12-5-2007

# Volumetric Visualization Of NEXRAD Level II Doppler Weather Data From Multiple Sites

Yi Ru
*Purdue University - Main Campus*, yru@purdue.edu

---

Ru, Yi, "Volumetric Visualization Of NEXRAD Level II Doppler Weather Data From Multiple Sites" (2007). *College of Technology Masters Theses.* Paper 10.
http://docs.lib.purdue.edu/techmasters/10

# VOLUMETRIC VISUALIZATION OF NEXRAD LEVEL II DOPPLER WEATHER DATA FROM MULTIPLE SITES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Yi Ru

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2007

Purdue University

West Lafayette, Indiana

# ACKNOWLEDGMENTS

I would like to thank my thesis committee – Dr. Gary R. Bertoline, Dr. Bedrich Benes, and Dr. Laura Arns for their enduring support, guidance and contributions. I am especially grateful to Dr. Bertoline for his trust, encouragement, and kindness during my graduate studies, and Dr. Benes for his long-term collaboration, attentive help, and additions to my work. Many thanks are given to Dr. Laura Arns for her advice and help from different perspectives and for the training on Virtual Reality instruments in the Envision Center for Data Perceptualization.

I would like also to express my gratitude to Dr. Carol X. Song and Lan Zhao in the Rosen Center for Advanced Computing (RCAC) Group for their kind help and support. I would like to extend my thanks to Dr. David Ebert for imparting and sharing his abundant knowledge on volumetric rendering techniques in his visualization class. Thanks to Dr. Mathew Huber for his invaluable feedbacks and suggestions.

I am also thankful to Leif L. Delgass for his quick responses and sharing his knowledge and experience with me. Thanks to all of my classmates and friends in the Envision Center and RCAC group for all their suggestions and encouragement. Thanks for the financial support by Envision Center and RCAC group.

Finally I would like to thank my husband, Chen for his unconditional care, tireless support and encouragement, and my parents and my sister for their endless love.

# TABLE OF CONTENTS

Page

Page

# LIST OF TABLES

Table Page

# LIST OF FIGURES

# ABSTRACT

Ru, Yi. M.S., Purdue University, December, 2007. Volumetric Visualization of NEXRAD Level II Doppler Weather Data from Multiple Sites. Major Professor: Gary R. Bertoline.

Weather visualization is critical for operational forecasters and weather researchers to analyze, monitor, and predict severe weather events such as storms, tornadoes, and hurricanes. Usually measured weather datasets such as NEXRAD Level II radar data contain some errors and have temporal aliasing issues, so that the raw data cannot represent the true meteorological patterns. Existing weather visualization packages usually ignore these problems, thus they fail to unveil some detailed information and cannot meet the needs of professional meteorologists. Most displays generated by those packages are limited to 2D images, 3D point clouds, or iso-surfaces. Those displays are not able to accurately represent the details of the data. The limitation of visualizing data from a single radar site at one single time step in most packages blocks desirable information in the datasets.

We developed an efficient and accurate visualization tool capable of displaying the long-track 3D NEXRAD Level II volumetric weather data from multiple sites. The data were captured and integrated from three sites (KEAX, KILX and KSLX), and rendered by using the hardware-accelerated volume-rendering technique. Customized transfer functions are also provided for users to map data values to optical properties such as color and opacity to create different views. We also implemented a compression algorithm to compress 3D textures in order to largely reduce the consumption of hard disk space and to enhance the rendering speed without degrading image quality. An example study shows the tracking of a 24-hour supercell storm observed on March 12, 2006, in the Midwest of the United States. Images and animations from our implementation coupled with results from a set of experiments demonstrate that our methods and approaches are fast and robust, making them suitable for processing and rendering huge amounts of Doppler Level II weather data from multiple stations.

# CHAPTER 1. INTRODUCTION

The occurrences of severe weather every year have caused many injuries and fatalities as well as severe damage to personal and public properties. Statistics [DisasterReport] show that on average tornadoes cause 1,500 injuries and 80 fatalities per year, thunderstorms result in 2,000 injuries and 150 deaths annually, and hurricanes account for 60 injuries and 17 deaths each year in the United States. Figure 1.1 shows a tornado and its damage in central Oklahoma on May 3, 1999 (on top row), and Hurricane Katrina and its aftermath on August 29, 2005 (on bottom row).

Weather forecasting and severe weather prediction are usually used to provide opportune warnings to the public that people can protect their lives and properties, change economic operations, and plan daily activities ahead of time. To accurately predict the weather, forecasters and researchers need to analyze data (precipitation, pressure, turbulence, wind speed, etc.) to understand how weather would change on our planet, forming and producing severe storms. An accurate and efficient weather data visualization tool or system can significantly help forecasters to gain fresh insights on atmospheric conditions and formations, hence to enhance the accuracy of their forecasting simulation models.

Figure 1.1. Examples of Tornado and Hurricane and the Aftermaths (Images Courtesy of National Oceanic & Atmospheric Administration (NOAA))

## 1.1. NEXRAD Level II Doppler Radar Data

Today the nationwide meteorological networks such as Doppler radar and satellite-borne sensors provide weather forecasters and researchers with more observational data than ever before. Particularly, the National Weather Surveillance 1988 Doppler Radar Network (WSR-88D, shown in Figure 1.2) - also known as the Next Generation Weather radar system (NEXRAD) - provides high spatial and temporal resolution information within 3D volumes on a continuous basis. The network comprises over 150 radar sites (Figure 1.3) across the United States and some overseas locations. The raw datasets produced by radars consist of three basic meteorological fields: reflectivity, radial velocity, and spectrum width. Reflectivity is related to the weather phenomena such as clouds and rain, while velocity is germane to wind, and spectrum width is relevant to turbulence. As a primary component of meteorologists' interest, reflectivity is the key point of the study.

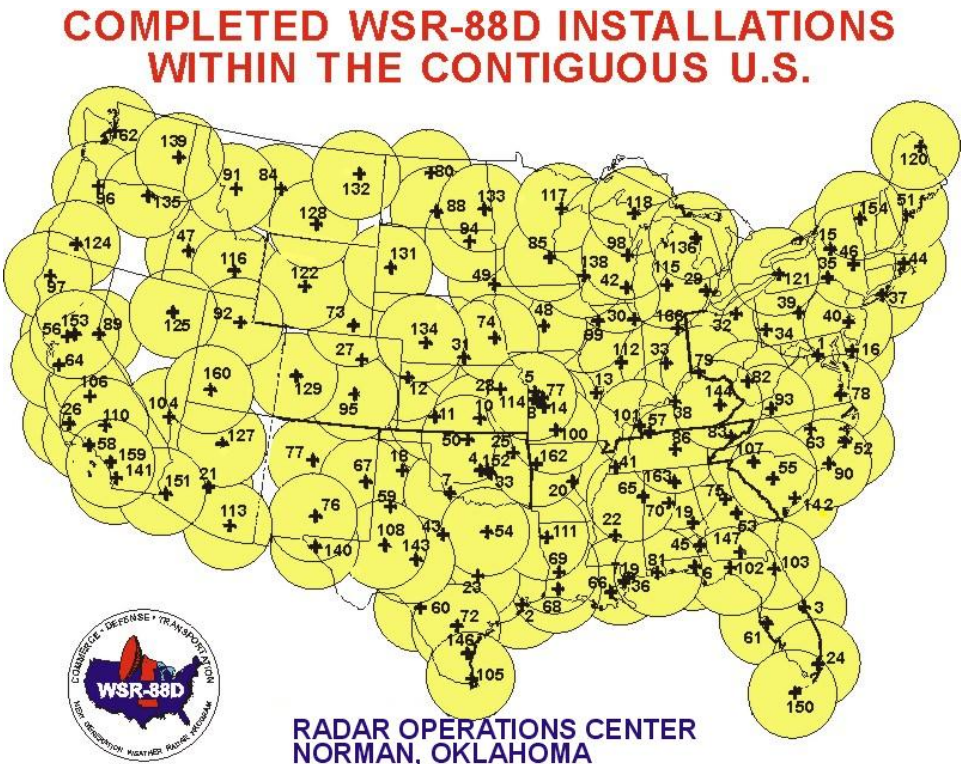

COMPLETED WSR-88D INSTALLATIONS
WITHIN THE CONTIGUOUS U.S.
WSR-88D
COMMERCE • DEFENSE • TRANSPORTATION
NEXT GENERATION WEATHER RADAR PROGRAM
RADAR OPERATIONS CENTER
NORMAN, OKLAHOMA

