University of Nevada,
Reno

Immersive Visualization and Analysis of Ground Penetrating Radar Data

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science

by

Matthew R. Sgambati

Dr. Daniel S. Coming, Thesis Co-Advisor
Dr. Frederick C. Harris, Jr., Thesis Co-Advisor

May, 2010

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

MATTHEW R. SGAMBATI

entitled

**Immersive Visualization and Analysis of
Ground Penetrating Radar Data**

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Dr. Daniel S. Coming, Co-Advisor

Dr. Frederick C. Harris, Jr., Co-Advisor

Dr. Nicholas Lancaster, Graduate School Representative

Marsha H. Read, Ph. D., Associate Dean, Graduate School

May, 2010

Abstract

Ground Penetrating Radar devices send pulses from a transmitter into the earth and reflected pulses are collected by a receiver. The data collected is used by geologists to obtain a view of terrain underground. This data is typically viewed using a desktop interface where the user usually interacts using a keyboard and mouse. Visualizing the data in a slice by slice 2D format can be difficult to interpret. Instead of performing the visualization and interaction this way, I designed a program for an immersive visualization environment that uses tracked input devices. The visualization is done using real-time, stereoscopic, perspective-corrected, slice-based volume rendering. To aid with the visualization the user can modify the display of the volume using integrated tools, such as transfer functions, lighting, and color maps. Users are also given data analysis tools that provide natural interactions. These tools allow users to take application-specific measurements such as dip, strike, other angles, and distances in 3D. Compared to typical desktop interface interactions, the 6-degree of freedom user interface provided by the immersive visualization environment makes it notably easier to perform the application-specific measurements.

Acknowledgments

This work is funded by the U.S. Army's RDECOM-STTC under Contract No. N61339-04-C-0072 at the Desert Research Institute.

I would like to thank my committee: Dr. Daniel Coming for his Lumberghness, Dr. Frederick C. Harris, Jr. for his Santa Claus laugh, and Dr. Nicholas Lancaster for his superhero nickname "Dr. Dune". Additionally, I would like to thank them for their constant dedication and support in helping me complete this thesis. I would like to thank all of my friends and family for helping me through my trying times and providing me with much needed support. Without them I would not be where I am today. Finally, I would like thank my sweet girls for their love and support. Without them I would not be the person I am today.

Contents

Abstract **i**

List of Figures **iv**

List of Tables **vii**

1 Introduction **1**

2 Background and Related Work **3**
2.1 Sand Dunes . . . 3
2.2 Technology for Sand Dune Visualization and Analysis . . . 4
2.2.1 Ground Penetrating Radar . . . 4
2.2.2 Brunton Compass . . . 6
2.3 Volume Rendering . . . 7
2.3.1 Indirect Volume Rendering . . . 8
2.3.2 Direct Volume Rendering . . . 8
2.4 Virtual Reality . . . 16
2.4.1 Depth Cues . . . 17
2.4.2 Stereoscopic Displays . . . 19
2.4.3 Input Devices . . . 21
2.4.4 Toolkits . . . 23
2.5 Toirt-Samhlaigh . . . 24
2.6 Related Work . . . 25

3 Visualizing and Analyzing GPR Data **27**

4 Software Specification and Design Process **29**
4.1 Requirements . . . 29
4.1.1 Functional Requirements . . . 29
4.1.2 Nonfunctional Requirements . . . 31
4.2 Use Cases . . . 31
4.3 System Overview . . . 31
4.3.1 Toirt-Samhlaigh with Expansion and Enhancements . . . 33
4.4 Iterative Design Process with a Researcher . . . 35

5 Implementation and Results **37**
5.1 Implementation 37
5.1.1 Visualization of GPR data 37
5.1.2 Menu System 41
5.1.3 User Tools 42
5.2 Results 44

6 Conclusions and Future Work **59**
6.1 Conclusions 59
6.2 Future Work 60

Bibiliography **63**

List of Figures

2.1 Location of a sand dune surveyed in the Namib Sand Sea [7]. 4
2.2 Researchers using a GPR unit in the field on a sand dune. 5
2.3 Example of GPR pulses traveling through the ground and reflecting off a surface [65]. 5
2.4 GPR profile gathered on the sand dune surveyed in Figure 2.1 [7]. . . 6
2.5 A Brunton Compass [9]. 7
2.6 Diagram showing the dip angle, strike line, steepest gradient, and dip direction [69]. 7
2.7 A ray cast through a volume [50]. 9
2.8 Example of splatting on a volume [32]. 10
2.9 A sketch of the shear-warp mechanism [66]. 11
2.10 Transforming a volume from object space to sheared object space for a parallel projection [34]. 12
2.11 Transforming a volume from object space to sheared object space for a perspective projection [34]. 12
2.12 Example of RLE runs; one for each major axis [66]. 13
2.13 Example of 2D-texture slices being composited into the final image [14]. 14
2.14 Programmable Graphics Pipeline [20]. 15
2.15 Fixed Functionality Graphics Pipeline [20]. 16
2.16 Example of linear perspective [70]. 18
2.17 Active stereo glasses. 19
2.18 Example of a fishtank VR system [5]. 20
2.19 Example of a HMD [5]. 20
2.20 Four-sided stereo projection system. 21
2.21 Six-sided stereo projection system. 22
2.22 Two 6-DOF head tracking devices [72]. 22
2.23 Two 6-DOF wand devices [72]. 23

4.1 Use Cases. 32
4.2 High level overview of the system. 33
4.3 Toirt-Samlaigh's subsystems with modified ones highlighted. 33

5.1 Creation of vectors used to calculate dip and strike. 38
5.2 An example of a GPR data set being rendering using Toirt-Samhlaigh. 40
5.3 The left image is the data before topographic correction is applied and the right image is after. 40
5.4 GPR data in columns before (left) topographic correction is applied and after (right). 41
5.5 Simple user interface created using Vrui. 42
5.6 Complex user interface created using Vrui. 42
5.7 Brunton Compass with user interface. 43
5.8 Distance Measurement Tool with user interface. 44
5.9 Surface visualization for a topographically correct GPR data set. . . . 45
5.10 Surface visualization for a topographically correct GPR data set being rendering with a transparency of 75%. 46
5.11 Surface visualization for a topographically correct GPR data set being rendering with a transparency of 50%. 46
5.12 Surface visualization for a topographically correct GPR data set being rendering with a transparency of 25%. 47
5.13 Distance Measurement Tool being used in a GPR data set with lines enhanced. 47
5.14 Brunton Compass Tool being positioned in the GPR data to take a measurement with snapping enabled. 48
5.15 Brunton Compass Tool being used in a GPR data set, which shows several saved measurements. 48
5.16 Brunton Compass Tool with several saved measurements and steepest gradient visible. 49
5.17 Brunton Compass Tool being used in a GPR data set with snapping disabled. 49
5.18 Brunton Compass Tool after a measurement has been taken with gradients enabled. 50
5.19 GPR data being rendered along with gradients. 51
5.20 Gradients being rendered only. 51
5.21 1D Transfer Function with a piecewise-linear function applied. 52
5.22 1D Transfer Function with a Gaussian function applied. 52
5.23 1D Transfer Function with multiple Gaussian functions applied. . . . 53
5.24 Clipping plane tool being used on GPR data. 54
5.25 Lighting being applied to GPR data with the user interface visible. . 54
5.26 Slicing Tool being used to view axis-aligned slices of a GPR data set. 55
5.27 GPR data before (left) the Slicing Tool is used as axis-aligned clipping planes and after (right). 55
5.28 Subsection of a GPR Data set. 56
5.29 Volume created from a linear sand dune slice. 57
5.30 Volume created from a linear sand dune slice 57
5.31 Slice version of GPR data set shown in Figure 5.29. 58
5.32 Slice version of GPR data set shown in Figure 5.30. 58

List of Tables

4.1 Functional Requirements. 30
4.2 Nonfunctional Requirements. 31

Chapter 1

Introduction

Sand dunes are constantly being studied by scientists, whether it's for better understanding of climate change and drought impacts or to understand their structure and composition. With the advancement of technology, scientists now have a way to obtain a picture of a sand dune's subsurfaces without having to damage the dune using boreholes. This advancement is called Ground Penetrating Radar (GPR). Along with using GPR to gather data about the dune's subsurfaces, scientists use tools for taking measurements regarding the surface's dip and strike angles, which describe the surface's orientation.

The data gathered by GPR requires special software in order to be visualized as 2D slice data or a 3D volume. There are many software programs that visualize GPR data [47, 63]; however, these applications have not been written to display GPR data in an immersive visualization environment (IVE) with tracked input devices. IVEs allow for user to get views of the data that desktop displays are not capable of, such as being able to view around the data and behind it without moving the data. In addition to visualizing the data, scientists need to perform analysis of it. Tools created for desktop displays are typically limited to the interactions provided by input devices designed for 2D interfaces, while tools created for an IVE can take advantage of its tracking abilities, which provides a very different way for interacting with the data.

This thesis describes a GPR enhancement to a volume rendering library called Toirt-Samhlaigh. It is intended to allow users to visualize and analyze GPR data

sets in an IVE. It allows for the user to load GPR data in Society of Exploration Geophysicists (SEG) Y [46] format, which is a widely used format. It provides the user with the ability to view the data from many different viewpoints and provides the user with many different tools for analysis of the data. Some tools are specific to the visualization of the volume, while others are specific to the geological analysis of the data.

The remainder of this thesis is mapped out as follows: Chapter 2 provides background information about volume rendering, ground penetrating radar, sand dunes, virtual reality, and related work. Chapter 3 explains the motivation for this thesis. Chapter 4 outlines the software specifications and design process followed for this project. Details of the project's implementation and the results obtained are given in Chapter 5. Finally, concluding remarks and future work are presented in Chapter 6.

Chapter 2

Background and Related Work

2.1 Sand Dunes

A dune is a hill of sand that was created by the aeolian process. These dunes can take on several different shapes: crescentic, linear, star, dome, parabolic, longitudinal, and transverse. There are different types of dunes, primarily inland, desert, and coastal. They come in three different forms: simple, compound, and complex. Simple dunes are ones that have a minimal number of slipfaces (downwind slope) to create its shape. Compound dunes are larger dunes created from superimposed smaller dunes that have a similar type and slipfaces. Complex dunes are just a combination of several dune types. Lancaster [35] provides more detailed information on sand dunes.

The dunes of interest for this thesis are linear sand dunes. This is because they are the most abundant type of desert sand dune that is poorly understood in terms of development and internal structure [6]. With the use of GPR, researchers were able to image the internal sedimentary structures and stratigraphy of a dune in the Namib Sand Sea, which have important implications for interpretations of paleoclimates and the rock record of eolian sandstones [7]. Figure 2.1 shows the location of the sand dune used to gather data.

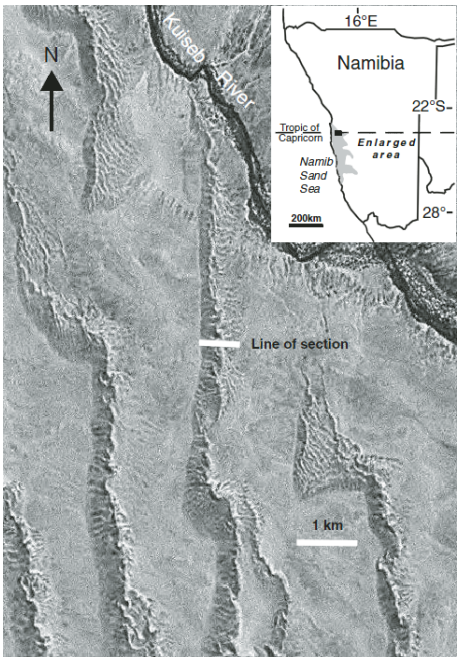


Figure 2.1: Location of a sand dune surveyed in the Namib Sand Sea [7].

2.2 Technology for Sand Dune Visualization and Analysis

2.2.1 Ground Penetrating Radar

Scientists in the field of stratigraphy (a branch of geology) need a non-destructive way of viewing underground surfaces. Ground penetrating radar (GPR) provides a way of achieving this goal. GPR is being used with increasing frequency because it yields images of the shallow subsurface that cannot be achieved by any other non-destructive method [25].

GPR uses a transmitting and a receiving antenna to send radar pulses/waves into the ground to gather information about the subsurface(s) features. The features that it aids in determining are stratigraphic architecture, sand-body geometry, and correlation and quantification of sedimentary structures [8]. An example of equipment containing these antennas is shown in Figure 2.2.

As seen in Figure 2.3, as the waves leave the transmitter and travel through

Figure 2.2: Researchers using a GPR unit in the field on a sand dune.

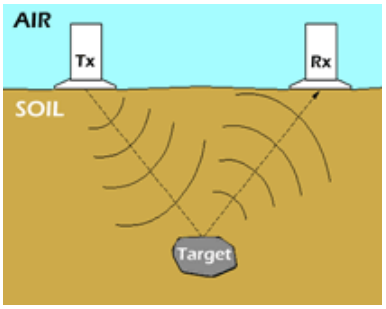


Figure 2.3: Example of GPR pulses traveling through the ground and reflecting off a surface [65].

the ground they reflect off of subsurface structures. The receiver then detects the reflected waves and records the information. These pulses do not only reflect off of objects in the subsurface, including cracks and voids, but they also reflect off of materials with different dielectric constants. This means that GPR can be used to detect subsurfaces along with changes in the type of subsurface material. It can detect these things because it is basically a map of the variation in ground electrical properties [23].

There are disadvantages to using GPR for data gathering. One is signal absorption due to high-conductive materials, such as clay or high salt-content soils [23]. Signal scattering is another problem, which is caused by subsurfaces with heterogeneous properties. Several other disadvantages include that gathered data is hard to interpret and that conducting a GPR survey requires a lot of power, which makes

long surveys harder to conduct.

Now that GPR has been explain it is import to understand how the gathering of data is performed, which is done through a GPR survey. A GPR survey is typically performed in a grid pattern, where the researcher determines the spacing for the x- and y-axis of the grid. The researcher then moves along this grid with the GPR equipment and takes readings at each interval. Figure 2.4 is an example of a GPR profile gathered on the sand dune in Figure 2.1 using such a grid pattern. Annan [2] provides details on the digital processing of GPR data.

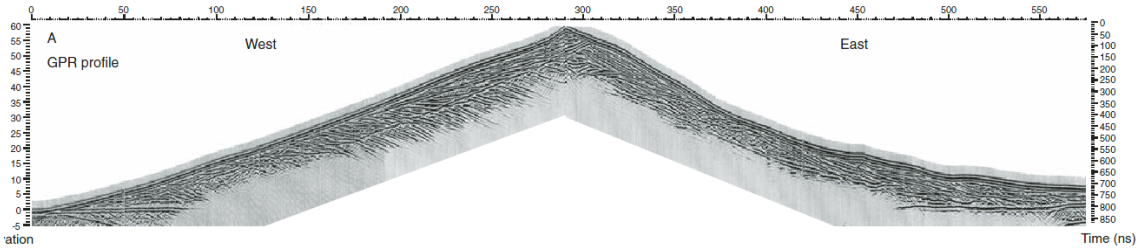


Figure 2.4: GPR profile gathered on the sand dune surveyed in Figure 2.1 [7].

2.2.2 Brunton Compass

A Brunton Compass, shown in Figure 2.5, is a tool used by scientists to determine the dip and strike angles of surface. The angle of dip is a measure of the surface's steepness relative to horizontal along it's gradient. The angle of strike describes the orientation of the surface relative to North. The strike is measured using the strike line. The strike line is a line on the surface that represents the intersection of a horizontal plane with the surface. The dip is measured using the steepest gradient from the strike line. Another way to represent strike is to use the dip direction. The dip direction is the gradient used for the dip measurement projected onto the horizontal surface that is used to create the strike line. This means that the dip direction is always 90° off the strike angle. Figure 2.6 is a diagram showing the dip angle, strike line, steepest gradient, and dip direction.

Figure 2.5: A Brunton Compass [9].

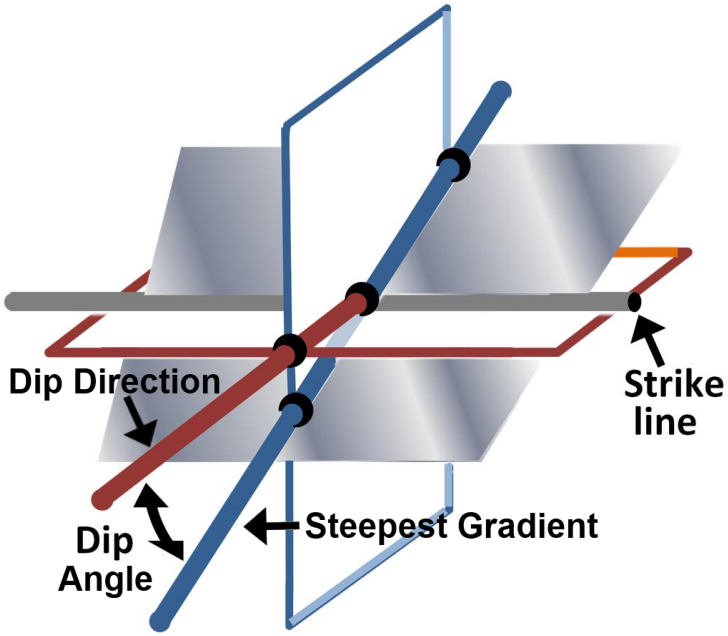


Figure 2.6: Diagram showing the dip angle, strike line, steepest gradient, and dip direction [69].

