4 coordinates of our tablet screen

To show a rendered slice of our virtual volume relating to where the tablet is, we needed to define our camera position. Figure 5.8 shows how we defined our camera in the middle of the tablet. We chose this location because different tablet devices have different camera locations and we decided to place it in the middle for our prototype. To account for a device's specific camera location, an offset can be used for exact location. With the middle of the tablet defined as our camera origin, we measured the the distance to the four corners of the screen. These 4 screen corners (points) will be used define our quad that OpenGL ES will render to. We measured the distance to these points because it was important to get the real scale of the volume data onto our screen.

These 4 points show in Figure 5.8 are used to multiply with the inverse pose matrix

provided by the Vuforia tracking SDK. These four points are given the Z coordinate 0 and are later translated 100 mm to account for the 100mm shift shown in Figure 5.4. Multiplying these 4 points with the inverse pose matrix moves these 4 points in accordance to the tablet device in our scene. This is because the inverse pose matrix shows the translation and rotation from the visual marker to the tablet. We then supply these 4 points as our quad to be rendered. Each of these four points has a X,Y, and Z value so they are able to access the glTexImage3d object in OpenGL ES 3.0. However, these 4 points are still in real-world coordinates (millimeters) and must be normalized to between 0 and 1 (texture coordinates).

5.2.4 Convert to Texture Coordinates

Texture Coordinates are between 0 and 1. To normalize the X, Y and Z axes we do the following calculations. We need to define the coordinate system for our volume data. To do this we set the origin in the middle of each axes (X, Y, Z).

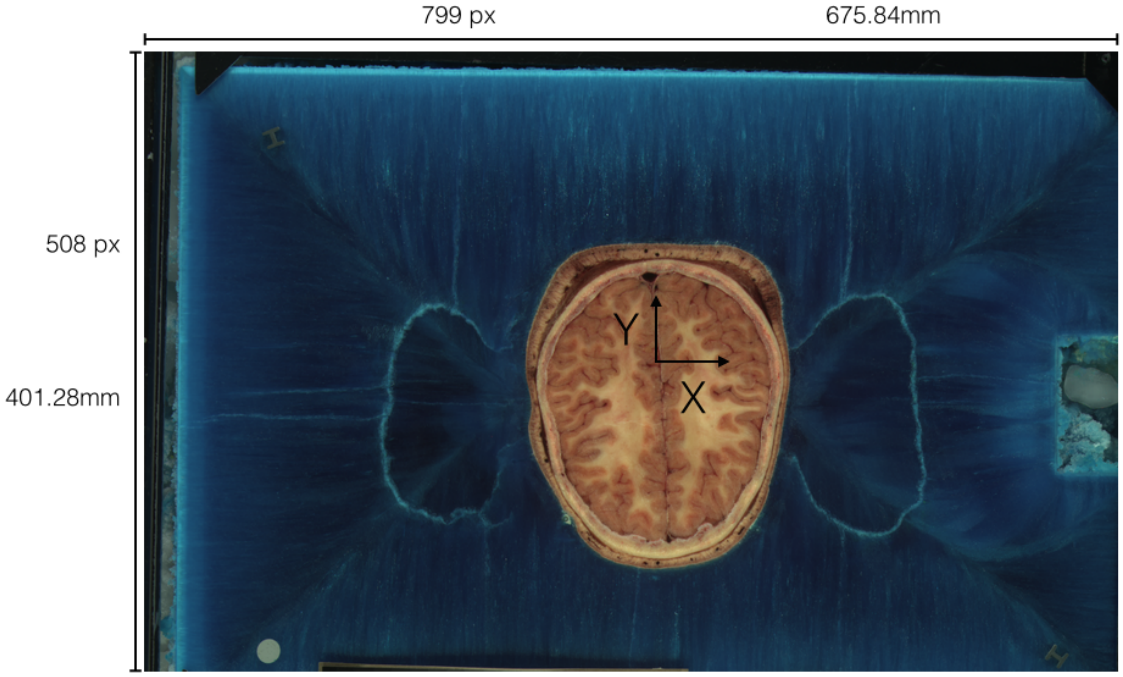


Figure 5.9: The coordinate system of our Visible Human images where we set the origin in the middle

Normalize X-axis

We are trying to normalize the X-axis to be between 0 and 1 instead of -675.84mm/2 and 675.84mm/2. The following shows how we were able to accomplish that. To get this origin coordinate for the X-axis, we use equation 5.1. In this equation, we take the maximum width for the X-axis from our volume data (Figure 5.9). We define the origin in the middle of this axis. Equations 5.2 and 5.3 show the mathematical steps we took to normalize this axis between 0 and 1. Equations 5.4-5.6 show the same steps, but with our Visible Human data's X-axis.

$$X = \frac{-xWidthmm}{2} ... \frac{xWidthmm}{2} \tag{5.1}$$

$$\frac{X}{xWidth\text{mm}} = -\frac{1}{2}...\frac{1}{2} \tag{5.2}$$

$$\frac{X}{xWidth\text{mm}} + \frac{1}{2} = 0...1 \tag{5.3}$$

In our scene:

$$X = \frac{-675.84}{2}...\frac{675.84}{2} \tag{5.4}$$

$$\frac{X}{675.84} = -\frac{1}{2}...\frac{1}{2} \tag{5.5}$$

$$\frac{X}{675.84} + \frac{1}{2} = 0...1 \tag{5.6}$$

Normalize Y-axis

We are trying to normalize the Y-axis to be between 0 and 1 instead of -401.28mm/2 and 401.28mm/2. To get this origin coordinate for the Y-axis, we use equation 5.7. In this equation, we take the maximum height for the Y-axis from our volume data (Figure 5.9. We define the origin in the middle of this axis. Equations 5.8 and 5.9 show the mathematical steps we took to normalize this axis between 0 and 1. Equations 5.10-5.12 show the same steps, but with our Visible Human data's Y-axis.

$$Y = \frac{-yHeight\text{mm}}{2}...\frac{yHeight\text{mm}}{2} \tag{5.7}$$

$$\frac{Y}{yHeight\text{mm}} = -\frac{1}{2}...\frac{1}{2} \tag{5.8}$$

$$\frac{X}{yHeight\text{mm}} + \frac{1}{2} = 0...1 \tag{5.9}$$

In our scene:

$$Y = \frac{-401.28}{2}...\frac{401.28}{2} \tag{5.10}$$

$$\frac{Y}{401.28} = -\frac{1}{2}...\frac{1}{2} \tag{5.11}$$

$$\frac{Y}{401.28} + \frac{1}{2} = 0...1 \tag{5.12}$$

Normalize Z-axis

We are trying to normalize the Z-axis to be between 0 and 1 instead of 0mm and 200mm. We are using 200 slice images, but this number comes from the amount of data these slices represent (200mm). To get this origin coordinate for the Z-axis, we use equation 5.13. In this equation, we take the maximum length (amount of data) for

the Z-axis. Equation 5.14 shows how we were able to define this axis between 0 and 1. Equations 5.15-5.16 show the same steps, but with our Visible Human volume data.