Figure 1.2. Network of WSR88D (Weather Surveillance Radar, 1988, Doppler) Radars (Image Courtesy of National Oceanic & Atmospheric Administration (NOAA))

Figure 1.3. A Doppler Radar Site at Jackson Kentucky (Image courtesy of NOAA)

## 1.2. Statement of the Problem

Developing such a weather visualization tool involves several problems: (1) The measured weather datasets usually contain some measurement errors. For example, NEXRAD Level II Doppler data is characterized by a large proportion of missing or invalid data points [Djurcilov 1999]. Existing weather visualization packages usually ignore these errors, thereby resulting in the missing of some detailed information and negatively affecting the accuracy of the data representations [Riley 2003]. (2) Radar datasets are captured and produced in 3D, however, most displays generated by existing visualization packages are limited to 2D images, 3D point clouds, or iso-surfaces, all of which cannot accurately represent the entire field. Figure 1.4 shows a typical 3D sweep perspective view of reflectivity data in the Integrated Data Viewer (IDV) [IDV], where the scalar field of reflectivity is rendered as 3D point clouds that do not represent the data very well. (3) Displaying multiple Doppler radar datasets and visualization of long-time datasets are very challenging. First, to display data from multiple sites, current approaches usually first render the data from every single site as a 2D image, and then mosaic images from all sites together, producing a less attractive visualization. Second, the radar data are renowned for having temporal aliasing issues.  Because radars at different locations are operated at different paces, resulting in asynchronous data files from different stations, thereby it is problematic to display 3D data of multiple radar sites simultaneously.  Finally, it is challenging to render long sequences of datasets in real time because large amounts of data are involved, leading to a heavy computation load.

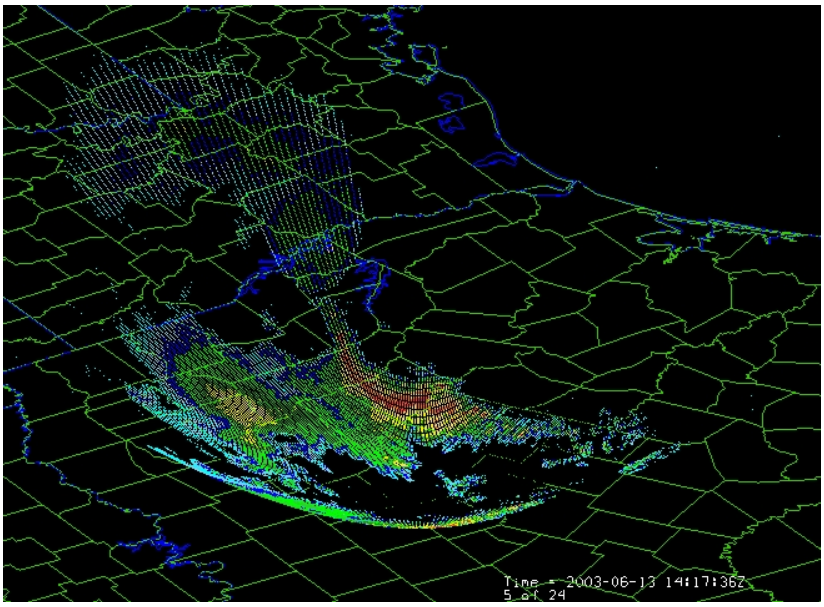


Figure 1.4.  A Typical 3D Radar Sweep Perspective View of Reflectivity Data Rendered as 3D Point Clouds in IDV (Image courtesy of IDV gallery)

In view of these technical issues, the problem of this study is to retrieve and integrate data from the multiple radar sites, and to display sequential 3D weather volumetric data in real time, providing detailed information to users especially for weather researchers.

## 1.3. Purpose of the Study

Currently the Rosen Center for Advanced Computing (RCAC) research group together with the Envision Center for Data Perceptualization at Purdue University is aiming at providing a real-time rendering interface for users to remotely access and to interactively visualize the Doppler Level II weather data from TeraGrid [TeraGrid]. The interface will be widely accessible to users ranging from ordinary people to educational and scientific research groups.

As one of the most significant parts of the research, the goal of this thesis work is the development of a real-time visualization system to render 3D Doppler radar data from multiple sites in real time. Visualization on supercell storms that were observed on March 12, 2006, in the Midwest of the United States is the subject of a case study to verify our approaches. The data were obtained from three Weather Surveillance Doppler radars – KEAX located in Kansas City, Missouri, KILX located in Lincoln, Illinois, and KSLX in Saint Louis, Missouri.

## 1.4. Delimitations

The study focuses on NEXRAD Level II Doppler radar data processing and rendering techniques. It does not handle datasets from other sources such as geostationary satellites. The field of interest is reflectivity. The observation is limited to visualization on data from three radar stations. The study does not process and display other fields such as spectrum width and velocity. Moreover, the research emphasizes hardware-accelerated volumetric rendering techniques rather than other rendering techniques such as iso-surfacing.

## 1.5. Summary

In this thesis, we introduce an interactive rendering system on visualizing Doppler weather data from multiple sites. The system integrates data from multiple sites into an entire volume at every timestamp and stores them as a 3D texture, which is a great advantage for creating weather animations where large amounts of data are involved.  Furthermore, we employ the hardware-accelerated texture-based volume rendering technique to speed up rendering processes.  Results from our implementations show that our processing and rendering approaches are fast and efficient.  Another contribution of this work is the improvement in storing and reading 3D textures.  Through compression and decompression procedures for 3D textures, required disk space for the texture storage is highly reduced, and the rendering performance is also improved, while keeping the quality of images unchanged.

The thesis is organized as follows: It starts with an introduction to motivations, problems, challenges, and objectives of this project, followed by detailed discussions of previous approaches, existing methods, and related work in Chapter 2.  Chapter 3 thoroughly describes schemes, methods, procedures, and implementations of our system.  Rendering images, experimental results, and analyses are presented in Chapter 4.  Chapter 5 concludes the contributions and limitations of our work, and suggests several possible future improvements and extensions to this work.

# CHAPTER 2. RELATED WORK

It is difficult and challenging to process huge amounts of Doppler radar datasets and to produce an accurate and efficient visualization. The literature review summarizes previous work from four aspects: weather visualization systems, volume rendering techniques, Doppler data processing techniques, and data compression algorithms.