2.3 Volume Rendering

A straightforward definition of volume rendering is the display of data sampled in three dimensions [67]. There are two main volume rendering methodologies: Indirect

Volume Rendering (IVR) and Direct Volume Rendering (DVR) [43]. Additionally, each algorithm from these two methodologies falls into one of three categories: object-order, image-order, and hybrid. Object-order operates by calculating the projection and contribution of every element to the pixels in the image plane [17]. Image-order operates on the opposite basis (pixels to elements). Finally, hybrid is just a combination of object-order and image-order. These algorithm classifications are described in more detail in [17, 37, 74]. These methodologies work on volumetric data, such as a 3D uniform grid of data or point cloud data, which is a collection of points in 3D space.

2.3.1 Indirect Volume Rendering

IVR takes volumetric data and converts it into polygonal iso-surfaces, which are generally computed using the Marching Cubes algorithm [39]. IVR makes several assumptions about the data that could prove to be problematic. The first is that a set of iso-surfaces exists for the data set. A second is that these iso-surfaces can, within a reasonable degree, be accurately rendered. This is a problem because the iso-surfaces could possibly be infinitely thin. Even if these two assumptions are proven valid, the possible complexity of the generated iso-surfaces could overwhelm the rendering capabilities of the graphics device. Another problem is the fact that when the object is complex or large, or the iso-surface is interactively varied, the repeated iso-surface generation overhead must be figured into the rendering cost [4]. Because of these issues, DVR may prove to be more efficient [51].

2.3.2 Direct Volume Rendering

DVR takes volumetric data and renders it directly to the screen, skipping the iso-surface conversion step. The main requirement for DVR is that every sample point of the data needs to be mapped to an opacity and color [15], which is done using a transfer function. Transfer functions range from very simple (a ramp) to extremely complex (a set of tables and functions), are single- ([28, 30]) or multi-dimensional ([30, 31, 54])

and can also be semi-automatically generated [29]. They are described in much more detail in [28] and Chapter 4 of [18]. There are five main DVR techniques: ray casting, splatting, shear-warp, texture mapping, and hardware accelerated. These techniques will be briefly described in the following sections, but references are provided that present much more in-depth explanations.

Ray Casting

Ray Casting is a very popular form of DVR, since it has seen the most publications over the years [43]. It is an image-order volume rendering algorithm that can produce low to very high quality images, although producing very high quality images is very computationally expensive [38]. It is performed by casting a ray from the eye (camera) through each pixel of the image plane, as shown in Figure 2.7.

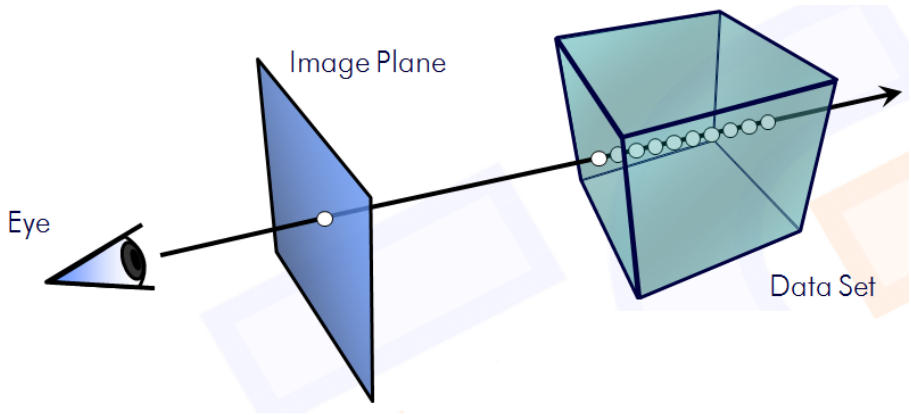


Figure 2.7: A ray cast through a volume [50].

For each ray, samples are taken at some predetermined intervals. These samples are then composited together using either a front-to-back or back-to-front algorithm [74], which then determines the pixel's final opacity and color. Very high quality images are achieved by performing trilinear filtering for each sample point and taking many sample points. Conversely, lower quality images can be produced by taking less samples and performing other filtering methods, such as bilinear filtering. The reason for this algorithm being so computationally expensive is the fact that the process just

described is done for every ray (each pixel), meaning that each ray is traversed and interpolated; samples are then taken, filtered, and composited. Interpolation is costly because at every sampling interval along each ray, eight samples need to be loaded in order to determine the sample value. Since this method can produce some of the highest quality images there has been a lot of research into minimizing the computational cost of this method. Some examples that help minimize this cost are early ray termination and using a spatial data structure [58]. Some additional examples that try to minimize this cost are [27, 38, 41, 52].

Splatting

Splatting [67] is an object-order volume rendering technique. It was designed to produce images of similar quality as Ray Casting, but at lower computational costs. The basic idea of splatting is very simple. It can be thought of as throwing a ball of paint onto a surface. Once it collides with the surface the paint splatters onto the surface. Combining all of the splattered paint makes up the final image, as shown in Figure 2.8.

Figure 2.8: Example of splatting on a volume [32].

Splatting is done by mapping every voxel to the viewing plane. Then a footprint is used to determine the extent of the influence of one data point with its surrounding points. A footprint is the integration of a reconstruction kernel, typically a Gaussian, that ultimately determines the shape of each splat and the strength of its influence [68]. This is where the speed of splatting comes in. These footprints can be precomputed, which speeds up the rendering by reducing the number of kernel integrations performed. However, to achieve ray casting quality images, a near perfect kernel needs to be picked. If the reconstruction kernel is selected so that the shape of the splat is too small, the final image will have gaps, and if it is too large, then the final image will be blurry. As with ray casting, research has been done in this area to try and resolve these shortcomings. Examples that try to resolve them can be found in [36, 44, 45, 75].

Shear-Warp

Shear-Warp is recognized as the fastest purely software-based volume rendering algorithm [66]. It is a hybrid algorithm that combines some of the better characteristics of object-order and image-order algorithms. Figure 2.9 illustrates the shear-warp mechanism.

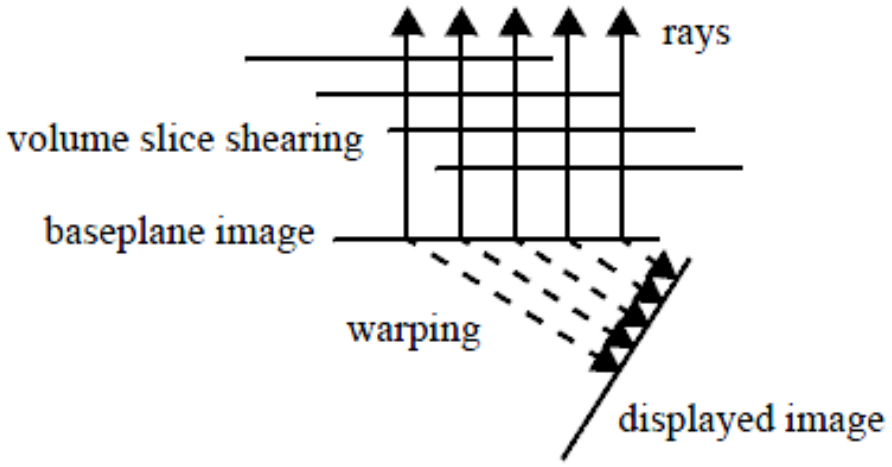


Figure 2.9: A sketch of the shear-warp mechanism [66].

It works by first creating a *sheared object space* [34], which is a coordinate system that provides a simple mapping to and from object space. An example of this space is shown in Figure 2.10. When performing a perspective projection, each slice must be scaled and sheared, as shown in Figure 2.11. Once the volume has been transformed, the slices are then composited together in front-to-back order using the "over" operator [34]. This compositing results in an intermediate image that is in sheared object space. Finally, the intermediate image is then transformed back into image space by warping it, which produces the final image.

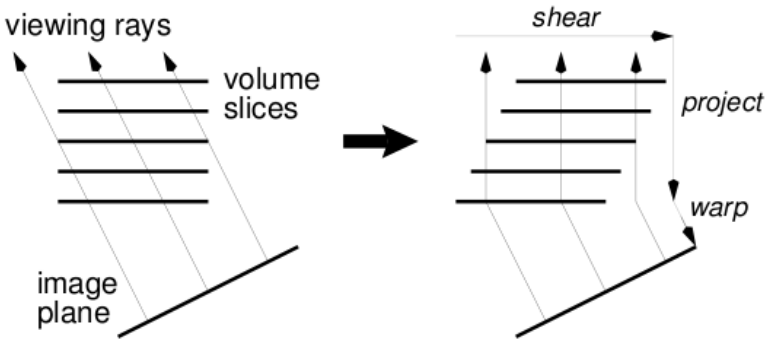


Figure 2.10: Transforming a volume from object space to sheared object space for a parallel projection [34].

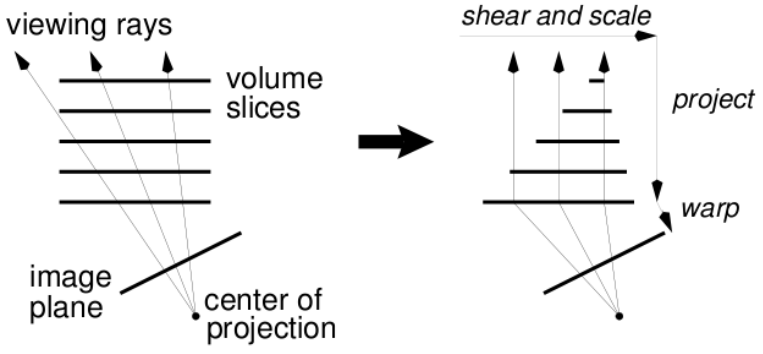


Figure 2.11: Transforming a volume from object space to sheared object space for a perspective projection [34].

One feature that makes shear-warp faster than other algorithms is that it uses run-length encoded (RLE) voxel and pixel runs. This allows for opaque pixels and transparent voxels to be efficiently skipped [66]. However, there are several shortfalls with shear-warping. One is that since interpolation occurs on a per slice basis, bilinear interpolation is performed instead of trilinear interpolation, which causes aliasing and staircasing effects depending on the viewing angle. Another is due to the fact that RLE is used, an encoding is created for each major axis, which causes it to use three times the amount of memory (shown in Figure 2.12).

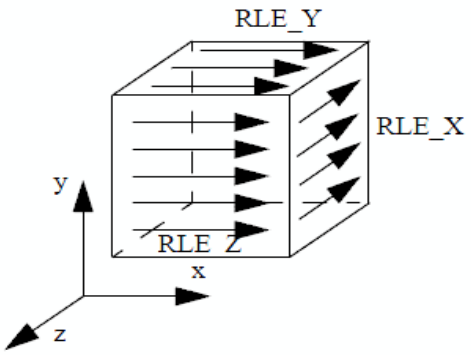


Figure 2.12: Example of RLE runs; one for each major axis [66].

Like the other algorithms, research has been done to combat these shortfalls and can be found in [59, 60, 66, 73]. The last one is an example of a combination of DVR algorithms, shear-warp and ray casting.

Texture Mapping

Graphics hardware has the ability to quickly perform texture mapping and interpolation, which is how the texture mapping algorithm came about. The texture mapping algorithm takes advantage of these abilities in the hardware. There are two types of texture mapping algorithms that will be discussed: 2D-texture mapping [10] and 3D-texture mapping [71]. Some examples of the texture mapping algorithm can be found in [13, 14, 71].

The 2D-texture mapping technique works by splitting the volume up into a bunch

of axis-aligned slices and storing them as 2D-texture images. Once these slices are created they are then composited in a back-to-front order resulting in the final image, as shown in Figure 2.13.

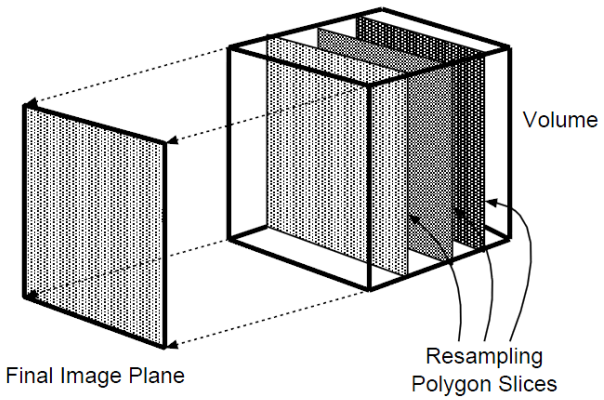


Figure 2.13: Example of 2D-texture slices being composited into the final image [14].

There are several advantages and disadvantages to this approach. Some advantages are this algorithm's simplicity and the fact that it uses 2D-texture images so the graphics hardware performs the interpolation, which is bilinear. The disadvantages are that since it has to create images for each major axis, it requires three times the memory; this is similar to the run-length encoding of shear-warp. Flickering can also be a problem when it needs to switch between which set of axis-aligned textures to use. It also suffers from aliasing when the data is magnified since the distance between the created slices is constant. There are ways to help alleviate this problem at the expense of more memory. One way is to compute intermediate slices that decrease the distance from one slice to another.

The 3D-texture mapping technique takes the data and stores it as a 3D-texture. Since it is a 3D-texture the graphics hardware can perform trilinear interpolation, as opposed to bilinear, which is required for the creation of viewport aligned slices. This eliminates several of the issues of 2D-texture mapping, such as having multiple copies of the data and flickering. These issues are overcome in 3D-texture mapping because it only needs one copy of the data and does not need to switch between different sets

of axis-aligned textures. However, there still are disadvantages. If the data set being used is too large to be stored on the graphics hardware, then some sort of bricking mechanism must be used. A brick mechanism is where the 3D-texture is broken up in pieces (bricks) that are stored on the system memory and transferred to the graphics hardware memory as needed. A problem with bricking mechanisms is that they are limited by the bandwidth between the system memory and the graphics hardware. Also, similar to the reconstruction kernels of shear-warp, correct brick sizes must be picked because they impact performance. If the bricks are too large, they won't fit into the graphics hardware's memory cache, and if they are too small, they will increase the memory footprint as well as the number of intersection tests.

Hardware Accelerated

The idea behind hardware accelerated volume rendering is to use the programmable graphics pipeline (shown in Figure 2.14), as opposed to using the fixed functionality graphics pipeline (shown in Figure 2.15). The programmable pipeline allows for the use of shaders. Shaders are a set of software instructions that are executed in place of the fixed functionality instructions. There are several different types of shaders:

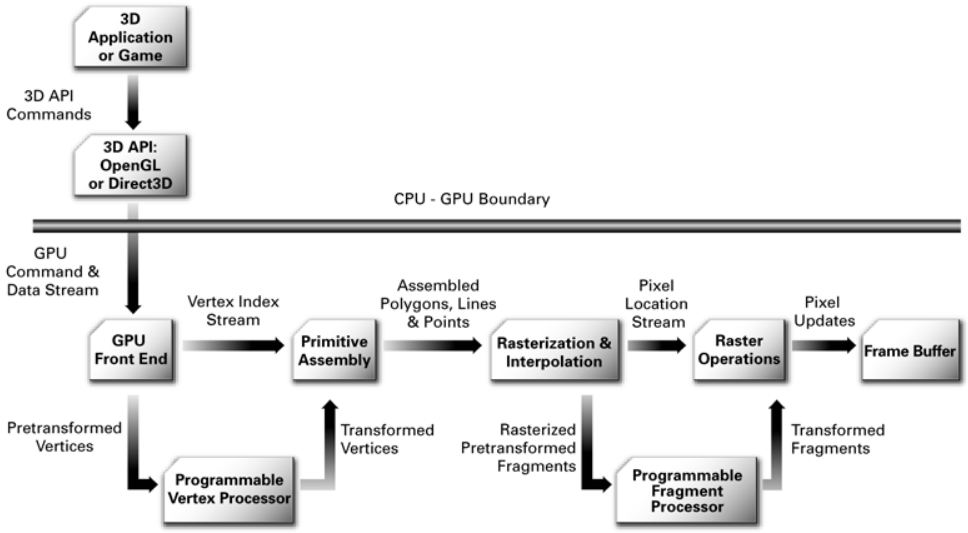


Figure 2.14: Programmable Graphics Pipeline [20].

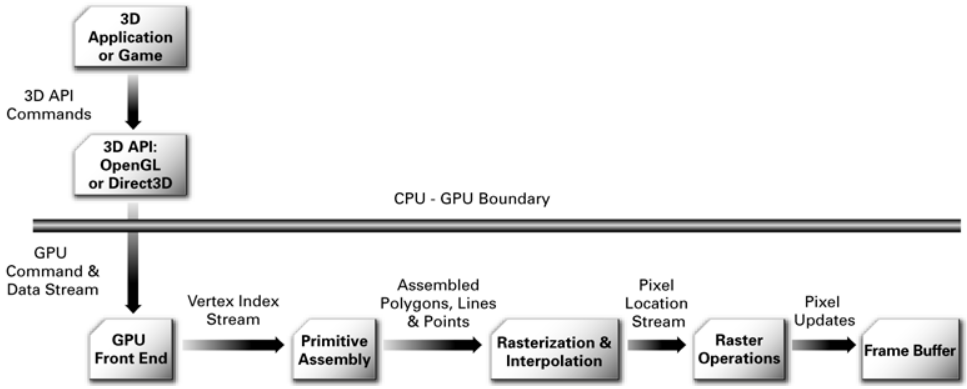


Figure 2.15: Fixed Functionality Graphics Pipeline [20].

geometry, vertex, fragment, and tessellation.

Through the use of the programmable pipeline many new techniques have been developed to help speed up volume rendering. These hardware accelerated techniques have also been applied to the volume rendering algorithms described previously. Several examples of hardware acceleration enhanced volume rendering algorithms can be found in [11, 16, 19, 55].

