REAL-TIME VISUALIZATIONS OF OCEAN DATA COLLECTED BY THE NORUS GLIDER

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Daniel Medina

June 20010

COMMITTEE MEMBERSHIP

TITLE:	Real-Time Visualizations of Ocean Data Collected By The NORUS Glider
AUTHOR:	Daniel Medina
DATE SUBMITTED:	June 20010
COMMITTEE CHAIR:	Zoë Wood, Ph.D.
COMMITTEE MEMBER:	Chris Clark, Ph.D.
COMMITTEE MEMBER:	Aaron Keen, Ph.D.

Abstract

Real-Time Visualizations of Ocean Data Collected By The NORUS Glider

Daniel Medina

Scientific visualization computer applications generate visual representations of large and complex sets of science data. These types of applications allow scientists to gain greater knowledge and insight into their data. For example, the visualization of environmental data is of particular interest to biologists when trying to understand how complex variables interact. Modern robotics and sensors have expanded the ability to collect environmental data, thus, the size and variety of these data-sets have likewise grown. Oftentimes, the collected data are deposited into files and databases where they sit in their separate and unique formats. Without easy to use visualization tools, it is difficult to understand and interpret the information within these data-sets.

NORUS, the North America-Norway educational program, has a scientific focus on how climate-induced changes impact the living resources and ecosystems in the Arctic. In order to obtain the necessary science data, the NORUS program utilizes the Slocum Glider, a form of Autonomous Underwater Vehicle (AUV). This thesis aims to create a compelling, efficient, and easy to use interactive system for visualizing large sets of science data collected by the Slocum Glider. This goal is obtained through the implementation of various methods taken from scientific visualization, real time rendering, and scattered data interpolation. Methods include visualizations of the surrounding terrain, the ability to map various science data to glyphs, control over color mapping, scattered data interpolation and interactive camera control.

Acknowledgements

I would like to thank each of the NORUS partner institutions:

1. Norwegian University of Science and Technology (NTNU)
   Department of Biology

2. University Centre of Svalbard (UNIS)
   Section of Arctic Biology

3. California Polytechnic State University (Cal Poly)
   Biological Sciences Department

4. Rutgers University, Institute of Marine Coastal Sciences
   Coastal Ocean Observation Lab

The science data for the Svalbard, Norway mission was collected and made available by the Rutgers University, Coastal Ocean Observation Lab (COOL).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The oceans are an important focus of scientific study and exploration. They cover over 70% of the earth's surface, are a significant source of food for a large part of the planet, and due to the impact of global warming, are increasingly central to predicting how our climate will evolve during the next century. An ocean is a complex, highly interconnected envrionment. A full description of an ocean must include not just physical characteristics, such as temperature and currents, but chemical, biological, and even geological parameters [7].

Scientific visualization computer applications generate visual representations of large and complex sets of science data. These types of applications allow scientists to gain greater knowledge and insight into their data. For example, the visualization of environmental data is of particular interest to biologists when trying to understand how complex variables interact. Modern robotics and sensors have expanded the ability to collect environmental data, thus, the size and variety of these data-sets have likewise grown. Oftentimes, the collected data are deposited into files and databases where they sit in their separate and unique formats. Without easy to use visualization tools, it is difficult to understand and

interpret the information within these data-sets.

NORUS, the North America-Norway educational program, has a scientific focus on how climate-induced changes impact the living resources and ecosystems in the Arctic. In order to obtain the necessary science data, the NORUS program utilizes the Slocum Glider, a form of Autonomous Underwater Vehicle (AUV). This thesis aims to create a compelling, efficient, and easy to use interactive system for visualizing large sets of science data collected by the Slocum Glider. This goal is obtained through the implementation of various methods taken from scientific visualization, real-time rendering, and scattered data interpolation.

This project is an implementation of a real-time scientific visualization system desgined to visualize ocean science data collected by the NORUS glider. The system was written entirely in C++, using OpenGL and GLSL for graphics and Qt for rapid development of a platform independent graphical user interface. It allows for the real-time rendering of offline ocean data containing upwards of 250,000 measurement points across more then 20 different sensors. Within the context of this paper, real-time or real-time rendering refers to the system's ability to render approximately 30 to 60 frames per second (fps).

The primary dataset available during development was from a glider deployment in a fjord of Svalbard, Norway. The glider's mission spanned from longitude 13.3042° East to 16.6875° East and latitude 78.1042° North and 78.7042° North. The 17 day mission began on June 30, 2009, during which time the glider collected 242,693 measurements for each of the on-board sensors.

The system uses a generic mesh loader to generate an accurate and compelling environment. The look of the environment is obtained through the use of multi-texturing to produce detailed terrain, a quasi-realistic ocean surface with reflec-

tion, refaction, and specular highlighting, and simple animated clouds generated by 3-D perlin noise. Also, various user configured scientific visualizations allow the user to analyze the measurements taken by the glider. They include glyphs that display the discrete data measurements and interpolated visualizations such as isosurfaces, gradient planes, and volume slices that provide interpolated values across continous surfaces. An interactive camera control allows the user to freely move through the environment and a virtual glider retraces the physical glider's mission path. Finally, an intutive graphical user interface provides the user with the highest possible level of customizability across the entire system.

Initial feedback from NORUS biologists working on the project conveyed enthusiasm for a tool that allowed them to visualize their data in a virtual environment [1]. Many of the tools included in the system were defined and refined through feed-back from biologists about what they would like to be able to see and what parameters they needed to be able to control [2].

The system presented in this thesis addresses the problem of creating an interactive underwater visualization system capable of handling very large and diverse sets of science data collected by AUVs such as the Slocum Glider. Beyond that, the system also provides a high level of configurability for each individual visualization.

Chapter 2 describes previous work related to the system, concentrating primarily on different methods of scattered data interpolation and underwater visualization systems. Chapter 4 provides an in depth look at the primary algorithms used by the system. These include radial basis functions, marching cubes, and multitexturing. Chapter 3 provides an overview of the implemented system, outlining its various components and how each one is configurable by the user. Chapter 5 provides analysis of various visualizations generated by the system and

describes feedback received from the users. Chapter 6 summarizes the results of the project and outlines future work that could be done to extend the system. Appendix A provides a brief overview of the three exteranl libraries used by the system.

Chapter 2

Related Work

2.1 Underwater Scientific Visualization Systems

This section provides an overview of some of the known underwater visualization systems, concentrating primarily on systems that generate visualizations using real-time rendering.

The goal of research developed at the Naval Post Graduate School [9] was to create an underwater virtual world that could comprehensively model all necessary functional characteristics of the real world in real-time. The virutal world was meant to provide AUV developers with the complete functionality of a submerged environment in the laboratory. The virtual world needed to recreate every aspect of the environment external to the AUV. The idea was to generate interactions between software processes and vehicle hardware that comprehensively modeled what would happen in the real world. Also, the physical behavior and sensor interactions needed to be modeled and simulated exactly. Fulfilling each of these requirements would hopefully produce a virutal world whose differences

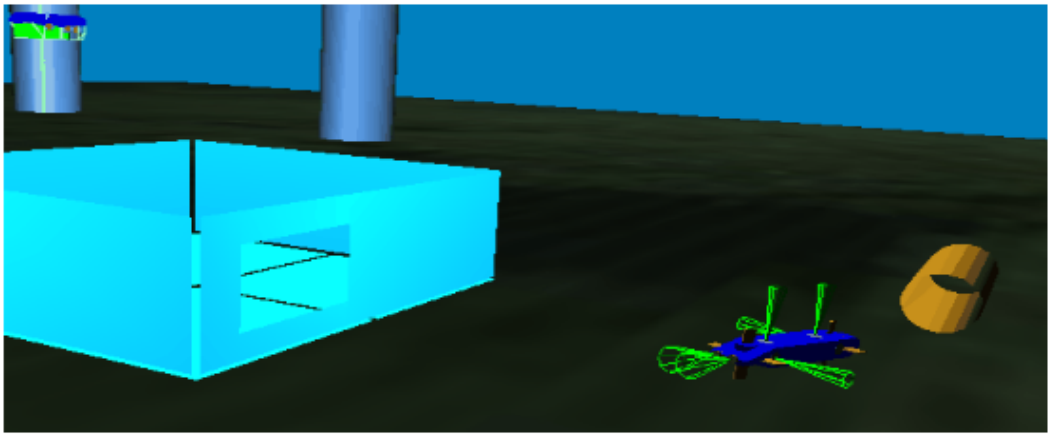

Figure 2.1: A view of the virtual world created in [9].

to the real world were transparent to the AUV's software. In addition to the virtual world, the project aimed to create a networked architecture that would enable multiple world components to operate collectively in real time, and also permit world-wide observation and collaboration.

Collaborative Ocean Visualization Environment (COVE) is a set collaborative tools used to support deep-water ocean observatories. These observatories allow hundreds of scientists from various fields to conduct experiments together, provide real-time sensor and data access though the internet, and create a vast archives of data [16]. COVE provides a common visual environment that cuts across the specifics of the multiple disciplines to provide an interactive workplace for individuals, groups, and the entire team. During its first and most extensive depolyment, COVE was sucessfully used to create the core cabling and instrumentation layout for an ocean observatory. Additional features of COVE include the generation of scientific visualizations for collected data in almost real-time and a novel environment for planning ship routes. A view of COVE is shown in figure 2.2.

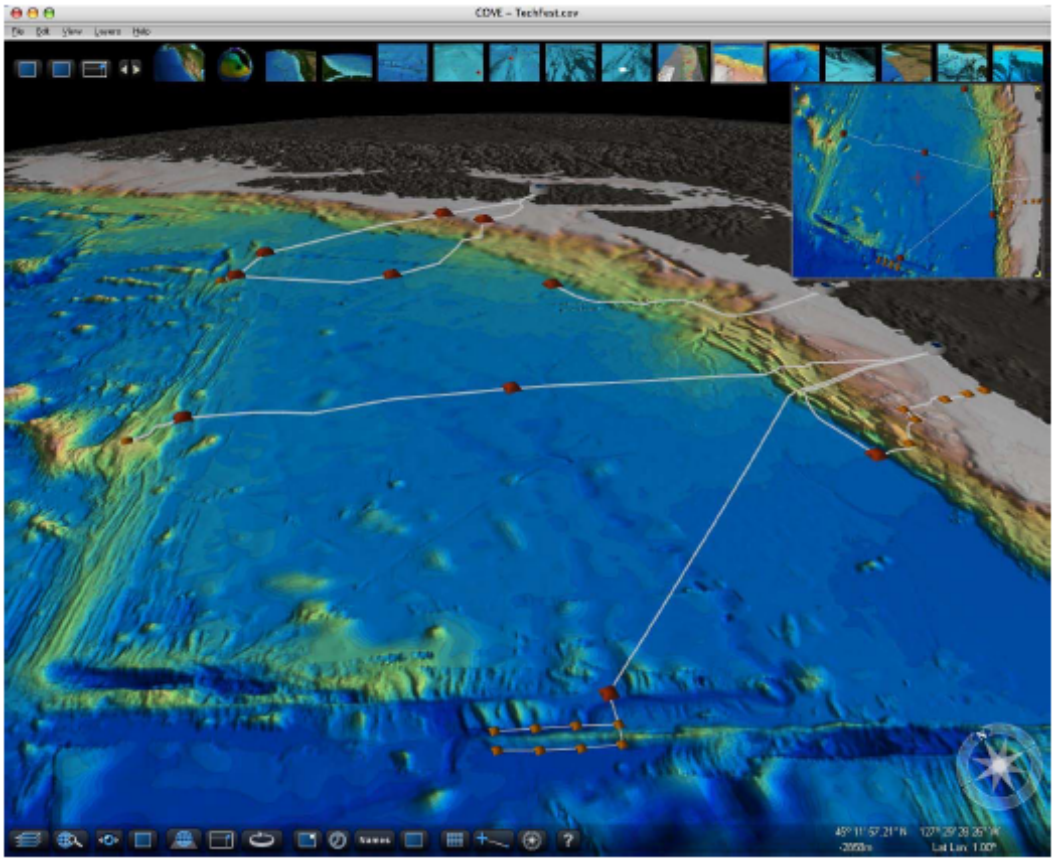

Figure 2.2: A view of COVE, the Collaborative Ocean Visualization Environment .

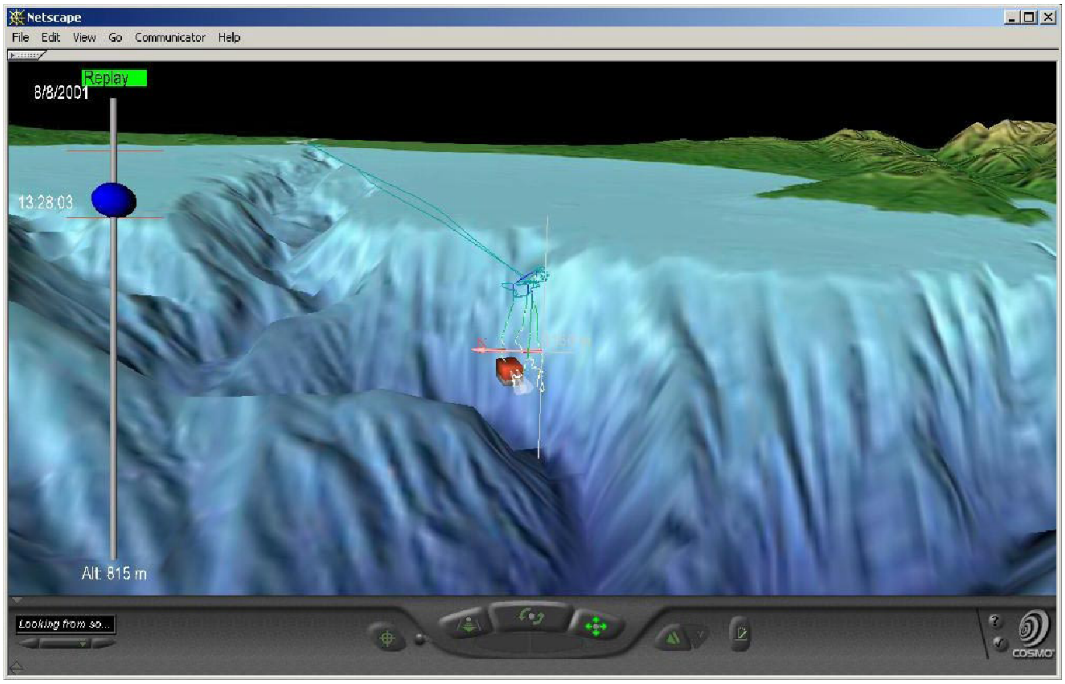

Figure 2.3: A visualization of an ROV mission generated by [26].

The research presented at the seventh International Conference on 3-D Web Technology [26] was aimed at creating 3-D oceanographic data visualizations for the web. Designed by the Monterey Bay Aquarium Research Institute (MBARI) in 1999, the goal was to synthesize various ROV-generated data sets for visualization in a common, compelling, efficient, and easy-to-use system. Integration of the visualizations into MBARI's database of dive information would make the data more understandable and useful. Figure 2.3 shows a visualization generated for an ROV mission. Research presented at the same conference two years later [27] provides an updated investigation of the original system [26] two years after its initial deployment.

Chapman, Wills, Stevens, and Brookes [10] present a system that generates post-survey visualizations of underwater pipelines using real-time rendering. The system provides the user with a 3-D underwater visualization environment that

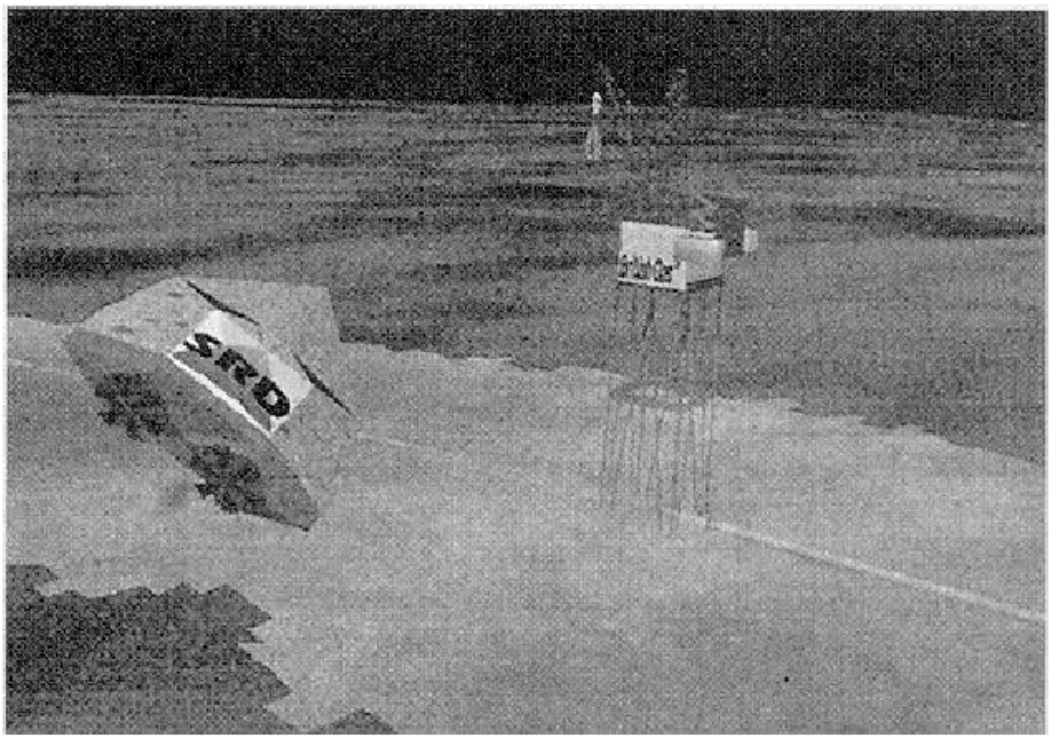

Figure 2.4: A view of the system used to generate visualizations of underwater pipelines.

allows them to pilot a virtual underwater vehicle around an accurate seabed model. The two case studies, visualizing pipeline dredging and pipe restoration visualization, were implemented using real survey data. A view of the system is presented in figure 2.4.

Research presented by the same authors describes a system that genereates real-time visualizations of the clear-up operation of a former U.S. Nuclear Submarine Base located in Holy Loch, Scotland [11]. Using the Whole Field Modeling System, the paper describes a system that is capable of generating accurate real-time visualization of a large number of varying parameters such as remotely operated vehicles, cranes, barges, grabs, magnets, and detailed seabed topography. The system has improved the field staffs spatial and temporal awareness

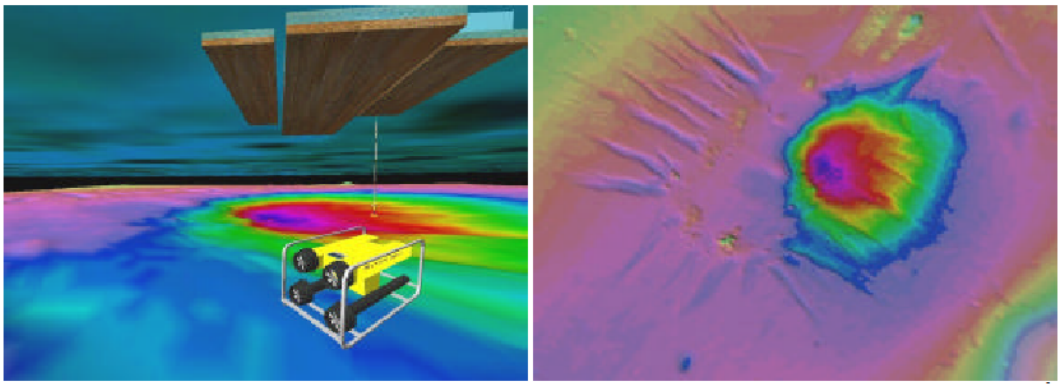

Figure 2.5: Views of the system described in [11]. The image on the left shows a view of flying alongside the ROV under a barge. The image on the right shows a bathymetric depth coloured view of the seabed.

of the underwater environment and facilitated decision-making within the complex offshore working environment. Figure 2.5 shows two views of the system generated by the system.