## 2.1. Weather Visualization System

For a long time, researchers have been working on developing weather visualization systems to help weather forecasters to better understand the data in order to provide timely and accurate weather forecasts. The IDV package developed by Unidata, funded primarily by the National Science Foundation (NSF) is an open source Java-based software framework for analyzing and visualizing geo-science data. The software can read different data formats, including satellite imagery, gridded data (for example, numerical weather prediction model output), surface observations, balloon soundings, the National Weather Service (NWS) WSR-88D Level II and Level III radar data, and NOAA National Profiler Network data, and display them in 2D or 3D fashions. It can also allow users to change the color tables or earth maps with known formats. The National Climate Data Center (NCDC) also provides some visualization tools to display Level II radar data. The NOAA NCDC Java NEXRAD Viewer and Data Exporter [JavaNEXRADViewer] are specific to load Level II, Level III, and Stage III radar data into an Open GIS [OGC]-compliant environment. As part of the Collaborative Radar Acquisition Field Test (CRAFT) [Kelleher 2007] project, the Interactive Radar Analysis System (IRAS) [Priegnitz 1995] can read and display

Level II radar data via the Internet in real time. All of these application Program Interfaces (APIs) are written entirely in the Java programming language, and both IDV and the NEXRAD Java Viewer are launched via Java Web Start.

In 1990, Hibbard et al. presented the first version of Vis5D system [Hibbard 1990] for interactive visualization of large grid datasets from numerical weather models. In Vis5D, the data are constructed in the forms of 5D rectangular grid of points. The five dimensions include three spatial dimensions, time dimension, and one dimension from the physical variables. For example, the three spatial dimensions of meteorological data are altitude, latitude, and longitude, and the variable dimension can be one of temperature, pressure, moisture, or three wind vector components. The system enables users to interactively visualize iso-surfaces, contour-line slices, colored slices, and volume renditions of the data in real time. Moreover, Vis5D supports comparison of multiple datasets. Additional datasets can be imported to the system at any time where the new datasets can either be overlaid in the current display or displayed as a series of 3D spreadsheets. Other features of the system include wind trajectory tracing and text annotations. Extensions of the system include an enhanced version called Vis5D+ [Vis5D+] and the Cave5D system [CAVE5D] that runs in a CAVE-like virtual reality environment [CAVE].

Later, Hibbard proposed a Java class component library - VisAD [Hibbard 1998]. The main feature of VisAD is to provide geographically distributed users with interactive and collaborative visualizations of a shared set of numerical data and computations. Using VisAD, the four major components - data, display, user's interface, and computational objects - can be linked together from different locations on the network. The tool includes a mathematical model, processing numerical data with a number of data formats such as netCDF, HDF-5, HDF-EOS, Vis5D and JPEG; and a display model, managing interactive visualization such as 3D viewpoints, animations, iso-surfaces and scalar maps.

Researchers at Georgia Institute of Technology developed a system [Jiang 2001] with real-time acquisition, organization, and visualization of atmospheric datasets in a geospatial environment. In the system, Doppler data, high-resolution terrain, and associated data such as buildings and maps are merged and displayed together in an existing real-time environment called VGIS [Lindstrom 1996]. The system can operate on a personal computer, and the display is viewed on a monitor or large-screen projection.

Focusing on local weather forecasting, IBM Thomas J. Watson Research center developed a meso-scale numerical prediction and visualization system called “Deep Thunder” [Treinish 2004] which is a complementary tool to the NWS. The deep thunder project was developed on the basis of previous research projects done at IBM, involving the use of the IBM Open Visualization Data Explorer (OpenDX) [OpenDX]. It was first used during the 1996 Atlanta Olympic Games and has been improved by adding more features and functions since then. For example, the idea of task-specific visualization design [Treinish 1998] was proposed to match diverse users’ goals. Usually in weather forecasting, some applications are built on a common framework to avoid the re-design of interfaces and content elements. However, the limitations of the framework, such as the lack of focus on the interface, may result in more troubles and challenges, precluding a novice from comfortably operating the system in a limited time period. Instead of using a general-purpose tool directly, Treinish suggested that an appropriate design for a specific task could be developed ahead of time. Using these design elements, specialized interfaces and tools are developed to improve and refine the general framework. Other new technologies include (1) a web-based approach to enable users to remotely access 3D visualizations of operational meso-scale weather models [Treinish 2002] and (2) multi-resolution visualization techniques to provide more detailed and coherent visualization of weather models [Treinish 2000].

Although these powerful tools are being utilized in many weather visualization applications, due to the inherent limitations of these tools, they are not yet able to fully accommodate the needs of some users. For example, most APIs are written in Java language, which makes the data retrieving and processing comparatively slow. They usually cannot represent datasets with multi-fields at the same time. An example is the Vis5D system, which can only handle one variable dimension at a time. Additionally, most representations are limited to 2D images or 3D point clouds. Many tools only can represent the dataset from one single site or simply mosaice 2D images for the datasets from multiple sites. Hence, the visual representations generated by most tools are not accurate. Another issue is that existing systems only handle the data generated at one time step. They cannot visualize long-time sequential datasets.

In order to provide accurate rendering and multi-field visualization, Riley et al. presented a new visualization system [Riley 2003] [Riley 2004] to produce a realistic representation based on the particles’ optical properties such as extinction and scattering. The system can visualize volumetric weather data with multiple fields, including cloud water, ice, rain, snow, and graupel. Song et al. proposed an integrated atmospheric visual analysis and exploration system [Song 2006]. The system supports a variety of rendering techniques, including physics-based atmospheric rendering, illustrative rendering, and particle and glyph rendering to provide users with flexible and efficient data analysis tools. Schpok et al. [Schpok 2003] described a software package called “Swell”, a multi-level, interactive, volumetric cloud modeling and animation system using an interactive cloud modeling tool. The system has three major components: a high-level modeling and animation system, a renderer, and an interface. Volumetric implicit functions [Ebert 1997] are used for generating high-level clouds and volumetric procedures based on noise and turbulence simulations for low-level clouds detail. The rendering of clouds is based on a modified slice-based volume rendering scheme.

However, despite the improvement from the previous applications, these systems only focused on data from a single radar site, and therefore fail to provide users with a large view of the weather display.

Some of the new techniques are under development to provide 3D views of data from multiple sites. In 2005, Ueng et al. proposed a system to specifically visualize the Doppler radar data through a three-pass technique [Ueng 2005]. In the first pass, the data is re-sampled and filtered to create a multiple-resolution data structure. In the second pass, cloud velocities are computed, and reflectivity data is interpolated by calculating gradients and intensities of each voxel. In the third stage, the reflectivity and the velocity are rendered by using the splatting volume-rendering technique [Westover 1990]. The authors also provided a method to synchronize the data from multiple radar sites by means of the calculated velocity from each site. The drawback of this system is that it ignores the effect of rain and evaporation by dropping the vertical component of the velocity. A new technique called the Vortex Objective Radar Tracking and Circulation (VORTRAC) [Harasti 2006] was developed and is currently tested for analyzing hurricane strength at the National Hurricane Center (NHC) in Miami, Florida. It is typically designed for tracking the central pressure and the radius of maximum wind of land-falling tropical cyclones. By using a series of algorithms, data from Doppler radars was transformed into a detailed 3D view of an approaching hurricane every six minutes. VORTRAC is a fusion of several single-Doppler radar data quality control and wind analysis methods.

## 2.2. Volume Rendering Techniques in Weather Visualization

Rendering is a significant part in a visualization system. To make the scene more realistic, most researchers adopted the method of volume rendering

[Drebin 1988] - a method of representing, displaying, and manipulating objects in the forms of sampled data in three or more dimensions. There is a huge pool of literature on volume rendering techniques [Lichtenbelt 1998] [Engel 2006]. In this section, we primarily focus on applied volume-rendering techniques in weather visualization.

Generally, in weather visualization, as in many other fields of visualization, volumes are most often represented as a grid composed of a group of volumetric elements called voxels. To synthesize an image of a volume, four general approaches – image-ordered ray casting [Levoy 1988], object-ordered splatting [Westover 1990], shear-warping [Lacroute 1994], and texture-based volume rendering [Wilson 1994] are commonly used to render the scene from 3D volumetric data.