More clarification on several of the previous topics can be found in Part III of the Overview of Volume Rendering chapter in The Visualization Handbook [26].

2.4 Virtual Reality

The field of Virtual Reality (VR) is based on the notion that one can be mentally immersed in a virtual world through some kind of feedback; whether this feedback be visual, haptic, olfactory, or auditory. Immersion can be described as one's state of mind in which the perception of the virtual world and real world is lost, and a virtual world is a depiction of the rules and relationships governing a collection of objects in a space [62]. Interaction with a virtual environment also helps with immersion. Since this thesis only focuses on the use of visual feedback and interactivity, they will be described in more detail in the following sections along with the different types of displays, input devices, and toolkits used to create a virtual environment.

2.4.1 Depth Cues

In order to be able to understand how one can be mentally immersed in a virtual world, we first need to understand how humans use depth cues to perceive depth. A depth cue is an indicator which allows a human to perceive information regarding the relative distance of objects [62]. There are three types of depth cues: monoscopic, stereoscopic, and motion.

Monosopic depth cues provide information from only one eye (camera). The kind of information gathered in a monoscopic depth cue can be interposition, size, linear perspective, and motion parallax [62]. Interposition can be described as one object being partially occluded by another, which lets us know that the partially occluded object must be further away. Size depth cues come from objects appearing larger the closer they are to the eye and smaller the further away they are. Linear perspective is where objects appear to converge the further away they are from the eye because they travel along a linear path, as shown in Figure 2.16. Motion parallax is where objects that are further away from the eye appear to move slower than closer objects, when the eye is moving.

Stereoscopic depth cues provide information from two eyes (cameras). The images received by each eye only differ in the fact that they are different perspectives of the same location. It is this difference in perspective that gives us the spatial information to determine depth. This spatial information is obtained from the parallax between the objects in the images and parallax is the apparent displacement in the position of an object when viewed from two different perspectives. The parallax is greater for objects that are closer to the eye and smaller for objects that are further away from the eye.

Motion depth cues are similar to motion parallax, which was described above, except that the eye is stationary. One thing to note is that motion depth cues are not always present. That is why monoscopic and stereoscopic depth cues are important.

Now that depth cues have been explained, and since this thesis focuses on visual

Figure 2.16: Example of linear perspective [70].

feedback, the two ways of achieving stereo vision should be explained, which are active and passive stereo. Active stereo is achieved by using a projector that has a refresh rate of at least 120Hz, 60Hz for each eye. It alternates between the left and right eyes and the user wears special glasses (shown in Figure 2.17) that are synchronized to the output signal of the projector. When the left eye image is displayed the glasses turn the right lens opaque and vice versa. It is this toggling between the images for each eye that creates the stereoscopic depth cues, if rendered with the correct perspectives for each eye, necessary to determine depth. Passive stereo uses two projectors, one for the left eye's image and one for the right eye's image. Each projector has a unique polarized filter and the user wears a pair of glasses that match the polarization of the filters. This way the left eye views the image from the left projector and the right eye views the image from the right projector even though both images appear on the screen at the same time and similar to active stereo, it is because of these two

Figure 2.17: Active stereo glasses.

different images that stereoscopic depth cues are created. Active stereo provides a strong sense of depth, which is why most stereoscopic displays use active stereo [62].

2.4.2 Stereoscopic Displays

Stereoscopic displays are usually capable of displaying two different images at a rate of least 60Hz each. This means that each eye will receive one of the two images, which then creates the stereoscopic depth cues explained in the previous section. There are several types of VR systems: fishtank, Head Mounted Display (HMD), and projection-based.

Fishtank VR systems use an ordinary desktop computer that has some VR graphics hardware attached, which is shown in Figure 2.18. These systems have the advantage of being relatively inexpensive, but they are limited compared to the other VR systems because of their stationary placement and screen sizes [62].

HMDs worn on the user's head, as shown in Figure 2.19. The advantage of a HMD is that they are not stationary like fishtank systems and they are less expensive than larger projection systems. They also typically have a better field of regard (FOR) when compared to projection systems. FOR is a percentage of all possible directions that a user can move their head without looking away from the VR display.

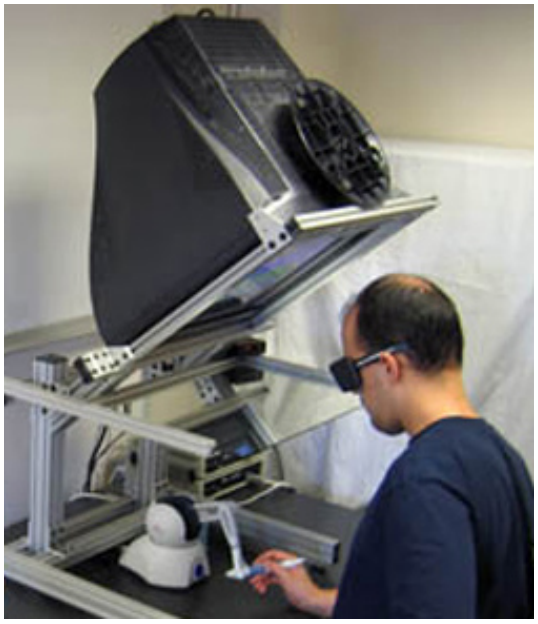


Figure 2.18: Example of a fishtank VR system [5].

Figure 2.19: Example of a HMD [5].

Projection-based displays use projectors, screens, and typically mirrors. They range from one-sided to six-sided projection systems. One thing to note is that the FOR advantage of HMDs is no longer relevant when using a six-sided projection system. One main advantage of a projection-based system is that more than one user can view the system at a time. Projection-based systems also perform perspective correction when rendering, which means that the user's orientation and position are taken into account when the scene is rendered. Figures 2.20 and 2.21 are examples of a four- and six-sided active stereo projection system, respectively.

Figure 2.20: Four-sided stereo projection system.

2.4.3 Input Devices

In order for the user to be able to interact with the virtual world and for it to be drawn correctly, the VR system needs to have some kind of input devices. These devices

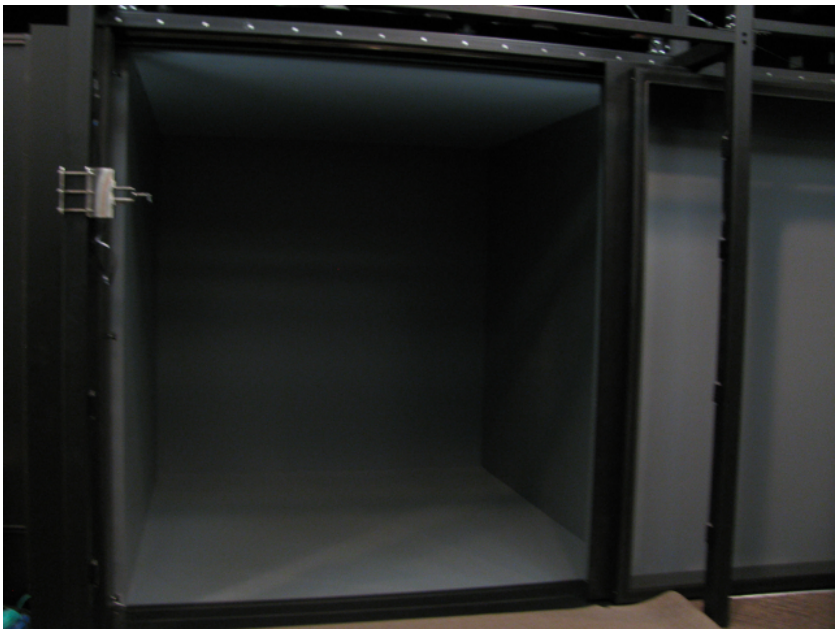


Figure 2.21: Six-sided stereo projection system.

need to let the VR system know the user's position and orientation, along with what the user is trying to do. There are many types of input devices. They range from joysticks to gamepads. They can be tracked or untracked devices. Typically tracked devices have six-degrees of freedom (6-DOF), meaning their position and orientation is known. One common tracked device that is very important for a projection-based system is a head tracking device. These devices tell the VR system their position and orientation, which is then used to display the virtual world from the correct position and perspective. An example of a head tracking device is shown in Figure 2.22.

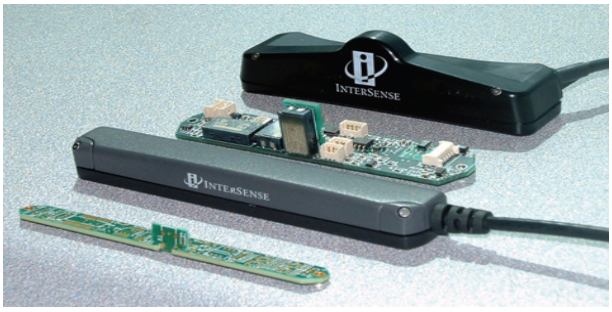


Figure 2.22: Two 6-DOF head tracking devices [72].

Another common input device is called the wand. A wand is a 6-DOF tracked device with several input buttons and a joystick. An example of two wands is shown in Figure 2.23. Using these kinds of devices the user can interact with the VR system and the VR system can display the virtual world correctly for the user.

Figure 2.23: Two 6-DOF wand devices [72].

2.4.4 Toolkits

VR Toolkits are a very important part of a VR system because of the specialized hardware, devices, and unique environments. They are designed to help the user deal with these items. There are many VR toolkits out there that try to handle the creation of stereoscopic images, setup of the VR environment, and device interfaces. FreeVR [61], VR Juggler [12], Virtual reality user interface (Vrui) [33], and Hydra [24] are all examples of VR toolkits. However, since this thesis uses a volume rendering library built on Vrui, it will be the only one explained with some detail.

Vrui is a VR development toolkit written by Oliver Kreylos [33]. Vrui's main goal is to shield the developer from a particular configuration of a VR environment, which makes it so that applications can be developed in a quick, portable, and scalable fashion. There are three main components to Vrui: display abstraction, distribution abstraction, and input abstraction.

Display abstraction is done by providing OpenGL rendering contexts for each screen, such that rendering a model in user-specific coordinates renders that model with the correct orientation and position on all rendering screens.

Distribution abstraction is done by hiding the detailed aspects of distribution. There are three types of distribution: split first, split middle, and split last. Split first replicates the application on all machines and distributes the synchronized input device data to all machines. Split middle is where a shared data structure is used to pass data from the main simulation/application machine to all rendering machines. Split last is a broadcast of the OpenGL API call stream from the main simulation/application machine to all rendering machines. Also, the VR toolkit should hide the type of system it is being run on. For example, a shared-memory multi-CPU machine should appear no different than a cluster of machines.

Input abstraction is done by hiding the differences in hardware. This is achieved by allowing the user to specify the application's input requirements at a high level and then allow the user to map input devices to those requirements. An example of this is that in simulation mode the mouse acts as an input device and when running in an IVE the wand acts as an input device, but no additional work is done by the application programmer.

Vrui also provides an integrated user interface. It allows for developers to easily create a menu system in a 3D virtual environment. There are not many VR toolkits that have a menuing system and it was one of the main reasons why Vrui was the chosen VR toolkit for the volume rendering library used in this thesis.

2.5 Toirt-Samhlaigh

Toirt-Samhlaigh is a volume rendering library mainly written by Patrick O'Leary [49, 48] while at Desert Research Institute (DRI). It uses a 3D-texture mapping, hardware accelerated DVR type of algorithm that performs slice-based rendering, which uses ideas from C. Rezk-Salama and A. Kolb [57]. It uses a heterogeneous data structure with 3D bricks of volume data stored in 3D textures, which uses the algorithm by D. Ruijters and A.Vilanova [56]. The 3D bricks contain octrees, and ocnodes that are empty after the transfer function is applied are skipped. It uses shaders along with 3D-textures for the rendering of volumes and performs view frustum culling to help

speed up the rendering process. It performs back-to-front order rendering to acquire the final image displayed. It is heavily integrated with Vrui, described previously, which allows it to run on many different types of hardware ranging from laptops to IVEs. Also, because of the fact that Toirt-Samhlaigh contains many features that meet the Functional Requirements, outlined in Chapter 4, and is integrated with a VR toolkit, it was chosen as the volume rendering library used in this thesis.

Toirt-Samhlaigh has many features that aid in the visualization of volumes and a few key features will be described here. One main feature of Toirt-Samhlaigh is that it has a 1D and 2D transfer function. A transfer function is a function or set of functions which map data values of interest into colors and opacities. The 1D transfer function has two types of functions: piecewise-linear and Gaussian. Along with transfer functions, Toirt-Samhlaigh provides directional lighting that has a user interface for manipulating its position and color. There are many color maps that can be applied to the data to change the range of colors displayed. The slice factor can be changed during the execution of the program, which determines the distances between the slices that are drawn through the volume. Toirt-Samhlaigh provides a clipping tool that allows users to place an arbitrary clipping plane. Additionally, users can change the orientation and position of the volume. Finally, it can load .raw and .dat data formats.

2.6 Related Work

As of this writing there are many GPR visualization software packages out there. They all perform many of the same tasks, with the differences between them ranging from display options to tools and formats supported. Here are a few of the current GPR visual softwares.

GPR-SLICE, which can read and display GPR data [22], is a ground penetrating radar imaging software designed for creation of 2D/3D images for use in a variety of fields. It was created by Dean Goodman and has been in production since 1994. It runs on a typical Windows desktop machine.

Ground Vision and Easy 3D were created by Malå GeoScience [40]. Ground Vision is used for the acquisition of data, including support for GPR data, and Easy 3D is the visualization software for Ground Vision. It provides the ability to have a 2D/3D visualization of GPR data. They run on a typical Windows desktop machine.

Another program called Easy 3D was created by AEGIS Instruments [1]. It provides the ability to visualize GPR data from a single channel or multi channel GPR system. It has several functions to help in the aid of viewing the GPR data, such as a filtering option, and it can create a 3D visualization. It runs on a typical Windows desktop machine.

Chapter 3

Visualizing and Analyzing GPR Data

There are many applications out there that can visualize GPR data. Several of these were described in Chapter 2, and they have lots of options and abilities. They all suffer from the same problem: they are all desktop applications, which rely on desktop input devices for interaction with the data as well as visualization. The interpretations of this data are unfortunately based on the user's subjective analysis of GPR reflection patterns, which makes this analysis dependent on the interpreter's expertise and experience [42].

This thesis provides a way to break away from this problem by giving the user the ability to take advantage of the visualization and more natural interactions provided by an IVE with tracked input devices. When looking at the results from Prabhat [53], the scientists perfered using a projected-based IVE over a Fishtank IVE and desktop display, to perform tasks on the visualization of their confocal microscope data. An IVE allows the user to become visually immersed with the data they are viewing. It allows the user to walk around and through the data, which provides views not possible on a desktop environment. When an IVE is combined with tracked input devices, it provides the user with the ability to perform interactions not possible on a desktop display.

My goal is to allow a user to view GPR data as either a 2D slice or 3D volume. I want to provide the user with many tools and menus for manipulating the view and

look of the volume. I will also provide the user with some GPR specific tools, such as a Distance Measurement Tool and Brunton Compass Tool. The Brunton Compass allows the user to take dip and strike measurements of subsurfaces and then view those measurements as a non-uniform vector field. This provides the user with a unique view of the subsurfaces structure.

The goal of this thesis is to provide the user with a system that can immerse them in GPR data sets, with unique tools for data analysis. Performing the rendering and interaction using a system that works in an IVE with tracked input devices helps achieve this goal.

Chapter 4

Software Specification and Design Process

In this chapter, a software specification will be outlined through the use of the systems functional and nonfunctional requirements, a use case diagram and a system overview diagram. Additionally, the iterative design process used to help shape this project and lessons learned from it will be described and discussed. The software engineering and practices used by this thesis were obtained from [3] and [64].

4.1 Requirements

Before the engineering of the system could be conducted the requirements of the system needed to be defined. These requirements determine the functionality of the system and how the system's behavior. They are separated into two categories: Functional and Nonfunctional Requirements.

4.1.1 Functional Requirements

The goal of this project was to create a system that provides the users with the ability to visualize and analyze a GPR data set. The functional requirements, listed in Table 4.1, reflect this goal. Most of them were created during the initial phase of the iterative design process, described later, and several requirements were added during the subsequent iterations of this project.

F01	The program shall be able to load GPR data in SEG Y revision 1 format.
F02	The program shall allow the user to turn the data visualization on and off.
F03	The program shall allow the user to change the orientation and position of the data.
F04	The program shall perform topographic correction to loaded GPR data if a topographic file is provided.
F05	The program shall provide the user with multiple transfer functions to allow for the manipulation of the data visualization.
F06	The program shall allow the user visualize the surface of data, if a topographic file is provided.
F07	The program shall allow the user to turn the display of the surface visualization on and off.
F08	The program shall allow the user to modify the opacity of the surface visualization.
F09	The program shall allow the user to be able to view axis-aligned slices of the data.
F10	The program shall allow the user to use the axis-aligned slices as clipping planes.
F11	The program shall allow the user to take dip and strike measurements.
F12	The program shall allow the user to take distance and angle measurements.
F13	The program shall allow the user to place an arbitrary clipping plane.
F14	The program shall allow the user to modify the lighting.
F15	The program shall allow the user to load and save transfer functions.
F16	The program shall allow the user to load and save lighting.
F17	The program shall allow the user to save distance measurements.
F18	The program shall allow the user to load and save dip and strike measurements.
F19	The program shall be able to visualize the dip and strike measurements as a vector field.
F20	The program shall allow the user to turn the display of the dip and strike measurements on and off.
F21	The program shall provide the user with a graphical user interface to aid in the use of the program's functionalities.

Table 4.1: Functional Requirements.