While these related works provide solutions for their setting, the system presented in this paper is designed specifically to generate real-time visualizations for ocean science data collected by the NORUS glider. Additionaly, the system was designed from the ground up, without the use of existing visualization libraries, to achieve a system with the maximum amount of configurability possible.

2.2 Visualization Toolkits

The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, image processing and visualization. VTK has an extensive information visualization framework, has a suite of 3D interaction widgets, supports parallel processing, and integrates with various databases on GUI toolkits such as Qt and Tk. It is also cross-platform and runs on Linux,

Windows, Mac and Unix platforms [44].

VTK was not chosen as a basis for the implementation of the interactive visualization system because of the limitations associated with it. Maintaining real-time rendering with large sets of data and high levels of detail is difficult with the additional overhead associated with the toolkit.

2.3 Scattered Data Interpolation

Since the system requires the visualization of continous data, but only takes as input discrete data points gathered by the NORUS glider, scattered data interpolation is essential for constructing continous functions from the discrete data samples. The problem of generating a continuous interpolation function for scattered data is encountered frequently in a wide variety of scientific disciplines and has been researched extensively. Despite considerable efforts, no generic method has yet been found that solves the problem of scattered data interpolation across all domains, and it is quite possible that no such method exists. This section provides an overview of some popular and effective methods of scattered data interpolation. Due to the extent of research done within the field, this is by no means an exhaustive review. Several survey papers exist that provide a more in depth look at a wider variety of methods [4] [14] [22] [37].

The Shepard's Method is quite possibly the most well known technique used to generate interpolants for scattered data. It was the first of many methods to utilize inverse distance weighting, an approach where a sample point's influence on the interpolated value is inversely related to the distance from the input position

[42] [4] [14]. In its original form, it is defined as

$$s(x) = \sum_{i=1}^{N} w_i(x) f(x_i) where w_i(x) = \frac{||x - x_i||^{-p}}{\sum_{j=1}^{N} ||x_i - x_j||^{-p}}. \quad (2.1)$$

To avoid the division by zero that occurs at a sample point, s can be extended to require that $s(x_i) = y_i$.

The equation defines an interpolating function that is the weighted average of the value at each sample point. This method is global because the evaluation of the interpolant requires the evaluation of a function on all given sample points. Global techniques are expensive for large number of sample points, but it is possible to apply them to overlapping subsets, and blend the solutions into a single interpolant for the whole set. There are variants to the technique, such as the Modified Shepards Method [20], which modifies the weighting function to take into account only the points lying in a disc within a specified radius, centered at the point at which the interpolant is evaluated. The modification improves the method both in addressing shape preservation and in making it local to a neighborhood of points [15] [32] [38].

Thin-plate interpolation is another approach to solving the scattered data interpolation problem. It uses an energy function $E(f)$ that measure the smoothness of a function f. The energy function is basically a measure of the aggregate curvature of $f(x)$ over the region of interest. The thin-plate solution to an interpolation problem is the function $f(x)$ that has the smallest possible value of E [43].

Another method for constructing an interpolant is to consider it to be a piecewise union of patches (usually low degree multivariate polynomials) joined with certain continuity. Examples of this approach are those based on Spline and Bezier patches [6] [13], which are extensively used in the area of geometric design

and give a user freedom to model and change the shape of an object [38].

For this project, we found that scattered data interpolation was best achieved using radial basis functions. Radial basis functions (RBFs) are a simple and useful tool for interpolating data in almost any number of dimensions. Given a set of data points with associated values, RBFs construct a smooth and continous function which interpolates the values at each data point. For a detailed description on radial basis functions, refer to section 4.1.1.

Chapter 3

System Overview

3.1 Environment

Within the system, the environment has been broadly defined to include all graphical components that are not scientific visualizations. This includes the terrain, interactive camera, ocean surface, clouds, heads-up display, compass, glider, etc.

3.1.1 Interactive Camera

The interactive camera allows the user to freely navigate through the environment. The w, a, s, and d keys move the camera forwards, left, backwards, and right respectively. The mouse allows the user to rotate the camera and look around the environment.

3.1.2 Terrain

The terrain for the mission's extent is generated from bathymetry data retrieved from the National Geophysical Data Center (NGDC) [30]. NGDC provides global bathymetry at resolutions of 30 arc-seconds and greater. The primary issue is that regions around Svalbard do not have publicly available bathymetry data that is of high resolution, resulting in strict limitations on the available detail for the under water terrain. Figures 3.1 and 3.2 provide different views of the terrain for the Svalbard mission.

The bathymetry data is loaded into a generic mesh loader that converts the data into a renderable triangle mesh. The triangle mesh is then multitextured to produce a detailed terrain mesh.

Generic Mesh Loader

In order to provide support for various formats of bathymetry data, the mesh loader was developed to be format agnostic. It accepts the bathymetry data and computes the necessary values from the data.

Multitexturing

Section 4.3 provides a detailed description of what multitexturing is and how it works. Multitexturing is used within the application to add additional detail to the terrain surface. Also, it provides clear differentation between surfaces that lie above and below the sea level.

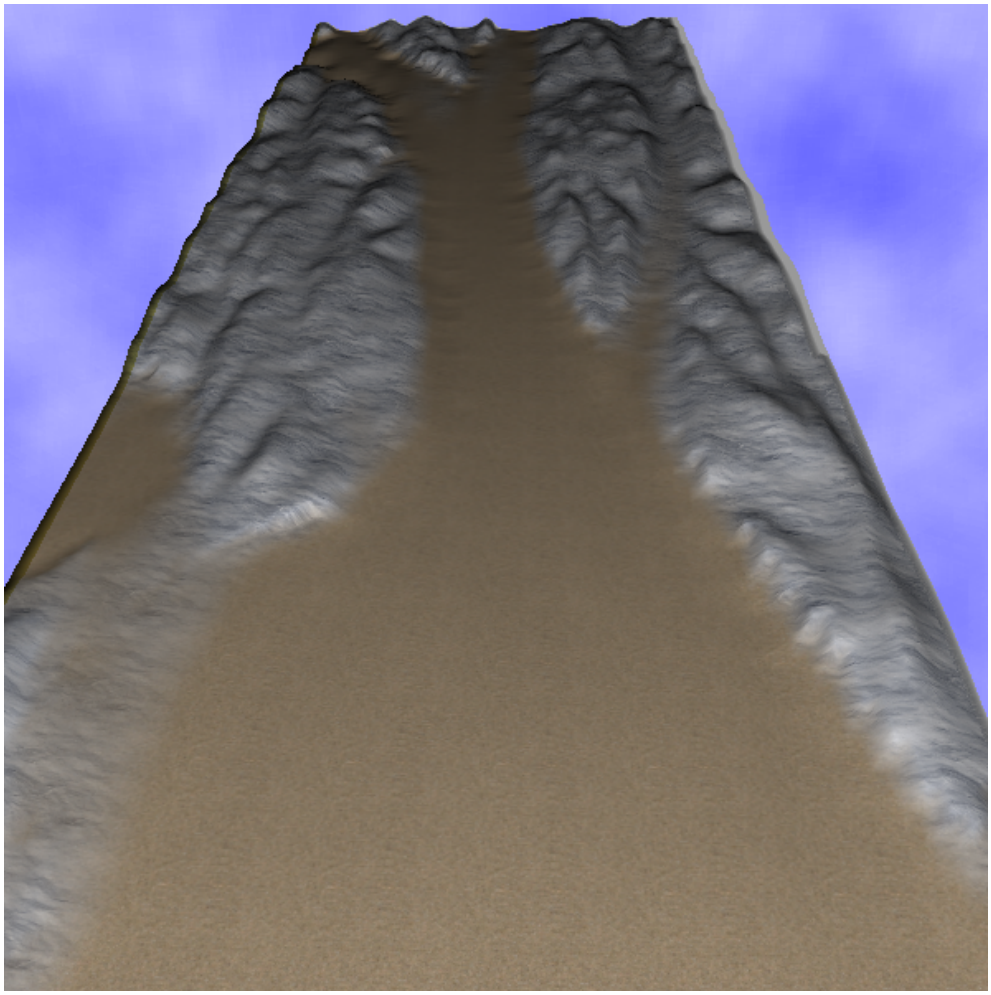

Figure 3.1: This image shows an aerial view of the Svalbard fjord.

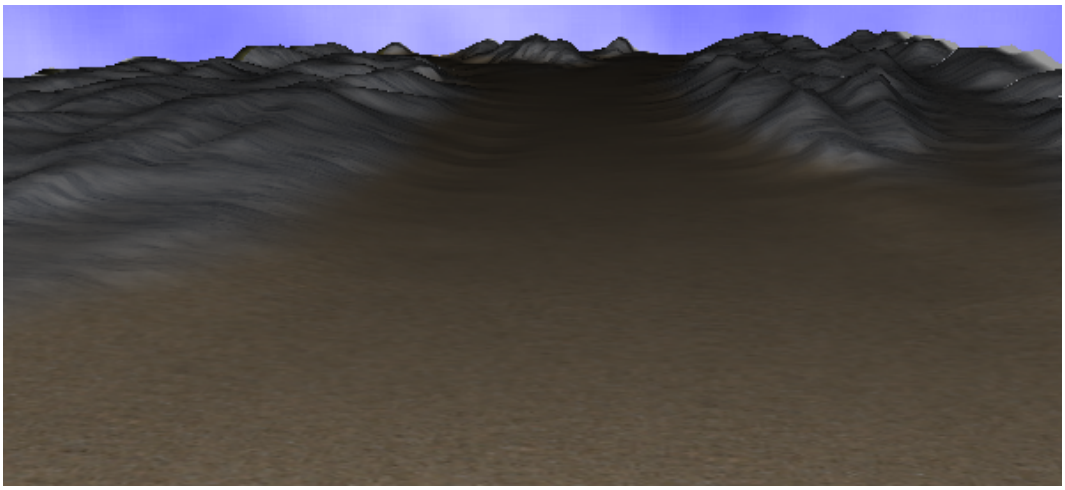

Figure 3.2: This image shows a ground view of the Svalbard fjord.

User Configuration

Within the *Configure Window* under the *Terrain* tab, The user is provided with the functionality to scale the sea floor by a factor of between one and twenty. With a scale of one, the terrain is rendered to scale, with the mountains and sea floor at their appropriate heights and depths. Larger scales allow the user to investiage specific regions carefully without having to get as close to the visualizations or glyphs. Also, a larger scale allows the user to navigate through the data with greater ease. Figure 3.3 shows two images of the same region, with the image on the right having a much larger scale.

3.1.3 Ocean Surface

A primary goal for the application was to produce an environment with an appealing look. In an attempt to further that goal, a compelling and quasi-realistic ocean surface was implemented using the techniques described in [19] and [8]. Using a normal map, dudv map, two additional rendering passes, and

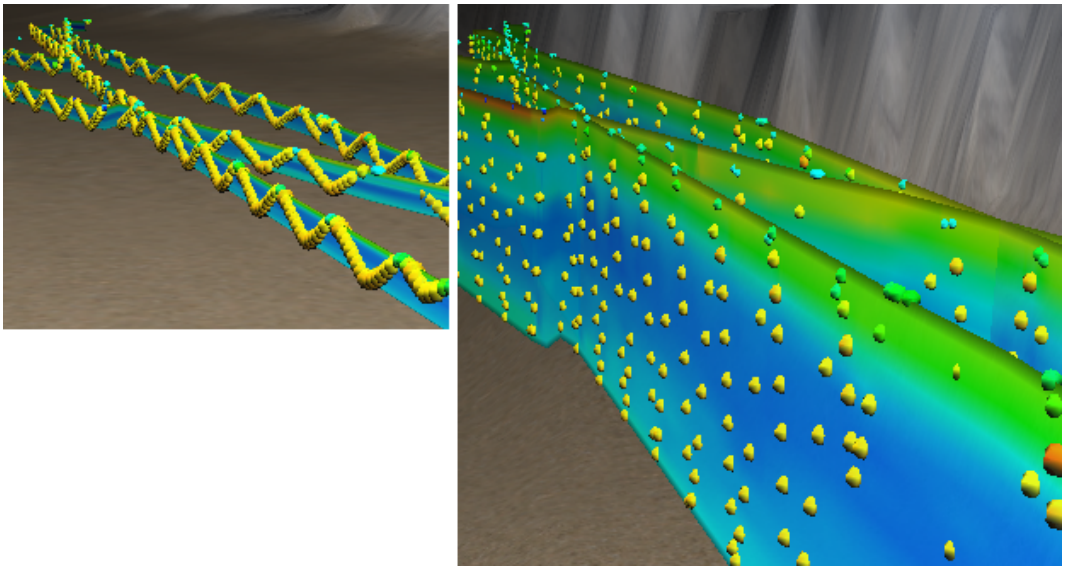

Figure 3.3: Both images show the same region. The image on the left has a scale factor of 1 while the image on the right is scaled to a factor of 11.

GLSL shaders, it is possible to render a compelling ocean surface with only a single quad.

Reflection

Reflections were implemented in an attempt to create a more realistic and appealing ocean surface. This adds an additional rendering pass where only the terrain above sea level is rendered. The terrain is reflected about the y-axis and rendered to a texture that is then mapped to the quad serving as the ocean surface. Since the majority of the reflected terrain is occluded by the terrain itself, the reflection is only visible for the terrain closest to the water's edge. The visibility of the reflection is also view dependent. Figure 3.4 shows a view of the reflection of the surrounding terrain onto the ocean surface.

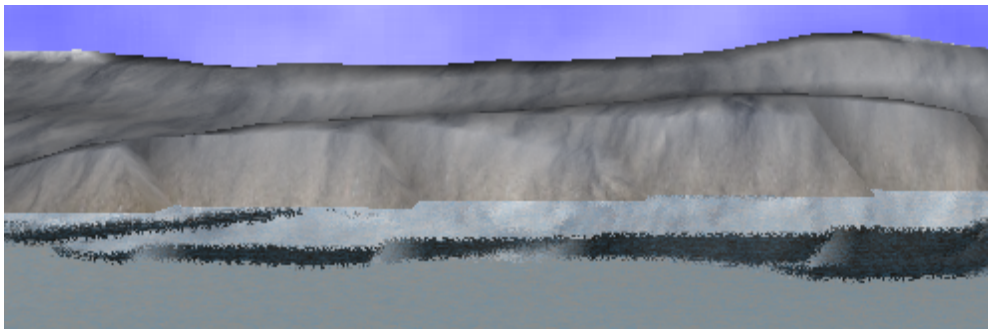

Figure 3.4: A view of the reflection of the surrounding terrain onto the ocean surface.

Refraction

Refraction refers to the turning or bending of light waves as they pass through a substance of differing density. The system uses a dudv map to approximate the refraction that occurs as light passes through the ocean surface. As seen in figure 3.4, the effect is especially noticable at sharp edges in the reflected surface. The refraction produces the discontinuites or perturbations in the reflected surface. A time based variable that modifies the refraction produces the appearance of a dynamic surface that is constantly changing.

Specular Highlights

The specular highlights on the surface of the water imitates the effect of light reflecting off the ocean surface. The specular highlights along with the refraction of the surface produces the shimmering effect that results from the random change in the angle of the reflection of light as the water moves. Figure 3.5 contains a view of the specular highlights within the environment.

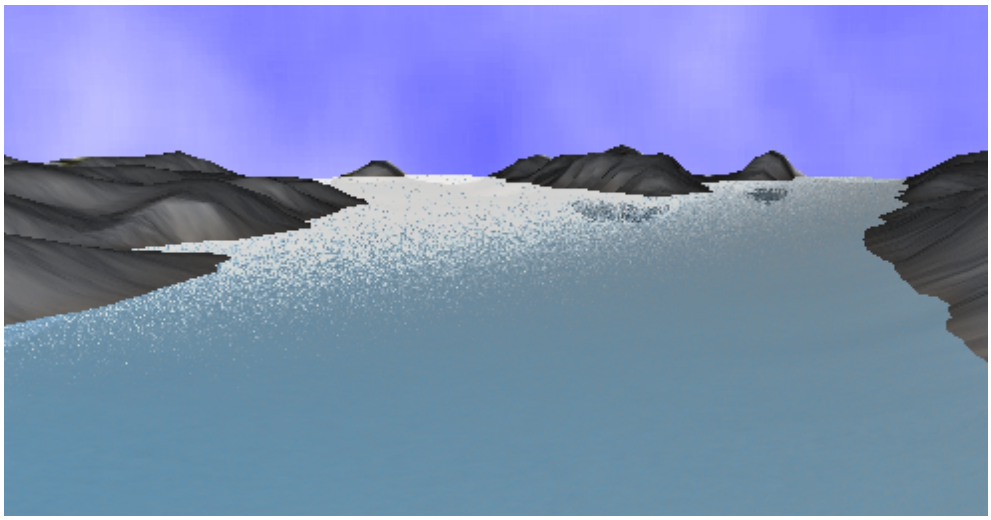

Figure 3.5: A view of the specular highlighting that occurs on the ocean surface.

3.1.4 Clouds

The clouds serve as a simple method for producing a more appealing environment for the user. The animation of the clouds moving slowly across the expanse of the sky assists an environment that is constantly changing. Figure 3.6 shows a view of the clouds within the environment. The methods for constructing and rendering the clouds are described below.

First, a 3-D Perlin noise function is generated that the application samples at various positions to create a 3-D texture. The texture is then mapped to a very large sphere that the terrain is contained within. A simple GLSL fragment shader performs the cloud animation by offsetting the texture by a factor of the time from the application's initialization.

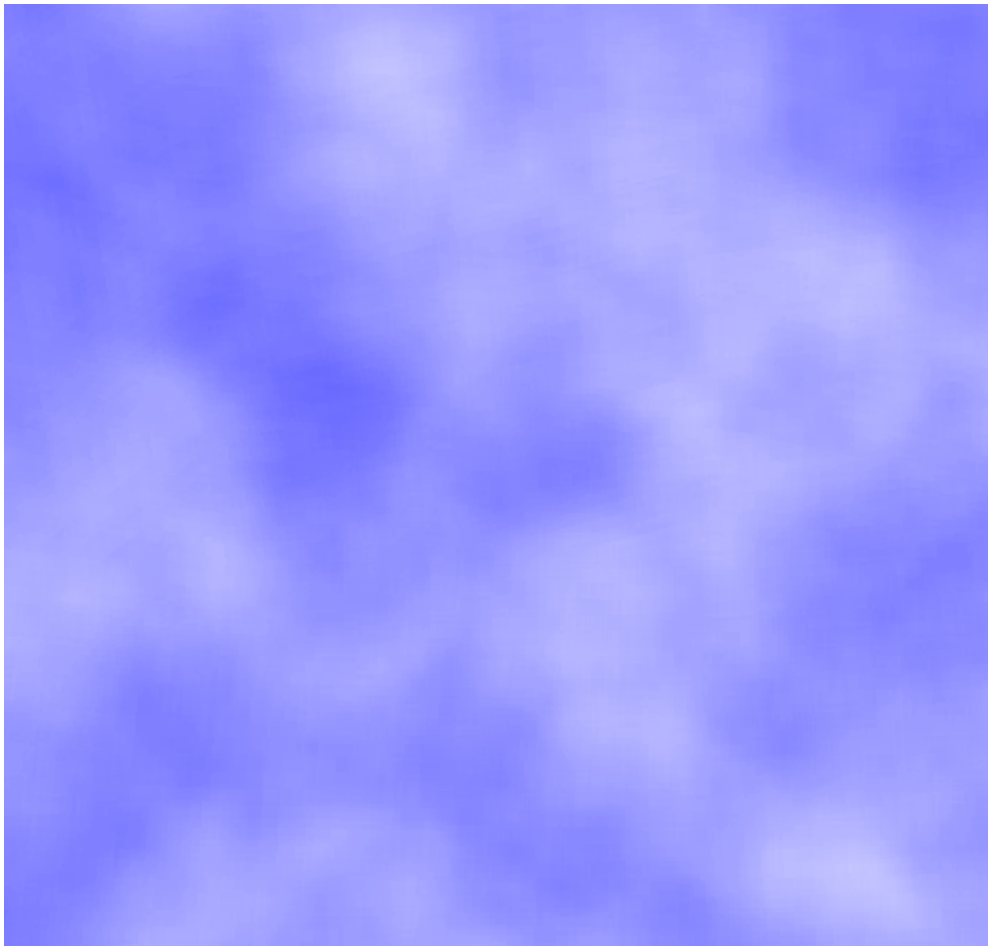

Figure 3.6: A view of the clouds generated by a 3-D Perlin noise function.