However, to generate photo-realistic images, more physical properties of the cloud material, such as light absorption and scattering, have to be considered. Hence, two major tasks are usually taken into account in a weather data visualization system: integrating the effects of optical properties along the path through the cloud volume, and incorporating the complex light scattering with the medium. To complete these two tasks, Kajiya et al. [Kajiya 1984] used ray tracing methods, dealing with both single and multiple scattering. Nishita presented global illumination approximation techniques, accounting for multiple anisotropic scattering and skylight on the cloud color [Nishita 1996]. Those methods are important for realistic rendering but are very time-consuming. There have been many efforts to approximate the physical properties of clouds while reducing the amount of computations. One of the possible approaches is to use 3D textures to render the volume density to accelerate the rendering process. Dobashi [Dobashi 2000] presented a simple and computationally inexpensive method for near real-time animation of clouds. The rendering method introduced in the paper is based on the splatting algorithm that can quickly render clouds as billboards, and calculate shadows and shafts of light through the clouds by using graphics hardware. Harris [Harris 2001] [Harris 2002] presented a method for realistic real-time rendering of constant-shape clouds for games. The rendering approach is based on the splatting representation of particles and the shading method described by Dobashi et al. [Dobashi 2000]. Harris extended Dobashi's model by simplifying light scattering as multiple forward scattering and anisotropic first-order scattering. This approach not only retains the realistic rendering effects, but also makes the rendering speed much faster than Dobashi's. Harris also presented a physically-based, visually-realistic interactive cloud simulation system [Harris 2003]. The clouds are represented using a "flat" 3D texture – a 2D texture that contains the tiled slices of a 3D volume. To render complex scenes containing gaseous phenomena such as clouds, fog and smoke, Ebert et al. proposed a fast and efficient method [Ebert 1990], combining scanline A-buffer techniques to render surface-modeled objects with volume-rendering techniques to render volumetric modeled objects. To speed up rendering scenes with atmospheric data, Jang et al. proposed a splatting technique, using volume rendering integrated with a level of detail [Jang 2002]. The level of detail is selected automatically based on the current view of the data. To achieve this, the authors used an adaptive tree structure that preserves the details of the non-uniform data at the lowest level of the tree. The method is applied on the NEXRAD Doppler data to produce interactive visualizations in real time.

Other research using the splatting volume-rendering method include: (1) an interactive, multi-level, cloud modeling and rendering system [Rana 2004], where the system adopts a two-level modeling approach - the high-level clouds are created from cubes, and the low-level clouds are generated from 2D textured billboards, and (2) a cloud system [Wang 2004] that can simulate a number of different cloud types that are manually designed and fine-tuned by artists in the 3ds Max environment – a 3D graphic application developed by AutoDesk Media and Entertainment. Recent advancements include a hardware-accelerated technique [Engel 2001] employed in weather visualization systems, and a photorealistic-rendering technique in weather visualization, which provides weather forecasters and researchers with more detailed representations of the data [Kniss 2003].

## 2.3. Doppler Data Processing

Recently Doppler NEXRAD Level II radar data are widely used in weather forecasts and analysis such as tracking formation and the path of severe storms. To make full use of Doppler data, weather researchers did numerous work on processing precipitation reflectivity and radial velocity datasets. For example, Chen et al. presented local least square and regularization frameworks [Chen 2001A] [Chen 2001B] for computing 3D velocity from 3D radial velocity. Qiu et al. [Qiu 2001] extended a 2D tracking algorithm to 3D, and applied it to 3D Doppler reflectivity data. Zhou et al. [Zhou 2005] proposed a method to calculate wind field velocity from the data provided by a single Doppler radar site. This approach is mainly based on a simplified assumption for the motion of the wind field. Furthermore, since a full 3D wind field can offer a better understanding of the atmospheric data for meteorologists, Laroche et al. proposed a variational method to construct a 3D wind field [Laroche 1994]. Gao et al. [Gao 2001] retrieved the 3D wind field from a single Doppler data based on the improvement of a simple adjoint method [Qiu 1992]. Djurcilov et al. discussed techniques for visualizing datasets with large number of missing values [Djurcilov 1999]. None of these techniques, however, was developed for the purpose of volume-rendering the data.

## 2.4. Data Compression Algorithm

Nowadays data compression is widely used in Computer Science and Information Technology because of its desirable features for data storage and data communication. With data compression algorithms, the data files are shrunk down to smaller sizes, which could save expensive resources such as disk space and transmission bandwidth.

There are two typical type of data compression: lossy and lossless [Lynch 1985] [Storer 1988]. Namely, the lossless compression algorithms, also referred as redundancy reduction algorithms, can reconstruct the original data by finding repetitive patterns in the data, encoding and decoding them in an efficient way. Obviously, the algorithms will have very poor performance when the redundant patterns do not exist or are not easily discovered. On the contrary, lossy compression algorithms cause loss of data, which means the data or messages could not be recovered after applying lossy algorithms. However, the algorithms are still applied when some loss is acceptable or when lossless compression could not accomplish the compression task.

Lossless data compression algorithms include: run-length encoding (RLE), prediction by partial matching (PPM), Burrows-Wheeler transform (BWT), and Huffman entropy encoding etc. As one of the simplest lossless compression algorithms, RLE algorithm is very suitable for messages or files that contain large runs of consecutive identical data values. The algorithm simply replaces repeated values with the value and the length of the run.

Lossy data compression algorithms include: fractal transform, wavelet compression, vector quantization and so on. S3 Texture compression (S3TC, DXTn) belongs to the lossy data comression group, which is originally developed by S3 Graphics. Ltd [Iourcha 1997]. It consists of a group of related image compression algorithms. Its desirable features including fixed-rate data compression together with the single memory access make it very suitable for compressing textures in image processing applications. Volume texture compression (VTC) algorithm is simiar to S3TC compression, but optimized for processing 3D textures, which could save some bus bandwidth and memory at the cost of image quality.

However, data compression can cause extra work. Specifically, compressed messages or files must be decompressed before they are processed or viewed. Therefore, the design of data compression involves trade-offs among various factors: the degree of compression, the required quality of data, and the time and spaces required to process the data.

## 2.5. Conclusion

Current weather visualization systems enable meteorologists to interactively visualize iso-surfaces, contour-line slices, color slices, and volume renderings of the data in real time by a large variety of rendering techniques such as volume rendering, illustrative rendering, glyph rendering and photorealistic rendering. Based on previous studies, it is believed that future visualization systems are likely to be improved by adding advanced transfer functions, providing flexible ground mapping projection, and enhancing properties of multi-fields. Future breakthroughs of weather visualization also rely largely on the technical advancement of hardware. Visualization of multiple Doppler radar datasets is capable of providing a larger view with higher resolution in large-region weather studies. Despite recent technological improvements, visualization of sequential long-time datasets still remains largely unsolved.

# CHAPTER 3. PROCEDURES

The NEXRAD Level II Doppler data visualization system discussed in this thesis consists of two major components: data processing and rendering that are aligned in a subsequent order. Each component contains a number of inter-connected computational operations, which are presented in greater detail in the following sections. A synopsis of the entire setup and data flow is given in Figure 3.1.