4.1.2 Nonfunctional Requirements

The nonfunctional requirements, listed in Table 4.2, were created to describe the system's behavior. They outline the technologies used in development, as well as the platform supported, and reflect the constraints and goals of the system. An example is to maintain interactive frame rates (greater than 30 frames per second) which are important in a VR system because it helps avoid user fatigue, the disruption of motion parallax, and the usability of the system.

NF01	The program shall maintain interactive frame rates.
NF02	The program shall be written in C/C++.
NF03	The program shall use a hardware accelerated, texture mapping DVR algorithm for its rendering.
NF04	The program shall use Vrui and Toirt-Samhlaigh for its prototype application.
NF05	The program's rendering algorithm shall use OpenGL.
NF06	The program shall be implemented on the Linux platform.

Table 4.2: Nonfunctional Requirements.

4.2 Use Cases

A use case diagram was created to present a graphical overview of the systems functionality. This diagram is shown in Figure 4.1. The use cases were derived from the functional requirements in Table 4.1. The use case diagram shows how each actor affects the system. For example, time only affects the rendering of the system, while the user can affect the system in many ways. These range from changing the position and orientation of the data to the use of various tools.

4.3 System Overview

The system consists of one main system, Toirt-Samhlaigh, which is made up of several subsystems and uses a VR toolkit. Toirt-Samhlaigh deals with everything from the volume rendering to tool creation, while Vrui is the VR toolkit used by Toirt-

Figure 4.1: Use Cases.

Samhlaigh for dealing with the VR environment and its menuing capabilities. An overview of this can be seen in Figure 4.2.

Figure 4.2: High level overview of the system.

4.3.1 Toirt-Samhlaigh with Expansion and Enhancements

In this section an overview of Toirt-Samhlaigh is presented, which describes its many subsystems, and is followed by an explanation of what subsystems were expanded and enhanced during this project.

Toirt-Samhlaigh, described in Chapter 2, is a volume rendering library which we expanded. It is composed of many subsystems that determine its functionality. The purpose of these subsystems is to make the system easier to manage and provides its structure. An outline of its subsystems is shown in Figure 4.3.

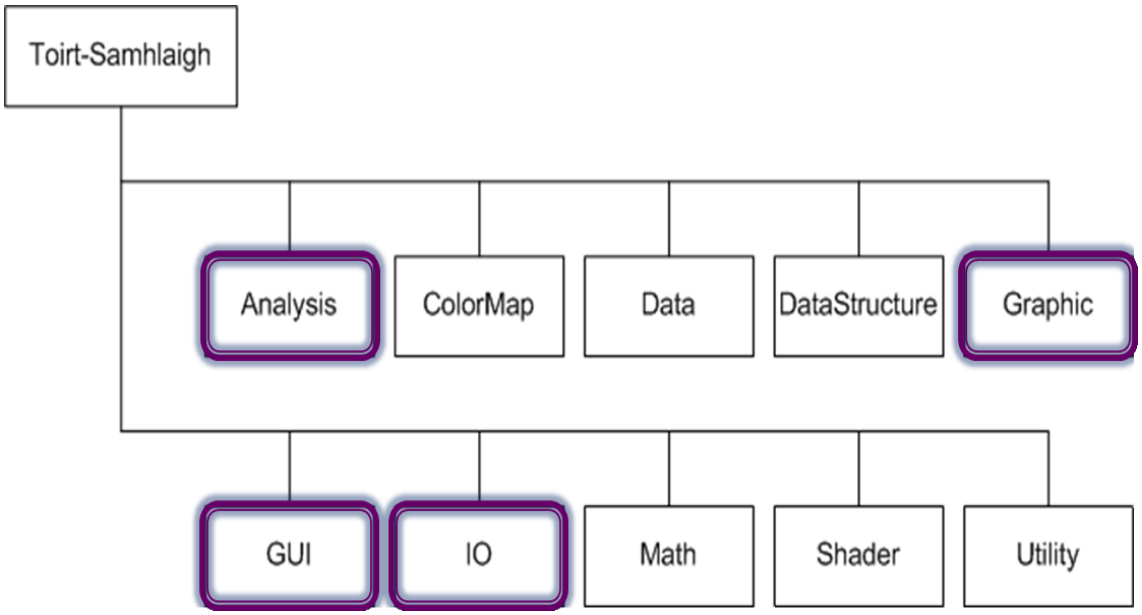


Figure 4.3: Toirt-Samlaigh's subsystems with modified ones highlighted.

A brief description of Toirt-Samhlaigh's subsystems follows:

- **Analysis**: This subsystem contains the tools used by the system to assist with the analysis of the volume. The Brunton Compass, Clipping Plane, and Distance Measurement Tool are some of the classes contained in this subsystem.

- **ColorMap**: This subsystem contains the components necessary to create the color maps used in several GUIs. Some of the items that make up this subsystem are Control Points, a Store class, and several callback classes that deal with changes to the color maps.

- **Data**: This subsystem is composed of the classes that contain the data used by the system, such as the Volume class.

- **DataStructure**: The pieces of this subsystem are ones which help improve the speed of the system and assist in the rendering of the volume. These pieces range from the Bricks used for rendering to OctTrees and OctNodes.

- **Graphic**: This subsystem holds everything that deals the direct rendering of the volume. It contains the Scene class, which contains all the pieces necessary to render the volume (such as shaders), a class used for frustum culling, the surface visualization class, and the texture class. FreeImage [21] is used by the texture class to load texture data.

- **GUI**: This subsystem contains all of the components necessary to create the many widgets used by the system, along with several of the widgets themselves. Some of the widgets included are 1D Transfer Function, Slices, Shading, and Lighting.

- **IO**: All of the components contained in this subsystem are used for loading the many different formats supposed by the system. A few of these formats are .raw and .dat. Also, the SEG Y file loader is contained here, which allows for the loading of SEG Y slice or volume files.

- **Math**: This subsystem contains the components that deal with the linear algebra math of the system, such as points, vectors, and matrices.
- **Shader**: The classes of this subsystem are the ones used for handling shaders. A few things contained are a shader manger, loader, program, and error checker.
- **Utility**: This subsystem holds all the classes that contain commonly used functions. Examples of these are a string tokenizer class, which converts a several data types into a string, and a class used to throw run time errors with custom messages.

The subsystems that were expanded and enhanced during this project were highlighted in Figure 4.3. Most of the enhancements described here are covered in more detail in Chapter 5. The Brunton Compass and Distance Measurement Tools were added to the Analysis section. The Graphic subsystem had modifications done to the DataItem structure in the Scene class, which is used by each rendering context. It also had a surface visualization class added to it. This class is used for creating and rendering the surface of a GPR data set when a topographic file is provided. It also has a texture class used for loading textures into OpenGL. Several modifications were done to the GUI subsystem. Mainly most of the components here had a save and load functionality added along with a few minor tweaks. Finally, the IO subsystem was modified to allow for the loading of the SEG Y data file format.

4.4 Iterative Design Process with a Researcher

Now that the software design of this project has been explained, the iterative design process of the project should be explained. This project was conceived by a researcher at DRI. Since this project was developed at DRI it allowed for an iterative design process to be performed. Meetings were held weekly to determine the current progress of the project and to provide much needed feedback. During one of these meetings the current status of the project would be looked at, along with where to go next and

what things could be added or removed. The feedback gained from these meetings was instrumental in the development of this project. It shaped the project into its current form. There were several important advantages to this iterative design process. It allowed for tools to be continually tested and modified to the researcher's satisfaction. It allowed for extraneous features to be cut from development, which allowed for more focus to be put towards what the researcher considered more important features.

Along with helping shape development, it taught the developer many important things. The first is that what the developer finds easy to understand will not necessarily be simple to understand for the user. Interface design is best done by allowing the user to provide input before creating an initial design. Getting feedback on an essential component during its implementation can save valuable time compared to finishing the implementation and then getting feedback that alters the design of it. Another important thing learned from this is that it is always better to err on the side of caution and not assume that someone has an understanding of the subject matter being discussed because this can save valuable time.

Chapter 5

Implementation and Results

In this chapter, Section 5.1 will cover the implementation of the system, which is explained through describing the visualization of GPR data, the menu system designed, and tools created. Along with these, Section 5.2 contains the results acquired from the creation of these components and some analysis of GPR data.

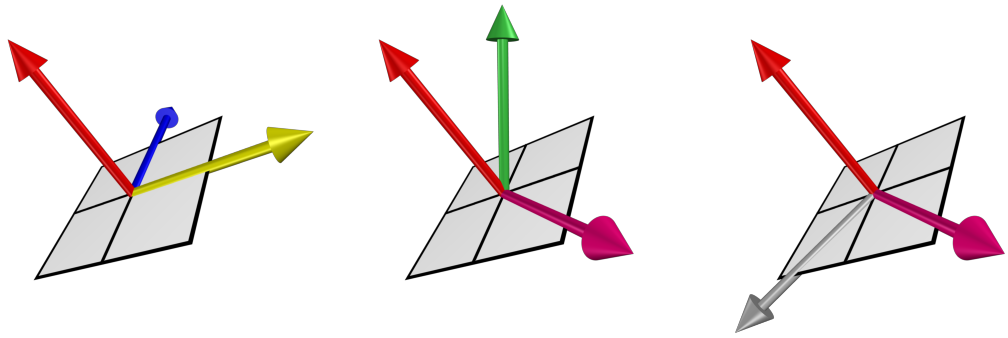
5.1 Implementation

5.1.1 Visualization of GPR data

Toirt-Samhlaigh was the volume renderer chosen for this project. It contains many components which will not be described in detail here. Of interest are those which were modified and/or added to Toirt-Samhlaigh in order to complete the Functional Requirements. The first thing created to enhance Toirt-Samhlaigh was a SEG Y file loader. Then the various tools available in Toirt-Samhlaigh were tested to verify that they worked with this newly loaded data. The slicing tool of Toirt-Samhlaigh was adjusted to handle the scaling of the GPR data and it was modified to double as a clipping plane tool. Next topographic correction was added into the SEG Y data loader. Then surface visualization was created and transparency was then added to the surface visualization, which proved to be rather tricky. This was because depth writing is turned off when the volume is rendered and if depth writing is turned on, it makes it impossible to see the surface since the volume is a cube and the surface is rendered inside this cube. The way around this was to first render the surface before

the volume is rendered and have front face culling on and then render the surface again after the volume was rendered, but this time have back face culling on.

Once the surface was done two tools were created: the Distance Measurement Tool and the Brunton Compass Tool. The Brunton Compass Tool seemed like it would be a simple tool to create; however, it turned out to be a rather complex tool. The first issue was to get the correct dip and strike measurements. These measurements are calculated by first taking the forward and right vectors of the plane and transforming them using the volumes current transformation. Once this is done these vectors are then normalized and then the cross product of them is taken to get the normal vector of the transformed plane and normalized again. Next the up vector of the volume is crossed with the normal vector of the plane, which gives the strike line of the surface. Now that the strike line has been calculated, all that is left is to calculate the steepest gradient. This is done by crossing the strike line with the normal vector of the plane. Now that the strike line and steepest gradient have been calculated, the dip and strike measurements can be determined. The angle of dip is calculated by performing the inverse cosine of the dot product between the strike line rotated 90° and the gradient. The angle of strike is then measured by performing the inverse cosine of the dot product between the strike line and the north vector. Figure 5.1 outlines the creation of the vectors used to calculate the dip and strike angles.



(a) (b) (c)

Figure 5.1: Creation of vectors used to calculate dip and strike.

Figure 5.1(a) shows the plane's forward (blue), right (yellow), and normal (red) vectors. Figure 5.1(b) shows the plane's normal (red) vector, volume's up (green) vector, and the strike line (magenta) vector. Figure 5.1(c) shows the plane's normal (red) vector, strike line (magenta) vector, and steepest gradient (gray) vector. Once this was completed the visualization of these measurements as a non-uniform vector field was done. Finally, a load and save functionality was added to many of the tools provided by Toirt-Samhlaigh and to the two tools described above.

Volume Rendering

SEG Y format files are currently the only data that can be read into the GPR portion of Toirt Samhlaigh. A data loader was developed that is specifically designed to read in this SEG Y format. From this data, either a single SEG Y file can be read in and duplicated to create a volume or a list of files can be read in to create a volume. After reading in the data, the effective dimensions of the slices are standardized by finding the smallest number of traces and samples and scaling the rest of the slices to these values. Creating the volume from a single file is done by creating a single slice of voxels and then stacking two copies of the slice together to form the 3D volume. The same applies for creating a volume from several files except for a slice is created for each file and then they are stacked together to form the 3D volume, as shown in Figure 5.2. Once this volume is loaded, Toirt Samhlaigh renders it by drawing planes through the volume that are orthogonal to the direction of the camera. Then for every fragment of the plane that intersects the volume a texture lookup is performed along with a interpolation step to determine the color of the fragment. Also, these planes are rendered from back to front to handle alpha blending.

Topographic Correction

The reason for applying topographic correction is that when the data is collected, the GPR device stores the data as if the terrain is flat when in reality it is not. To account for the slope of the sand dune, which helps get more accurate dip/strike measurements

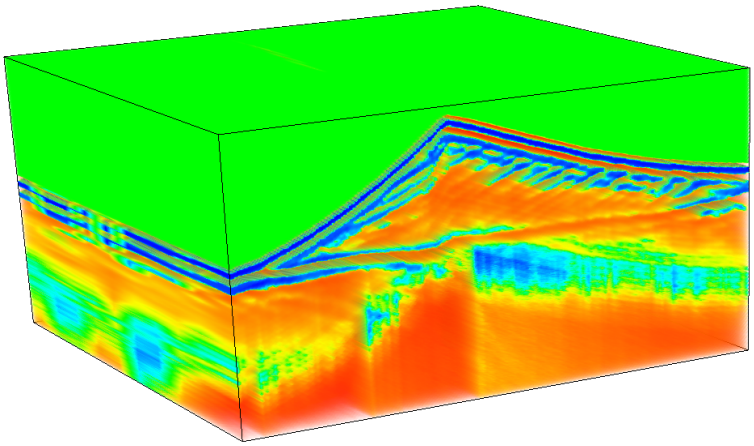


Figure 5.2: An example of a GPR data set being rendering using Toirt-Samhlaigh.

and easier layer visualization, a topographical correction must be applied. To do this the topography data is read into a 2D grid structure. The format of the topography data is x-coordinate, y-coordinate, elevation. The topography data includes elevation values at regular intervals as well as a sample at the peak. Since there are samples at the peak, the topography data cannot be stored as a texture, but the samples at the peak make it so that the peaks are not missed during interpolation. When provided with a topography file in this format corresponding to the GPR data, a topographic correction can be applied to that data, as shown in Figure 5.3. Then when reading in the SEG Y file(s), linear interpolation is performed between samples to determine how much to vertically shift the placement of the data value in the volume. Figure 5.4 shows how the topographic correction is applied to the volume data.

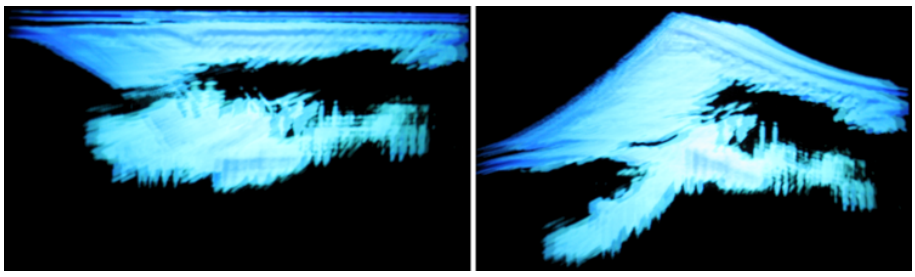


Figure 5.3: The left image is the data before topographic correction is applied and the right image is after.

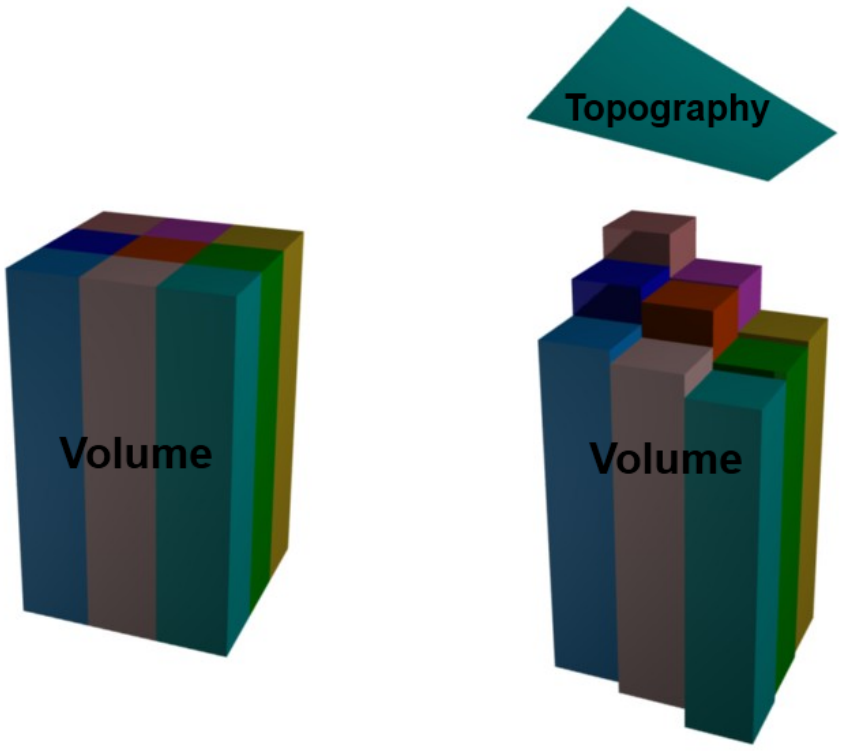


Figure 5.4: GPR data in columns before (left) topographic correction is applied and after (right).