Figure 3.7: A view of the compass.

3.1.5 Heads-up Display

The heads-up display provides the user with updates on the status of visualizations being constructed in the background. Section 3.3.2 describes why it is necessary for the construction of the interpolated visualizations to occur within threads. The heads-up display serves as a means for the application to provide the user with continual updates on the construction status. Figure 3.13 provides a view of the heads-up display.

Compass

The compass provides directional orientation within the scene. Its used primarily for assisting the user in specifying spatial constraints for visualizations. Figure 3.7 provies a view of the compass.

3.1.6 Glider

The glider within the application provides a simple 3-D representation of the physical glider as it travels along the mission path. Its purpose is to give the user an idea of how the glider moves while collecting science data. Figure 3.8 provides a view that shows what the glider looks like.

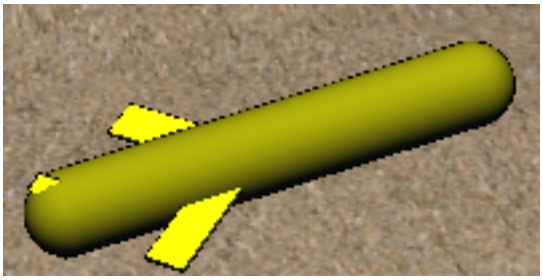

Figure 3.8: A view of the glider.

Follow Glider

The follow glider feature, which can be found within the *Controls* menu, attaches the camera to the glider. The features provides a third person view as the glider moves along its mission path.

Glider Speed

The speed of the glider can be adjusted within the *Controls* menu. Adjusting the glider's speed effects the amount of time, or frames, it takes for the glider to move between the control points of the approximated mission path discussed in Section 3.1.7.

3.1.7 Mission Path

An approximation of the glider's mission path is constructed for use in the animation of the computer generated glider and the construction of the mission defined gradient planes. The approximated mission path is constructed by locating the indicies within the mission data that correspond to the glider's local minimum and maximum depths. Since the glider's motion is consistent, an adequate approximation of its path can be constructed by connecting sequential

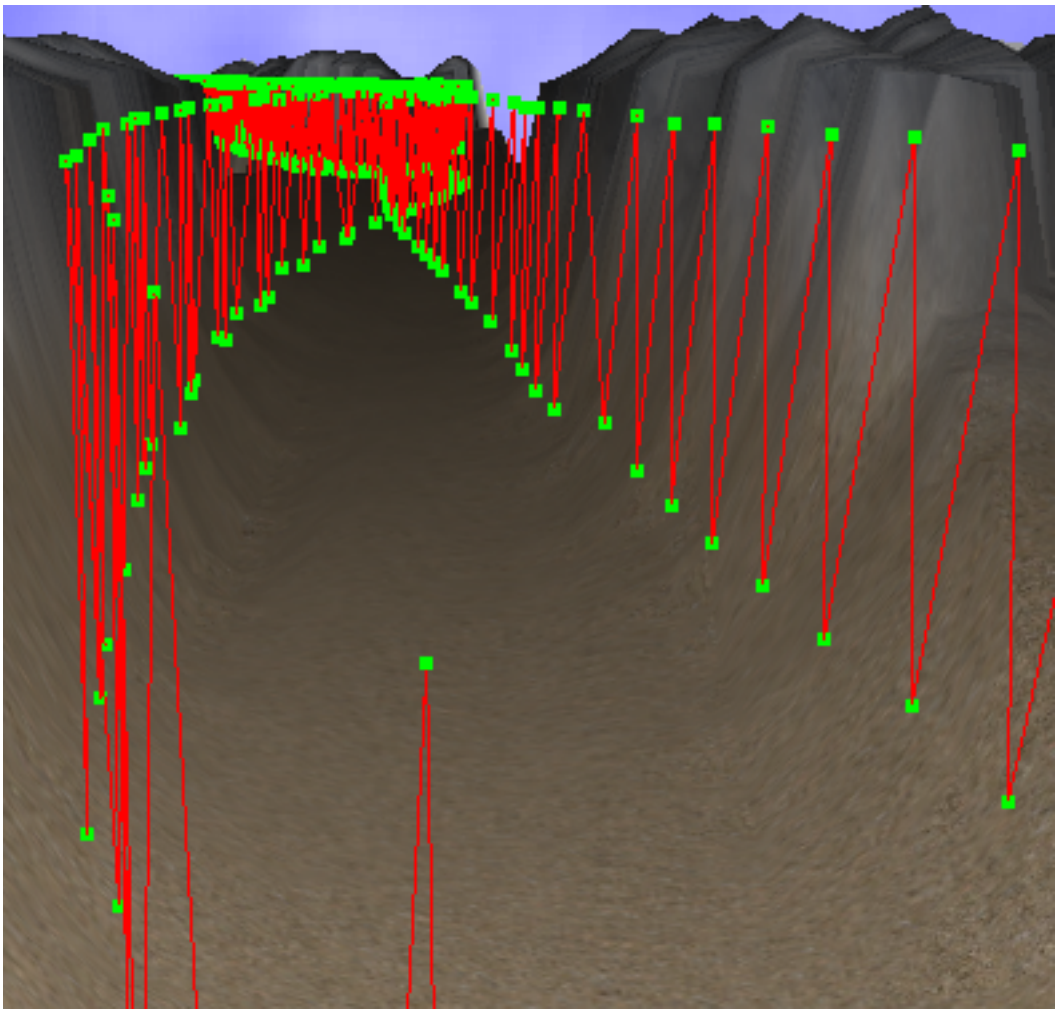

Figure 3.9: This figure shows the a portion of the approximated mission path constructed for the Svalbard mission.

local minimums and maximums. Figure 3.9 shows a portion of the approximated mission path constructed for the Svalbard mission.

3.2 Scientific Visualizations

The primary goal of the project was to generate scientific visualizations that would allow the biologists associated with NORUS to locate areas of interest

within the dataset as well as determine coorelation between variables. The biologists are interested in understanding how variables such as salinity, chlorophyll, turbidity, and temperature interact with eachother.

3.2.1 Spatial Data Structure

A spatial data structure is one that organizes geometry in some n-dimensional space. One primary use of spatial data structures is to accelerate queries about whether geometric entities overlap [3].

A common query used during the construction of a radial basis function is to find all data points within a specific spatial region. In order to accelerate these queries, a bounding volume hierarchy using axis-aligned bounding boxes was implemented. Every bounding volume contains references to each of the mission data points within the volume's extent. In order to assist in the construction of radial basis functions with overlapping sample points, a bounding volume also contains a reference to all adjacent bounding volumes of equal size.

3.2.2 Lookup Table

The lookup table allows for quick conversion from scalar values to their corresponding RGB color values. The construction of a lookup table requires that both a hue and value range are specified, generating a mapping between the scalar values and their corresponding color. Any scalar value passed to the lookup table that falls outside the value range is assigned either the minimum or maximum color associated with the lookup table.

3.2.3 Glyphs

Glyphs are simple objects used for the visualization of discrete data samples. Within the application, glyphs are represented as low resolution spheres whose color and size are mappable to different variables. The user is provided with a configuration window that provides the functionality to customize all aspects of how the glyph visualize discrete data. Figure 3.10 shows a view with the glyph size and color being mapped to different variables.

Variable Mapping

Glyphs are useful for mapping multiple variables to a single object. The ability to independently manipulate a glyph's size, color, orientation, and transparency provides the opportunity to map multiple variables to a single glyph. In order to avoid information overload, only the size and color of the glyphs within the application can be mapped to variables.

User Configuration

One key goal of the project is to provide the highest possible level of customizability so that the biologists have control over how the visualizations are generated and shown. Figure 3.11 shows the configuration window that helps to obtain this goal.

1. Variable Selection

   Variable selection allows the user to update the variables that are being mapped to either size and/or color of the glyphs.

2. Size Configuration

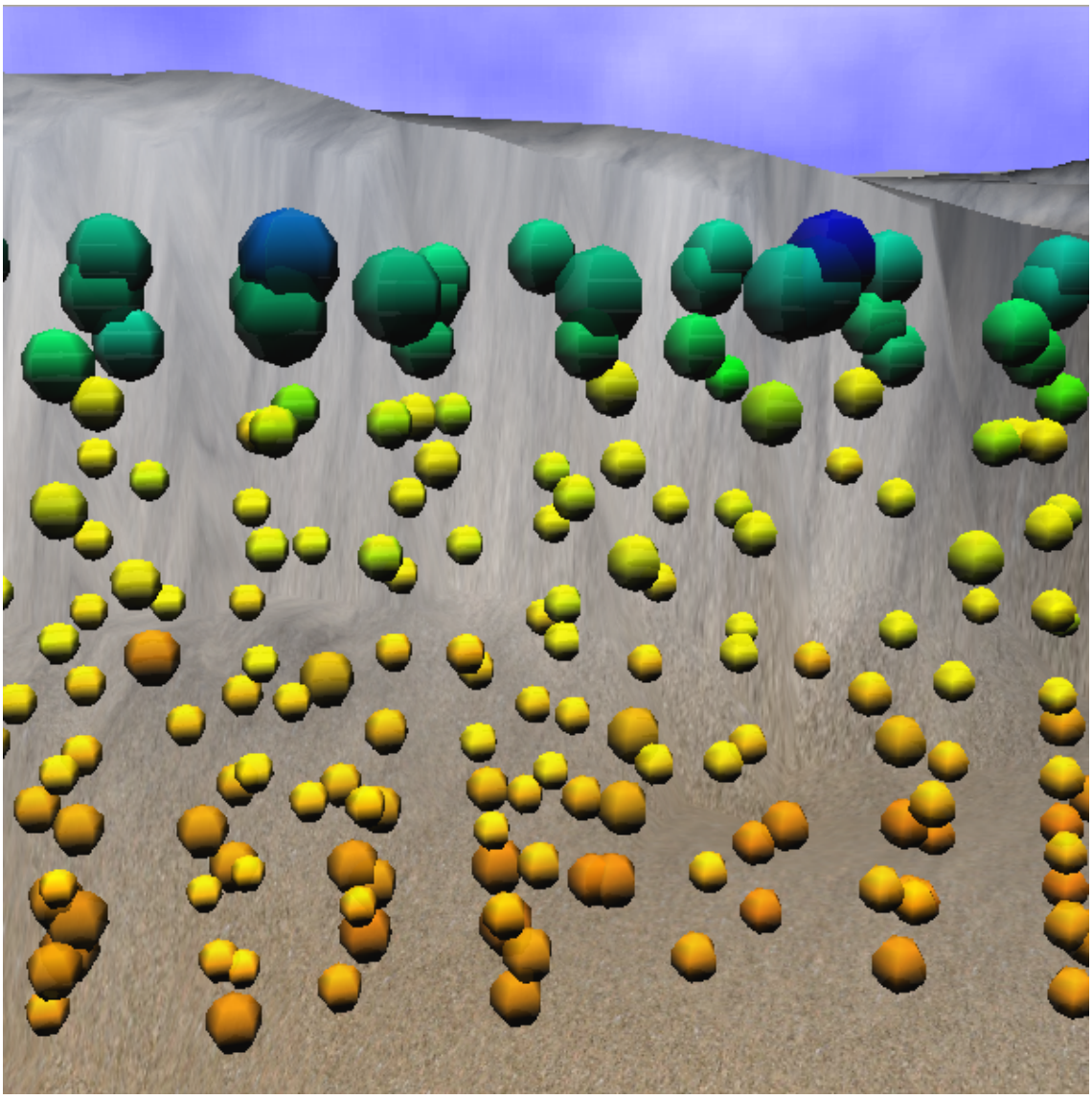

Figure 3.10: A view with glyph size being mapped to chlorophyll levels and color being mapped to salinity levels. The smaller sized glyphs correspond to lower levels of chlorophyll concentrain. The color scale goes from blue to red, where the blues correspond to lower values.

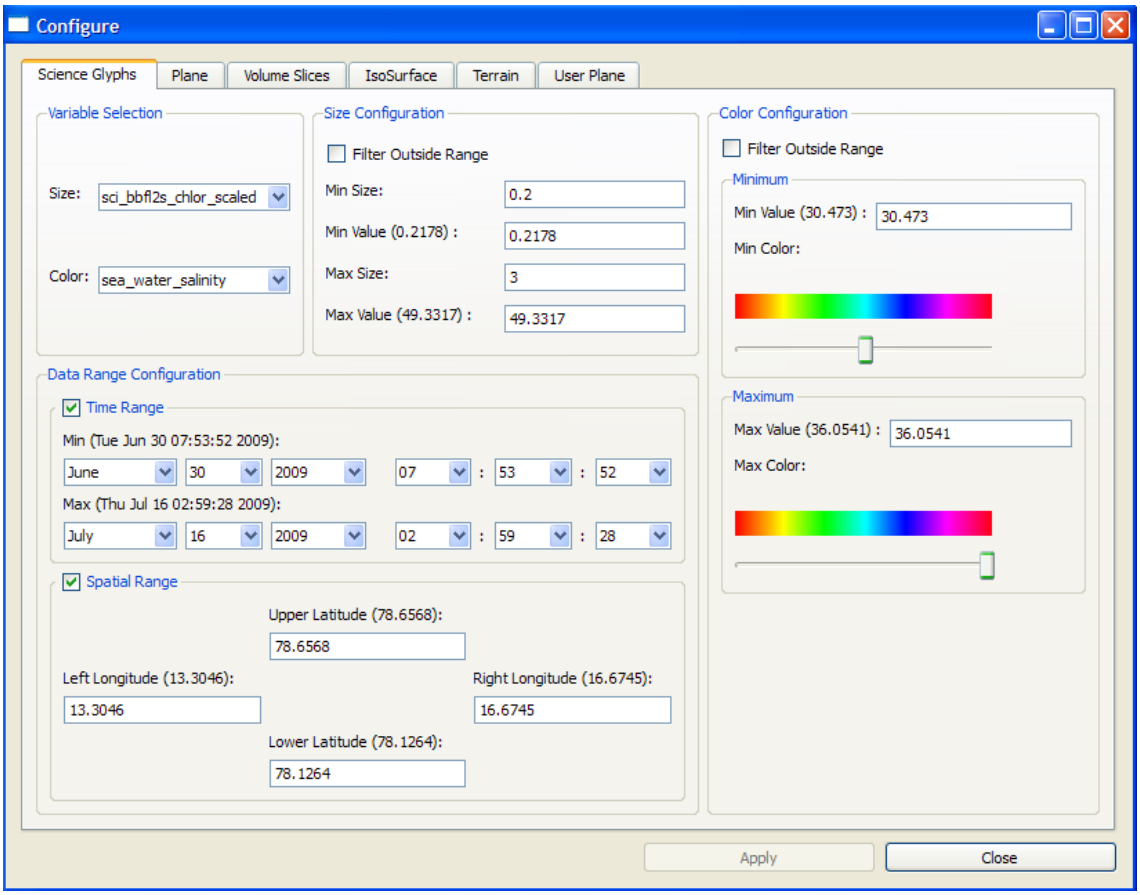

Figure 3.11: This figure shows the configuration window available to the user to manipulate how science data is displayed on the glyphs.

(a) Size

The minimum and maximum size specify constraints on the glyph size. The *value* is mapped to the *size* so that glyphs representing the minimum or maximum value are rendered at the corresponding minimum or maximum size. Any values in between are sized accordingly.

(b) Value

The minimum and maximum value specify constraints on the data values that the glyphs are able to represent. Since the glyph *size* is directly mapped to the *value*, whenever the value range is constrained there will be glyphs with values that lie outside the value range. This will result in glyphs with different *values* being mapped to the same *size*. This problem can be avoided by utilizing the *Filter Outside Range* feature. *Filter Outside Range* automatically filters glyphs whose data value lies outside the user defined value range. The mission minimum and maximum values for the chosen variable appear in the label name, *Min Value (X.XX)*.

3. Color Configuration

(a) Color

The minimum and maximum color specifies a color scale for the glyphs. The *value* is mapped to the *color* so that glyphs representing the minimum or maximum value are rendered with the corresponding minimum or maximum color. Values between the minimum and maximum are weighted by their distance from the minimum value and colored by selecting the color with an equivalent distance from the minimum color.

(b) Value

The minimum and maximum value specify constraints on the data values that the glyphs are able to represent. Since the glyph *color* is directly mapped to the *value*, whenever the value range is constrained there will be glyphs with values that lie outside the value range. This will result in glyphs with different *values* being mapped to the same *color*. This problem can be avoided by utilizing the *Filter Outside Range* feature. *Filter Outside Range* automatically filters glyphs whose data value lies outside the user defined value range. The mission minimum and maximum values for the chosen variable appear in the label name, *Min Value (X.XX)*.

4. Data Range Configuration

Glyphs can be filtered by time and/or space. When either of these constraints is adjusted, any glyphs that lay outside the range are not rendered.

(a) Time Range

In order to specify a time constraint for the glyphs, the checkbox next to the label *Time Range* must be checked. The default minimum and maximum time values correspond to the start and end time of the glider mission.

(b) Spatial Range

In order to specify a spatial constraint for the glyphs, the checkbox next to the label *Spatial Range* must be checked. The default latitude and longitude coordinates correspond to the glider's extent during the course of the mission.

3.3 Interpolated Scientific Visualizations

The majority of the visualizations generated by the application involve the use of radial basis functions as interpolators. The use of scattered data interpolation allows the application to generate compelling visualizations across surfaces where the glider may not have traveled.

3.3.1 Radial Basis Functions as Interpolators

Depth Constraints

Severe depth constraints are placed on the data samples used in the construction of the radial basis functions. These constraints arose from biologist input on the rate of change for the underwater variables of interest. Specfically, they noted that the variables of interest change approximately 100 times faster in the vertical direction then in the horizontal direction.

When constructing an interpolated visualization, an interpolation function is generated for each depth value associated with the visualization. In constructing the interpolators, the application attempts to use only data samples within half a meter up or down, and will only increase the depth if it did not find enough data samples. Using this method, the interpolation function is influenced primarily by data samples at the same approximate depth.

Basis Function Selection

The selection of a basis function and the corresponding user defined parameter is not a particularly easy choice to make. Many popular basis functions such as B-splines and Gaussian distribution functions *seem to have an at best shaky*

mathematical justification [14]. Also, the basis functions are often sensitive to a user defined parameter whose appropriate value will oftentimes depend upon the function values and point distributions. The goal is to select a basis function that best fits the surface.

Figure 3.12 provides a comparison of three popular basis functions with the shifted log ($\sqrt{log_{10}(r^2 + B^2)}$) that was ultimately chosen as the most desirable basis function. Each of the other basis functions produces interpolation functions that create local minimums and maximums that do not correspond to the discrete data samples. The polyharmonic spline, shown in figure 3.12B, produces values which increase as the distance increases, generating an interpolation function whose intial points have more influence the further away the input point is. The gaussian, shown in figure 3.12C, generates values that approach 0 as the distance increases. The problem is that due to the path of the glider, there are regions outside the influence of any particular point, resulting in regions of correct values primarily where the glider traversed. The multiquadric, shown in figure 3.12A, produces an interpolation function which is more accurate than the two previously described, but still does not adequately fit the surface. One major problem is that regions with sparse data samples, such as the bottom left quarter of the figure, produce innacruate results. The shifted log, shown in fgure 3.12D obviously generates the best interpolation function for the surface. The growth of the basis function is severly dampened by the log and square root. This generates a radial basis function whose weight coefficients fall into a reasonable spread, generating a smooth interpolation function.