## 3.1. Data Processing

The raw weather datasets used in this study are referred to as NEXRAD Level II Doppler data. The data, captured by a given radar system, are typically compressed using some compression algorithms and are stored as a series of binary files. In addition, the radar data are captured by radars at different time, at different locations, and stored in their respective local spherical coordinate systems. It would be beneficial to process the data before the data can be directed into our system. Therefore, in the data processing stage, the compressed radar data comprised of sample coordinates and sample values are first retrieved from the data files captured in approximate same time. Then local spherical coordinates of the samples are converted to global geographic coordinates, and the sample values are aligned by linear interpolation from multiple sites according to the designated time. Finally, the values are re-sampled into the rectilinear grid structure that is useful for volumetric rendering, and the formed 3D volumetric data are compressed and stored on the hard disk.

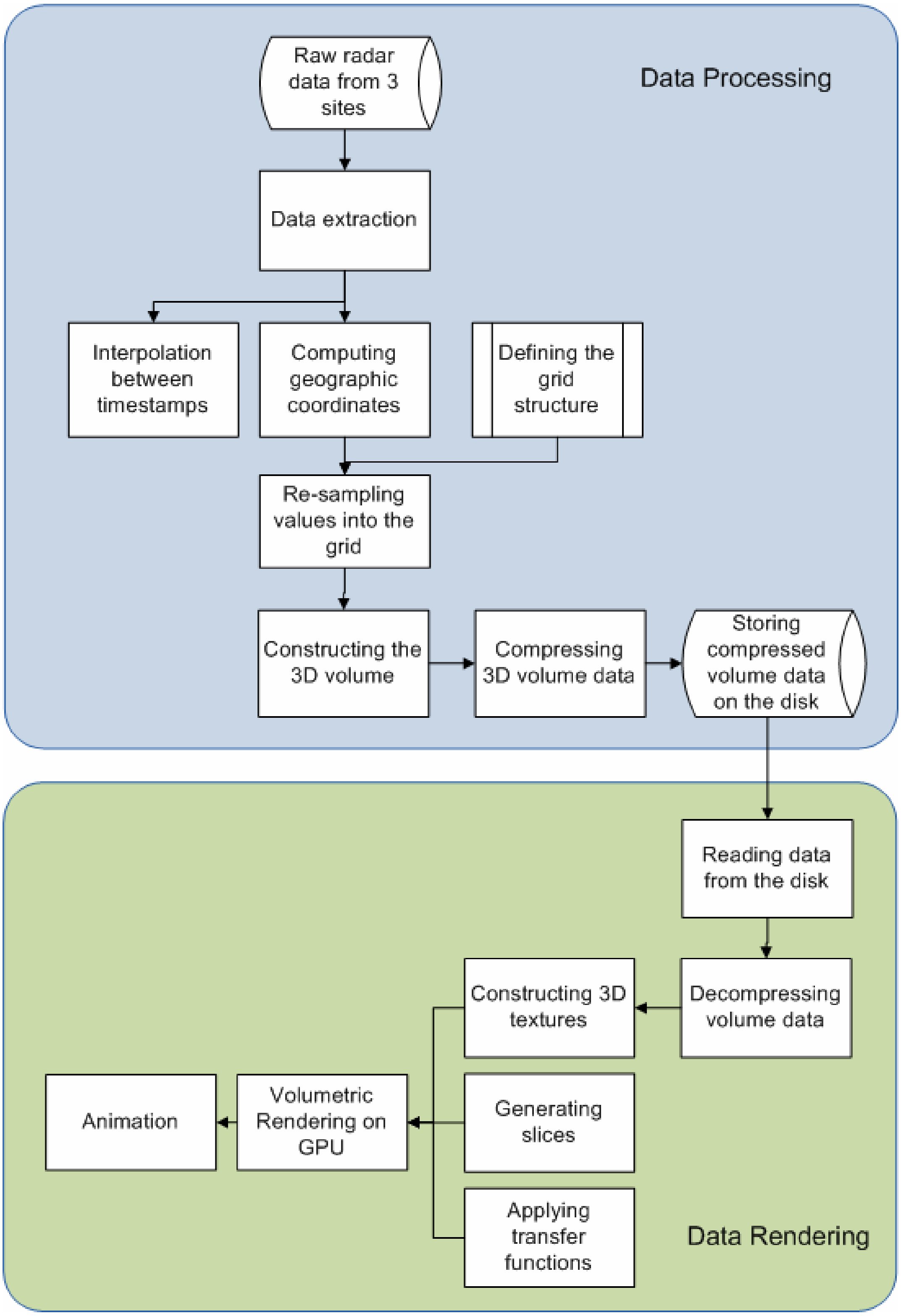

Raw radar data from 3 sites
Data Processing
Data extraction
Interpolation between timestamps
Computing geographic coordinates
Defining the grid structure
Re-sampling values into the grid
Constructing the 3D volume
Compressing 3D volume data
Storing compressed volume data on the disk
Reading data from the disk
Constructing 3D textures
Decompressing volume data
Animation
Volumetric Rendering on GPU
Generating slices
Applying transfer functions
Data Rendering

Figure 3.1. Overview of the Visualization System

### 3.1.1. Radar Data Extraction

The NEXRAD Level II Doppler data are collected as radars go through a programmed set of movements, which involve a continuous rotation over 360° in azimuth and a simultaneous increase in elevation by 1° to 3° per complete sweep [Huber 2007]. The spatial resolution is 1 kilometer for reflectivity and 0.25 kilometer for velocity and spectrum width in range; 1° for all the three fields in azimuth. The radar makes up to 20 azimuthal scans in elevation ranged from 0.5° to 19.5° determined by the Volume Coverage Patterns (VCPs). Usually radars do more sweeps for severe storms and fewer sweeps for clear weather. Figure 3.2 is a graphical representation showing the structure for reflectivity of Doppler data. Velocity and spectrum width components also have similar structures. Radars produce the raw datasets on a continuous basis, making the temporal resolution around 5 to 6 minutes in severe weather.

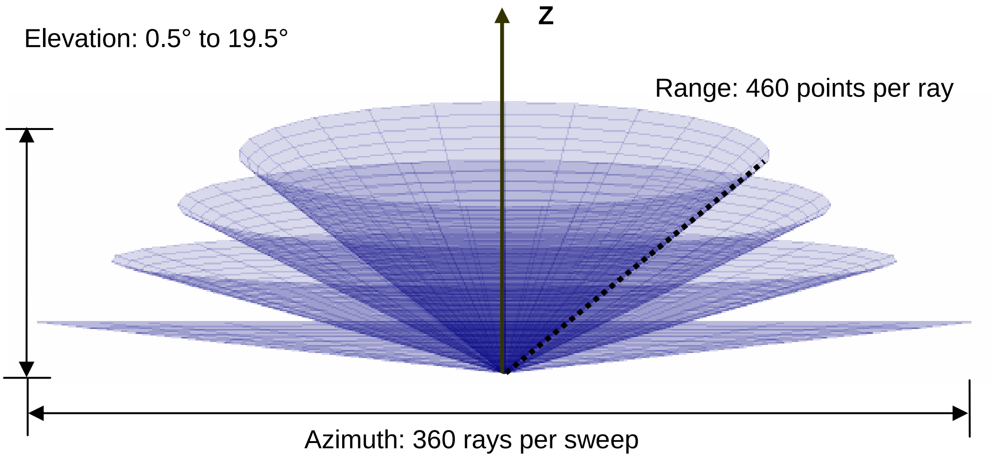


Figure 3.2. 3D Structure of NEXRAD Level II Doppler Radar Data (Reflectivity)

Once raw radar datasets were obtained, they were processed by using the TRMM Radar Software Library (RSL) [TRMM]. The library allows for the retrieval of some components such as reflectivity, from compressed Doppler data files. The retrieved data were stored in the computer's main memory for processing.