$$Z = 0...zLength\text{mm} \tag{5.13}$$

$$\frac{Z}{zLength\text{mm}} = 0...1 \tag{5.14}$$

In our scene:

$$Z = 0...200 \tag{5.15}$$

$$\frac{Z}{200} = 0...1 \tag{5.16}$$

5.2.5 Walkthrough of Algorithm

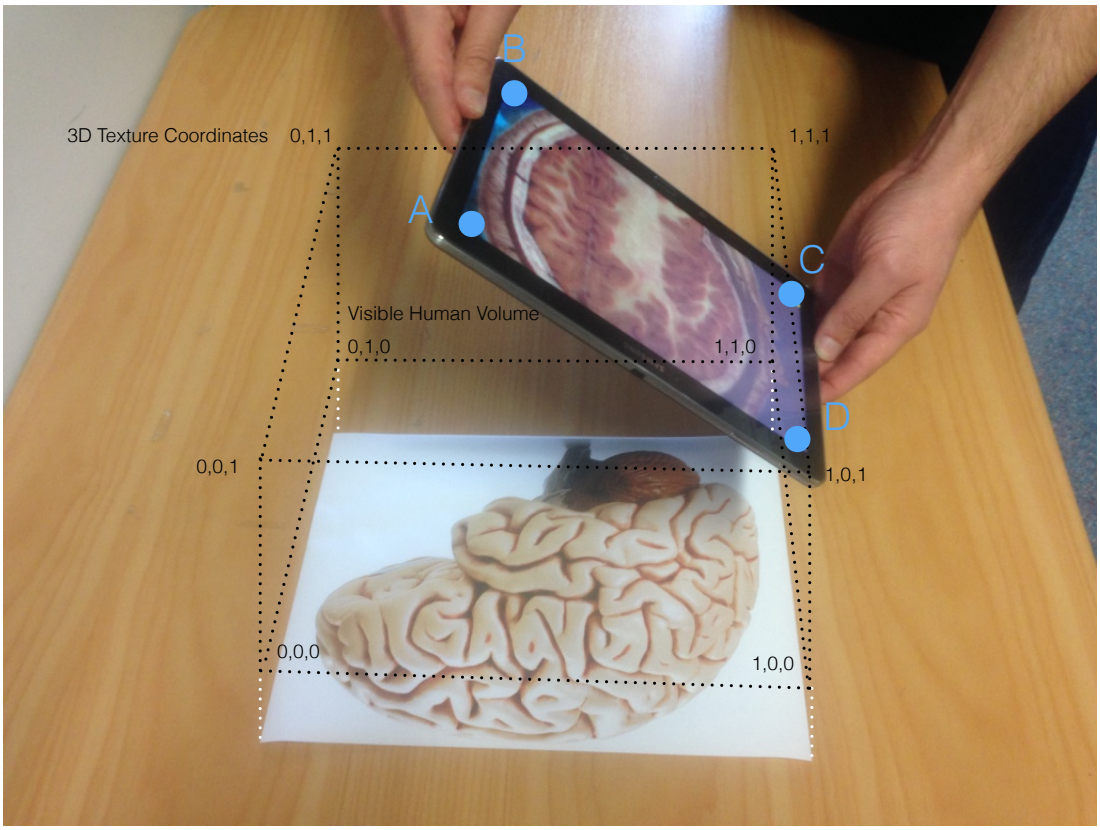


Figure 5.10: Example scenario that shows the quad we will render inside our Visible Human volume data

To show a walkthrough of the process of displaying each slice, we will show how each point in our quad is calculated. We will start for position A, but the same algorithm is used for points B,C and D. The only difference is the starting coordinates which are shown in Figure 5.8. The starting coordinates for point A will be 107.5mm (x), 67.5mm (y), 0mm (z) from Figure 5.8. The 0mm on the z axis is from placing the tablet ontop of the marker to start (it will be moved 100mm above later).

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_x & 0 & 0 & T_x \\ 0 & R_y & 0 & T_y \\ 0 & 0 & R_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 107.5\text{mm}_X \\ 67.5\text{mm}_Y \\ 0\text{mm}_Z \\ 1 \end{bmatrix} \tag{5.17}$$

X′, Y′and Z′are the new coordinates of our Point A after it has been multiplied by the inverse pose matrix provided from the Vuforia tracking SDK (Equation 5.17). These new coordinates are the real-world position of Point A in relation to the visual marker. These coordinates now need to be normalized to texture coordinates which is done by the steps explained above in 5.3.7.

$$X'' = \frac{X'}{675.84} + \frac{1}{2} \tag{5.18}$$

$$Y'' = \frac{Y'}{401.28} + \frac{1}{2} \tag{5.19}$$

$$Z'' = 1.0 - \frac{Z'}{200} \tag{5.20}$$

X″, Y″and Z″are now converted to texture coordinates for point A. We subtract our normalization for the Z-axis from 1.0 because we stored our virtual volume upside down (ie. first slice on bottom). The same process is done for points B,C and D of our quad. Again, the only difference is the starting X,Y and Z for each point from Figure 5.8.

5.3 Summary

We presented our rendering approach to display our slice of volume data based on where the tablet is inside the volume. We define our volume data that is mapped into the user's environment as our virtual volume. This virtual volume is based on the same physical dimensions and resolution of the Visible Human dataset. This is done to show the slice in real scale, as it would have been seen in the actual volume. We then use the Vuforia tracking SDK to get the tablet's position in this virtual volume. We use OpenGL ES 3.0 to render this slice of the virtual volume. This requires converting different coordinate systems and the math that was required was presented as well. Our interface was able to browse the volume data at 30fps on average. The end result is that the user is able to explore the virtual volume interactively and in real time.

Chapter 6

Evaluation

Contents

In this chapter, we present the user evaluation of our application and the results. We set out to evaluate the utility and usability of our application. Since our application is an early stage prototype, we wanted to get feedback as soon as possible about its design and possible uses for future directions. We wanted to approach domain experts because their knowledge of their domain would allow us to gain high quality feedback. This also meant that the feedback would be significant and allow us to not need a large number of participants for meaningful feedback.

We came in conversation with the Anatomy department at Otago University and received interest in the application's possible usefulness from them. We decided to follow this up by conducting an evaluation of the application that consisted of six participants that had neural anatomy expertise, including four neural anatomy professors. To evaluate our interface, we designed and conducted an interview with each participant. Careful planning went into the design of the interview to allow for the participant to fully express their thoughts. We had an idea of some general aspects of our application that we wanted feedback on, but we didn't want to restrict the participants feedback to just these areas. As this is a prototype, we wanted as much feedback as possible. We hope the work presented in this chapter can be applied to other areas which are beyond the scope of this thesis.

6.1 Design

The goal of our experiment was to evaluate the utility and usability of our application. As we believe our interface to be novel, we had no prior research to guide us on how to evaluate our interface.

6.1.1 Type of evaluation

We looked to the book by Purchase (2012) for advise on what types of user evaluation were available to us and how to run them. This book led us to our first decision of whether to conduct a quantitative (represented by numbers) or qualitative (not represented by numbers) user study. This distinction is important because it determines how the data is collected and analyzed. Vaterlaus and Higginbotham (2011) explained the goals of quantitative vs qualitative research, "Qualitative research questions are used to seek understanding of phenomena that are not fully developed, where quantitative methods are used to test hypotheses". With this in mind, we decided upon a qualitative user evaluation. We chose this because our interface is novel and we felt we needed more general information about the application itself from an outside perspective. We felt we didn't have enough information to form hypotheses that could be evaluated.

We next had to decide how to conduct our qualitative evaluation. We considered a comparative study that would have users use our interface and compare it against another interface that is used to browse volumetric data. The problem we found with this method of evaluation is how to choose the other interface to compare it against. We believe there is no "standard" interface for exploring volume data on either the tablet or desktop computer. We concluded that the results would be of no use if we couldn't prove the "comparative" interface was a good interface.