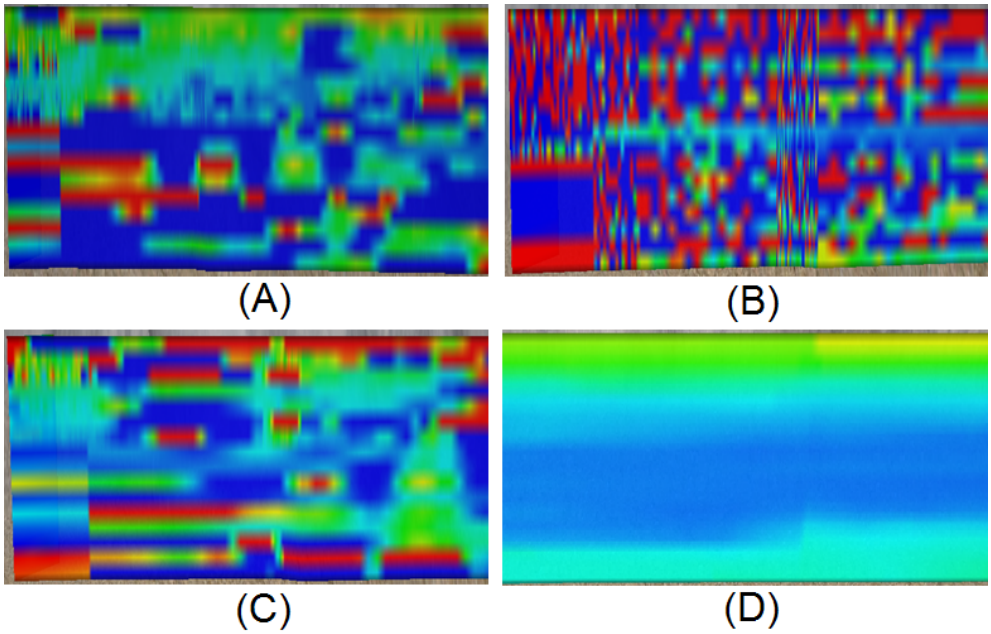

Figure 3.12: A comparison of radial basis function interpolation with different basis functions. (A) shows the result of a multiquadric ($\sqrt{r^2 + B^2}$) basis function. (B) shows a polyharmonic spline (r^{2n-1}). (C) shows a gaussian (e^{-Br^2}) basis function. (D) shows the shifted log ($\sqrt{log_{10}(r^2 + B^2)}$)

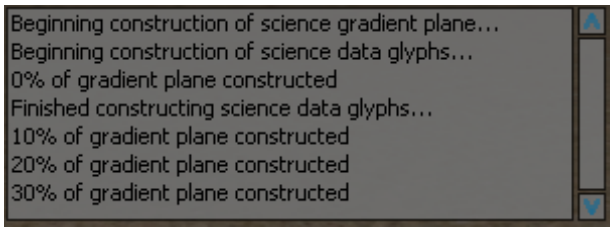

Figure 3.13: The HUD providing incremental updates on the status of the construction of the mission defined gradient plane.

3.3.2 Threaded Construction

The construction of the radial basis functions requires the inversion and multiplication of matrices to solve for a linear system. Depeneding on the amount of initial data samples provided, these can be rather lengthy calculations. In order to allow the user to continue to navigate through the data while a visualization is being constructed, the construction of any visualization requiring radial basis functions occurs within a thread. The appropriate locks are made to the graphical user interface while the visualization is being constructed, and the visualization is automatically enabled once construction finishes. A heads-up display, or HUD, provides updates on the status of the visualization's construction. Figure 3.13 shows the HUD providing incremental updates for the construction of the mission defined gradient plane.

3.3.3 Gradient Planes

The gradient planes are 2D planar surfaces rendered in the 3D environment. Using a radial basis function interpolator, the application generates smooth color gradients across the surface of the plane. In order to avoid hard transitions in the color gradient when spatial constraints require a new interpolator, the application

uses overlapping data samples. Whenever a new interpolator is generated, the application retrieves the data samples for the current bounding volume as well as those for each of the adjacent bounding volumes. The data samples are then filtered based on the depth constraints described in Section 3.3.1, and then the new interpolator is generated with the remaining data samples. By utilizing overlapping data samples, smoother transitions occur between regions using different interpolators. Figure 3.14 shows a comparison between the same gradient planes generated with and without overlapping sample points. The image on the left shows the hard transition that occurs without overlapping sample points. For the region to the left of the hard transition, the glider only traveled to approximately half of its maximum depth, producing a dark blue region that signifies a lack of data samples. By constructing the interpolation function with overlapping data samples, the system is able to interpolate more appropriate values for regions lacking adequate data samples. Each gradient plane is constructed so that its spans vertically from the sea level to the deepest depth reached by the glider.

Mission Defined Gradient Planes

Section 3.1.7 describes how the control points for the mission path are generated. The application utilizes the x and y coordinates of the control points when constructing the mission defined gradient planes. At each control point, we generate two new triangles, which can be viewed as a single quad, in the following manner. The first vertex uses the x and y coordinates of the current control point and the deepeset depth reached by the glider. The second vertex uses the same x and y coordinates with its depth set to the sea level. The next two vertices have similar depths, but use the next control points x and y coordinates. This is done for every control point except the last. Figure 3.15 shows how the mission defined

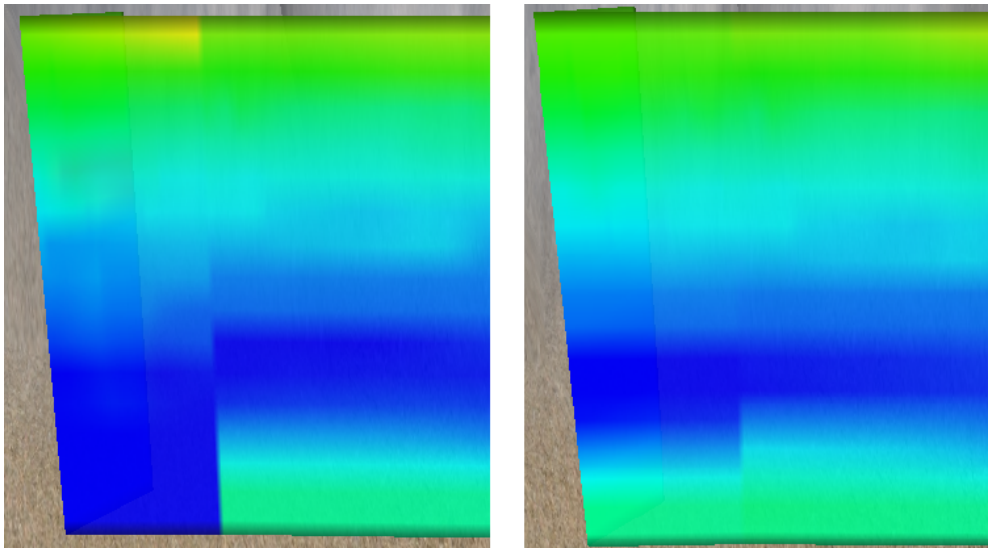

Figure 3.14: The image on the left shows the construction of the gradient planes without using overlapping sample points. The image on the right shows the same region with the gradient planes generated using overlapping sample points.

gradient plane corresponds to the control points of the mission path. In order to obtain an adequate resolution in the color gradient that spans the surface, each quad is subdivided into 256 equal sub-quads. An interpolated scalar value is then generated for each vertex of the sub-quads. Section 3.3.3 describes how the application utilizes texture mapping to minimize the number of triangles, or quads, required to display the gradient plane at an adequate resolution.

User Defiend Gradient Planes

The user defined gradient planes are generated in much the same manner as the mission defined gradient planes, with two primary differences. The first of which is that rather then utilizing the control points for the entire mission, the user defined gradient planes allow the user to specify the location of each control point. This allows the user to generate gradient planes that do not corre-

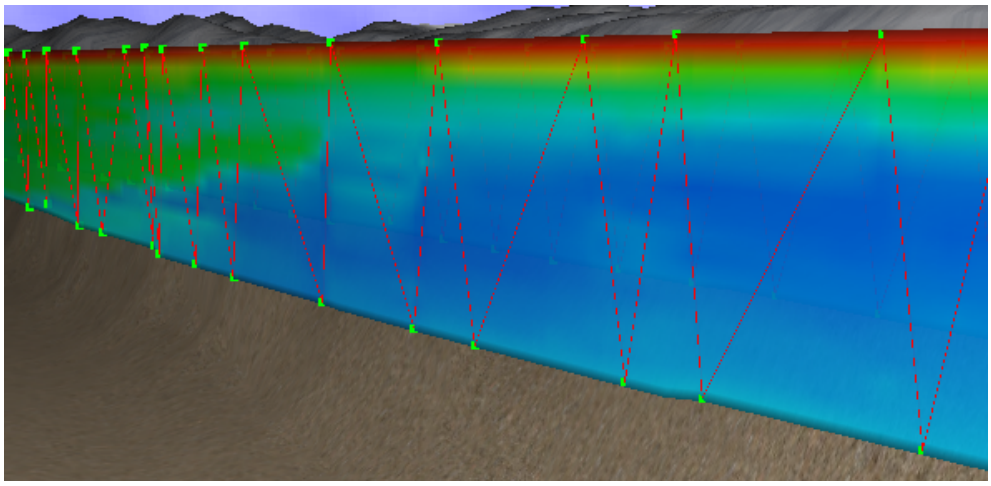

Figure 3.15: The image shows how the mission defined gradient plane corresponds to the control points of the mission path.

spond to the mission path taken by the glider. Figure 3.16 shows a user defined gradient plane that does not correspond to the mission path. The second primary difference is that the user defined gradient planes allows the user to specify time constraints on the data samples used to generate the radial basis functions. Throughout the course of a mission, the glider will oftentimes pass the same region multiple times in an attempt to determine how the time of day and the passage of time affect the measured values. The ability to specify time constraints on the data samples used in the generation of the radial basis functions allows the user to construct seperate visualizations for each glider pass.

Texture Mapping

Simply put, texture mapping is the process of mapping a function onto a surface in 3-D [18]. In order to minimize the number triangles required to render the gradient planes at an adequate resolution, a simple form of texture mapping was implemented into the system. Section 5.1.2 describes how control points are

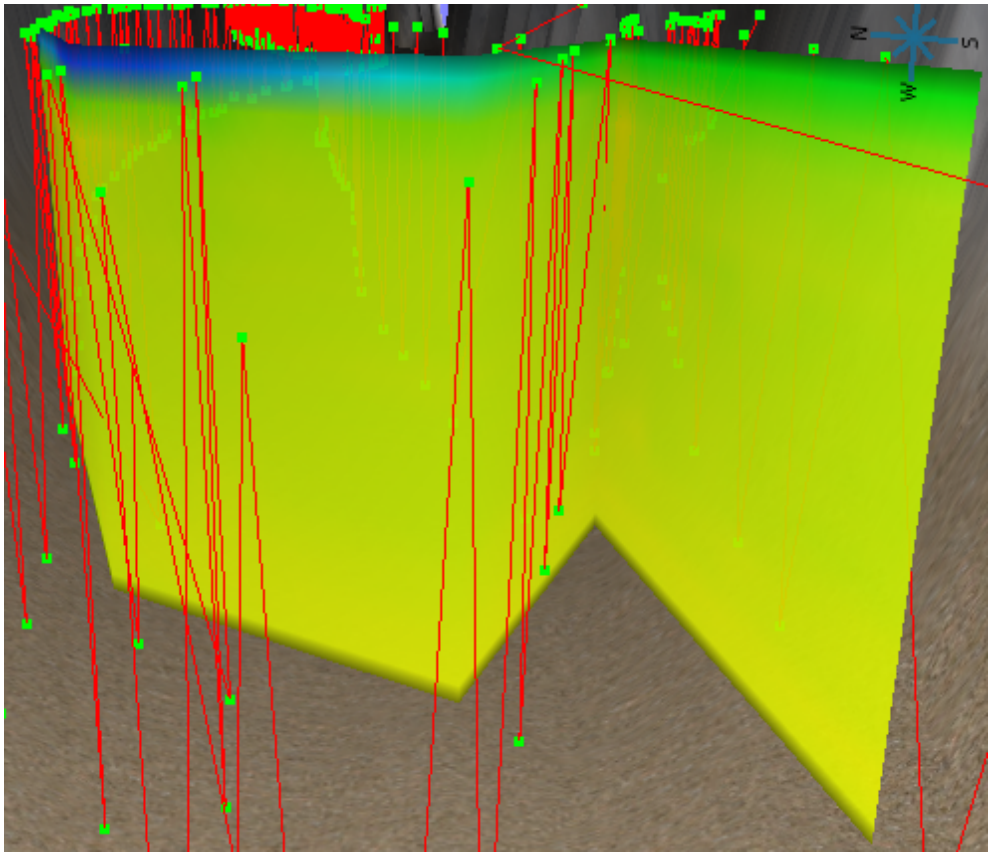

Figure 3.16: The image shows a user defined gradient plane that does not correspond to the mission path.

used to generate the vertices of the quads that define the gradient planes and how those quads are subdivided in order to obtain an adequate resolution for the color gradient. Rather then store and render 256 quads per control point, the application generates textures that store the RGB value at each vertex, reducing the amount of vertex storage to two times the number of control points.

Depending on the number of control points, it is sometimes necessary to generate multiple textures in order to accomodate the destired resolution. When this occurs, the textures are stiched together so that transitions between textures are not discernable. The mapping between vertex position and texture coordinate occurs as follows. The t value alternates between 0.0 and 1.0, depending on the position depth. The s value is the distance from the texture's first control point as a percentage of the total distance.

Texture Generation

In order to continue to provide a responsive and user configurable application, an additional feature was implemented that allows real-time texture generation to handle user configuration changes of the gradient planes. The only configuration change that does not occur in real-time is the selection of a new variable. When a new variable is selected, it is necessary to generate an entirely new set of interpolation functions, resulting in the re-creation of the entire surface.

Whenever a configuration change occurs that does not involve the selection of a new variable, the following steps occur in the generation of the new textures:

1. Delete the previous textures.

   The previous textures are stored in the graphics card's memory. At this point, the application makes the appropriate openGL function calls to delete

the previous textures from memory.

2. Generate a new lookup table with the updated color values and range.

   The new lookup table allows the application to easily generate the RGB values for each scalar value stored in the gradient plane.

3. Create and store the new textures.

   Using the scalar values stored in the gradient plane, the application generates the new textures for the updated configuration. Once the textures have been generated, they are once again stored on the graphics card's memory.

User Configuration

In order to maintain the highest possible level of customizability, all aspects of the user and mission defined gradients planes are customizable. Figure 3.17 shows the configuration window for the mission defined gradient planes.

1. Color Range

   The minimum and maximum color specifies a color scale for the gradient planes. The *value* is mapped to the *color* so that scalar values representing the minimum or maximum value are rendered with the corresponding minimum or maximum color. Values between the minimum and maximum are weighted by their distance from the minimum value and colored by selecting the color with an equivalent distance from the minimum color.

2. Color Value Range

   The minimum and maximum value specify constraints on the data values that the gradient plane is able to represent. Since the *color* is directly

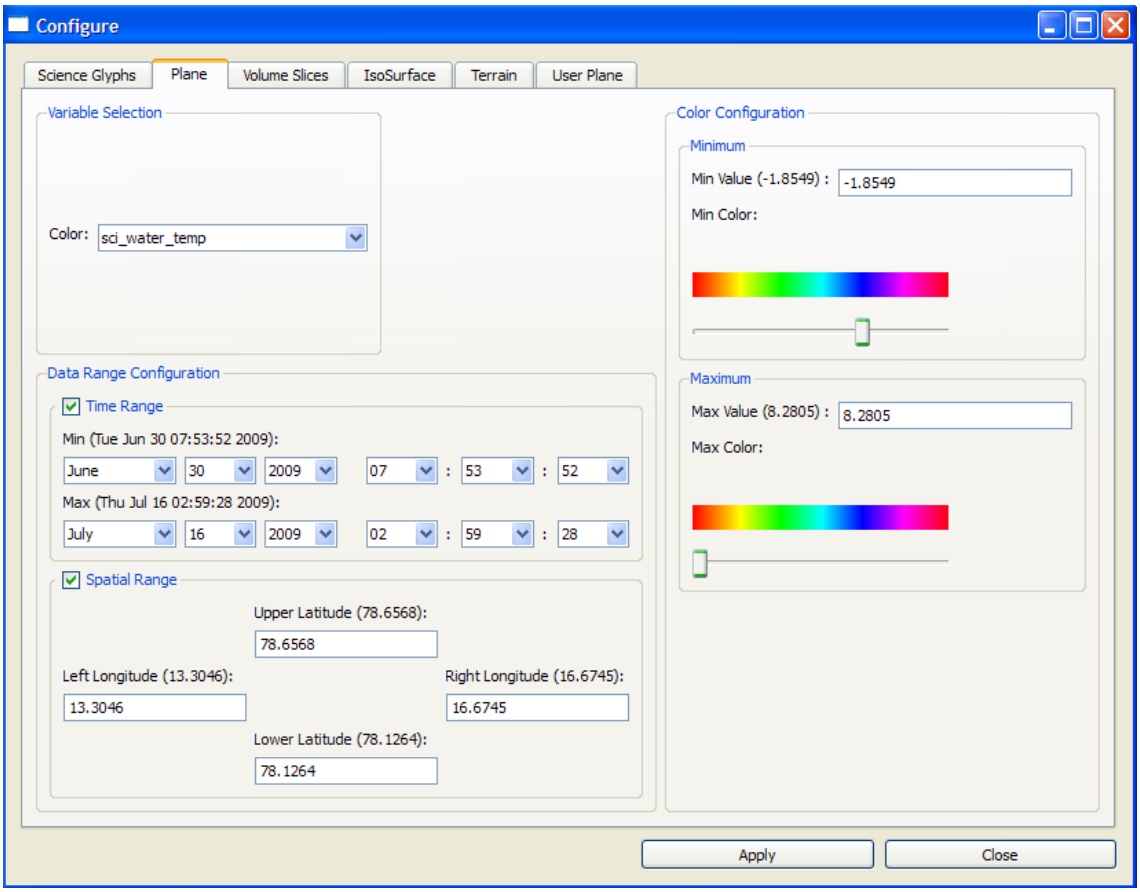

Figure 3.17: The image shows the configuration window for the mission defined gradient planes.

mapped to the *value*, whenever the value range is constrained there will be scalar values that lie outside the value range. This will result in texels with different *values* being mapped to the same *color*. The mission minimum and maximum values for the chosen variable appear in the label name, *Min Value (X.XX)*.

3. Time Constraints

   In order to specify a time constraint for the gradient planes, the checkbox next to the label *Time Range* must be checked. The default minimum and maximum time values correspond to the start and end time of the glider mission.

   As described in Section 3.3.3, the time constraint for the user defined gradient planes specifies constraints on the data samples used to generate the radial basis functions.

4. Spatial Constraints

   In order to specify a spatial constraint for the mission defined gradient planes, the checkbox next to the label *Spatial Range* must be checked. The default latitude and longitude coordinates correspond to the glider's extent during the course of the mission.

   Since the user specifies the control points for the user defined gradient planes, no spatial constraints are provided.

3.3.4 Volume Slices

Volume slicing provides a method for visualizing interpolated values across planar surfaces. The primary benefit that volume slicing provides over gradient

planes is the ability to generate different orientations of the 2-D planar surfaces. The volume slices configuration allows the user to specify that the slices be oriented in either the X-Y, X-Z, or Y-Z planes. Additionally, the user is able to specify the volume extent that the slices span, the resolution of each slice, and the variable used to generate the interpolation functions.

Radial Basis Function Construction

Section 3.3.1 describes in detail the depth constraints used to generate the radial basis functions used as interpolators. Briefly, an interpolation function is generated for each depth value associated with the volume slices. In constructing the interpolators, the application attempts to use only data samples within half a meter up or down, and only increases the depth if it did not find enough data samples. Using this method, the interpolation function is influenced primarily by data samples at the same approximate depth.