### 3.1.2. 3D Rectilinear Grid

To ensure an efficient volume rendering in the rendering stage (Figure 3.1), all data need to be organized and re-sampled in a rectilinear grid structure which contains a collection of cells arranged on a regular lattice [Schroeder 2003]. Since this study involves the raw data from three geographic sites – KILX, KEAX and KLSX, the grid needs to cover all three sites and to store the data in memory in an efficient way. To accomplish this task, a bounding box that contains all data from three sites was generated first. Due to the unique spatial distribution of the radar data, there are more data points distributed in the X-Y plane than in the Z direction (the elevation direction) in the grid. Therefore, the grid (Figure 3.3) is designed to have a non-uniform structure such as 256 by 256 by 128, which provides the grid with 128 layers of X-Y planes, each consisting of a 2D uniform grid of 256 by 256. This structure is also referred as "semi-regular" because though the topology of the grid is regular, the points arranged are partially regular.

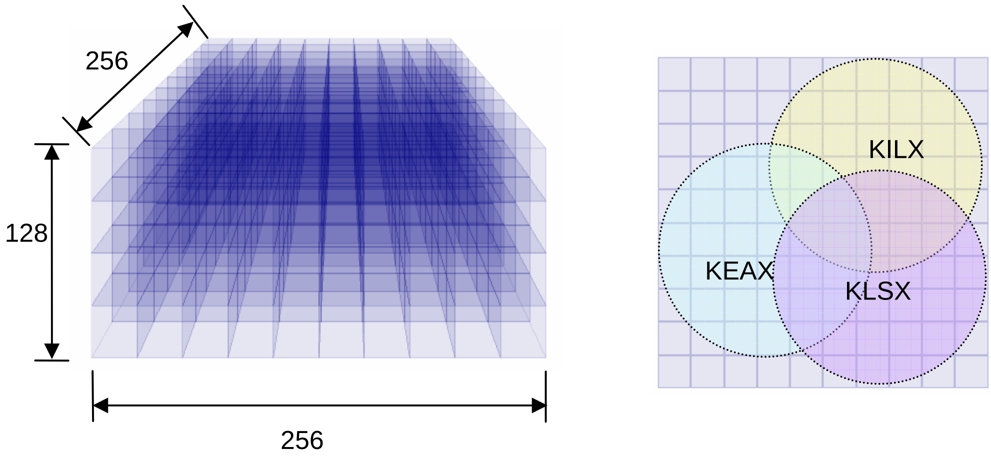


Figure 3.3. 256x256x128 Grid Structure (left) and Bounding Box (right)

### 3.1.3. Conversion from Spherical Coordinates to Geographic Coordinates

The completion of the 3D grid is followed naturally by re-sampling of radar reflectivity data values into it. For each site, the TRMM library stores the data in the local spherical coordinates (azimuth, elevation and range) with the origin placing on the location of each radar station. If only the data from a single site is considered, the values can be directly re-sampled into the grid. However, to integrate the data from the multiple sites, a global coordinate system has to be used.

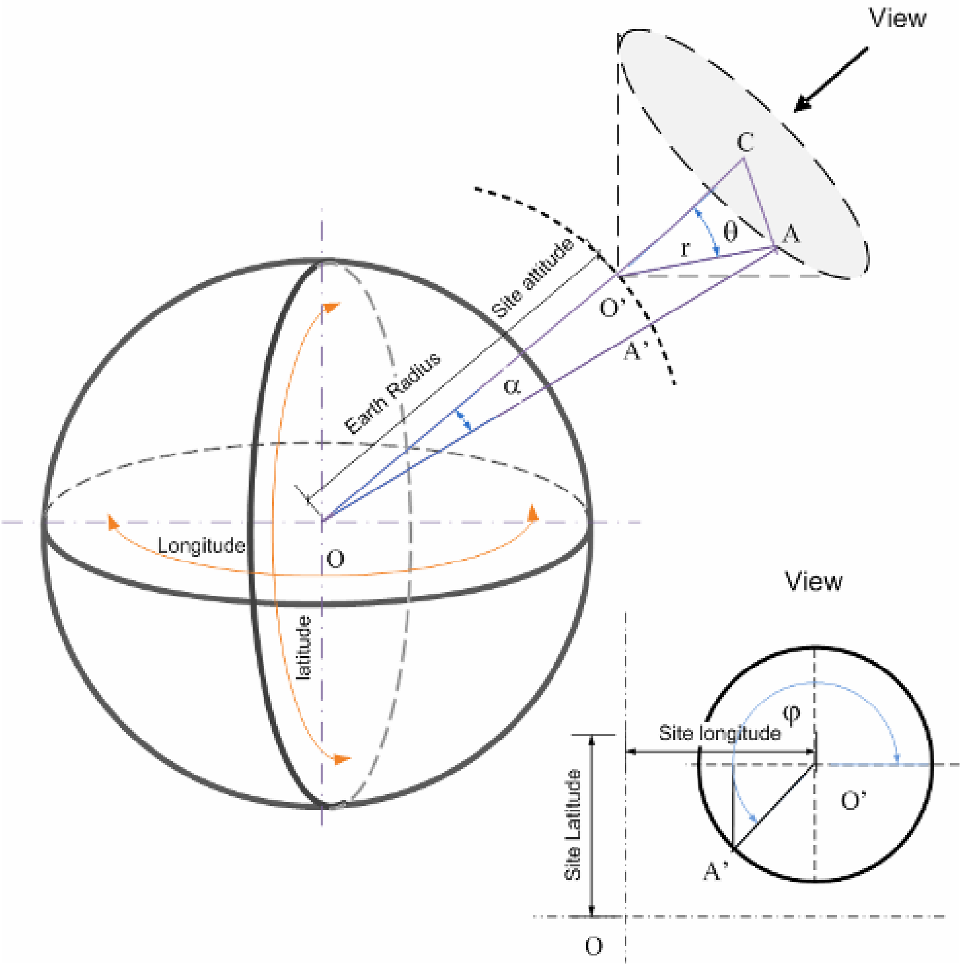


Figure 3.4. Computing Geographic Coordinates of Sample Points

Here, coordinates of each sample point are converted from the local spherical coordinate system to the global geographic coordinate system prior to the re-sampling operation. This conversion is graphically interpreted in Figure 3.4.

Specifically, given the geographic coordinates of the location of a radar site $O'$, denoted as $(lon, lat, alt)$, and the local spherical coordinates of a sample point $A$, denoted as $(r, \theta, \varphi)$, the geographic coordinates $(lon', lat', alt')$ of the sample point $A$ is defined as in equation 3.1:

$$\begin{aligned} lon' &= lon + O'A'\cos\varphi / lonScale; \\ lat' &= lat + O'A'\sin\varphi / latScale; \\ alt' &= OA - R_e, \end{aligned} \qquad \text{Eq. 3.1}$$

where $R_e$ is the approximated earth radius ($\approx 6{,}367{,}450.0\text{m}$), $latScale$ (the scale in latitude) is the width per latitude degree denoted as $R_e \cdot \pi / 180$, and $lonScale$ (the scale in longitude) is the width per longitude degree defined as $R_e \cos(lat) \cdot \pi / 180$.