We, therefore, chose to conduct this qualitative user evaluation by having an interview with participants to gain feedback about our application's usability and utility. To gain advice about the different available interview techniques, we consulted Turner III (2010). They explained that there are three types of interviews that can be used in qualitative studies: informal, general and standardized. Informal interviews are conducted by "spontaneously" coming up with questions for the subject to answer. This means that every interview is different for each participant and it can be hard to draw patterns and conclusions across all the interviews. General interviews are more structured than informal, but still offer flexibility to the interviewer. The interviewer can word the

questions as they see fit which leads to a lack of consistency in the way questions are posed. However, this way allows the interviewer to build a rapport with the interviewee and choose the questions to ask depending on their previous responses. The idea is that you are following a certain theme of questions, but have the flexibility to word the questions as the interviewer sees fit. This allows the interviewer to "ensure the same general areas of information are collected from each interviewee" and, therefore, able to draw conclusions from these same general areas. Lastly, standardized open-ended interviews are extremely structured in terms of wording of the questions. This means that each interviewee is asked the same identical questions, however, they are worded in a way that allows for responses to be open-ended. This allows for the interviewees to "contribute as much detailed information as they desire and it allows the researcher to ask probing questions as a means of followup". This method relies on the participants fully expressing their viewpoints or feeling for each open ended question. The result being that there is a lot of "rich and thick qualitative" data that the researcher must sift through to accurately reflect the interviewees overall perspective. However, Turner points out that this method reduces researcher bias in the study because they follow a strict wording of the questions. We decided to conduct a standardized open-ended interview. We chose this method because the strictness of the wording of the questions ensured that the evaluation would be conducted in a consistent manner. This allowed us to touch on common themes we wanted answers on, while providing us the freedom to probe further depending on what the interviewee response is. This is useful for us because we don't know what to expect with user responses about our application and this allows us to gain a concrete understanding of what they think.

6.1.2 Evaluation Questions

With our qualitative interview method chosen, we had to design our questions and figure out what we wanted to find out about our application.

Usability

To design the usability questions we looked to the work by Nielsen (1994) which defines usability by five attributes:

- Learnability - How easy is the system to learn so that the user can get work done right away?
- Efficiency - How quickly can users perform tasks once they learned the system?
- Memorability - How easy it is to remember how to use the system?
- Errors - How many errors do users make and how easy is it for them to recover from?
- Satisfaction - How does the system feel to use, do they enjoy/like it?

We decided to focus for the usability portion of the interview on learnability, efficiency, and satisfaction. Memorability was not feasible given we were interviewing them only once and errors are difficult to judge among participants because each participant will use the application differently (no set tasks). The open ended questions we created for usability are shown in table 6.1 on page 63.

We formed these questions with the three attributes (learnability, efficiency, satisfaction) from the definition of usability provided by Nielsen (1994). We made the questions open ended to adhere to the standardized interview method we are following. The goal of these questions was to provide a conversation starting place that allowed flexibility to probe followup questions depending on the interviewees answers.

Utility

We consulted Patton (2008) for guidance on how to design the utility portion of our evaluation. This book defines utilization-focused evaluation as "evaluation done for and with specific intended primary users for specific, intended uses". It defines the focus of utilization-focused evaluation as "evaluation is on intended use by intended users". With this in mind, we had to narrow our focus on our intended users. We

Table 6.1: The open ended usability based questions for our user evaluation

Question	Aim
Talk me through what you got out of using the system?	
	Satisfaction
	Enjoyment
	Works way you want it to work
	Recommend to a friend
	Pleasant to use
How easy was it to learn how to use this system?	
	Learnability
	Intuition
	Difficulty
	How long to become skillful at
How did you find the efficiency of the interface?	
	Moving through the data
	Effort to use

decided to focus on neural anatomy domain experts as our intended users for the evaluation. We chose them because we had Visible Human slices corresponding to the brain readily available and within our GPU memory constraints (of how many slices we can display). We also had an informal interview with a member of the anatomy department who expressed interest in the application and its usefulness for education of the neural anatomy. The questions we created are shown in table 6.2 on page 64.

Our goal was to get feedback from neural anatomy domain experts about how they viewed the interface and what possible uses, if any, they could see this interface being useful in. We wanted their feedback on our visualization of the neural anatomy and how the physicality of moving through it affected their experience with the data. We also were curious to see what they didn't like about the interface and what changes could be made in the future. We'd like to stress that their feedback is coming from a neural anatomy point of view, but we believe their feedback can be applied to other domains.

Table 6.2: The open ended utility based questions for our user evaluation

Question	Aim
Could you see this interface being useful?	
	How so?
	Why?
	Use Case
	Problem with existing practices
	Other datasets/domains useful
Can you see any advantages or disadvantages of this type of interface?	
	Physically moving through the data
	Compared to other interfaces
	Compared to other datasets of volume data

6.2 Evaluation Process

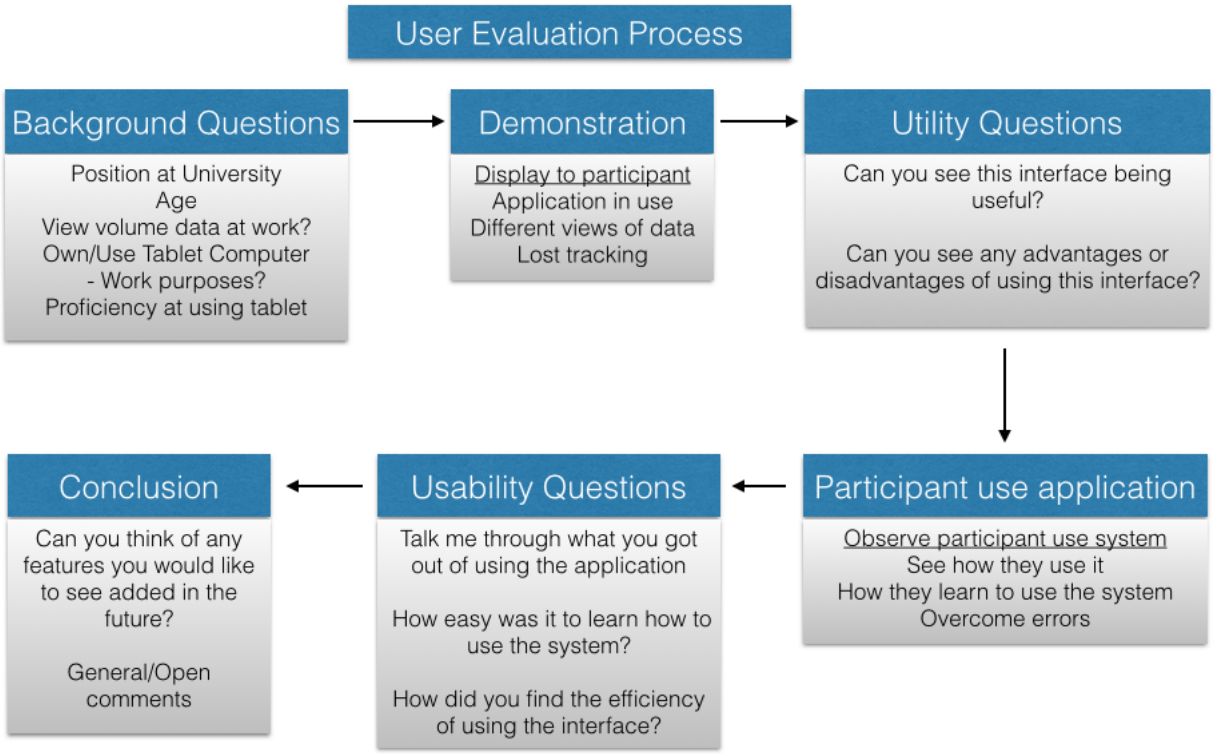


Figure 6.1: The process of our qualitative user evaluation

With our intended users established and our interview questions formed, we needed to figure out the process of the interview itself. We decided upon some background questions to start the interview with to determine how experienced they are with viewing volume data and using tablet computers. This could factor into the learnability portion of the interview. We then wanted to give the interviewee a demonstration of the application. This was structured in a a step-by-step manner so that each interviewee received the same demonstration. After our demonstration (and before they used the application), we asked our 2 open-ended utility questions. This was done to not overwhelm the interviewee and allow them to focus their thoughts on the interface itself (with no first hand knowledge of the usability). The interviewee was then allowed to use the device and the interviewer helped only if they could not overcome an error. After they determined they were acquainted with the application, we moved on to the usability questions. The usability questions were then followed with our concluding questions. These questions consisted of asking if they could think of any features they wanted added/removed and any general remarks about the concept of the slicing, our application or the way the evaluation was conducted. This process is shown in

figure 6.1. The interview sheet we created and used is shown in Appendix A.3.