User Configuration

As with each previous visualization, the volumes slices have been designed to have the maximum amount of configurability possible. Figure 3.18 shows the configuration window for the volume slices.

1. Volume Specification

   The configuration window provides an interface for specifying the extent used to generate the volume slices. When the user selects the *New...* button, they are prompted to select four points within the scene that specify the initial volume selection. Figure 3.19 shows the wireframe box generated after the initial volume is selected by the user. The spinners within the

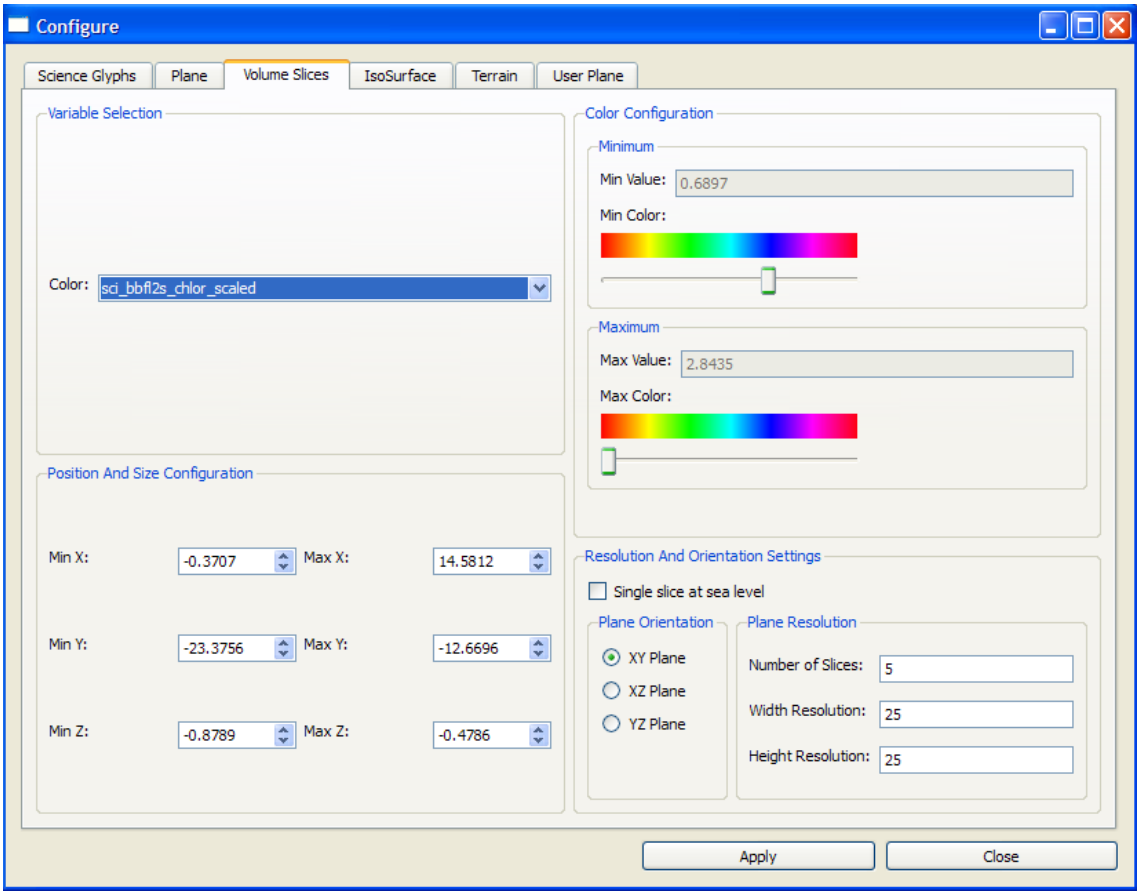

Figure 3.18: This image shows the configuration window for the Volume Slices.

configuration window allow fine tune adjustments to the initial volume that ensure that the user is able to generate slices in the desired position.

2. Plane Orientation

   The application provides three different planar orientations when generating the volume slices. The orientations are the X-Y, X-Z, and Y-Z planes. The various orientations allow the user to generate interpolated 2-D surfaces across regions that the glider did not travel. Figure 3.20 shows an example of the three different orientation of the volume slices.

3. Color Mapping

   The color mapping for the volume slices occurs in much the same manner as each of the previous visualizations. Refer to section 3.3.3 for a detailed description on how the color mapping occurs.

   One unique component for the volume slices is that the minimum and maximum color values are not editable until the volume slices have been generated. When the volume slices are generated, the minimum and maximum values are determine for the specified volume, and they are used as the default values. After the initial construction, the color values as well as the color range are editable in real-time.

4. Resolution

   The resolution of the volume slices is a user configurable property. This allows the user to specify both the number slices to generate and the resolution of each individual slice. The volume slices are not textured in the same manner as the gradient planes, so specifying a higher resolution results in more geomtery being drawn on the screen. It is important to keep

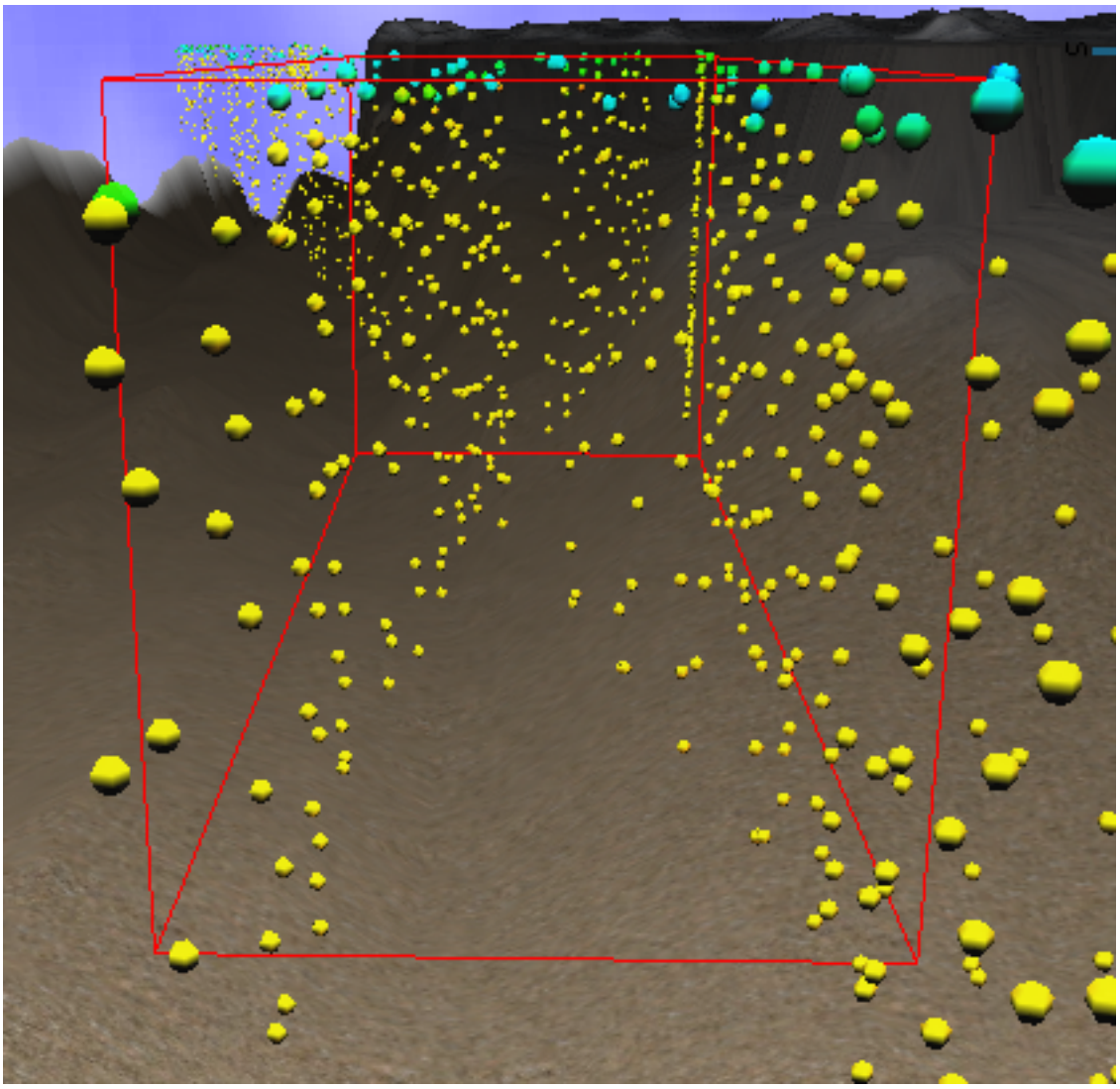

Figure 3.19: This image shows the wireframe box that is a visual representation of an initial volume selection for Volume Slices.

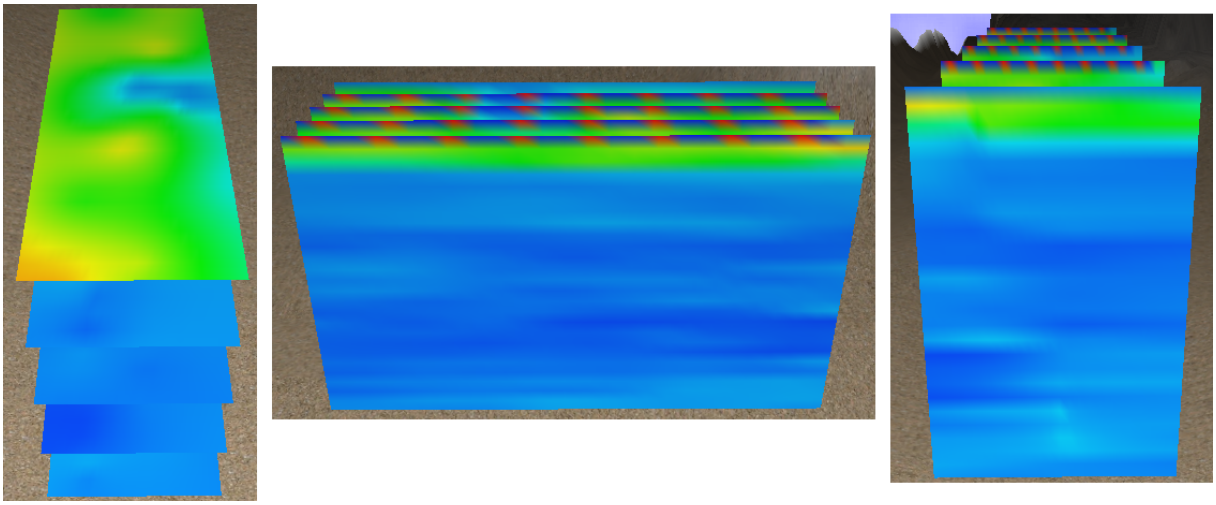

Figure 3.20: These image shows the three different orientations for the volume slices. The image on the left is showing chlorophyll being mapped to volume slices in the X-Y plane, the image in the center is the same variable mapped to volume slices in the X-Z plane, and the image on the right is showing the same thing but with images in the Y-Z plane.

this trade off in mind, especially on slower machines. An additional feature added for convenience is the ability to specify the generation of a single slice at sea level with a single click. Enabling the *Single slice at sea level* checkbox sets the min and max depth to be equal, the slice orientation to the X-Y plane, and the number of slices to one.

3.3.5 Isosurfaces

An isosurface is a 3-D surface that represents points of a predefined constant scalar value within a volume. The algorithm used to generate isosurfaces is discussed in Section 4.2. Isosurfaces provide a compelling method for visualizing interpolated values using 3-D surfaces. Also, the ability to map colors to the isosurface provides an effective means of visualizing the coorelation between

two different variables. When generating an isosurface, the user can specify the volume extent, isovalue, surface resolution, and the color mapping.

Radial Basis Function Construction

Refer to sections Radial Basis Functions As Interpolators and 3.3.4 for detailed descriptions on how the radial basis functions are generated.

One unique component for the construction of the isosurface interpolators is that the use of multiple variables requires the construction of two sets of interpolation functions. One set is used to generate the isosurface and the second is used to generate the colors for the surface.

Variable Coorelation

An interesting attribute of the isosurface is its ability effectively map multiple variables to a single visualization. It allows the user to easily determine the values of the variable mapped to color at positions of the specified isovalue. Figure 3.21 shows an isosurface generated using chlorophyll and color gradient generated from the temperature within the volume.

User Configuration

The isosurface provides an interface that allows the user to fully customize the surface's configuration. Figure 3.22 show the configuration window for an isosurface.

1. Isovalue

   The isovalue is the primary attribute of an isosurface. It specifies the scalar

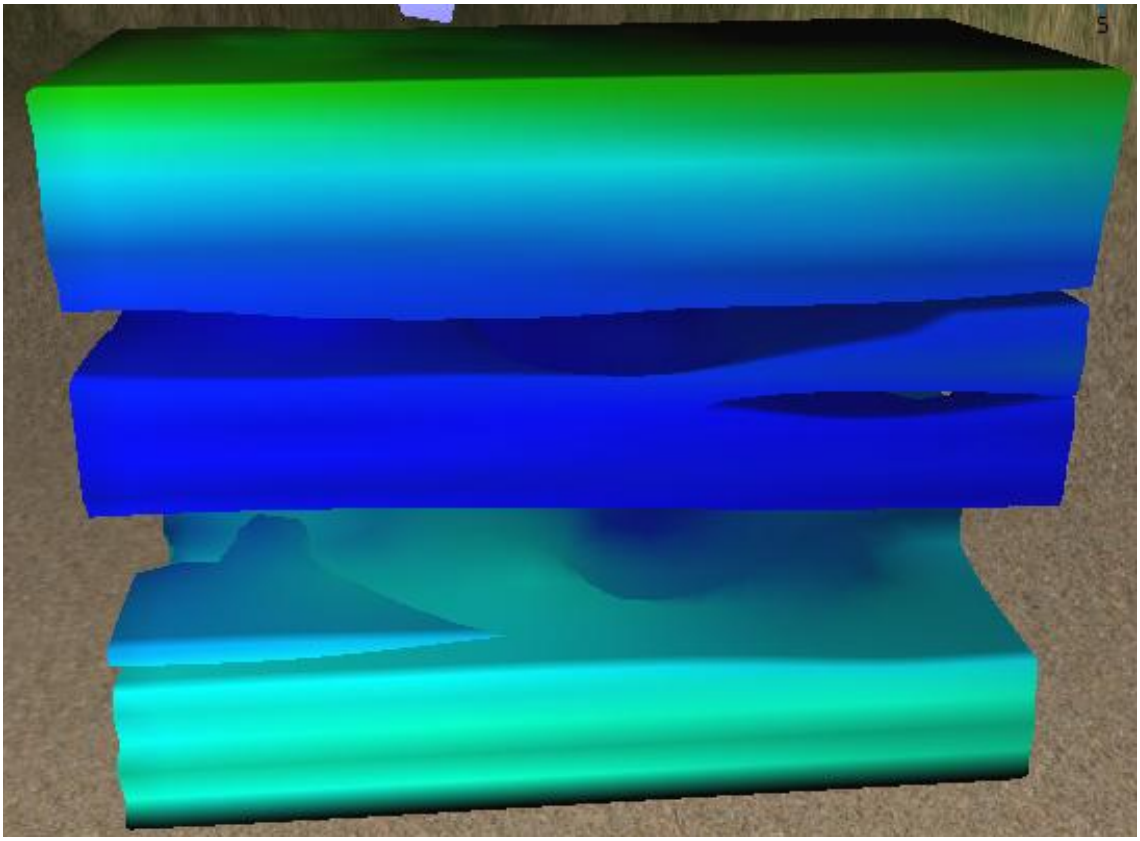

Figure 3.21: This image shows an isosurface generated chlorophyll with the color gradient generated from temperature.

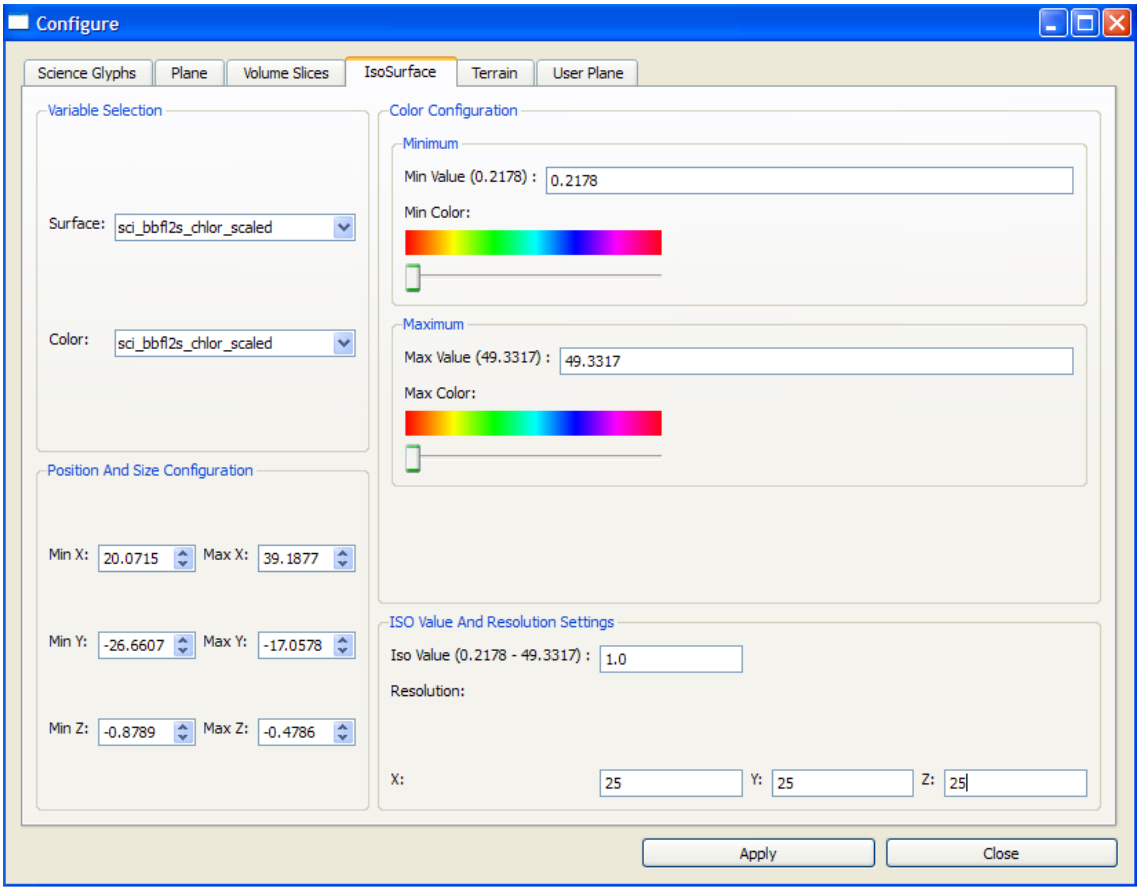

Figure 3.22: This image shows the configuration window for the Isosurface.

value for which the 3-D surface is generated. The application provides the functionality to update the isovalue in real-time after the initial surface generation. This allows the user to quickly change the isovalue, generating vastly different surfaces for the user to examine.

2. Resolution

   The resolution of the isosurface is a user configurable property. Higher resolutions result in denser grids from which to generate the 3-D surface. As the grid becomes denser, the amount of geometry being drawn to the screen also increases, resulting in a slower rendering time. It is important to keep this trade off in mind, especially on slower machines.

3. Color mapping

   The color mapping for the isosurface occurs in much the same manner as each of the previous visualizations. Refer to section 3.3.3 for a detailed description on how the color mapping occurs.

   One unique component for the isosurface is that the minimum and maximum color values are not editable until the isosurface has been generated. Once the isosurface has been generated, the minimum and maximum values are determine for the specified volume, and they are used as the default values. After the initial construction, the color values as well as the color range are editable in real-time.