5.1.2 Menu System

To help the user better visualize the readings from the tools described in the next section, to simplify the actions performed using the tools of this project, and to provide easy interaction with the system, a menu system was used. The menu system used for this project was Vrui. Vrui was used because it was already integrated into Toirt-Samhlaigh. As described in Chapter 2, Vrui provides an easy-to-program user interface library. Figures 5.5 and 5.6 are examples of some of the user interfaces created for this system.

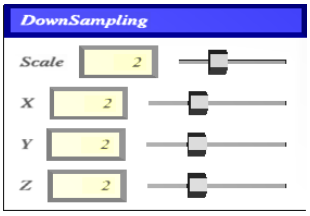


Figure 5.5: Simple user interface created using Vrui.

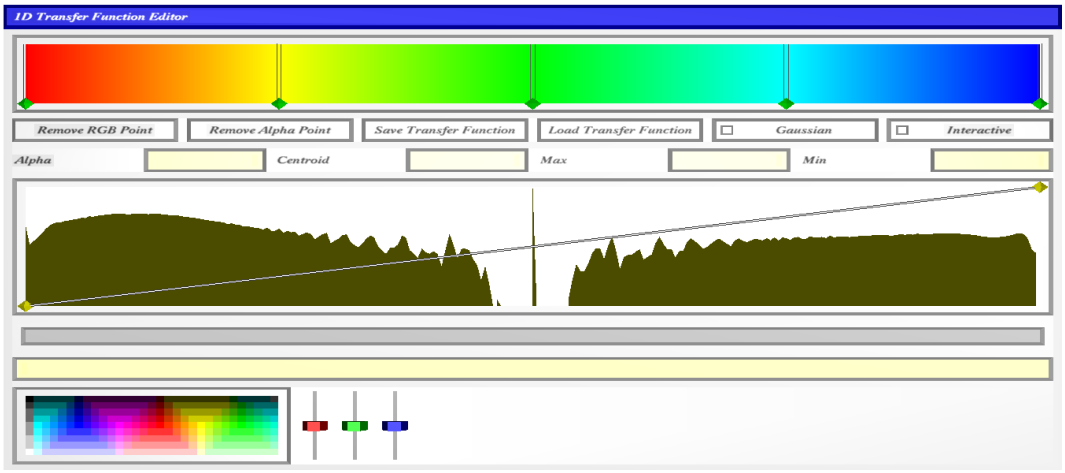


Figure 5.6: Complex user interface created using Vrui.

5.1.3 User Tools

In this section two of the main tools created for GPR data analysis are explained: Brunton Compass Tool and Distance Measurement Tool

Brunton Compass Tool

In order to measure dip and strike it was necessary to create a Brunton Compass Tool (shown in Figure 5.7). The angle of dip is a measure of a surface's steepness relative to horizontal along it's gradient. The angle of strike describes the orientation of the surface relative to North. This tool provides the user with a plane that can be placed in the VR environment. Based on the orientation of the plane the user is provided with the dip and strike, along with the coordinates of the plane's center. All measurements are calculated with respect to the transformation of the volume. In other words, when

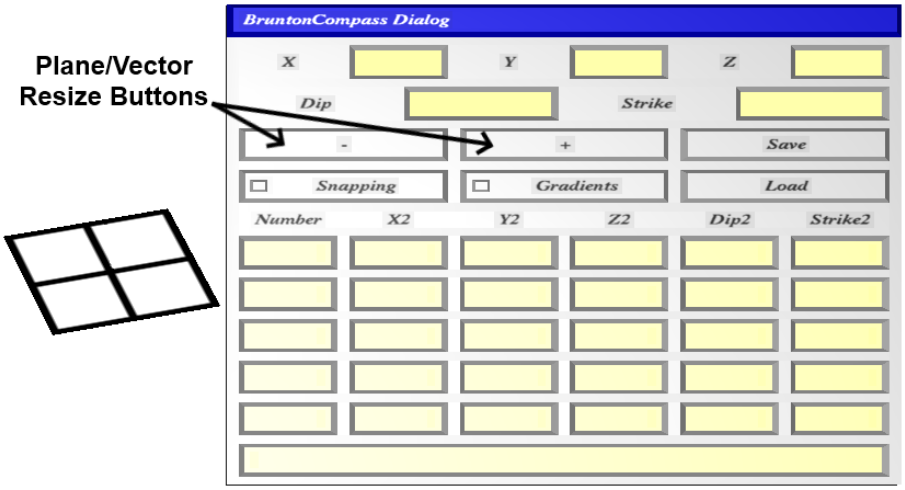


Figure 5.7: Brunton Compass with user interface.

the volume is transformed the North and vertical vectors are transformed along with it. This is helpful for when the volume's orientation is modified by the user. For ease of use, the user has the ability to increase or decrease the size of the plane, save its current values, and choose between having the plane snap to the 6-DOF input device or perform transformations relative to the input device. The plane always moves relative to the 6-DOF input device; however, with snapping enabled, the plane is transformed to the position and orientation of the device before each movement. The benefit of using a 6-DOF input device is that it allows the user to place the plane at any orientation and position in the data. The measurements that were taken, either during this session or saved and loaded from a previous session can be viewed as a non-uniform vector field, which provides the user with an unique view of the GPR data. To help the user easily visualize the values from the Brunton Compass Tool, a user interface was created that displays the following information: current measurement (dip, strike, plane center), last five saved measurements, plane/vector resize buttons, toggle snapping button, toggle gradient button, a 'Load' button, and a 'Save' button. Additionally, the plus and minus buttons on the Brunton Compass Tool user interface are used to increase and decrease the size of the Brunton Compass plane and the vectors in the non-uniform vector field.

Distance Measurement Tool

To allow the user to take measurements from one user selected point in 3D space to another, I created a Distance Measurement Tool (shown in Figure 5.8). The user creates a start and end point using the 6-DOF input device to take a measurement. There are two methods for creating the points using the input button: press and release or press, hold, and release. The second method for creating one of the points allows the user to drag the point to the desired location and if the point is the end point, the user will see the line being continually updated and drawn. The user can then repeat these steps as many times as desired. A marker is drawn at each point and when both points are defined a connecting line segment is drawn. The user has the ability to label what type of measurement they are taking, whether it is a horizontal x, horizontal y, or vertical z and can save the current measurement. The distance calculated is the Euclidean distance and the labels have no affect on how the measurement is calculated. Similar to the Brunton Compass Tool, a user interface was created so the user can visualize the measurements being taken. The following information is displayed through this interface: start point (X, Y, Z), end point (X, Y, Z), Distance, buttons for labels, and a 'Save' button.

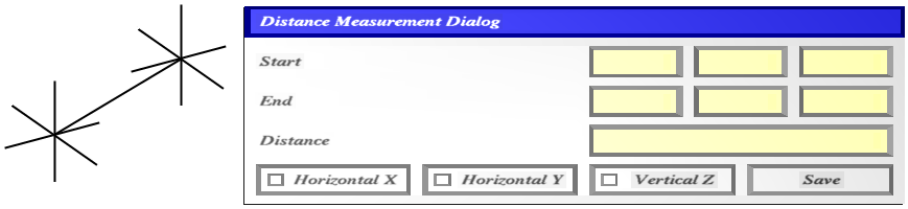


Figure 5.8: Distance Measurement Tool with user interface.

5.2 Results

To help with the visualization and analysis of the GPR data, the user can perform many modifications to the data. The tools and features described later provide explanations of how the modifications and tools assist with the visualization and analysis

of the GPR data. An example showing a GPR data set being rendered with Toirt-Samhlaigh is shown in Figure 5.2. This data has been topographically corrected and also has a surface that can be rendered (shown in Figure 5.9).

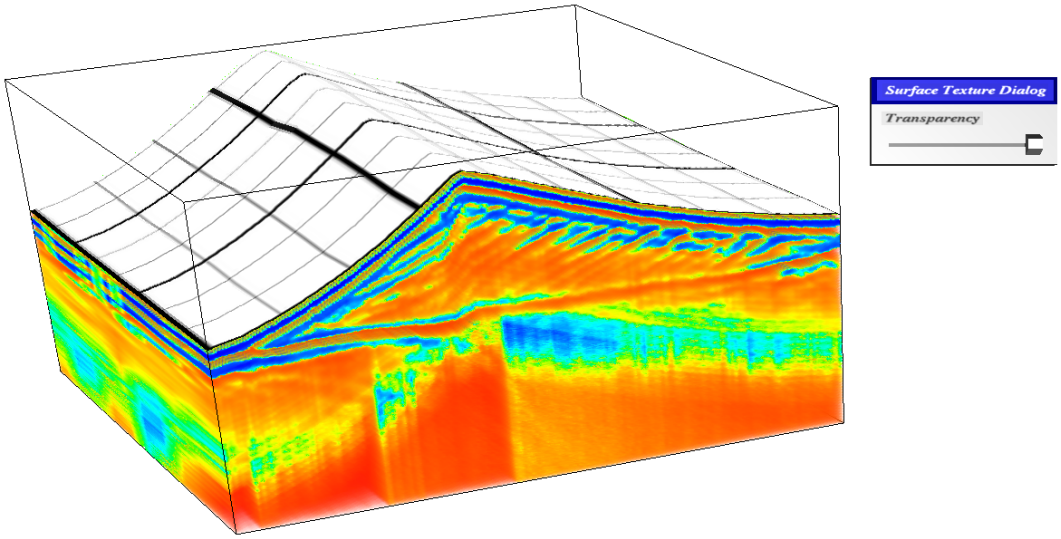


Figure 5.9: Surface visualization for a topographically correct GPR data set.

The transparency of the surface can be modified by the user, ranging from 100% transparent, slider all the way to the left, to 100% opaque, slider all the way to the right. Figures 5.10–5.12 are examples of the surface rendered at different transparency levels. Being able to see the surface is useful for the research because it allows them to determine where exactly they are working at in the data.

The two tools described in the previous section are currently the main way to assist in the analysis of the GPR data. The Distance Measurement Tool is useful when the user needs to determine the distance between items in the data. An example of the Distance Measurement Tool being used in a GPR data set is shown in Figure 5.13. The Brunton Compass Tool simulates its real life equivalent by allowing the user to take dip and strike measurements. Figures 5.14–5.17, are examples of the Brunton Compass Tool in use.

Once dip and strike measurements have been taken the user can view them using a non-uniform vector field (gradients). Figures 5.18–5.20 are examples of this non-

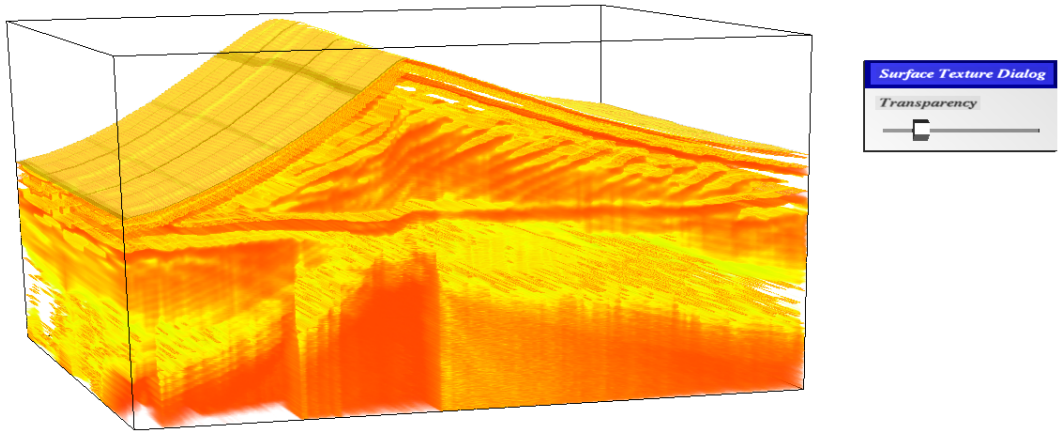


Figure 5.10: Surface visualization for a topographically correct GPR data set being rendering with a transparency of 75%.

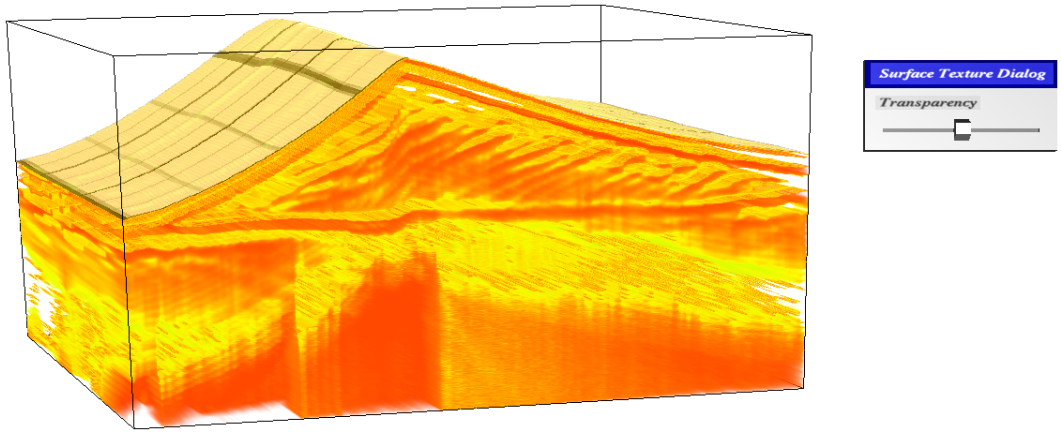


Figure 5.11: Surface visualization for a topographically correct GPR data set being rendering with a transparency of 50%.

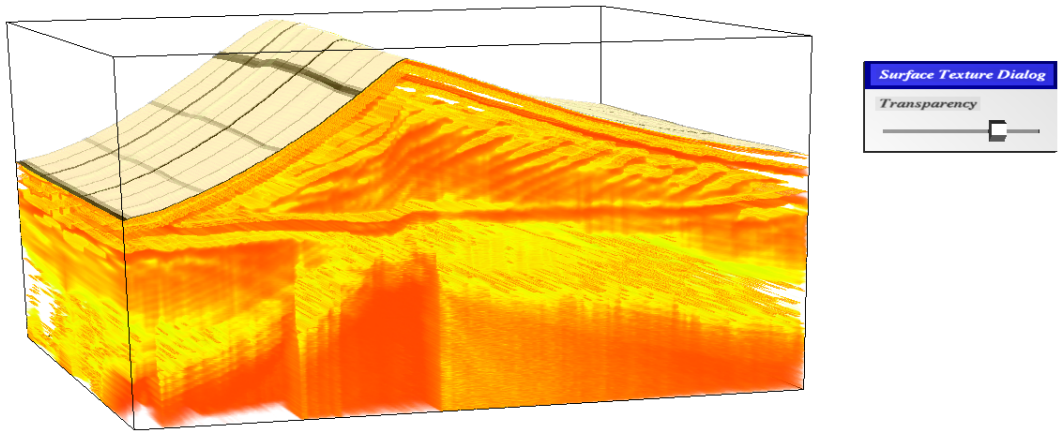


Figure 5.12: Surface visualization for a topographically correct GPR data set being rendering with a transparency of 25%.

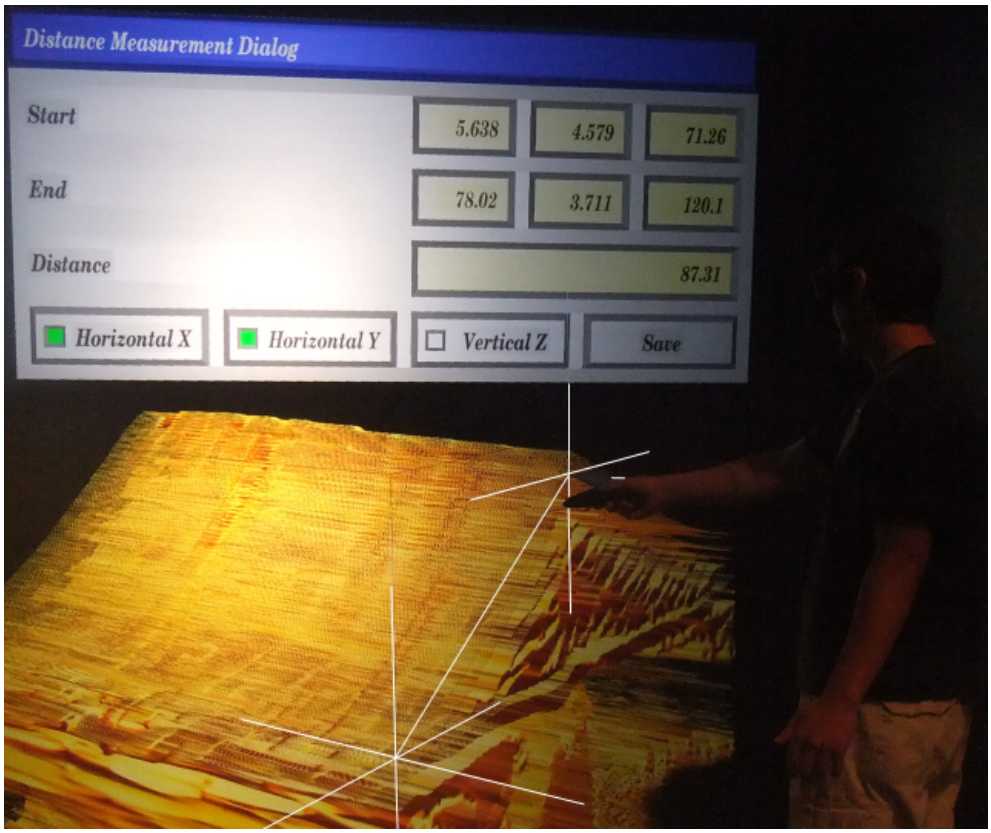


Figure 5.13: Distance Measurement Tool being used in a GPR data set with lines enhanced.