With neural anatomy domain experts as our intended users, we had to prepare the slices of the Visible Human that they will view. We chose to provide 20 cms of brain data which are the slices from the top of the head to around the bottom of head. The slices were selected at 1mm intervals to provide as much depth resolution as we could.

6.3 Results

With our evaluation process determined, we now needed to reach out to neural anatomy domain experts and convince them to be apart our evaluation. We only wanted neural anatomy domain experts as they were our intended user for the utility portion of the interview.

6.3.1 Participants

We contacted the Anatomy department at Otago University and they supplied contact information for professors/staff with neural anatomy expertise. We were able to recruit four neural anatomy professors and scheduled interviews at their convenience. A 6th (final) year medical student and a PhD candidate from neuro psychology were also recruited. These two student participants both had experience with neuro anatomy from their studies and provided actual student feedback about the application. The age range for our participants ranged from 26-52 years old. The mean age of participants was 41.6 years old and the median age was 44.5 years old. Five (A, B, C, D, E) of the six participants answered that they were proficient at using a tablet computer.

Table 6.3: Participants for our evaluation and their assigned reference letter

Participant	Occupation
A	Senior Lecturer - Anatomy
B	Associative Professor - Anatomy
C	Associative Professor - Anatomy
D	Associative Professor- Anatomy
E	Final Year Medical Student
F	PhD Candidate - Neural Psychology

6.3.2 Data Collection

Data collection was primarily done by audio recording our interview with each participant (after obtaining their permission). The interviewer also took brief notes during the interview, but their focus was on the conversation itself. The recordings were analyzed and transcribed quickly after each interview so that they could be as accurate as possible. With our qualitative data now collected, it was time to analyze the data and draw conclusions.

6.3.3 Data Analysis

Purchase (2012) defines Data Analysis as the "process of summarising raw data into a useful form". For our data analysis, we are going to summarize the participant responses and draw upon common themes across the interviews. Purchase (2012) points out that there is some subjectiveness to this process, but we tried to keep it as objective as possible.

Utility

A common theme that three (A,B,C) of the four neural anatomy professors brought up is that one of the major challenges for students learning the neural anatomy is the "3D relationship between anatomical structures". They explain this stems from how neural anatomy is taught. Neural anatomical structures are taught to students by a number of different resources. The ones that were consistently brought up are 2D diagrams with labels of the structures on them, 2D physical slices of actual brain matter, 3D visualize reconstructions and brain dissections. They explain that these resources do a good job of teaching the 2D aspect of the structures, but the students struggle to combine these 2D resources to create a 3D mental model of the structures. Three (A,B,C) of the four neural anatomy professors expressed their feelings that they believe being able (by using our interface) to view the structures at different angles and being able to physically explore them would be helpful for students to understand the three dimensionality of the structures. The fourth neural anatomy professor (D) found that viewing the volume data on a slice by slice basis on a constant axis provided the necessary 3D perception for students and that our interface didn't add any 3D feel to the volume data. This brings us to how our participants said they usually view volumetric data.

Four (A,C,E,F) of the six participants described a similar desktop interface that

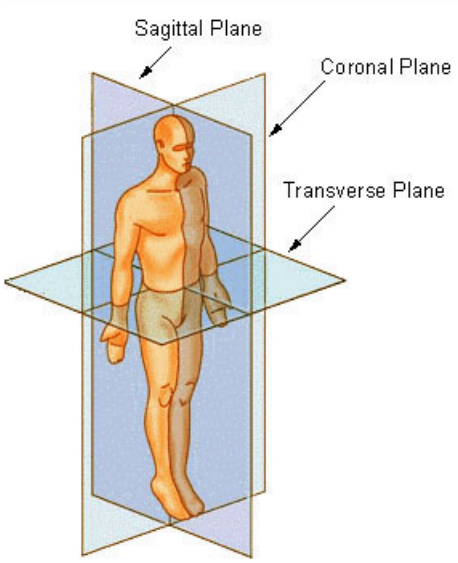


Figure 6.2: Three commonly viewed axes for volumetric data: Coronal, Sagittal and Traverse ©

they are used to using to interact with volumetric data. They described a program that allowed you to scroll through on 3 different axes (Figure 6.2) which are the Coronal, Sagittal, and Traverse planes. By scrolling on each axis, you are able to view a different slice of the volumetric data. These axes are fixed and don't allow any rotation, they just allow movement along one of these three axes. Participant E described our interface as "Pure 3D" compared to the "3 sections of 2D" that these three axes provides. Participants B and D described viewing volumetric data on physical slices. These are sections of a real brain that have been sliced and encapsulated in plastic. They show the neural anatomy through a series of slices. Participants B and C expressed that these physical slices do not allow you to understand the three dimensionality of the structures because you are left guessing what is in between the individual slices.

Participants B and E were quick to point out other available applications that they believed were similar. One such application was called 3D Brain[1] from DNA Learning (Figure 6.3). This is a mobile application that shows the brain and labels structures/regions. This particular application was brought up by Participant B as something that is recommended to students. This application can be useful in showing the relationships between structures. The interface to interact with these 3D reconstructions is that you are able to rotate the brain, but are not able to move through it. Parts of the brain become transparent to gain depth through the brain to deeper

[1]https://www.dnalc.org/resources/3dbrain.html

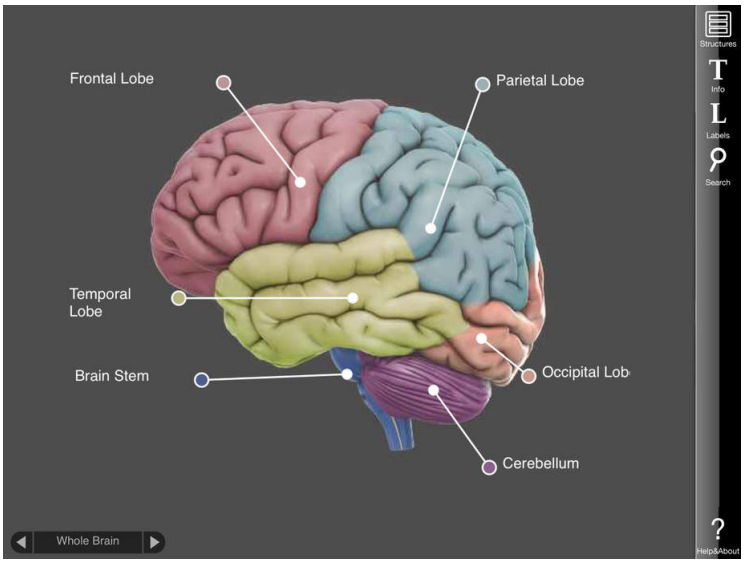


Figure 6.3: Example of available neural anatomy visualization mobile application, 3D Brain, which shows brain regions in false colours with labels

structures, but you are always observing it from afar. A drawback that Participants B and E brought up about these types of applications is that it isn't real data, it is a reconstruction and, therefore, not what the students are going to see in an actual brain. Participants B, E and F liked the "realness" of our application (the dataset) compared to 3D reconstructions they have experienced before.