Chapter 4

Algorithms

4.1 Scattered Data Interpolation

Scattered data interpolation is the broad category for methods that construct a continuous function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ given N sample values $f_i \in \mathbb{R}$ at N scattered data points $\mathbf{x}_i \in \mathbb{R}^n$. The methods of particular interest are those that construct the continous function as a linear combination of N basis functions, where $f(\mathbf{x}) = \sum_{i=0}^{N} \lambda_i \phi(||\mathbf{x} - \mathbf{x}_i||)$. Methods of this form use radially symmetric basis functions which are centered at each of the data points [17] [29].

4.1.1 Radial Basis Functions

Radial Basis Functions (RBFs) are a simple and useful tool for interpolating data in almost any number of dimensions. Given a set of data points with associated values, RBFs construct a smooth and continous function which interpolates the values at each data point. The procedure is to represent the function as a sum of radial basis functions each weighted by a coefficient with the addition

of a linear polynomial term. It is a requirement that the interpolated surface honors the original data points, so $f(\mathbf{x}_i) = f_i$ for $i = 1$ to N. This leads to a simple system of linear equations where we solve for the coefficients of the basis functions and polynomial terms [5].

Radial Basis Functions are represented mathematically as

$$f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi(||\mathbf{x} - \mathbf{x}_i||) + P(\mathbf{x}) \tag{4.1}$$

where the function is a sum of the basis functions, ϕ, weighted by a coefficient λ_i with the addition of a polynomial $P(\mathbf{x})$ in the components of $\mathbf{x}$. The addition of the linear polynomial term gives the RBFs linear precision, allowing the system to realize an affine transformation. Generating a solution for the RBFs can be reduced to solving a system of linear equations.

For radial basis functions whose scattered data points $\mathbf{x}_i \in \mathbb{R}^3$, the linear system can be written in standard matrix form, $Ax = b$, as

$$\begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} & 1 & x_1 & y_1 & z_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{N1} & \cdots & \phi_{NN} & 1 & x_N & y_N & z_N \\ 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ x_1 & \cdots & x_N & 0 & 0 & 0 & 0 \\ y_1 & \cdots & y_N & 0 & 0 & 0 & 0 \\ z_1 & \cdots & z_N & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.2}$$

where $\phi_{ji} = \phi(||\mathbf{x}_j - \mathbf{x}_i||)$. The first N rows of matrix A contain the expansion of equation 4.1 without its coefficients. The last 4 rows of A consist of four additional side constraints

$$\sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N} \lambda_i x = \sum_{i=1}^{N} \lambda_i y = \sum_{i=1}^{N} \lambda_i z = 0 \tag{4.3}$$

which provide the necessary equations for a fully determined linear system. For a discussion on how these side constraints are generated, refer to [29]. The weight and polynomial coefficients are located in matrix x, and matrix b contains each equations associated value. Solving for the unknowns in equation 4.2 requires the inversion of matrix A, leading to the requirement that A not be singular.

The effectiveness of radial basis functions in scattered data interpolation relies heavily upon the selection of proper basis functions. For a discussion on the basis function used in this application, refer to section 3.3.1.

4.2 Marching Cubes

Marching cubes is a high resolution 3-D surface construction algorithm. Specifically, it uses a constant value, or *isovalue*, to create a triangle mesh from 3-D data. When the marching cubes algorithm was originally published, it was used primarily for constructing 3-D surfaces from medical imaging data such as magnetic resonance (MR), computed tomography (CT), and single-photon emission computed tomography (SPECT) [23]. Since its original publication, it has been adapted to solve countless other 3-D surface construction problems [33] [34] [28] [21] [41] [25]. Its popularity is due to the simplicity of the algorithm and the ease of implementation.

Constructing the 3-D surface with the marching cubes algorithm consists of two primary steps. First, the surface, or *isosurface*, corresponding to the isovalue is located and then triangulated. Second, surface normals are calculated for each triangle of the isosurface so that the surface will be properly illuminated. Since each cube has eight verticies and each vertice can be either inside or outside the surface, there are a total of $2^8 = 256$ ways that a surface can intersect a

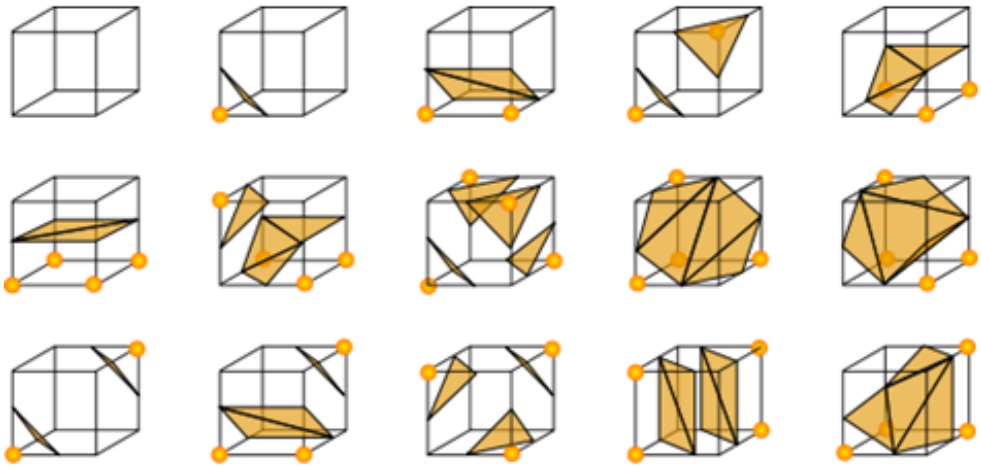

Figure 4.1: The 14 unique triangulations of a cube. Each of the 256 possible cube configurations can be obtained from these unique 14.

cube. With the use of two different symmetries of the cube, the 256 cases can be reduced to 14 unique triangulation patterns. First, the topology of the triangulated surface is unchanged if the relationship of the surface values to the cubes is reversed. Complementary cases, where vertices greater than the surface value are interchanged with those less than the value, are equivalent. This means that only cases with zero to four vertices greater than the surface valu need to be considered, reducing the number of cases to 128. Using rotational symmetry, the 128 cases can be reduced to 14 unique triangulations [23]. Figure 4.1 depicts the 14 unique triangulations.

Figure 4.2 provides the vertex and edge numbering order presented in the original paper. Using this numbering order, marching cubes precalculates a table consisting of the 256 possible edge configurations. The edge table is constructed so that the indexing method depicted in figure 4.2 corresponds to the appropriate edge configuration. Starting with a grid of voxels or cubes, the marching cubes algorithm proceeds as follows:

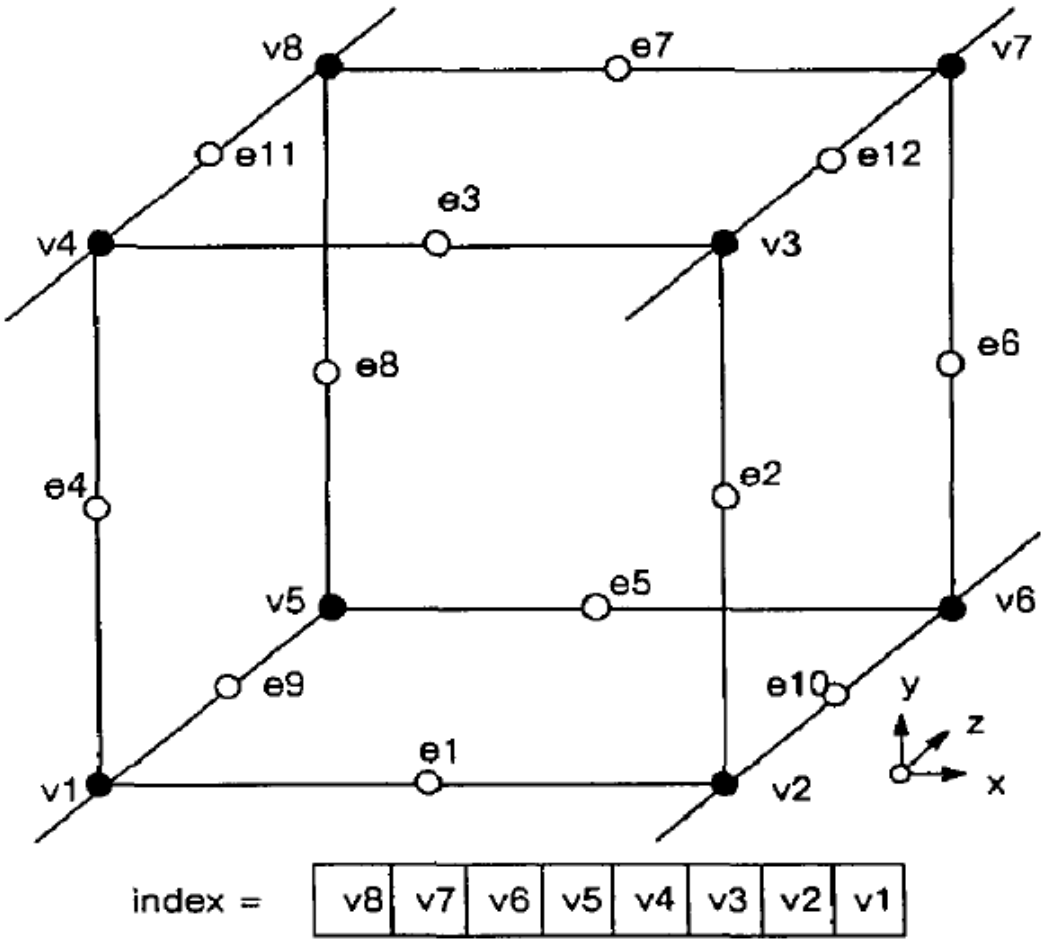

Figure 4.2: Cube Numbering

1. Iterate, or *march*, over each cube, determining edge intersections by generating an index into the edge table based on each vertices state (inside or outside the surface).

2. Using the values of the verticies at each intersected edge, use linear interpolation to estimate the position of the surface-edge intersection.

3. Calculate a unit normal at each cube vertex using central differences. Interpolate the normal to each triangle vertex.

   For a more detailed description on why it is possible to use central differences to calcualte vertex normals, refer to [23]. For the purposes of generating the 3-D surface, it is enough to know that the normal of each cube vertex is calculated as follows:

$$N_x(i,j,k) = \frac{D(i+1,j,k) - D(i-1,j,k)}{\Delta x}$$
$$N_y(i,j,k) = \frac{D(i,j+1,k) - D(i,j-1,k)}{\Delta y}$$
$$N_z(i,j,k) = \frac{D(i,j,k+1) - D(i,j,k-1)}{\Delta z}$$

   The normals of each triangle vertex are calculated using linear interpolation between the two edge adjacent cube vertices.

4. Render the triangle vertices using the calculated normals.

4.3 Multitexturing

Multitexturing is the combination of two or more textures applied to a single primitive. Proper application of multitexturing can greatly increase the level

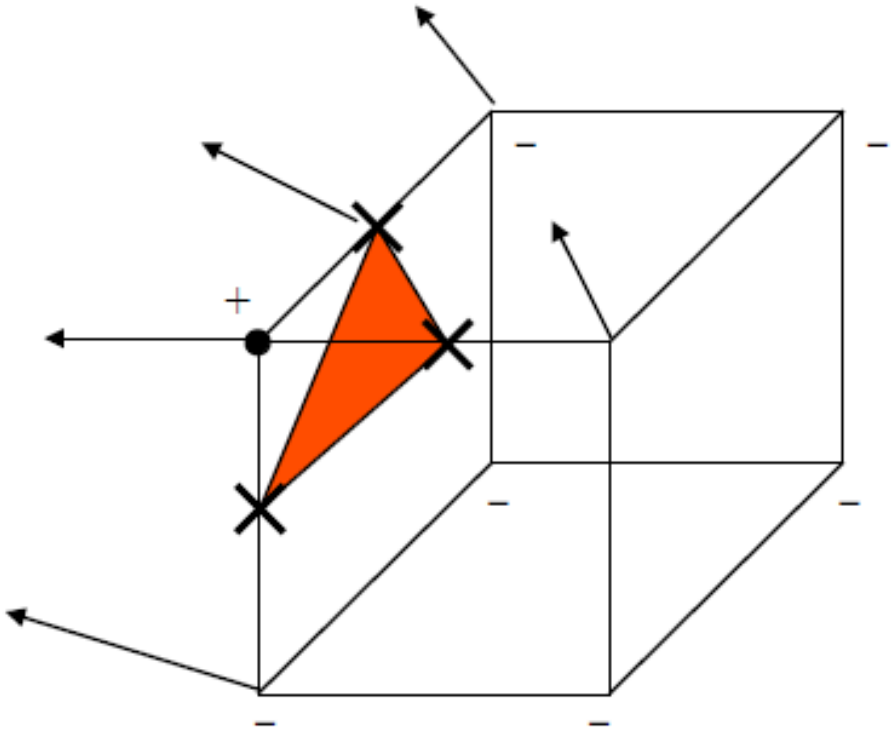

Figure 4.3: A depiction of how the vertex normals for the triangulated surfaces are calculated.

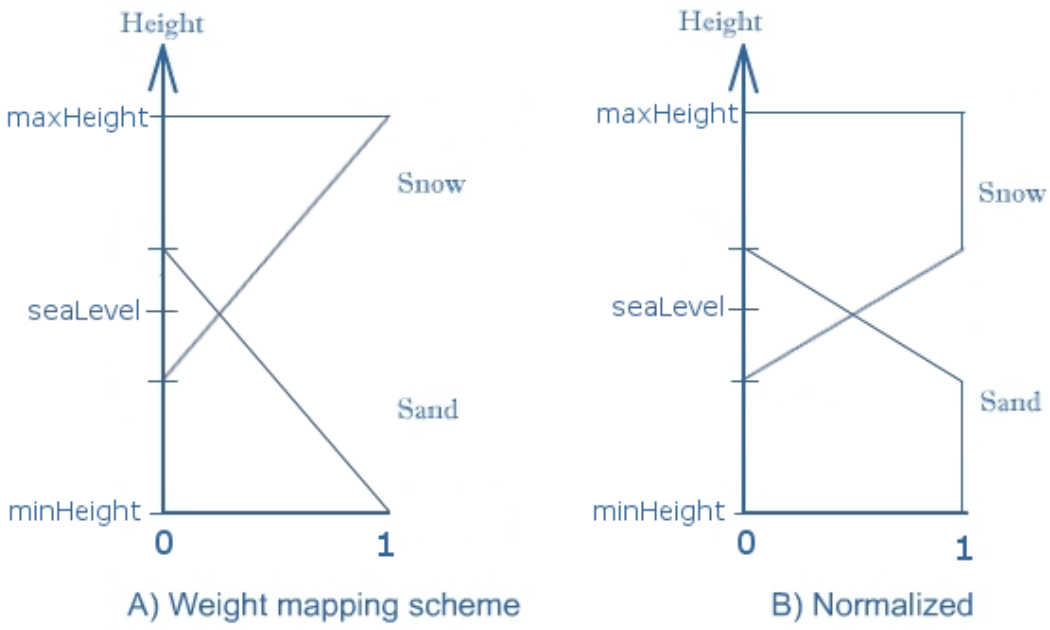

Figure 4.4: (A) shows the simple mapping scheme for two textures. (B) shows the normalized mapping scheme.

of detail within an environment without adding any additional geometry to the scene. Hardware multitexturing utilizes the graphics card's texture units to allow the hardware to apply a combination of multiple textures in a single pass, resulting in a rendering time that is noticably faster then previous techniques [3] [35].

The specific application of multitexturing discussed here is the use of weight coefficients to determine a texture's contribution at each vertex. Using weight coefficients that are calculated as a factor of the vertex height allows for smooth transitions between region specific textures. For example, the application uses a snow texture for regions above sea level and a sand texture for all terrain under the sea level; with smooth transitions between the two [39].

In order to generate the weight coefficients for the textures at each vertex, a

mapping scheme must be defined that maps vertex height to a weight coefficient. Figure 4.4a shows the ideal mapping scheme for the two textures described above. As figure 4.4a shows, the minimum and maximum heights receive a weight value of 1.0 for the sand and snow textures respectively. As the height approaches the sea level, the vertex is assigned a weight between 0.0 and 1.0 for both textures. The simple mapping scheme shown in figure 4.4a can be defined mathematically for each vertex as follows:

$$sandWeight = 1.0 - clamp(\left|\frac{(vertex.z - 0.0)}{(2.0/3.0 * (maxHeight - minHeight))}\right|, 0.0, 1.0)$$

$$snowWeight = 1.0 - clamp(\left|\frac{(vertex.z - maxHeight)}{(2.0/3.0 * (maxHeight - minHeight)}\right|, 0.0, 1.0)$$

The weight for the sand texture scales between 1.0 and 0.0 for vertex.z values between 0.0 and 2/3 of the total height span. Similarly, the weight for the snow texture scales between 1.0 and 0.0 for the vertex.z values between the maxHeight and 1/3 of the total height span.

The problem with this simple mapping scheme is that the total vertex weight for heights between the minimum and maximum will not add up to 1.0, resulting in dark primitives being rendered for values close to the middle. This can be corrected by normalizing the weights at each vertex so that the sum of the texture weights equal 1.0. Figure 4.4b shows the original mapping scheme with normalized weights. Using the previously defined sand and snow weights, the normalized mapping scheme can be defined mathematically as follows:

$$totalWeight = sandWeight + snowWeight$$

$$sandWeight = sandWeight/totalWeight$$

$$snowWeight = snowWeight/totalWeight$$

Figure 4.5: A view of the smooth transition that occurs between textures.

For the sake of simplicity, only two textures were used in the mapping scheme, but it can easily be extended to include more then two textures. Figure 4.5 provides a view of the smooth transition that occurs between textures when the weights are defined appropriately.

Chapter 5

Results

5.1 Scientific Visualizations

This section provides analysis of the scientific visualizations generated by the system. The focus is put primarily upon validating the interpolated values generated by the radial basis functions and showing how user configurable properties can be changed to highlight areas of interest.

5.1.1 Glyphs

Figure 5.1 provides a view of the glyphs with both size and color mapped to chlorophyll. Both are using a minimum value of 0.2178 (global minimum) an a maximum value of 4.0. The sizes range from a scale of 0.2 to 1.0 and the colors range from a dark blue (0.667) to red (0.0). The maximum was lowered from the global maximum of 49.3317 in order to provide more variation in the glyphs. Since the vast majority of chlorophyll measurements lie between 0.2178 and 8.0, the regions with excessively high chlorophyll levels most likely constitue an error

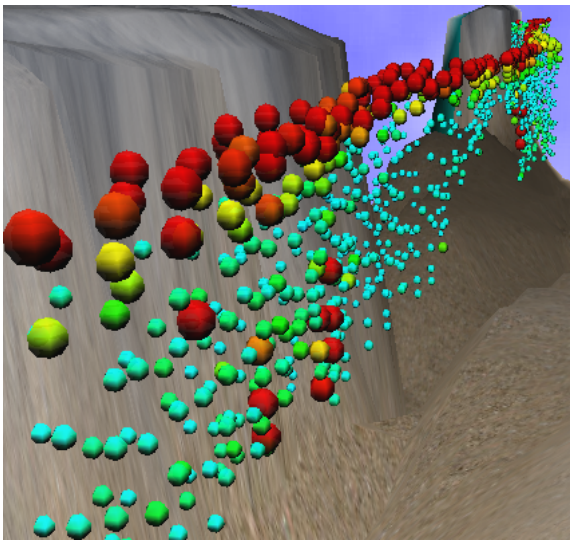

Figure 5.1: A view of the glyphs with both size and color mapped to chlorophyll.