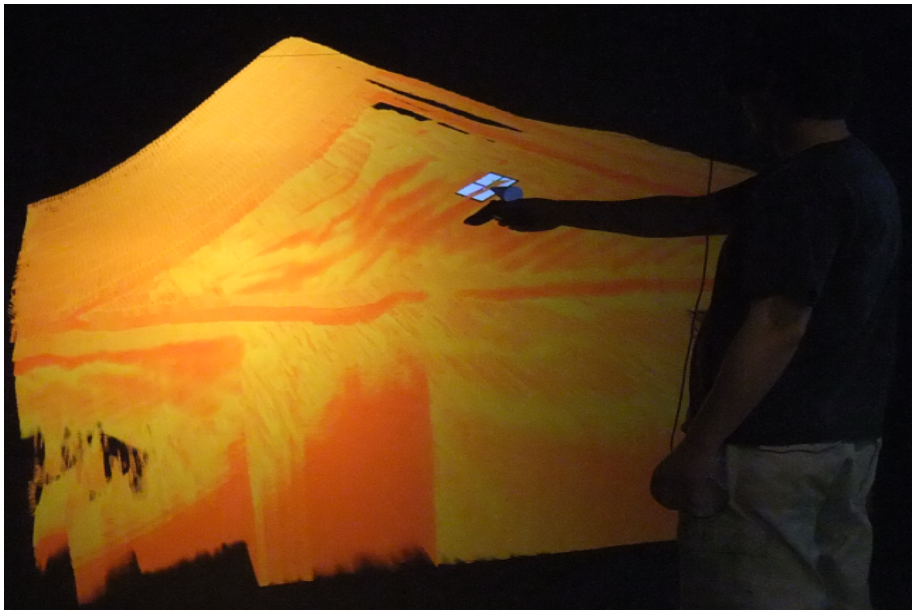


Figure 5.14: Brunton Compass Tool being positioned in the GPR data to take a measurement with snapping enabled.

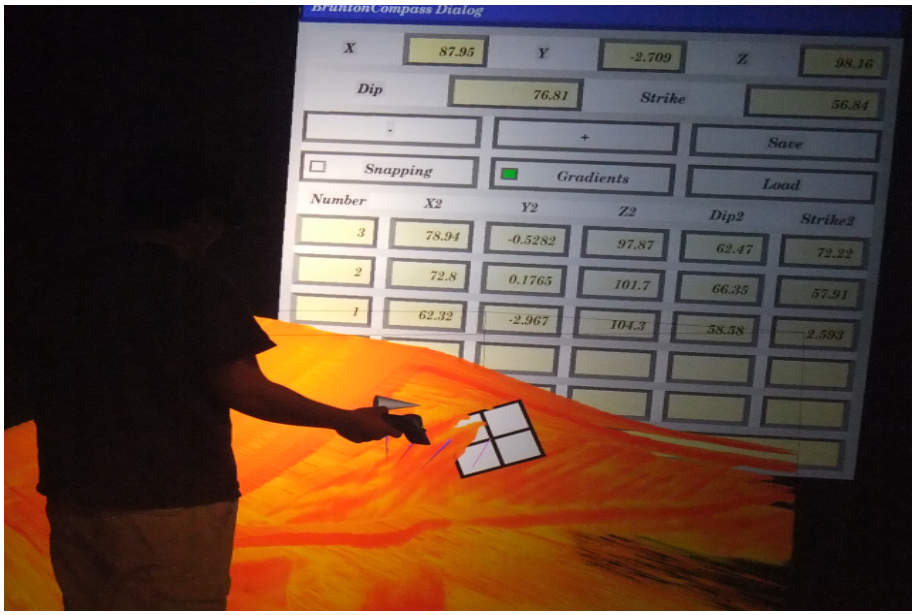


Figure 5.15: Brunton Compass Tool being used in a GPR data set, which shows several saved measurements.

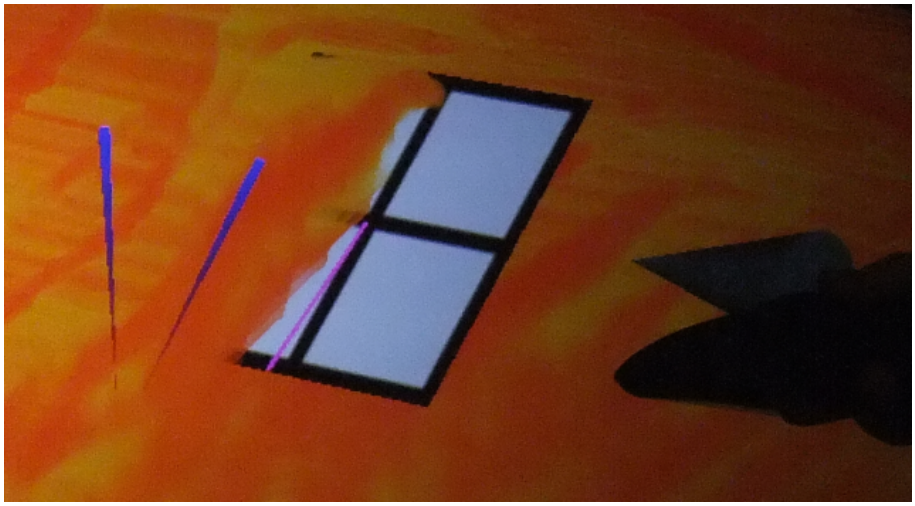


Figure 5.16: Brunton Compass Tool with several saved measurements and steepest gradient visible.

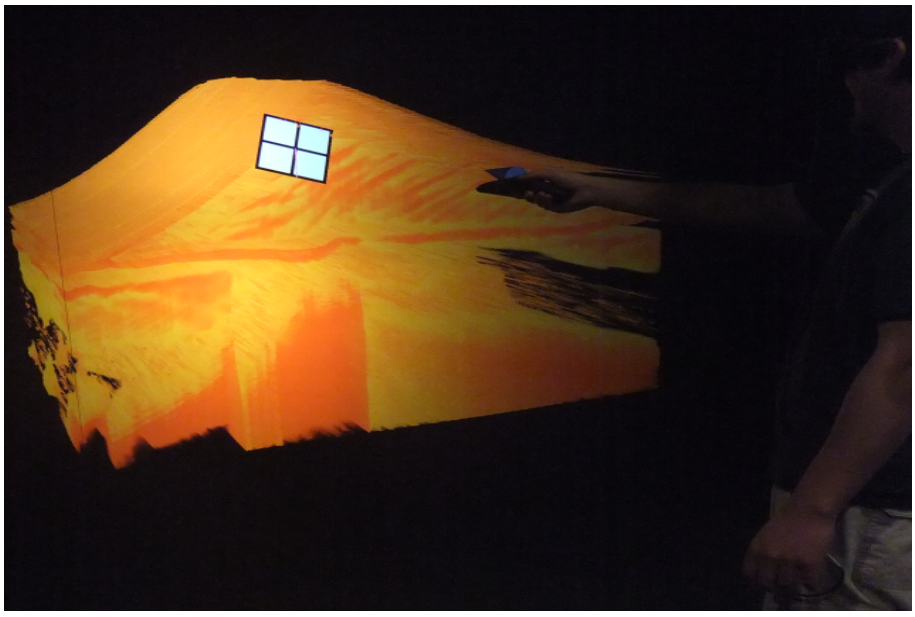


Figure 5.17: Brunton Compass Tool being used in a GPR data set with snapping disabled.

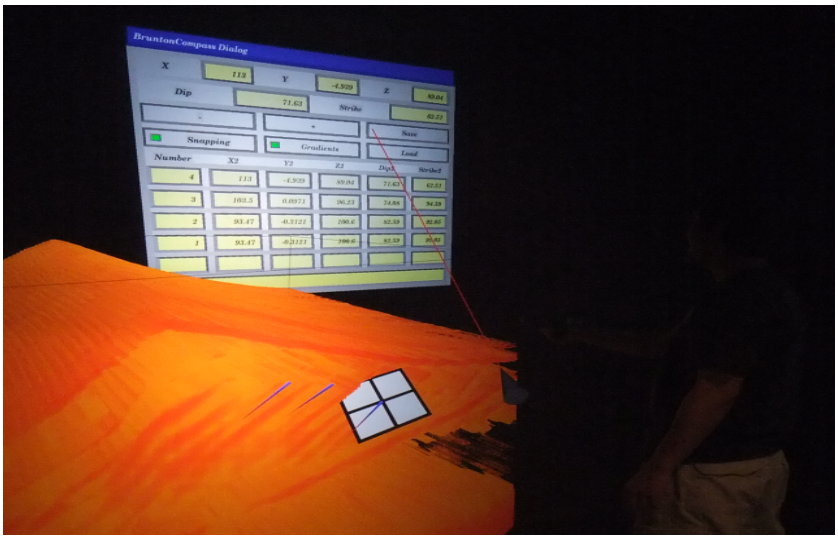


Figure 5.18: Brunton Compass Tool after a measurement has been taken with gradients enabled.

uniform vector field. Figure 5.20 is a good example of the usefulness of the Brunton Compass Tools gradients because they provide the user with an easier to view outline of the GPR data subsurfaces structure. One thing to note about the non-uniform vector field that renders just the gradients is that the strike lines are missing. The strike lines are not rendered because they ended up becoming visual clutter. The Brunton Compass in Figures 5.14–5.20, is shown towards the front and surface of the volume to make it easier to see; however, it is typically used below the surface since its purpose is to help better understand the subsurface's structure.

The 1D Transfer Function provided by Toirt-Samhlaigh is a very useful tool for manipulating the view of the GPR data. It allows the user to display certain ranges of color in the data. The user can also change the colors shown in the data using the color map at the top of the 1D Transfer Function user interface. The 1D Transfer Function can either have a piecewise-linear or Gaussian function for editing opacity. The piecewise-linear function is shown in Figure 5.21 and the Gaussian function is shown in Figure 5.22. Multiple Gaussian functions can be applied at the same time, which is shown in Figure 5.23. Also, the user can save and load these transfer functions.

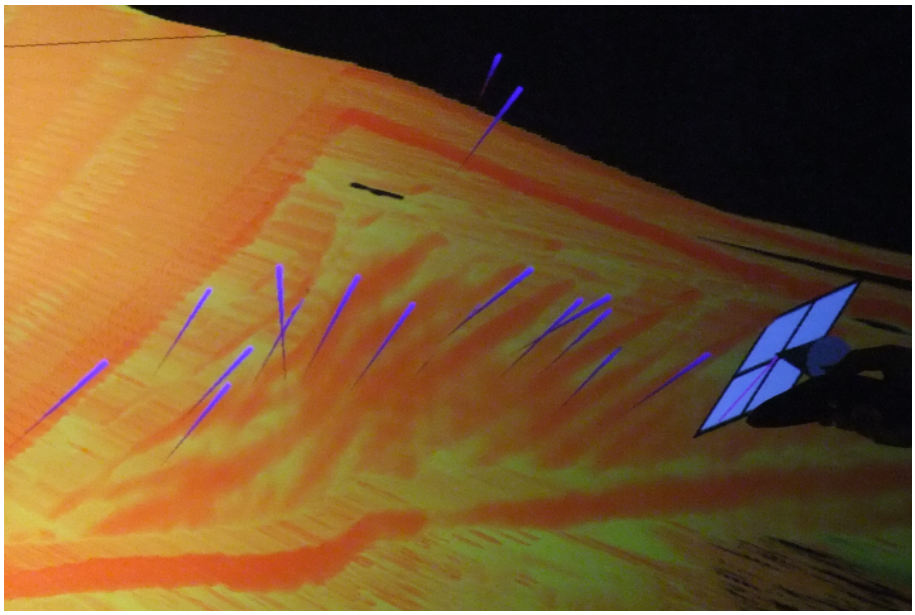


Figure 5.19: GPR data being rendered along with gradients.

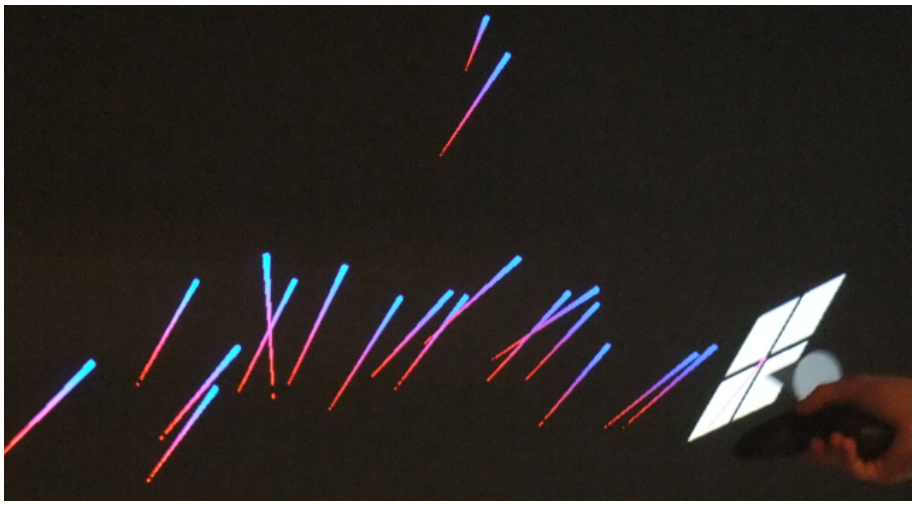


Figure 5.20: Gradients being rendered only.

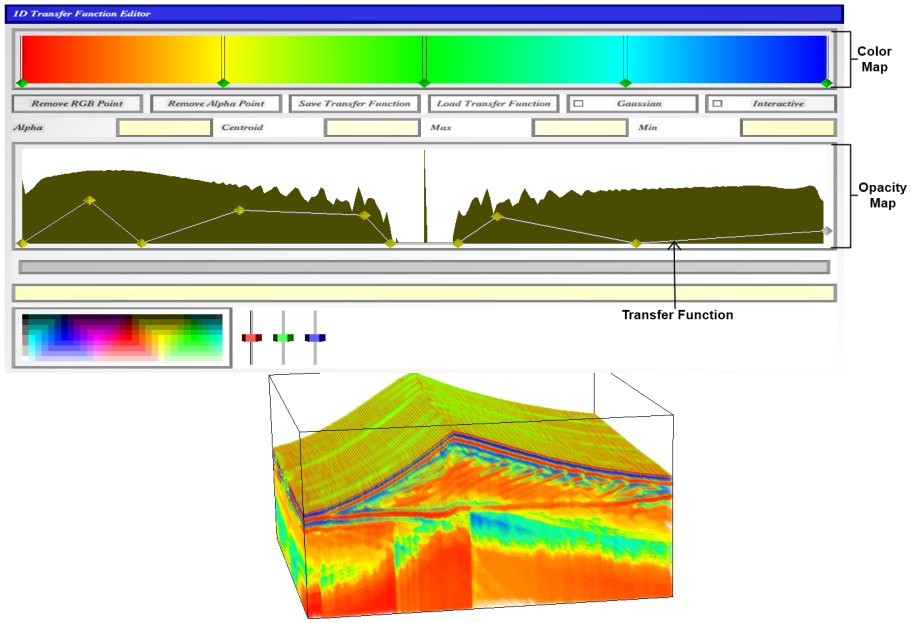


Figure 5.21: 1D Transfer Function with a piecewise-linear function applied.

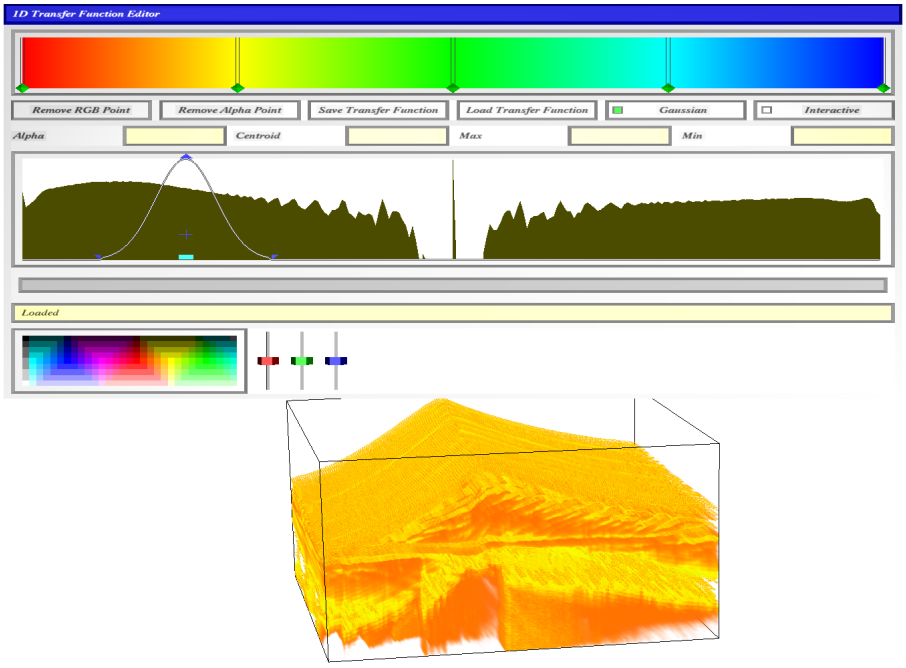


Figure 5.22: 1D Transfer Function with a Gaussian function applied.