We believe one possible use of this interface (based on the opinion of Participants A, B, C and E) is in our applications ability to convey the three dimensionality of structures to users. Participants D and F said that current available technology adequately satisfies this perception. Participant D made the distinction that some prior knowledge of the neural anatomy is required before this application would help with the three dimensionality aspect. They pointed out that when students are just learning about the neural anatomy, they have no idea what/where anything is, so they must have some prior knowledge of the structures for this to have effect on that 3D perception.

When asking the participants about the usefulness of the application, five (A,B,C,E,F) of the six participants said they could see it being useful in teaching/education. Participant D said they were not sure it added anything of value from the resources they have available now. Of the five who said it could be useful in teaching, the areas they identified that it could be useful in are neuro anatomy, pathology, psychology and neuro surgical planning. Participants A, C and E brought up how this application would be useful especially for medical students. They explained that in a clinical setting they are going to see a lot of volume data, particularly MRI and CT scans. The Visibile

Human dataset looks similar to what they would see in clinical practice, so having experience viewing this type of data would be beneficial for learning how to interpret the volume data they will readily encounter in clinical practice.

When asked to identify any advantages or disadvantages, both answers varied widely. On the advantages with this type of interface, Participants A and C liked the "quickness" of being able to browse the volumetric data. They commented that it allows you to get a quick idea of relationship between anatomical structures. Participants E and F felt that by having the application on a tablet, it is an advantage that you can easily bring the tablet anywhere and use the application there. This could be useful for students studying at home or having clinicians easily bring the volume data with them on the move.

Participants C and D found that a disadvantage of this particular interface is that it is hard to keep the current slice that they are viewing still. Interestingly, Participants B, E and F expressed that they felt the sensitivity of the application was satisfactory. It seemed from observing the participants that some participants wanted to view still slices of the data and others enjoyed that the slice they were viewing deviated slightly. It seemed to give those participants a real-time interactive feel for the data, while others perferred a static, still view of the data. Participant C suggested adding a pause feature to freeze the current slice of the data they are viewing. This could be a compromise to allow both groups of participants to view the data in a way that they want to. Participant F found a disadvantage is that it is physically demanding to hold the tablet in the required stance. Participants B and F brought up a disadvantage of this interface is that they didn't know where they were at times and both suggested a reference map to help them keep track of where they are in the volume data.

All six participants expressed a disadvantage in our interface that you can't view the data from "full" frontal or Coronal (from Figure 6.2). They wanted more views of the data. When you angle the tablet device you are able to get some coronal views, but it is never completely coronal because it loses tracking of the marker. At the conclusion of the interview, many of the participants asked how the visual marker worked. After learning how it worked, Participants A, C, D and E expressed they would like a second "vertical" marker to give them the ability to view the data from coronal plane. Another solution that Participant F came up with is to rotate the volume data so that they use the same interface (with one marker), but the data has been rotated.

Participants C and F expressed that they felt our interface was novel. They expressed that they have never seen something like this before. Participant C thought

that the novelty of the interface would add to the experience of using it. They felt that students would find it new and exciting. Participant D felt that students would find the interface "fun". Both these participants, C and D, felt that these qualities would add to the user experience of students and make them more likely to use the application.

Usability

Answers from the usability portion of the interview were generally positive and indicate a user interface that could be useful in the future for different contexts. Four (A, B, C, E) of the six participants expressed that they found the experience of using the application satisfactory. We believe the novelty of the application as well as the simplicity of the interface contribute to this. The two participants (D, F) who didn't express satisfaction seemed to be bothered by the lost tracking errors.

Interestingly, all six expressed that it was very easy to learn how to use. The general consensus was that it took a few movements to get used to the tracking and the movements that were possible (given the one marker). Four (A, B, C, E) of the six participants felt it was intuitive to use and were able to use right away (after watching the demonstration). Participant A summarized this by saying how the application used our natural 3D perception and this felt more natural than using a mouse/keyboard to interact with volume data. It should be noted that two (A, C) of our participants had experience with augmented reality applications (which use a visual marker), so that would factor in to how quickly they were able to pick up the movements allowed.

When asked about the efficiency of the interface, five (A, B, C, D, E) of the six participants found it "easy" to physically browse the volume data. Participant F felt it was strenuous to constantly hold the device with two hands while browsing the data. The five participants who found it "easy" to move through the data felt it didn't require physical effort to move through the data. Five (A, B, C, E, F) of the six participants reacted favorably to the response time of the application to their movements. Three (A, C, E) of the six specifically mentioned that the sensitivity was satisfactory. We received no complaints about the response time from any participants and all seemed satisfied by the real-time aspect of the interface.

Opinions were generally positive about the resolution of the dataset. The participants only referred to the X and Y(2D) resolution of the dataset, no participant mentioned any concern about the depth (Z) resolution. Five (A, B, C, D, E) of the six participants expressed that they felt the current resolution was acceptable for students.

When pressed for why students, they explained that for students all they need to see are the structures that they are likely to need to know (membranes and capsules). Participant A felt the current resolution wasn't good enough to see veins/arteries which they felt would be useful. Participant C felt the resolution would have to be increased for use with neuro scientists.

Suggested Features

Suggested features were relatively consistent throughout the participants interviews. All six participants expressed that labels could be useful. They explained that they wanted anatomical structures to be labeled and that this could help students learn the structures. Two participants (E, F) stressed that there must be no errors in the labeling and that careful consideration must be point to exactly the correct structure. These two participants explained that in their experience with computer visualizations with labels, there have been incorrect/not exact labeling and this has led to student confusion. Participant A suggested adding pins that would provide a similar educational effect. Participant B added that being able to highlight structures in a color could be useful to students as well to allow them to track structures in 3D.

Two participants (A, C) said adding some kind of examination tool would be useful. They said this could be implemented using pins or question marks on certain structures and having the students identify the relevant structure/region. Participant C expressed that students like doing these type of quizzes and being able to compare their results with other students. Implementing this would require mapping coordinates to specific structures and could be feasible to implement.

The zooming feature didn't seem as useful as we initially thought. 2 participants (B, C) said the zoom feature was not useful at all. Participant C said it was because of the pixellation when you zoom in and Participant B said the current view of the data was how they would want to view the data on. Three participants (A, E, F) expressed they would prefer a pinch zoom instead of the slider. Participant C said they would prefer using the side buttons to zoom as touching the screen while holding the device with one hand would be difficult. When observing the participants use the device, they all seemed satisfied at the standard "real scale" of the data and hardly used any other zoom level.

6.4 Summary

We conducted a six participant user evaluation that included four neural anatomy professors, one sixth (final) year medical student and one PhD candidate from neuro psychology. Their responses for the utility of the interface centered largely around education. Five of the six participants said they could see it being useful in education, particularly the learning of neural anatomy. They conveyed that one of the main problems in the teaching of neural anatomy is for students to understand the three dimensionality of the anatomical structures. This is because of how they are taught, which is mostly by 2D resources (such as diagrams, slices, pictures). Three of the four neural anatomy professors believed that this interface (with its ability to physically interact with the data in 3D) would help students with this 3D perception of the anatomical structures. While our study focused only on neural anatomy, we believe these findings can applied to other domains where it is important/difficult to understand the three dimensionality of the volume data.

Participants expressed that it is especially useful for medical students because they will be working with volume data, mainly MRI/CT scans. They believed that this interface would help them be able to interpret the volume data that they will readily work with in a clinical setting. Being exposed to this type of data early in their education will make them more familiar with this type of data and be able to better understand it.

One of the big disadvantages of our interface that participants found was the limits of viewing the data. They expressed that they are used to viewing volumetric data along 3 axes (Figure 6.1 on page 65) and that our interface didn't allow them to go "full frontal" or view along the coronal plane. This is something that should be investigated in the future. This might be able to be achieved by rotating the data or by adding a second, vertical marker to gain this movement through the volume data.