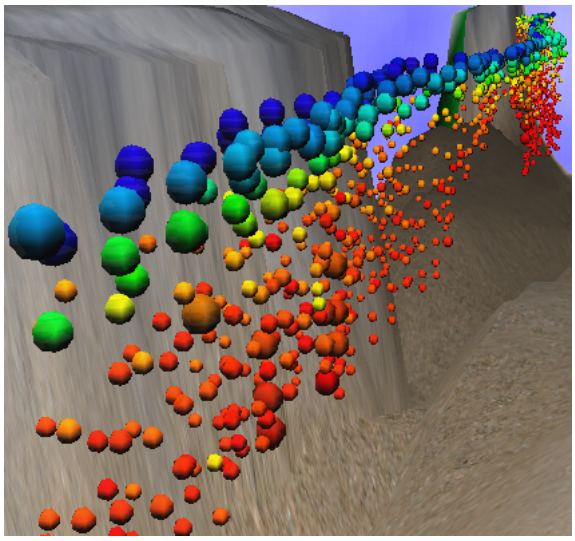

Figure 5.2: A view of the glyphs with size mapped to chlorophyll and color mapped to salinity.

in measurement or a region of interest.

Figure 5.2 provides a view of the glyphs with size mapped to chlorophyll and color mapped to salinity. The size values are identical to those in figure 5.1. The color values range from a minimum of 32.473 and a maximum of 34.5, which constrain the range by 2.0 and 1.5541 respectively. The colors range from dark blue (0.667) to red (0.0), with dark blue corresponding to the lowest values. Simple analysis shows that the salt content is lowest at the surface of the water and increases steadily until approximately one third of the glider's maximum depth, where it stays fairly consistent. For the most part, there appears to be an inverse correlation between chlorophyll levels and salt content, with regions of variability.

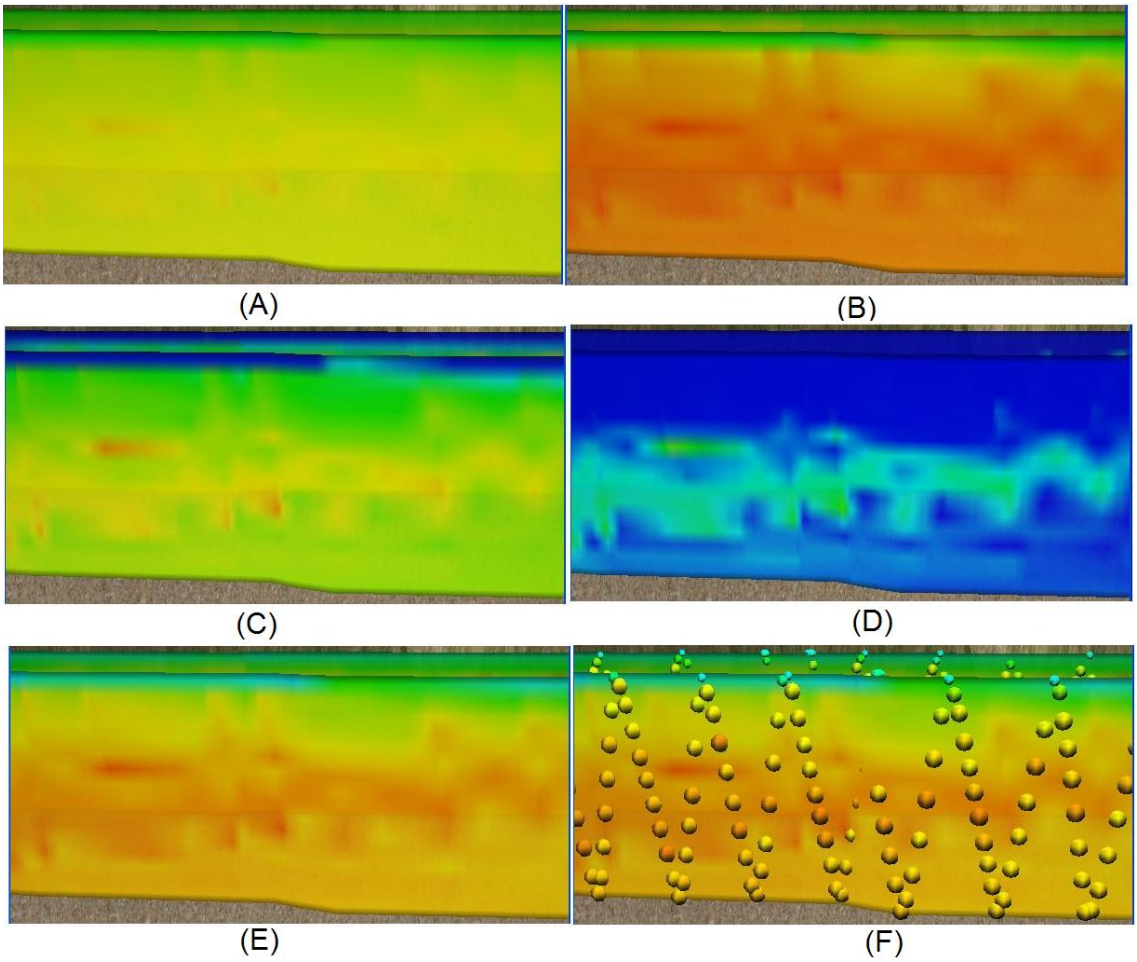

Figure 5.3: A sequence of images that show how altering the minimum and maximum value range can highlight different regions of the mission defined gradient plane.

5.1.2 Mission Defined Gradient Planes

Figure 5.3 provides a sequence of images that show how altering the minimum and maximum value range can highlight different regions of the mission defined gradient plane. In every image, the gradient plane is being mapped to salinity with a minimum color of dark blue (0.667) and a maximum color or red (0.0). Figure 5.3A has a minimum value of 30.473 and a maximum value of 36.0541, figure 5.3B has a minimum value of 31.473 and a maximum value of 35.0541, figure 5.3C has a minimum value of 33.473 and a maximum value of 35.0541, figure 5.3D has a minimum value of 34.473 and a maximum value of 35.0541, and both figure 5.3E and figure 5.3F have a minimum value of 32.473 and a maximum value of 35.0541. Figure 5.3F shows the glyphs being mapped to with the same value and color range as the gradient plane to show the coorelation between discrete data sample values and the interpolated values generated by the interpolation function. With default values shown in figure 5.3A, the areas of higher values are barely visible. By constraining the minimum and maximum value range in different ways, it is possible to highlight the areas of higher values within the specific region. Figure 5.3E produces the best looking results, with the regions of higher values clearly visible.

Figure 5.4 provides a view of the temperature mapped mission defined gradient planes. The gradient planes are being rendered with a value range of -1.8549 (default) to 8.2805 (default) and a corresponding color range of dark blue (0.667) to red (0.0). The specific region shows an interesting striping pattern where there are regions of lower temperature above regions of higher temperature. The glyphs in the lower image show how the interpolated values of the gradient plane correspond to discrete data samples.

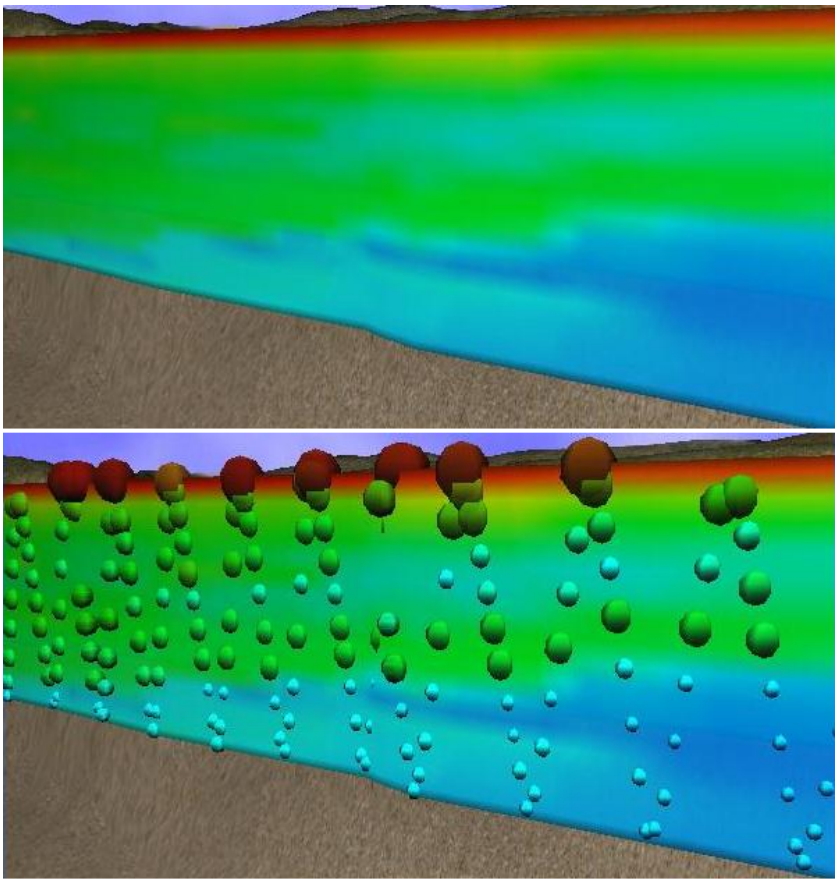

Figure 5.4: This view of the temperature mapped mission defined gradient planes shows a region with interesting temperature values. The glyphs in the lower image show how the interpolated values coorelate to discrete measurements.

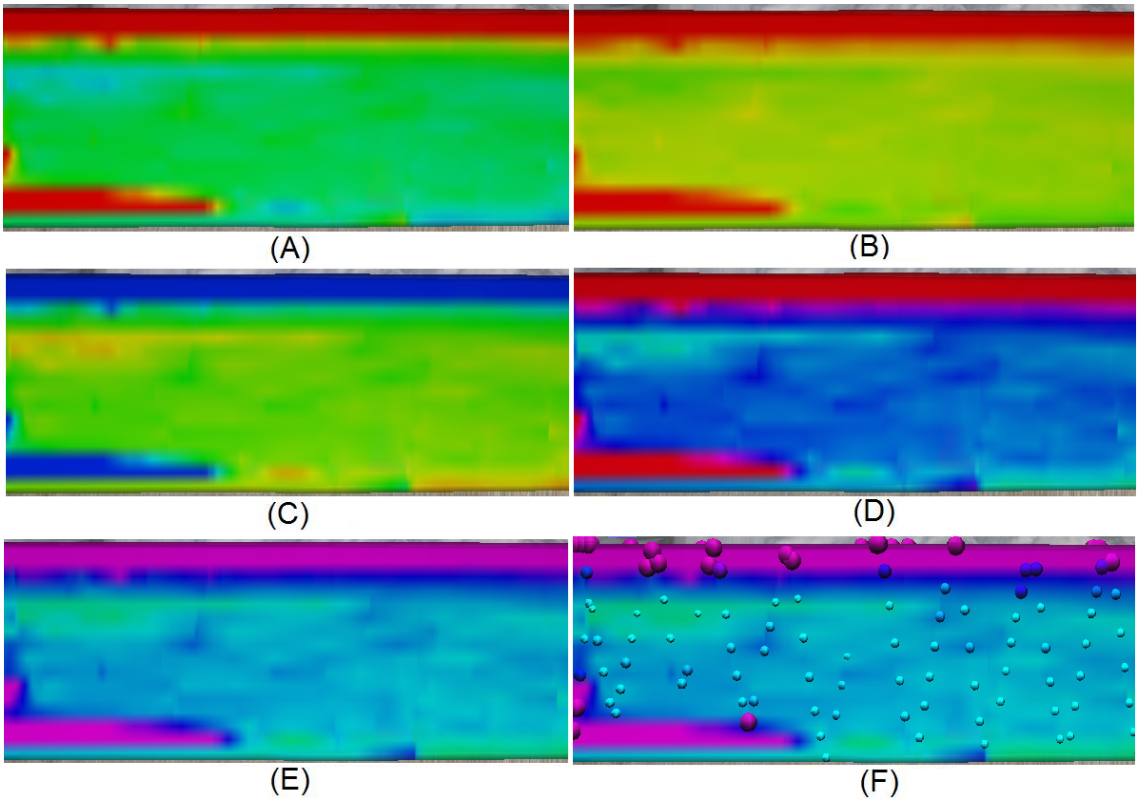

Figure 5.5: A sequence of images that show how altering the minimum and maximum color range can highlight different regions of the mission defined gradient plane.

Figure 5.5 provides a sequence of images that show how altering the minimum and maximum color range can highlight different regions of the mission defined gradient plane. In every image, the gradient plane is being mapped to chlorophyll with a minimum value of 0.2178 (default) and a maximum value of 4.0. Figure 5.5A has a minimum color or dark blue (0.667) and a maximum color of red (0.0), figure 5.5B has a minimum color of green (0.333) and a maximum value of red (0.0), figure 5.5C has a minimum color of red (0.0) and a maximum value of blue (0.667), figure 5.5D has a minimum color of blue (0.667) and a maximum value of red (1.0), and both figure 5.5E and figure 5.5F have a minimum color of sky blue (0.5) and a maximum value of purple (0.8). Figure 5.5F shows the glyphs being mapped to with the same value and color range as the gradient plane to show the coorelation between discrete data sample values and the interpolated values generated by the interpolation function.

5.1.3 Volume Slices

Figure 5.6 provides a view of a chlorophyll mapped volume slice along the ocean surface. The slice is being rendered at a resolution of 50x50 with a value range of 0.8712 to 3.4606 and a corresponding color range of dark blue (0.667) and maximum color of red (0.0). The glyphs represent discrete data samples used to generate the interpolation function for the region. The important thing to note is the coorelation between color of the glyphs and the color of the volume slice. Since the data samples within the selected region are fairly spare, there is some error in the generated interpolation function. The area labeled A shows how the interpolation function generates a smooth increase in values as the interpolated position moves further from the initial sample points. The area labeled B shows the opposite effect.

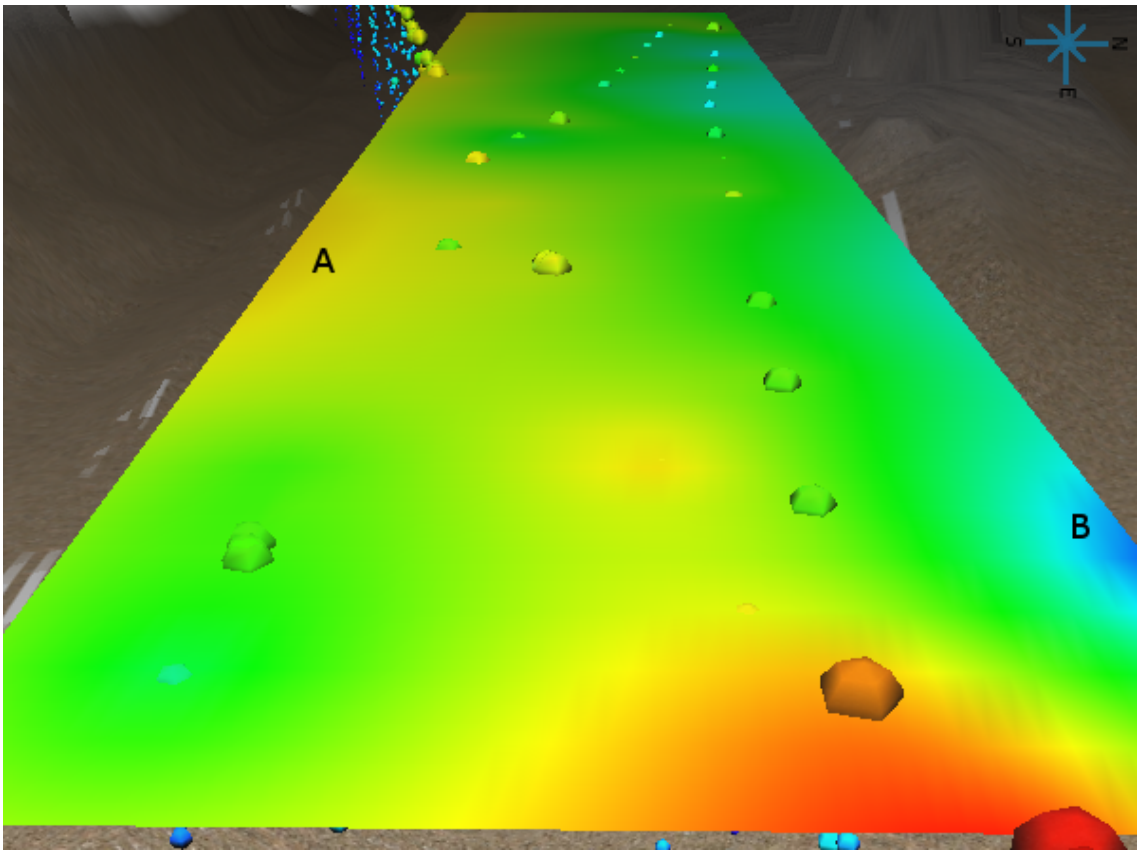

Figure 5.6: A chlorophyll mapped volume slice along the ocean surface.

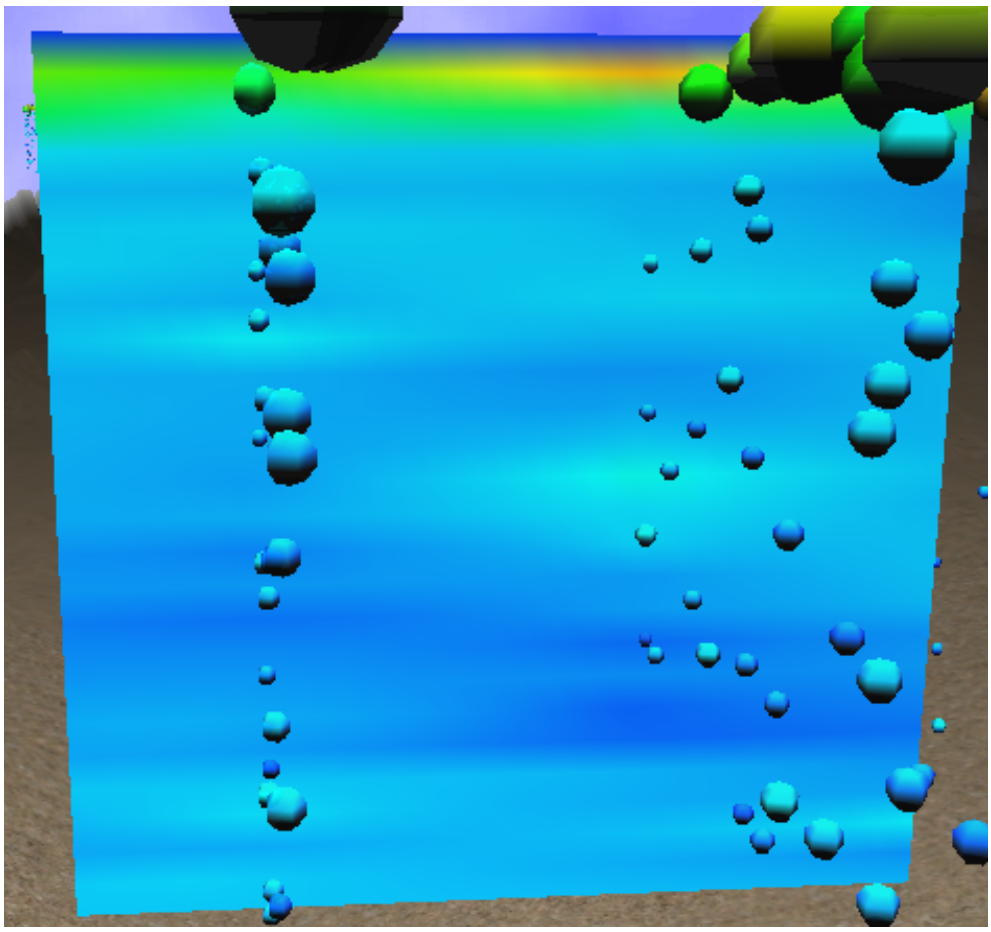

Figure 5.7: A chlorophyll mapped volume slice in the YZ plane.