$OA$ and $O'A'$ are defined in following equations:

$$\begin{aligned} OA &= \sqrt{OC^2 + AC^2} = \sqrt{(OO' + O'C)^2 + AC^2} \\ &= \sqrt{(OO' + r\cos\theta)^2 + (r\sin\theta)^2} \\ &= \sqrt{OO'^2 + r^2 + 2r \cdot \cos\theta \cdot OO'} \\ &= \sqrt{(R_e + alt)^2 + r^2 + 2r \cdot \cos\theta \cdot (R_e + alt)}; \\ O'A' &= OO' \cdot \alpha = (R_e + alt) \cdot \alpha \\ &= (R_e + alt) \cdot \arcsin(AC / OA) \\ &= (R_e + alt) \cdot \arcsin(r\sin\theta / OA). \end{aligned} \qquad \text{Eq. 3.2}$$

The resulting coordinates $(lon', lat', alt')$ are used to find the appropriate cell in the grid where the sample point should be placed. More specifically, the cell indices in x, y, z dimensions are solved by

$$index\_x = \frac{lon' - bbox\_\min X}{bbox\_x};$$
$$index\_y = \frac{lat' - bbox\_\min Y}{bbox\_y};$$
$$index\_z = \frac{alt' - bbox\_\min Z}{bbox\_z}.$$
Eq. 3.3

$bbox\_x$ , $bbox\_y$ , $bbox\_z$ represent the length of the side of the bounding box respectively. $bbox\_\min X$ , $bbox\_\min Y$ , $bbox\_\min Z$ denote the minimal values of the bounding box.

### 3.1.4. Integrating Data from Multiple Sites

From the work in previous sections, we located the cell in the grid which the current sample point should go to. The next step is to fill the reflectivity value at the sample point into the cell, where integrating data from multiple radar sites are coming onto the stage. Obviously and undeniably the large coverage of the Doppler radar network in the United States brings a lot of benefits, in the meantime, it results in a large overlapping region between adjacent radars, leading to ambiguous and sometimes perplexing results if the data are not properly operated. However, combining radar data from overlapping sites is a nontrivial task [Watson 1995]. First, radars perform scans in a local spherical coordinate system with a unique origin placed on its own location. Fortunately, we had solved this problem with the coordinates conversion illustrated in section 3.1.3. Further complicating the issue is the fact that radars at different location are operated at different tempos, which means the data from one site are collected at the time and rate different from another site. To resolve such complications and visualize a sequence of datasets from one time point to another time point, interpolations between two time stamps are necessary to reduce the temporal-aliasing issue and to synchronize the data from different sites. In our implementation, we define a series of time - $t_i$. For each time step

$t$ in $t_i$, we check the downloaded radar product files to find two files at time $t_a, t_b$ exactly before and after $t$. This is applicable because the downloaded file names contain the date and time information as well as the site name when the scan was performed. The sample value $v_t$ at time $t$ is given in the following equation reflecting linear interpolation:

$$v_t = v_{t_a} \cdot (1 - \frac{t - t_a}{t_b - t_a}) + v_{t_b} \cdot (\frac{t - t_a}{t_b - t_a}). \qquad \text{Eq. 3.4}$$

We conduct the above interpolation between the data from every site. For example, synchronizing the weather data at a specific time $t$ - 00:05:00 on 3/12/2006 from KEAX, KLSX and KILX - involves the steps illustrated in Figure 3.5. Leftmost boxes indicate searched files on the disk, where, for instance, the file name 6500KILX03122006_000306 contains information including the radar station name – “KILX”, the date – “03/12/2006”, and the scanning time – “00:03:06”.

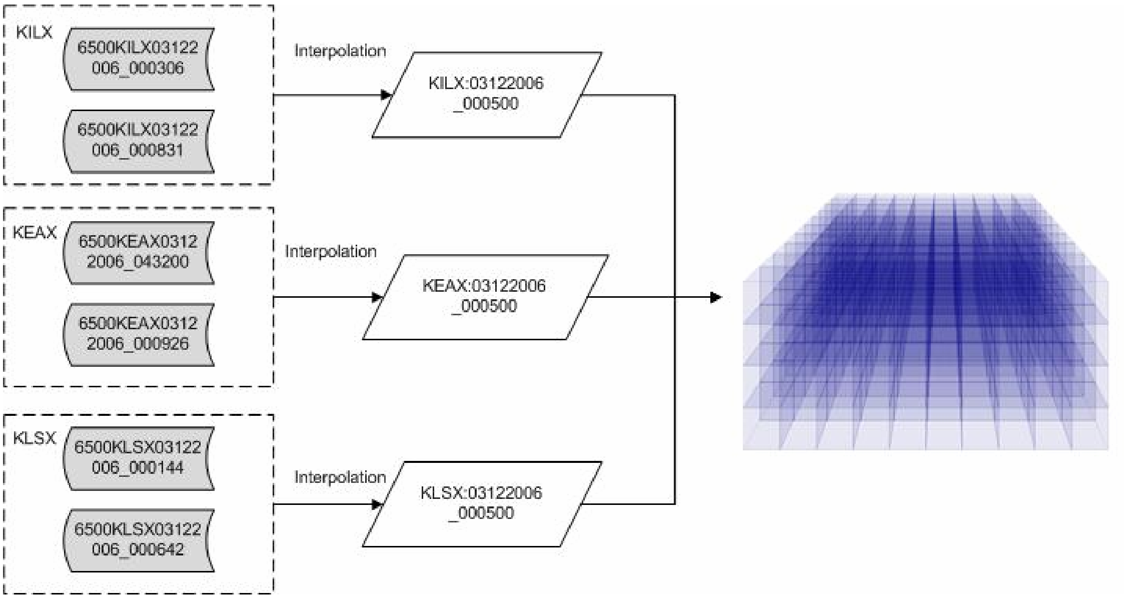


Figure 3.5. Interpolation between Different Timestamps

There are also some extreme values in the raw datasets because of measurement mistakes or malfunctions of radars. Hence, the data are also subjected to the noise attenuation method in which thresholds are set to remove extreme values created from measurement errors and to smooth the re-sampled data. Specifically, the data values are limited from 0 to 80dBZ. Resulting interpolated sample values are stored in the 3D grid structure described in section 3.1.2.

### 3.1.5. Constructing and Compressing the Volume

When the reflectivity values are re-sampled at each grid cell into the pre-defined 3D array, the volume is constructed. It is easy to conceive that, in partially overlapping regions, values obtained from more than one site are likely to be sampled to the same cell, which means some cells have duplicate values, whereas others only have a single value. To eliminate the potential error resulting from the redundant data sampling, we sum up all of values in the cell and then average them. Since the distribution of sample values may still be sparsely populated in the volume space, vertical interpolation is applied to fill the gaps in the volume. As in figure 3.6, given value $v_{k2}$ and $v_{k1}$, the value in between such as at $k$ could be obtained using the linear interpolation same as in Eq. 3.4. If all the cells under a known value $v_{k2}$ are empty, all of these empty cells are filled with the given value ($v_{k'} = v_{k2}$ as shown in the figure 3.6).

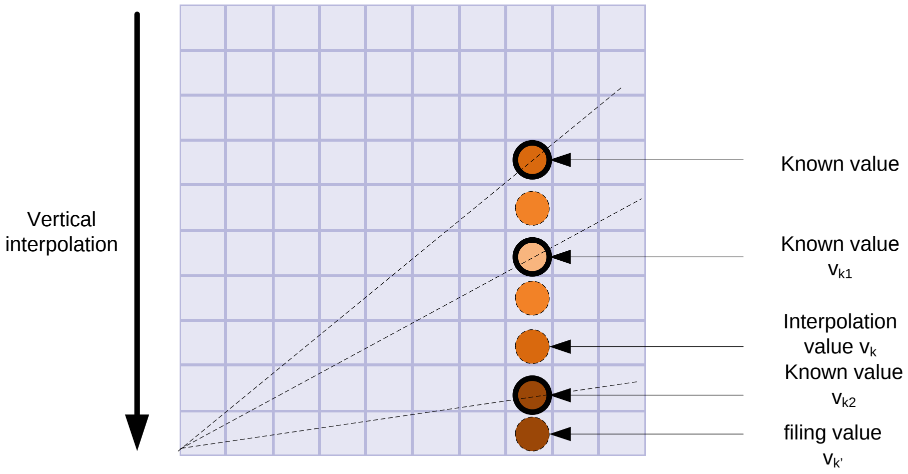


Figure 3.6. Vertical Interpolation

Displaying an animation may involve creation of a large number of images. For each image, we store the compressed 3D volumetric data in order to save the space on the hard disk and to reduce the number of disk Input/Output operations and frequent page faults during the rendering process, using the modified Run Length Encoding (RLE) compression algorithm. Because the Level II radar reflectivity data usually range from -35 dBZ to 80 dBZ, a 8-bit byte is adopted to represent the reflectivity value, and the first bit of the byte is set to 0 or 1 indicating if it is a repetition. If the value is repetitive, the first bit is set to 0, and the value is followed by a 32-bit unsigned integer to count the number of the repetition. If it is not, the first bit is set to 1 by taking it to the bitwise OR operation with 10000000(128). The compression algorithm is exemplified in the following illustration.