All six participants found the interface easy to learn. They found it intuitive because it uses your natural 3D perceptions to interact with the data. They didn't find it physically exhaustive to use and found it easy to move through the volume data. They found the response time of the application satisfactory and believed the resolution of the dataset was adequate.

Suggested features for the future were centered around labeling of structures. They conveyed that this would be useful for students learning neural anatomy structures. Adding a examination element would also be helpful for students. The zoom feature

didn't provide much and wasn't used often. Some participants felt lost in the volume data and suggested adding a reference map to let users know where they are in the volume.

We conclude that this interface could be useful in different contexts where understanding the three dimensionality of volume data is difficult/important. It seems to have use in neural anatomy education as well as medical purposes. Users seem to find the interface intuitive and easy to learn requiring minimal teaching time. A main disadvantage of this interface is the tracking element. It restricts the views of the data to the user and incorporating more movements (along the different axes) through the data would greatly enhance the user experience. Physically moving through volume data has its advantages in certain contexts, but more research is needed into other areas it could be useful.

Chapter 7

Conclusion

Volume data is 3D data that represents every unit in some given volume. It has been widely used in medical imaging, such as a MRI or CT scans, to visualize the human anatomy. This type of data is typically large in size and computationally expensive to process. This has largely constrained its use with desktop hardware. Mobile hardware has recently advanced to where it is possible to store and display volume datasets.

In this work, we present our novel approach to visualize volume data using mobile hardware. Mobile hardware offered us many unique interface possibilities compared to deskop hardware because of their size, input possibilities and components. We decided on an interface that uses the mobile hardware as an exploration tool to physically move through volume data. We wanted the user to be able to "feel" the volume data by placing the volume data in front of them. They would use the mobile hardware to slice through this placed (virtual) volume. What would be displayed to them is the slice of the volume data based on the mobile hardware's position inside the virtual volume. We wanted them to be able to view these slices interactively and in real-time to be able to "feel" the three dimensionality of the volume data.

In order to implement this interface, we had to deal with two main challenges. Firstly, How can we get the position of the mobile hardware? Secondly, How can we render our slice of the volume data? To accomplish this we drew from different areas of computer science. The two main areas were Augmented Reality (AR) and Volume Rendering (Computer Graphics). In AR, it is imperative to know the device's position and orientation to correctly augment the user's environment (as captured by the device's camera). This is typically accomplished by placing a visual marker into the user's environment and having the camera analyze it to get this position and orientation. We use an Augmented Reality SDK (Vuforia) to get the mobile device's

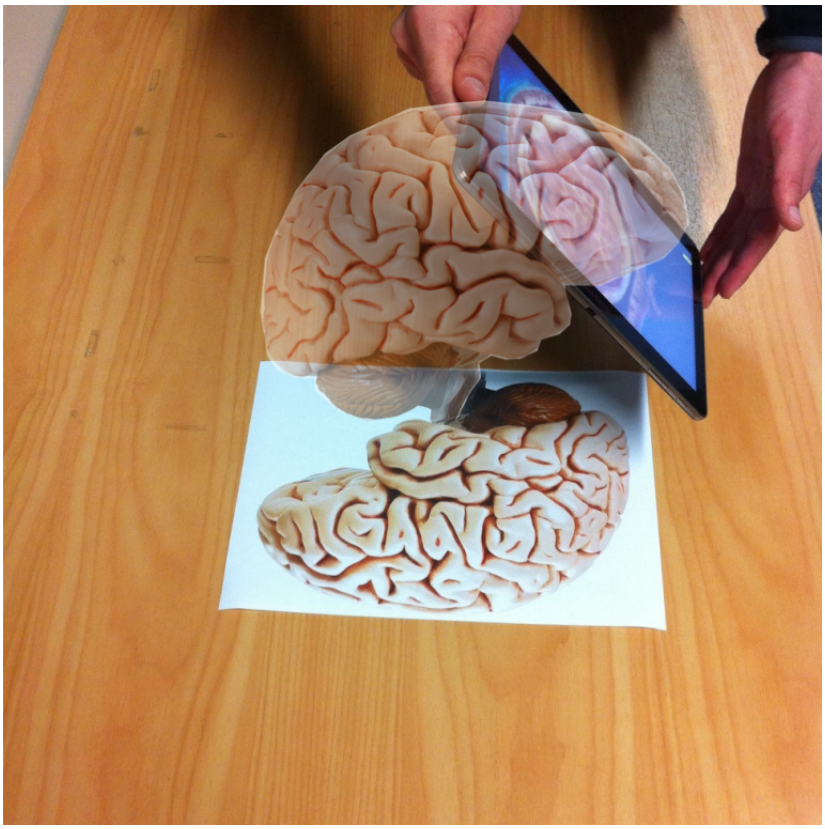


Figure 7.1: Our implemented interface concept that uses a visual marker to get the mobile hardware's position inside our virtual volume

position inside our virtual volume. This accomplishes our goal of "slicing" through the virtual volume. To display our volume data we looked to Volume rendering which refers to the techniques/algorithms used to visualize volume datasets. We used OpenGL ES to to render our virtual volume. As explained in the Rendering Chapter, we use this programming interface to interactively create/display slices of the virtual volume. This required converting from different coordinate systems, starting with the real-world coordinates of our marker tracker and ending with the texture coordinates of virtual volume (OpenGL ES). Our implemented concept is shown in Figure 7.1. Before we could evaluate the interface, we needed to select a volume dataset to display.

We chose the Visible Human Project dataset as our volume dataset to display with our interface. This volume dataset is a collection of images taken at regular intervals throughout a cadaver. It is commonly used to visualize the human anatomy. We had hardware limitations that limited the amount of volume data we could render and interact with. This led to us displaying the portion of Visible Human dataset containing only the neural anatomy. With our dataset chosen, we set out to evaluate

our interface.

We decided to perform an explorative study to get feedback about the usability and utility of our interface. We reached out to neural anatomy domain experts because of their expertise with our neural anatomy volume data. We were able to recruit six domain experts including four neural anatomy professors. We wanted domain expert feedback because they would provide high quality feedback about the utility aspect of our interface. We decided upon conducting a formal interview with each of our six participants. This interview process consisted of having the participants observe and use the interface and then asking them open-ended questions. The open-ended questions were asked with strict wording and structure to allow us to collect feedback about certain aspects of our interface, while allowing them the freedom to fully express their thoughts. Our interface is a novel, early prototype so we didn't want to limit the direction of any possible thoughts they had about our interface. With our explorative study design complete, we conducted our six participant evaluation.

The participants were in general agreement about different usability aspects of our interface. All six participants found the interface easy to learn. Four of the six participants found it intuitive to use. One commented on how it uses our natural 3D perception to browse the volume data. Five of the six participants found it easy to physically browse the volume data. Only one participant found using the mobile hardware to browse the data physically demanding. Participants seemed generally positive about the X and Y axes resolution and no one mentioned anything about the Z resolution. One main disadvantage that all six participants found with our interface is that you can't view the data from the front (Coronal) plane. They expressed how they would like to be able to move through this plane which isn't completely possible because you have to keep the visual marker in view of the camera.

Our utility evaluation responses were mostly centered around education. Five of the six participants believed it could be useful in teaching/education. The neural anatomy professors commonly brought up how difficult it is for students to understand the 3D relationship between anatomical structures. This is because they are typically taught these with 2D resources such as diagrams and brain slices. Three of the four neural anatomy professors felt that this interface would be helpful for students to understand the 3D aspect of anatomical structures. Some participants also mentioned how this interface could be helpful for medical students to be exposed to volume data they would likely to be seeing in medical practice at an early stage in their education.