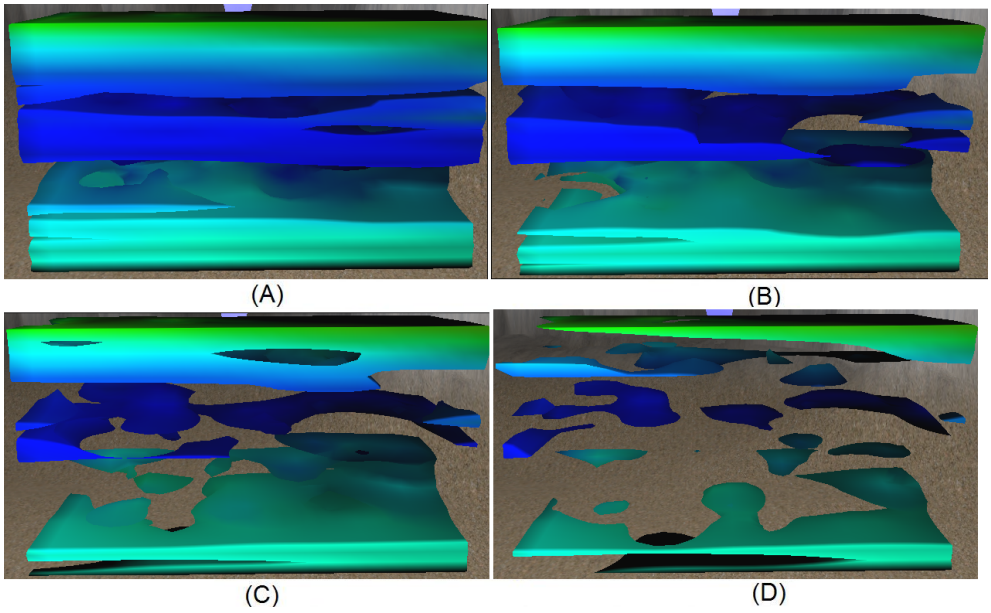

Figure 5.8: A sequence of views for a chlorophyll mapped isosurface with temperature mapped to color.

Figure 5.6 provides a view of a chlorophyll mapped volume slice along the YZ plane. The slice is being rendered at a resolution of 50x50 with a value range of 0.8712 to 3.4606 and a corresponding color range of dark blue (0.667) and maximum color of red (0.0). The slice generates interpolated values for the area inbetween the two distinct glider passes that are defined by the position of the glyphs. Also, the dark blue region at the top of the slice results from a lack of data samples at that depth.

5.1.4 Isosurfaces

Figure 5.8 provides a sequence of views for a chlorophyll mapped isosurface with temperature mapped to color. Figure 5.8A is defined with an isovalue of 0.9, and in each successive image the isosvalue is increased by 0.05, resulting in figure 5.8D being mapped to an isovalue of 1.05. The color of each isosurface is

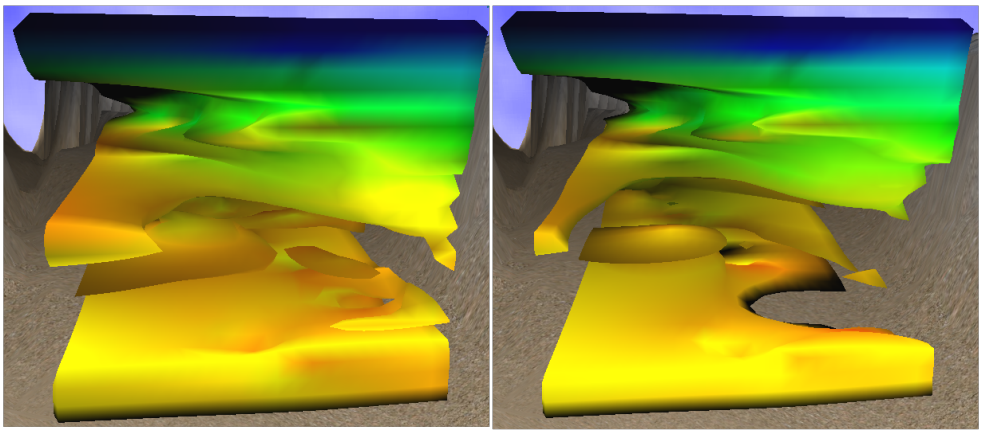

Figure 5.9: Two views for a chlorophyll mapped isosurface with salinity mapped to color.

mapped to the temperature within the region. The minimum and maximum color values are set to the corresponding global values of -1.8549 and 8.2805, while the color ranges from dark blue (0.667) to red (0.0).

Figure 5.9 provides two views for a chlorophyll mapped isosurface with salinity mapped to color. Figure 5.9A is mapped to an isovalue of 0.9 and figure 5.9B is mapped to 0.95. The color of both isosurfaces are mapped to the salinity levels within the region, with a color range of dark blue (0.667) to red (0.0).

5.2 Interpolated Visualization Creation Time

The primary bottleneck in constructing the interpolated visualizations is the construction of the radial basis functions. Specifically, finding a solution to the linear system requires the inversion of a matrix, which has a run time of $O(n^3)$. Table 5.1 shows how the creation time for a single volume slice oriented in the XY plane is effected by the number of data samples used in the creation of the radial basis functions.

Number of Data Points	Volume Slice Creation Time (ms)
202	217
438	1715
457	2176
523	2746
751	7373
789	8364
1292	3486
1415	50029
1463	50765

Table 5.1: A Table that shows how the number of data points effects the creation time of a volume slice. The volume slice is constructed with a resolution of 25x25 and is oriented in the XY plane.

When the volume slice is constructed, an interpolation function is generated for the slice's depth value. When constructing the interpolator, the application attempts to use only data samples within half a meter up or down, and will only increase the depth if it did not find enough data samples within range. Since the appliction applies severe constraints on the data points used for the construction of the interpolator, the problem illustrated in table 5.1 is mitigated. For a full description on the depth constraints used when generating the radial basis functions, refer to section 3.3.1.

Chapter 6

Conclusions

This thesis has presented a compelling, efficient, and easy to use interactive system for visualizing large sets of science data collected by the Slocum Glider. An interactive camera allows the user to navigate through accurate terrain generated from bathymetry data. The visualization system constructs glyphs, mission defined gradient planes, user defined gradient planes, volume slices, and isosurfaces. Although the glyphs are generated from discrete data measurements taken by glider, all other visualizations, known as interpolated visualizations, use scattered data interpolation to allow continous sampling over regions containing a discrete number of data measurements.

The system was developed iteratively to ensure that it would be a useful tool for the biologists involved in analyzing and interpreting the glider data. At the completion of each iteration, the system was presented to NORUS biologists to obtain user feedback and gather additional requirements for the next iteration. Overall, user feedback was extremely positive, and the biologists were excited to have an additional tool for visualizing the glider data.

On a personal note, I gained a lot of experience in the field of scientific visualization and computer graphics as a whole. Working on a project that users were excited to have the opportunity to use was a rewarding experience. At the completion of each iteration, the biologists were eager to provide us with feedback on the system and additional features they wanted to have implemented.

The following list outlines some future work that can be done to extend the system, making it more user accessible.

1. Support for Multiple Missions

   At this point, the system only allows the user to visualize a single mission at a time. It's reasonable to believe that the glider could complete multiple missions within the same region. Ideally, multiple missions would produce a better spread of data sampling, making it possible to construct more accurate visualizations.

2. Support for Multiple Instances of Interpolated Visualizations

   The system does not currently allow the user to generate multiple instances of the same type of interpolated visualization. For example, the user cannot have more then one isosurface being rendered at a time.

3. Exporting Data

   A feature that was requested by the biologists but never implemented is the funcionality to export the data for a specific region or interpolated visualization.

4. Glider

   Due to time constraints, two requested features involving the glider were not implemented. The first of these was to have the glider's speed be scaled

to a factor of real-time. This would allow the user to get a better idea of how quickly the glider traversed its mission path. The second feature was to have a toggle that would cause the mission defined visualizations to be constructed in the wake of the glider. So, rather then show the entire mission data at once, it would produce the effect of the visualizations being constructed as the glider made the measurements.

5. Terrain Optimizations

   At this point, the only optimzation being done on the terrain mesh is triangle stripping. A number of terrain optimzation algorithms exist that produce view-dependent triangle meshes in order to reduce the number of triangles needed to be rendered. One such algorithm, ROAM, produces optimal adapting triangle meshes in real-time [12].

6. Uncertainty Visualizations

   Since the construction of continuous functions from discrete data samples is inherently error prone, some form of uncertainty visualization would provide a means for the biologists to gauge the accuracy of the interpolated visualizations. Alex Pang's paper on "*Approaches to Uncertainty Visualization*" provides a survey of uncertainty visualization techniques, several of which could be used within the sytem.

7. In-Depth User Study

   Due to time constraints, an in-depth user study did not occur. A *final* system did not get released until approximately two weeks prior to the end of the project, leaving little time to conduct in-depth user testing.

8. Isosurface Transparencty

In their current state, the isosurfaces are being rendered at 100

9. Optimizations

   Storing the $A^{-}1$ matrix required to construct the radial basis functions would reduce the amount of processing time required to build the visualizations.

Appendix A

External Libraries

Several open source libraries were used to assist in the construction of the application. These libraries include Qt, LIBMATIO, and Newmat. Qt, a cross-platform application and UI framework, is used for its windowing system and built-in compatability with openGL. LIBMATIO, a MATLAB library, is used to parse the MATLAB version 5 MAT-files containing the formatted mission data. Newmat, a matrix library, is used during the construction of the radial basis functions used in scattered data interpolation.

A.1 Qt Version 4.5.3

Qt is a cross-platform application and UI framework [36]. It is a comprehensive C++ framework for rapid development of cross-platform GUI applications.

A.2 LIBMATIO Version 1.33

The data collected by the glider is converted to a standard format by the Rutgers University COOL Center operating out of New Jersey. The formatted data is exported as a MATLAB Version 5 MAT-file [40].

In MATLAB Version 5, a MAT-file is made up of a 128-byte header followed by one or more data elements. Each data element is composed of an 8-byte tag followed by the data in the element. The tag specifies the number of bytes in the data element and how these bytes should be interpreted; that is, should the bytes be read as 16-bit values, 32-bit values, floating point values or some other data type. By using tags, the Version 5 MAT-file format provides quick access to individual data elements within a MAT-file. You can move through a MAT-file by finding a tag and then skipping ahead the specified number of bytes until the next tag [24]. Figure A.1 provides a graphical representation of the mat-file format.

LIBMATIO Version 1.33 is an open source library that provides an API for parsing MATLAB Version 5 MAT-files. It provides all of the necessary data structures and functions to extract the formatted mission data from the Mat-file generated by the COOL Center citeLIBMATIO.

A.3 Newmat

Newmat is a C++ library that is intended for scientists and engineers who need to manipulate a variety of types of matrices using standard matrix operations. Emphasis is on the kind of operations needed in statistical calculations such as least squares, linear equation solve and eigenvalues [31].

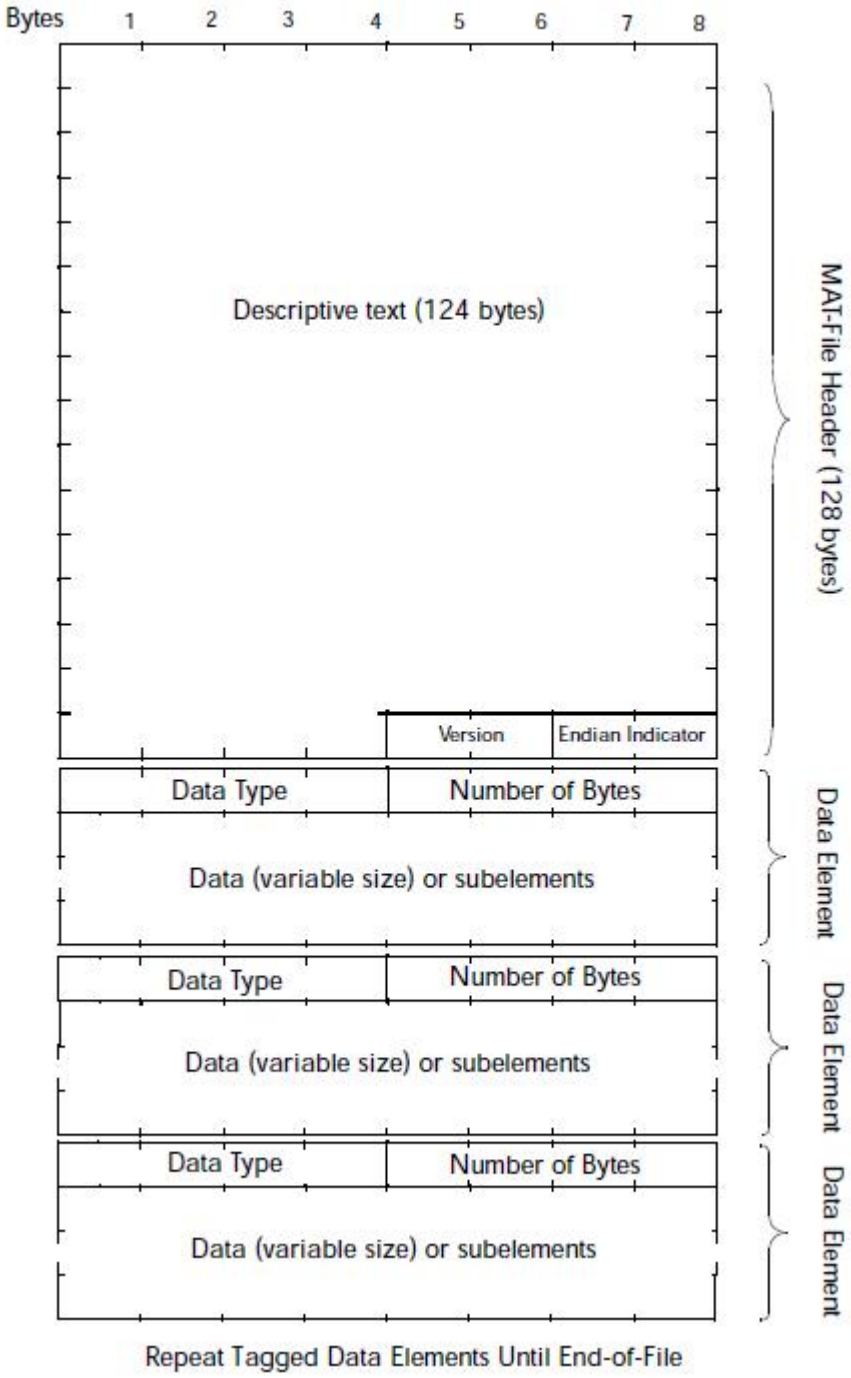

Figure A.1: This figure shows the graphical representation of the MATLAB Version 5 mat-file format.

Newmat is utilized during the construction of the radial basis functions used for scattered data interpolation. Construction of the radial basis functions require that a system of N linear equations be solved to generate values for the weight and polynomial coefficients. Specifically, it requires the inversion and multiplication of large matrices.

Bibliography

[1] Norus workshop, November 2009.

[2] Requirements gathering session with mark moline, phd, May 2010.

[3] T. Akenine-Moller and E. Haines. *Real-Time Rendering*. A K Peters, LTD., second edition edition, 2002.

[4] P. Alfeld. Scattered data interpolation in three or more variables. In *Mathematical Methods in Computer Aided Geometric Design*, pages 1–33. Academic Press, 1989.

[5] J. A. Baerentzen and K. Gebal. Shape reconstruction using compactly supported radial basis functions. Using compactly supported RBFs to reconstruct surfaces.

[6] R. Bartles. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, Inc., 1987.

[7] J. G. Bellingham and M. Godin. Exploring ocean data. *SIGMOD Rec.*, 37(2):78–82, 2008.

[8] Bonzai software water tutorial. http://www.bonzaisoftware.com/water_tut.html.

[9] D. P. Brutzman. *A virtual world for an autonomous underwater vehicle*. PhD thesis, Naval Postrgraduate School, December 1994.

[10] P. Chapman, D. Wills, P. Stevens, and G. Brookes. Whole field modelling (case study): effective real-time and post-survey visualization of underwater pipelines. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 445–448, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[11] P. Chapman, D. Wills, P. Stevens, and G. Brookes. Real-time visualization of the clear-up of a former u.s. naval base. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 505–508, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[12] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. Roaming terrain: real-time optimally adapting meshes. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 81–88, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[13] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide*. Fourth edition edition.

[14] R. Franke. Scattered data interpolation: Tests of some method. *Mathematics of Computation*, 38(157):181–200, 1982.

[15] R. Franke and G. Nielson. Smooth interpolations of large ata sets of scattered data. *International Journal for Numerical Methids in Engineering*, 15:1691–1704, 1980.

[16] K. Grochow. Cove: a visual environment for multidisciplinary science collaboration. In *GROUP '09: Proceedings of the ACM 2009 international*

conference on Supporting group work, pages 377–378, New York, NY, USA, 2009. ACM.

[17] J. C. Hart. Some notes on radial basis functions and thin plate splines. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 40, New York, NY, USA, 2005. ACM.

[18] P. S. Heckbert. Survey of texture mapping. In *IEEE Computer Graphics and Applications*, pages 56–67, November 1986.

[19] B. Humphrey. Realistic water using bump mapping and refraction. http://www.gametutorials.com/Articles/RealisticWater.pdf.

[20] M. A. Iyer, L. T. Watson, and M. W. Berry. Sheppack: a fortran 95 package for interpolation using the modified shepard algorithm. In *ACM-SE 44: Proceedings of the 44th annual Southeast regional conference*, pages 476–481, New York, NY, USA, 2006. ACM.

[21] G. Johansson and H. Carr. Accelerating marching cubes with graphics hardware. In *CASCON '06: Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, page 39, New York, NY, USA, 2006. ACM.

[22] G. G. L. L. L. Schumaker and C. K. Chui. Fitting surfaces to scattered data. In *Approximation Theory II*, pages 203–268, New York, NY, USA, 1976. Academic Press.

[23] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM.

[24] Mat-file format. http://maxwell.me.gu.edu.au/spl/matlab-page/matfile_format.pdf.

[25] S. V. Matveyev. Approximation of isosurface in the marching cube: ambiguity problem. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 288–292, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[26] M. P. McCann. Creating 3d oceanographic data visualizations for the web. In *Web3D '02: Proceedings of the seventh international conference on 3D Web technology*, pages 179–184, New York, NY, USA, 2002. ACM.

[27] M. P. McCann. Using geovrml for 3d oceanographic data visualizations. In *Web3D '04: Proceedings of the ninth international conference on 3D Web technology*, pages 15–21, New York, NY, USA, 2004. ACM.

[28] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 281–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[29] F. J. Narcowich. Notes on scattered-data radial-function interpolation. Notes on using radial basis function for scattered data interpolation.

[30] National geophysical data center. http://www.ngdc.noaa.gov.

[31] Newmat c++ matrix library. http://www.robertnz.net/nm_intro.htm.

[32] G. Nielson. Scattered data modeling. In *IEEE Computer Graphics and Applications*, January 1993.

[33] G. M. Nielson. Dual marching cubes. In *VIS '04: Proceedings of the con-*

ference on Visualization '04, pages 489–496, Washington, DC, USA, 2004. IEEE Computer Society.

[34] G. M. Nielson, A. Huang, and S. Sylvester. Approximating normals for marching cubes applied to locally supported isosurfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 459–466, Washington, DC, USA, 2002. IEEE Computer Society.

[35] Opengl. http://www.opengl.org.

[36] Qt. http://qt.nokia.com.

[37] J. R. R. R. E. Barnhill. Representation and approximation of surfaces. In *Mathematical Software III*, pages 69–120, New York, NY, USA, 1977. Ed. Academic Press.

[38] G. A. Ramos and W. Enright. Interpolation of surfaces vver scattered data. An investigation into differential equation interpolants.

[39] Riemer's xna tutorials. http://www.riemers.net/.

[40] Ru-cool slocum glider datasets. http://marine.rutgers.edu/ kerfoot/slocum/data/readme/.

[41] R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill. Octree-based decimation of marching cubes surfaces. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 335–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[42] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *ACM '68: Proceedings of the 1968 23rd ACM national conference*, pages 517–524, New York, NY, USA, 1968. ACM.

[43] G. Turk and J. F. O'brien. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.*, 21(4):855–873, 2002.

[44] The visualization toolkit. http://www.vtk.org.