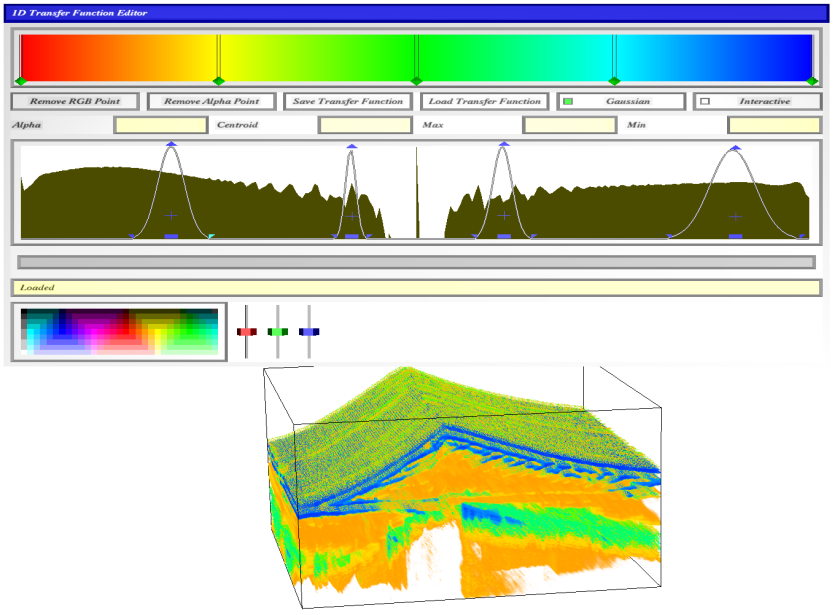


Figure 5.23: 1D Transfer Function with multiple Gaussian functions applied.

Another tool provided by Toirt-Samhlaigh is a clipping plane tool. This tool allows the users to attach a clipping plane to a 6-DOF tracked input device. The plane is created based on the orientation of the input device. Figure 5.24 is an example of this tool being used.

A useful feature provided by Toirt-Samhlaigh is the lighting feature. It allows the user to apply a directional light to the data. The user can change the color of the ambient, diffuse, and specular lighting, alter the direction of the light, and save and load lighting settings. Figure 5.25 shows an example of lighting being applied to GPR data, as well as the user interface for it. The blue dot in the top part of the lighting user interface represents the light's position.

Toirt-Samhlaigh also provides a slicing tool that allows the user to view axis-aligned slices of the GPR data. This tool was enhanced to also allow the user to treat these slices as clipping planes. These slices can be viewed on any combination of the axes at a time. Figure 5.26 shows the slicing tool being used in a GPR data set to view slices, while Figure 5.27 shows the slicing tool being used as axis-aligned clipping planes.

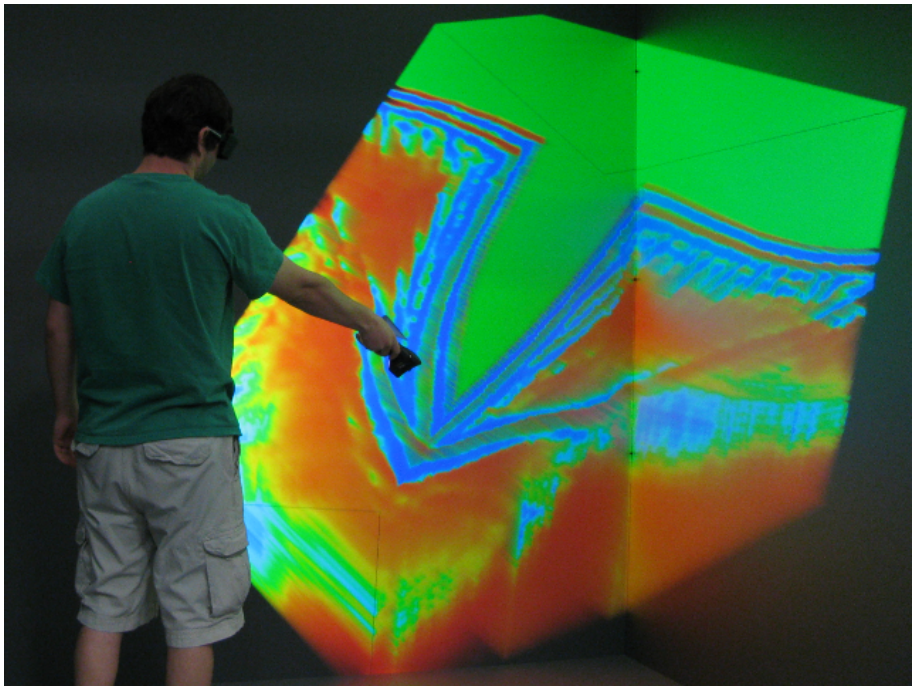


Figure 5.24: Clipping plane tool being used on GPR data.

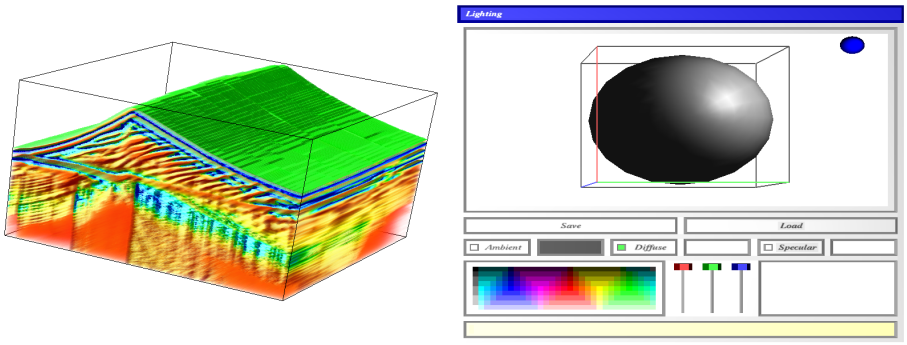


Figure 5.25: Lighting being applied to GPR data with the user interface visible.

Along with the GPR data set shown in the previous images, several other GPR data sets can be loaded as well. These GPR data sets are shown in Figures 5.28–5.30. The GPR data sets shown in Figures 5.29 and 5.30 are actually a single slice turned into a volume. Their slice versions can be seen in Figures 5.31 and 5.32, respectively.

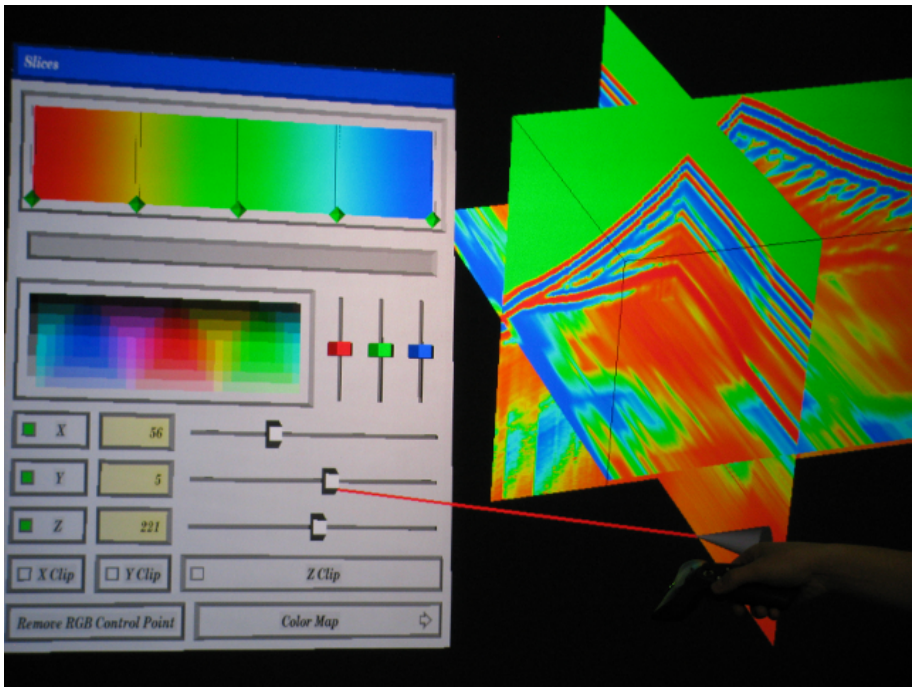


Figure 5.26: Slicing Tool being used to view axis-aligned slices of a GPR data set.

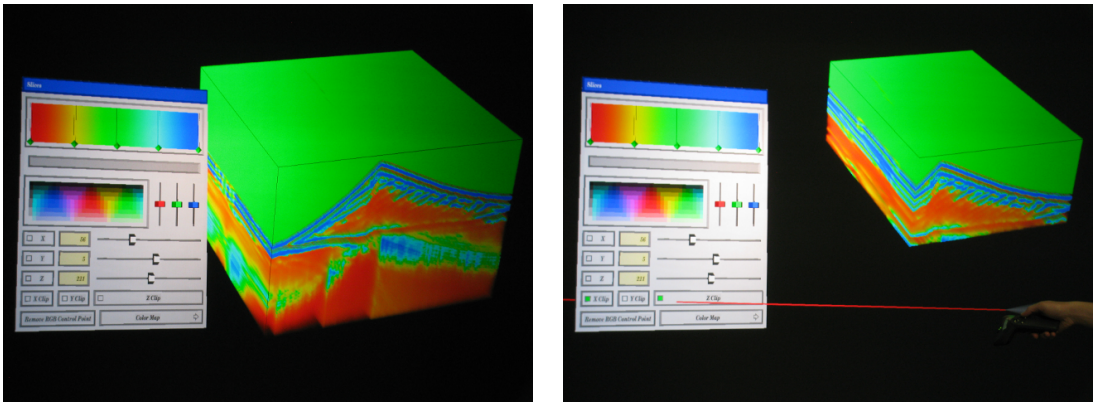


Figure 5.27: GPR data before (left) the Slicing Tool is used as axis-aligned clipping planes and after (right).

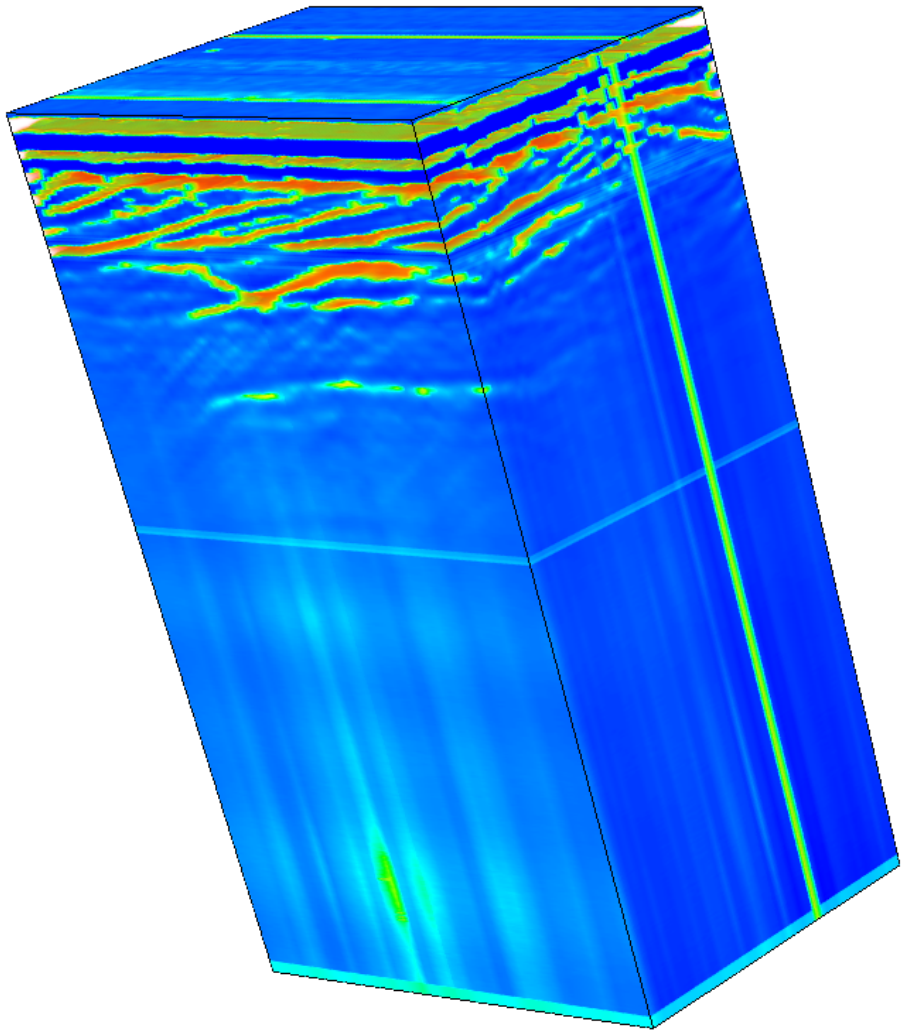


Figure 5.28: Subsection of a GPR Data set.

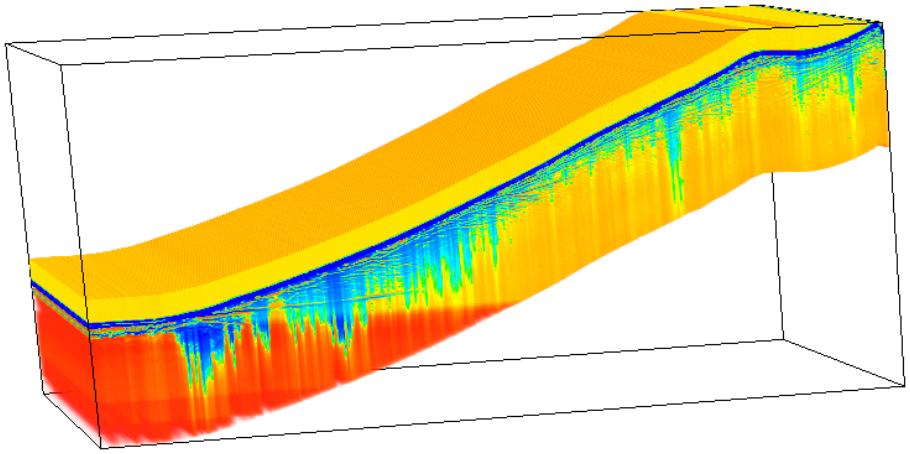


Figure 5.29: Volume created from a linear sand dune slice.

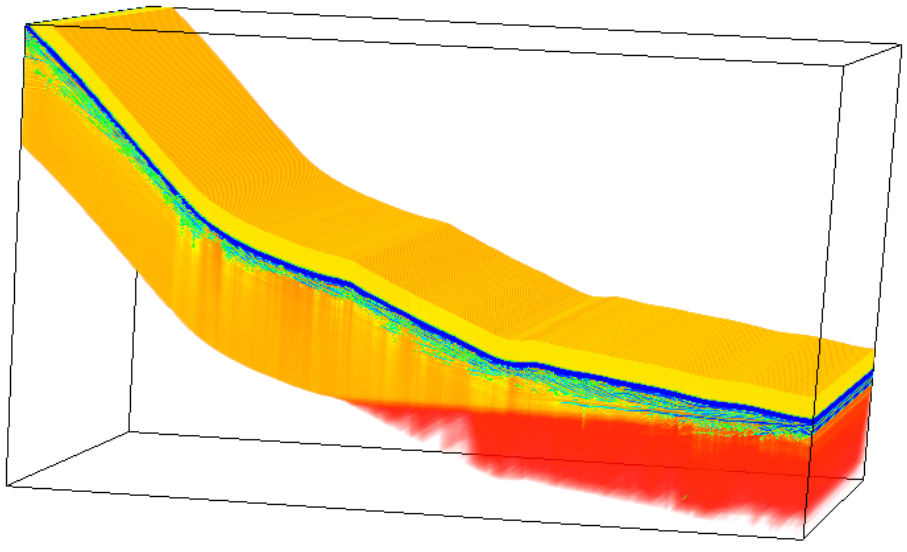


Figure 5.30: Volume created from a linear sand dune slice .

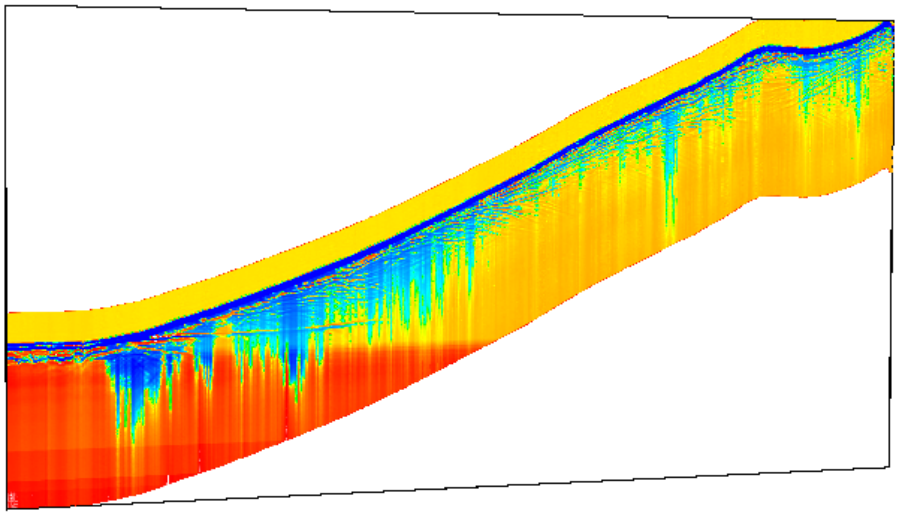


Figure 5.31: Slice version of GPR data set shown in Figure 5.29.