7.1 Future Work

We believe there is much potential work that could be done with the interface presented in this work. From the user evaluation, we found out that users would like more range of interaction with the volume data. Right now they are limited by having to keep the visual marker in view of the camera. One participant suggested having a interface feature that rotated the volume. This would allow them to physically explore different planes of the data, however, the problem of keeping the marker in the camera view would still remain. Four of the six participants expressed that a having a second, vertical visual marker could help gain more views of the volume data. This is an interesting suggestion and could be implemented because Vuforia allows for multiple markers. Along those lines, Vuforia has a feature called Extended Tracking. This feature maps the environment around the marker using features in the environment. Once the marker is out of view of the camera, Vuforia is able to keep tracking. We briefly experimented with this feature, although our environment was not optimal. It could be something to look into in the future as long as you have an environment with unique features.

We were limited by available space in the GPU memory for which we could store our virtual volume. Compression could be explored further in the future to store more volume data in memory. Kalman filtering could also be explored for caching of the volume data. We thought this could be useful because when the user is moving the device in a certain direction, it is likely they will keep moving it in that direction. The program could predict where the user is going and store that section of the volume data into GPU memory.

Participants in our evaluation brought up a couple features that they could potentially see being useful for our interface. All six participants brought up that labels could be useful. Labelling anatomical structures would allow students to use this interface earlier in their educational career (before they are taught the anatomical structures). Two participants brought up adding some type of examination tool to the interface. This could have "?" marks on certain structures and have the students fill them in. Three of the participants said they would have preferred a pinch zoom, which likely wouldn't be hard to implement. As this was a novel, early prototype interface, we believe there is much potential work that could be done around this interface. We were pleased with how our interface turned out and are excited to see what the future could hold for this type of interface and the different domains it could be used in.

Appendix A

Appendix

Video demonstration of our interface: https://www.youtube.com/watch?v=DNBosoSDKqs

A.1 Vertex Shader Code

```
#version 300 es
layout(location = 0) in vec2 a_Position;
layout(location = 1) in vec3 a_TextureCoordinates;
out vec3 v_TextureCoordinates;

void main()
{
    v_TextureCoordinates = a_TextureCoordinates;
    gl_Position = vec4(a_Position, 0.0, 1.0);
}
```

A.2 Fragment Shader Code

```
#version 300 es
precision mediump float;
in vec3 v_TextureCoordinates;
layout(location = 0) out vec4 fragmentColor;
uniform lowp sampler3D u_TextureUnit;

void main()
{

    if (v_TextureCoordinates.z > 1.0 || v_TextureCoordinates.z < 0.0 ||
        v_TextureCoordinates.x > 0.8 || v_TextureCoordinates.x < 0.25 ||
        v_TextureCoordinates.y > 0.9 || v_TextureCoordinates.y < 0.3) {
        fragmentColor = vec4(0.0,0.0,0.0,1.0);
    }

    else {
        fragmentColor =  texture(u_TextureUnit, v_TextureCoordinates);
```

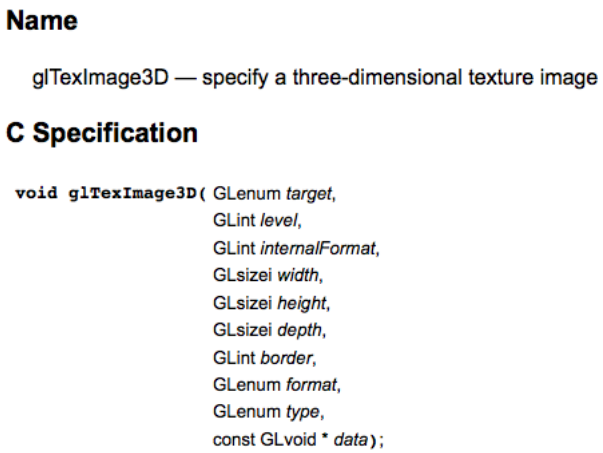
```
    }
}
```

A.3 Our tablet dimensions

Table A.1: Samsung Galaxy TabPRO Dimensions

	2013 Samsung TabPRO 10.1
Height	243.1mm
Width	171.7mm
Depth	7.3mm
Weight	477g

A.4 3D Texture



Name

glTexImage3D — specify a three-dimensional texture image

C Specification

```
void glTexImage3D( GLenum target,
                   GLint level,
                   GLint internalFormat,
                   GLsizei width,
                   GLsizei height,
                   GLsizei depth,
                   GLint border,
                   GLenum format,
                   GLenum type,
                   const GLvoid * data);
```

Figure A.1: 3D texture object in OpenGL ES 3.0 and its required input parameters to create such an object

A.5 User Evaluation Questionaire

Subject:

Stage 1: Introduction

- Introduction
- Context
- Purpose
- Schedule / Agenda / their questions at end
- Using recording - Ok?

Stage 2: Participant Background

What is your occupation at the university?

What is your age?

Do you work with visualised medical data?
- For example, MRI, X-ray, volumetric data

Do you own or regularly use a tablet computer?

- Do you use it at work?

- What do you use it for?

- Would you say you're proficient at using one?

Stage 3: Demonstration

- Explain they can ask questions, tell me when you're ready to move on

- Explain marker - orientation, pose, position in space

- Stand next to participant

- Start application

- Place application directly above marker (err on the side of too high)

- Explain that the volume is between 10cm and 30cm. Explain that above and below this is black.

- Move application down into the volume and then back up. Do it again at a faster speed.

- Move tablet to the front of the face. Tilt screen and show the eyes and nose. move through with this tilted motion.

- Move tablet through the side of the volume.

- Show off the zoom in feature on some specific part of the brain.

- Show off the zoom out feature to get a view of the whole brain.

- Explain that the application needs the tablet camera to be able to see the marker for it to work correctly.

- Show what happens when the marker loses tracking.

- Show how to reclaim tracking.

Stage 4: Utility Questions

Question: Could you see this interface being useful?
- How?

- Why?

- Use Case

- Problems with existing practices

- other datasets

Question: Can you see any advantages or disadvantages of this type of interface?
- compared to other interfaces for viewing volumetric data
- step back to other datasets
- physically moving through the data

Stage 5: User use System

- Start application for them
- Hand them tablet
- Let them play with it
- Help if they get stuck

Stage 6: Usability Questions

Question: Talk me through what you got out of using the system?
- satisfactory
- enjoyable
- works way you want it to work
- recommend to a friend
- pleasant to use

Question: How easy was it to learn how to use this system?
- learn
- intuitive
- difficult?
- remember to use it
- became skillful at it

Question: How did you find the efficiency of the interface?
- Moving through the data
- Effort to use

Stage 7: Conclusion

Question: Can you think of any features you would like to see added in the future?
- any features to take away or change?

Question: General Comments
- concept of slicing
- implementation
- procedure / experiment

References

Ayub, S., Bahraminisaab, A., and Honary, B. (2012). A sensor fusion method for smart phone orientation estimation. In *Proceedings of the 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool*.

Azuma, R. T. (1997). A survey of augmented reality. *Presence*, *6*(4), 355–385.

Baricevic, D., Lee, C., Turk, M., Hollerer, T., Bowman, D., *et al.* (2012). A hand-held AR magic lens with user-perspective rendering. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, 197–206. IEEE.

Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. (1993). Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 73–80. ACM.

Bornik, A., Beichel, R., Kruijff, E., Reitinger, B., and Schmalstieg, D. (2006). A hybrid user interface for manipulation of volumetric medical data. In *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on*, 29–36. IEEE.

Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., and Ivkovic, M. (2011). Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, *51*(1), 341–377.

Collins, J. P. (2008). Modern approaches to teaching and learning anatomy. *BMJ*, *337*.

Craig, A., McGrath, R. E., and Gutierrez, A. (2011). Technical Note: Augmented Reality Software Kits for Smart Phones.

Dahlstrom, E. and Brooks, D. C. (2014). with a foreword by Diana Oblinger. *ECAR Study of Faculty and Information Technology*, 3–3.