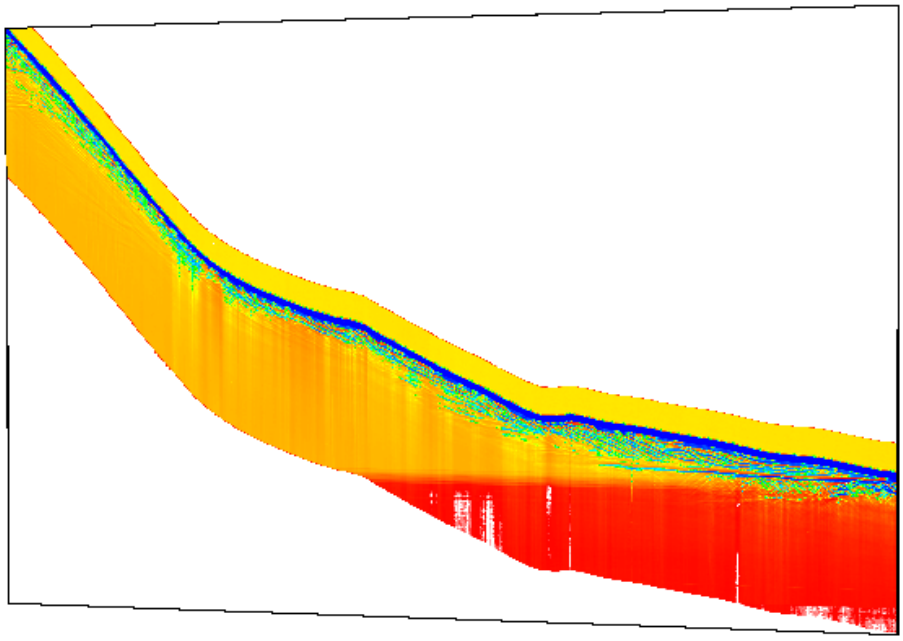


Figure 5.32: Slice version of GPR data set shown in Figure 5.30.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

There are many programs that render GPR data, [1, 22, 40]. These programs are bound to the limitations of performing visualization and analysis on a typical desktop display and input devices. This thesis presented a way of overcoming the limitations of that environment by creating a system that successfully allows for the visualization and analysis of GPR data sets in an IVE.

The IVE allows for the user to view the data from many different viewpoints and perform interactions not possible with a typical desktop display and input devices. The tracked input devices provided the user with more natural ways of interacting with the data as seen in the results section of Chapter 5. This thesis met all of the functional requirements outlined in Chapter 4, including providing the user with tools to analysis GPR data in an IVE.

I presented a system that allows for GPR data sets to be viewed in an IVE, along with giving the users the ability to perform analysis on this data. Two tools were created to perform this analysis, the Brunton Compass Tool and Distance Measurement Tool. The Distance Measurement Tool allows users to take various distance measurements, while the Brunton Compass Tool allows users to take dip and strike angle measurements. The surface visualization allows the user to get a better grasp on what data they are currently working with. The topographic correction performed on the data assists the user with understanding the shape of the dune. Additionally

the system provides many ways for the user to view and interact with the data, such as changing its orientation and position, apply lighting, and transfer functions.

The system created in this thesis is not only limited to GPR data however. The enhancements made to the original volume rendering system, Toirt-Samhlaigh, can be applied to other data types as well. The Brunton Compass and Distance Measurement tools, specifically, can be used with any of the data types that Toirt-Samhlaigh supports. Also, the saving and loading functionality added to many of Toirt-Samhlaigh components increases the user friendliness of the system for future users. This thesis ultimately provides the ground work for future work to be performed to increase the analysis capabilities of GPR data using this system.

6.2 Future Work

One of the major things that needs to be done to this project is to enhance the user friendliness of the project. Using the Vrui menu system a file select menu should be created and used to allow the user to select what data file to load. It should also be used to allow the user to select from any of the saved files for the lighting, transfer functions, color maps, measurements, and Brunton Compass files. This could be done by creating a GUI that has a drop down box with all of the files located in the current directory that have the correct file extension. Some sort of virtual keyboard should be developed so that the user can input the name of the file when going to save any of the above mention options. If Vrui is still being used, a virtual keyboard could be implemented by taking advantage of its menuing capabilities, such as buttons.

Currently in order to change the scale of the volume, the user must modify the startup script and restart the program. Thus, the ability to change the scale of the volume on any axis while the program is running should be implemented. Currently with the way Toirt-Samhlaigh creates the volume this could only be accomplished by reloading the entire data set, but with some hard effort a better solution could be found. One way to do this would be to recreate the volume that is created when the program is loaded; however, the creation time for this volume is rather long, which

would make it seem as if the program froze. Along with the ability to change the scale would be the ability to change the data set at any time during the operation of the program. This would add a much needed convenience to the program, so that the user does not have to restart it every time he/she wishes to change the data set. This can be done similar to the scaling, but a way to select which data set to load needs to be decided, such as a file selection menu.

Several tools could also be implemented to help aid in the analysis of the data. The first would be a tool that lets the user create a set of points defining a shape and then only rendering the data inside this shape. This would give users more refined views of areas of interest. A starting point for creating this tool would be to create two clipping planes that face each other and are only a small distance a part. The next tool would be one that allows the user to peel off layers of the data. The user would just need to select a certain location and the program would determine what layer it is a part of and allow the user to manipulate at will. This is different than allowing the user to see axis-aligned slices of the data since these layers would follow the contours of the data. In order for this tool to work, some form of automatic or semi-automatic segmentation of the data into layers needs to be performed. Another tool that could possibly be useful would be one that does some analysis of the data and generates iso-surfaces to help visualize the structure of the subsurfaces and it would also allow the user to modify these auto-generated iso-surfaces. A basis for this would be to take the segmented data and determine if they are within a certain thresholded distance of each other. Then combine them into one piece and create an iso-surface that represents the shape of the data, most likely using the Marching Cubes algorithm.

Along with these things the project would also benefit from replacing the volume renderer, Toirt-Samhlaigh, with one that takes better advantage of the ever increasing power of graphics hardware and implements newer DVR algorithms that prove to be faster than its approach to volume rendering. The ability to load more than just SEG-Y data file formats should be implemented, along with different types of data.

A file loader would need to be written in order to load other file formats or a file converter that converts other file formats into SEG Y is needed. Finally, the ability to have more than one volume loaded would be a useful feature, which could be done by having two instances of the volume class instead of one.

Bibliography

[1] AEGIS Instruments. Easy 3D - GPR Visualization Software. `http://www.aegis-instruments.com/products/brochures/easy-3d-gpr.html` (Accessed April 21, 2010).

[2] AP Annan. Practical processing of GPR data. In *Proceedings of the Second Government Workshop on Ground Penetrating Radar*, pages 26–28, 1993.

[3] J. Arlow and I. Neustadt. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition)*. Addison-Wesley, 2nd edition, 2005.

[4] D. Bartz and M. Meißner. Voxels versus polygons: A comparative approach for volume graphics. In *Proc. of Volume Graphics*, pages 33–48. Swansea, 1999.

[5] Christoph W. Borst. Cacs vr lab. `http://www.cacs.louisiana.edu/labs/vrlab/` (Accessed April 21, 2010).

[6] CS Bristow, SD Bailey, and N. Lancaster. The sedimentary structure of linear sand dunes. *Nature*, 406(6791):56–59, 2000.

[7] CS Bristow, GAT Duller, and N. Lancaster. Age and dynamics of linear dunes in the Namib Desert. *Geology*, 35(6):555–558, 2007.

[8] C.S. Bristow and H.M. Jol. An introduction to ground penetrating radar (GPR) in sediments. *Geological Society London Special Publications*, 211(1):1–7, 2003.

[9] Brunton Inc. Brunton geo pocket transit. `http://www.brunton.com/product.php?id=190` (Accessed April 27, 2010).

[10] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 symposium on Volume visualization*, pages 91–98. ACM, 1994.

[11] S.W. Chan. GPU-based volume rendering of medical images. *Honors Year Project Report, Department of Computer Science, School of Computing, National University of Singapore*, 2006.

[12] Carolina Cruz-Neira. The VR Juggler Suite. `http://www.vrjuggler.org/index.php` (Accessed April 21, 2010).

[13] Timothy J. Cullip and Ulrich Neumann. Accelerating volume reconstruction with 3d texture hardware. Technical report, Chapel Hill, NC, USA, 1994.

[14] Frank Dachille, Kevin Kreeger, Baoquan Chen, Ingmar Bitter, and Arie Kaufman. High-quality volume rendering using texture mapping hardware. In *HWWS '98: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 69–76, New York, NY, USA, 1998. ACM.

[15] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988.

[16] K. Eide. GPU-based transparent polygonal geometry in volume rendering. Technical report, 2005.

[17] T.T. Elvins. A survey of algorithms for volume visualization. *ACM Siggraph Computer Graphics*, 26(3):194–201, 1992.

[18] K. Engel. *Real-time volume graphics*. AK Peters Ltd, Natick, Massachusetts, 2006.

[19] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16. ACM, 2001.

[20] R. Fernando and M.J. Kilgard. *The Cg Tutorial: The definitive guide to programmable real-time graphics*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.

[21] FreeImage. The freeimage project. `http://freeimage.sourceforge.net/` (Accessed April 24, 2010).

[22] Dean Goodman. GPR-SLICE Software. `http://www.gpr-survey.com/` (Accessed April 21, 2010).

[23] S. Griffin and T. Pippett. Ground penetrating radar. *Geophysical and Remote Sensing Methods for Regolith Exploration, CRCLEME Open File report*, 144:80–89, 2002.

[24] Roger V. Hoang. Hydra. `http://www.cse.unr.edu/hpcvis/hydra/` (Accessed May 4, 2010).

[25] H.M. Jol and C.S. Bristow. GPR in sediments: advice on data collection, basic processing and interpretation, a good practice guide. *Geological Society London Special Publications*, 211(1):9–27, 2003.

[26] A. Kaufman and K. Mueller. Overview of volume rendering. In *The Visualization Handbook*. Elsevier, 2005.

[27] H.R. Ke and R.C. Chang. Ray-cast volume rendering accelerated by incremental trilinear interpolation and cell templates. *The Visual Computer*, 11(6):297–308, 1995.

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

[29] G. Kindlmann and J.W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 79–86, 1998.

[30] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of the conference on Visualization'01*, pages 255–262. IEEE Computer Society, 2001.

[31] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[32] Leif Kobbelt. RWTH Graphics: 2004. `http://www.graphics.rwth-aachen.de/index.php?id=28` (Accessed April 21, 2010).

[33] Oliver Kreylos. Vrui VR Toolkit. `http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/index.html` (Accessed April 21, 2010).

[34] P. Lacroute and M. Levoy. 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*, pages 451–458. ACM, 1994.

[35] N. Lancaster. *Geomorphology of desert dunes*. Burns & Oates, Routledge, London, 1995.

[36] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *ACM SIGGRAPH Computer Graphics*, 25(4):285–288, 1991.

[37] M. Levoy. A taxonomy of volume visualization algorithms and architectures. In *Introduction to Volume Visualization*, pages 6–12. ACM SIGGRAPH, 1991.

[38] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, 1990.

[39] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM, 1987.

[40] Malå GeoScience. Windows based acquisition and visualization software. `http://www.idswater.com/water/us/mala_geoscience/data_acquisition_software/85_0/g_supplier_5.html` (Accessed April 21, 2010).

[41] N. Max, P. Hanrahan, and R. Crawfis. Area and volume coherence for efficient visualization of 3D scalar functions. *ACM SIGGRAPH Computer Graphics*, 24(5):27–33, 1990.

[42] Alastair F. McClymont, Alan G. Green, Rita Streich, Heinrich Horstmeyer, Jens Tronicke, David C. Nobes, Jarg Pettinga, Jocelyn Campbell, and Robert Langridge. Visualization of active faults using geometric attributes of 3d gpr data: An example from the alpine fault zone, new zealand. *Geophysics*, 73(2):B11–B23, 2008.

[43] M. Meißner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 81–90, 2000.

[44] K. Mueller, T. Möller, and R. Crawfis. Splatting without the blur. In *Proceedings of the conference on Visualization'99: celebrating ten years*, pages 363–370. IEEE Computer Society Press, 1999.

[45] K. Mueller and R. Yagel. Fast perspective volume rendering with splatting by utilizing a ray-driven approach. In *Proceedings of the 7th conference on Visualization'96*, pages 65–72. IEEE Computer Society Press, 1996.

[46] E.M.W. Norris and A.K. Faichney. SEG Y rev 1 Data Exchange format1. *Technical Standards Commitee SEG (Society of Exploration Geophysicists)*, 2002.

[47] L. Nuzzo, G. Leucci, S. Negri, MT Carrozzo, and T. Quarta. Application of 3D visualization techniques in the analysis of GPR data for archaeology. *Annals of Geophysics*, 45(2), 2009.

[48] P. O'Leary, W. Sherman, A. Murray, C. Riesenfeld, and V. Peng. Enabling Scientific Workflows Using Immersive Microbiology. DVD created for and used in IEEE Visualization 2008 Conference: Workshop on Scientific Workflow with Immersive Interfaces for Visualization, Columbus, October 2008.

[49] Patrick O'Leary. Toirt-Samhlaigh. `http://code.google.com/p/toirt-samhlaigh/` (Accessed April 27, 2010).

[50] X. Papademetris, M Jackowski, A. Joshi, D. Scheinost, C. Lacadie, M. DiStasio, and L. Staib. Bioimage suite user's manual. `http://www.bioimagesuite.org/doc/` (Accessed April 20, 2010).

[51] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.P. Sloan. Interactive ray tracing for isosurface rendering. In *Proceedings of the conference on Visualization'98*, pages 233–238. IEEE Computer Society Press, 1998.

[52] Steven Parker, Michael Parker, Yarden Livnat, Peter-Pike Sloan, Charles Hansen, and Peter Shirley. Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):238–250, 1999.

[53] A.F. Prabhat, M. Katzourin, K. Wharton, and M. Slater. A Comparative Study of Desktop, Fishtank, and Cave Systems for the Exploration of Volume Rendered Confocal Data Sets. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):551–563, 2008.

[54] S. Roettger, M. Bauer, and M. Stamminger. Spatialized transfer functions. In *Proc. of IEEE/Eurographics Symposium on Visualization (EuroVis)*, pages 271–278. Eurographics Association.

[55] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser. Smart hardware-accelerated volume rendering. In *Proceedings of the symposium on Data visualisation 2003*, pages 231–238. Eurographics Association, 2003.

[56] D. Ruijters and A. Vilanova. Optimizing GPU volume rendering. *Journal of WSCG*, 14(1-3):9–16, 2006.

[57] C.R. Salama and A. Kolb. A vertex program for efficient box-plane intersection. In *Vision, Modeling, and Visualization 2005: Proceedings, November 16-18, 2005, Erlangen, Germany*, pages 115–122. IOS Press, 2005.

[58] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[59] JP Schulze, M. Kraus, U. Lang, and T. Ertl. Integrating pre-integration into the shear-warp algorithm. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 109–118, 2003.

[60] J.P. Schulze and U. Lang. The parallelization of the perspective shear-warp volume rendering algorithm. In *Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 61–69. Eurographics Association, 2002.

[61] William R. Sherman. FreeVR: Virtual Reality Integration Library. `http://www.freevr.org/` (Accessed April 21, 2010).

[62] William R. Sherman and Alan B. Craig. *Understanding Virtual Reality*. Morgan Kaufmann Publishers, San Francisco, CA USA, 2003.

[63] T. Sigurdsson and T. Overgaard. Application of GPR for 3-D visualization of geological and structural variation in a limestone formation. *Journal of Applied Geophysics*, 40(1-3):29–36, 1998.

[64] I. Sommerville. *Software Engineering*. Addison-Wesley, 8th edition, 2006.

[65] Subsurface Detection. Subsurface Detection. If it's in the ground, we'll find it. `http://www.subsurface.com.au/GPR.html` (Accessed April 20, 2010).

[66] J. Sweeney and K. Mueller. Shear-warp deluxe: The shear-warp algorithm revisited. In *Proceedings of the symposium on Data Visualisation 2002*, pages 95–104. Eurographics Association, 2002.

[67] L. Westover. Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, pages 9–16. ACM, 1989.

[68] Lee Westover. Footprint evaluation for volume rendering. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, volume 24, pages 367–376, New York, NY, USA, 1990. ACM.

[69] Wikipedia. Strike and dip. `http://en.wikipedia.org/wiki/Strike_and_dip` (Accessed April 21, 2010).

[70] William Vann. Edupic math images main. `http://www.edupic.net/math_pics.htm` (Accessed April 21, 2010).

[71] Orion Wilson, Allen VanGelder, and Jane Wilhelms. Direct volume rendering via 3D textures. Technical report, University of California at Santa Cruz, Santa Cruz, CA, USA, 1994.

[72] D. Wormell, E. Foxlin, and P. Katzman. Improved 3D Interactive Devices for Passive and Active Stereo Virtual Environments. In *Proceedings of the 13th Eurographics Workshop on Virtual Environments*, Weimar, Germany, 2007. The Eurographics Association.

[73] Y. Wu, V. Bhatia, H. Lauer, and L. Seiler. Shear-image order ray casting volume rendering. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 152–162. ACM, 2003.

[74] R. Yagel. Towards real time volume rendering. In *Proceedings of GRAPHICON*, volume 1, pages 230–241, Saint-Petersburg, Russia, 1996.

[75] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA volume splatting. In *Proceedings of the conference on Visualization'01*, pages 29–36. IEEE Computer Society, 2001.