Desjardins, F., van Oostveen, R., Muirhead, W., and Goodman, W. M. (2011). Tablet PCs and reconceptualizing learning with technology: a case study in higher education. *Interactive Technology and Smart Education*, *8*(2), 78–93.

Drebin, R. A., Carpenter, L., and Hanrahan, P. (1988). Volume rendering. In *ACM Siggraph Computer Graphics*, Volume 22, 65–74. ACM.

Friedman, J. H. and Silverman, B. W. (1989). Flexible parsimonious smoothing and additive modeling. *Technometrics*, *31*(1), 3–21.

Fröhlich, B. and Plate, J. (2000). The cubic mouse: a new device for three-dimensional input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 526–531. ACM.

Gallo, L., Placitelli, A. P., and Ciampi, M. (2011). Controller-free exploration of medical image data: Experiencing the Kinect. In *Computer-based medical systems (CBMS), 2011 24th international symposium on*, 1–6. IEEE.

Goble, J. C., Hinckley, K., Pausch, R., Snell, J. W., and Kassell, N. F. (1995). Two-handed spatial interface tools for neurosurgical planning. *Computer*, (7), 20–26.

Gordon, I. and Lowe, D. G. (2006). What and where: 3D object recognition with accurate pose. In *Toward category-level object recognition*, 67–82. Springer.

Gotow, J. B., Zienkiewicz, K., White, J., and Schmidt, D. C. (2010). Addressing challenges with augmented reality applications on smartphones. In *Mobile Wireless Middleware, Operating Systems, and Applications*, 129–143. Springer.

Guthe, S., Wand, M., Gonser, J., and Straßer, W. (2002). Interactive rendering of large volume data sets. In *Visualization, 2002. VIS 2002. IEEE*, 53–60. IEEE.

Ikits, M., Kniss, J., Lefohn, A., and Hansen, C. (2004). Volume rendering techniques. *GPU Gems*, *1*.

Interactive, H. (2013). National Report: Students in Grades 4-12. *Pearson Student Mobile Device Survey 2013*, 39.

Kalkofen, D., Mendez, E., and Schmalstieg, D. (2007). Interactive focus and context visualization for augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 1–10. IEEE Computer Society.

Kalkofen, D., Mendez, E., and Schmalstieg, D. (2009). Comprehensible visualization for augmented reality. *Visualization and Computer Graphics, IEEE Transactions on*, *15*(2), 193–204.

Kalkofen, D., Reitinger, B. a., Risholm, P., Bornik, A., Beichel, R., Schmalstieg, D., and Samset, E. (2006). Integrated medical workflow for augmented reality applications. In *International Workshop on Augmented environments for Medical Imaging and Computer-aided Surgety (AMI-ARCS), Copenhagen*.

Kindlmann, G. (2002). Transfer functions in direct volume rendering: Design, interface, interaction. *Course notes of ACM SIGGRAPH*.

Lacroute, P. and Levoy, M. (1994). Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 451–458. ACM.

Lang, P., Kusej, A., Pinz, A., and Brasseur, G. (2002). Inertial tracking for mobile augmented reality. In *Instrumentation and Measurement Technology Conference, 2002. IMTC/2002. Proceedings of the 19th IEEE*, Volume 2, 1583–1587. IEEE.

Lee, S. and Jeon, J. W. (2010). Evaluating performance of Android platform using native C for embedded systems. In *Control Automation and Systems (ICCAS), 2010 International Conference on*, 1160–1163. IEEE.

Lin, C.-M., Lin, J.-H., Dow, C.-R., and Wen, C.-M. (2011). Benchmark dalvik and native code for android system. In *Innovations in Bio-inspired Computing and Applications (IBICA), 2011 Second International Conference on*, 320–323. IEEE.

Looser, J., Billinghurst, M., and Cockburn, A. (2004). Through the looking glass: the use of lenses as an interface tool for Augmented Reality interfaces. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, 204–211. ACM.

McLachlan, J. C., Bligh, J., Bradley, P., and Searle, J. (2004). Teaching anatomy without cadavers. *Medical education*, *38*(4), 418–424.

Meißner, M., Pfister, H., Westermann, R., and Wittenbrink, C. (2000). Volume Visualization and Volume Rendering Techniques. In *EUROGRAPHICS2000*. Citeseer.

Mendez, E., Kalkofen, D., and Schmalstieg, D. (2006). Interactive context-driven visualization tools for augmented reality. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 209–218. IEEE Computer Society.

Milette, G. and Stroud, A. (2012). *Professional Android Sensor Programming*. Wrox Programmer to Programmer. Wiley.

Nguyen, K. G. and Saupe, D. (2001). Rapid high quality compression of volume data for visualization. In *Computer Graphics Forum*, Volume 20, 49–57. Wiley Online Library.

Nielsen, J. (1994). *Usability engineering*. Elsevier.

Noguera, J. M., Jiménez, J.-R., Ogáyar, C. J., and Segura, R. J. (2012). Volume Rendering Strategies on Mobile Devices. In *GRAPP/IVAPP*, 447–452.

Ong, J. (2014). Report: android reached record 85% smartphone market share in Q2 (2014). Xiaomi now fifth-largest vendor. *The next web*.

Patton, M. Q. (2008). *Utilization-focused evaluation*. Sage publications.

Purchase, H. C. (2012). *Experimental human-computer interaction: a practical guide with visual examples*. Cambridge University Press.

Qi, W. and Martens, J.-B. (2005). Tangible User Interfaces for 3D Clipping Plane Interaction with Volumetric Data: A Case Study. In *Proceedings of the 7th International Conference on Multimodal Interfaces*, ICMI '05, New York, NY, USA, 252–258. ACM.

Rigamonti, D. D., Bryant, H. J., Bustos, O., Moore, L., and Hoffman, H. M. (2000). Implementing anatomic visualizer learning modules in anatomy education. In *the Proceedings of The Third Visible Human Project Conference*.

Rodler, F. F. (1999). Wavelet based 3D compression with fast random access for very large volume data. In *Computer Graphics and Applications, 1999. Proceedings. Seventh Pacific Conference on*, 108–117. IEEE.

Rose, S., Potter, D., and Newcombe, M. (2010). Augmented Reality: A Review of available Augmented Reality packages and.

Sabatini, A. M. (2006). Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing. *Biomedical Engineering, IEEE Transactions on*, *53*(7), 1346–1356.

Slabaugh, G. G. (1999). Computing Euler angles from a rotation matrix. *Retrieved on August*, *6*(2000), 39–63.

Son, K.-C. and Lee, J.-Y. (2011). The method of android application speed up by using NDK. In *Awareness Science and Technology (iCAST), 2011 3rd International Conference on*, 382–385. IEEE.

Turner III, D. W. (2010). Qualitative interview design: A practical guide for novice investigators. *The qualitative report*, *15*(3), 754–760.

Van Krevelen, D. and Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*, *9*(2), 1.

Vaterlaus, J. M. and Higginbotham, B. J. (2011). Qualitative program evaluation methods. In *The Forum for Family and Consumer Issues*, Volume 16.

Viega, J., Conway, M. J., Williams, G., and Pausch, R. (1996). 3D magic lenses. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, 51–58. ACM.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. (2008). Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 125–134. IEEE Computer Society.

Waterston, S. and Stewart, I. (2005). Survey of clinicians' attitudes to the anatomical teaching and knowledge of medical students. *Clinical Anatomy*, *18*(5), 380–384.

Zamora, G., Wilson, M., Mitra, A., and Thoma, G. (2000). An innovative web-based system for high lossless compression and fast, interactive transmission of visible human color images. The Third Visible Human. In *Conference Proceedings. 2000; CDROM.*

Zickuhr, K. and Rainie, L. (2014). E-reading rises as device ownership jumps. *Pew Internet Research Project*.