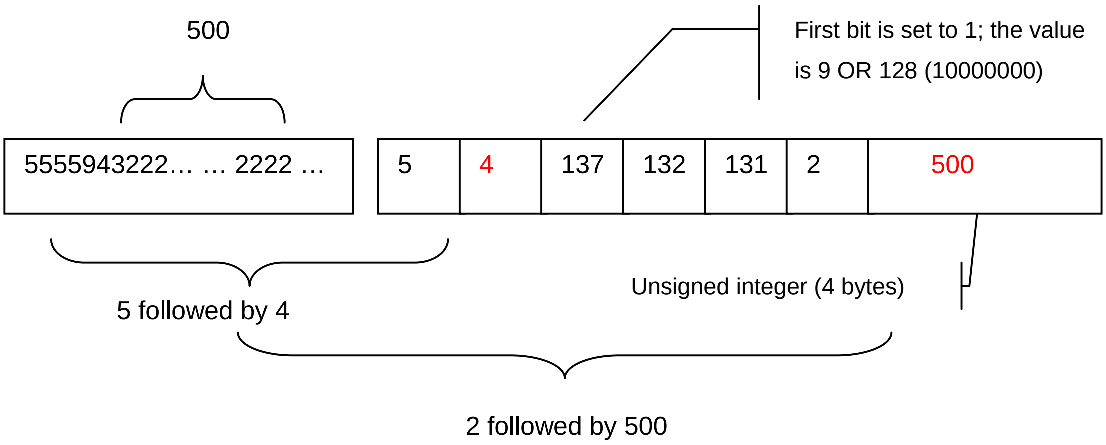

500
First bit is set to 1; the value is 9 OR 128 (10000000)
5555943222… … 2222 …
5
4
137
132
131
2
500
5 followed by 4
Unsigned integer (4 bytes)
2 followed by 500

Figure 3.7. Compressing the Data using Modified RLE Algorithm

## 3.2. Weather Data Rendering

A simple and commonly used rendering approach is to display the weather data with a polygon mesh colored by a 1D look-up table. Using this approach, the dataset from each site is represented as a number of layered cone-shaped objects as shown in Figure 3.8.

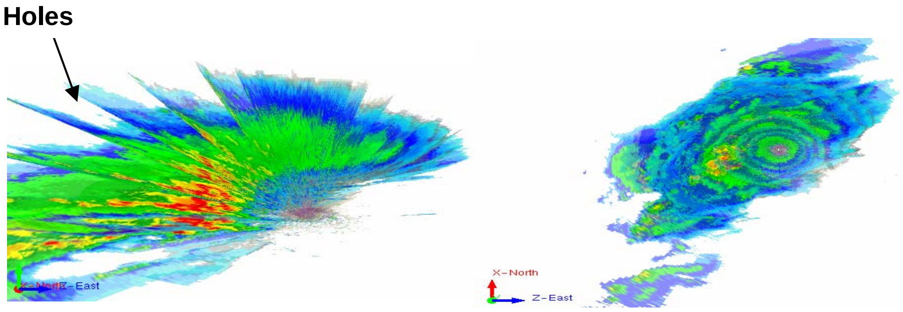


Figure 3.8. Representation of Polygon Meshes at a Site (The left image is a side view, and the right is the top view.)

This type of representation, simple and straightforward notwithstanding, becomes inadequate when data from multiple sites need to be handled. As discussed in section 3.1.4, generating a display covering areas of multiple sites needs to combine the data in overlapping regions. If the dataset at each site is represented by quadrilateral meshes – a boundary representation, combining the datasets becomes computationally expensive, and involves calculations such as complex intersection and union operations. Additionally, since the object is represented as a collection of connected surface elements, one cannot find any information once looking beyond the object surface. As shown in Figure 3.8, it is noticeable that the scene lacks the internal matter so that it leaves gaps between individual cone formations.

To overcome these limitations from the traditional surface mesh models, we adopt the texture-based volume rendering technique [Wilson 1994]. In essence, the texture-based rendering process consists of three steps: (1) generating the 3D texture, (2) slicing the 3D volume into polygons and mapping the texture onto the slices, and (3) compositing and texture mapping. A more detailed description of these steps is given in the subsequent sections.

### 3.2.1. Texture-based Volume-Rendering Technique

To attain more realistic scenes, direct volume rendering [Levoy 1988] has been used to display the high-quality visualization of the weather data. Generally, data values in the scalar filed are considered as the density of particles making up the volume in which light is emitted and absorbed. The technique maps the volume data to optical properties, and composites the optical values along the rays, to generate an image directly from the data. An optical model based on the physical light transport theory, is employed to measure the amount of light when it passes through a volume starting from $s_0$ and ending at $s_d$. (Figure 3.9)

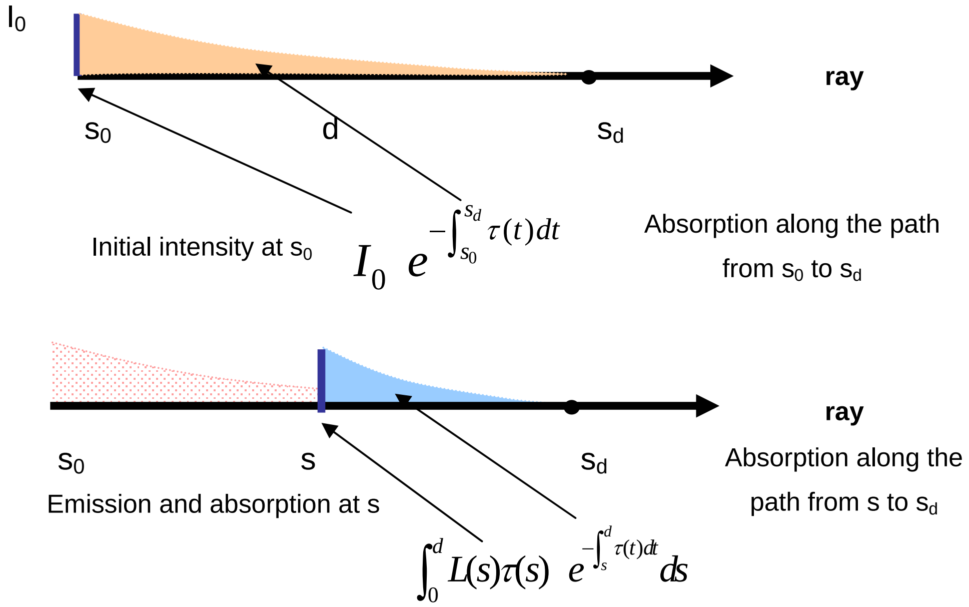

$I_0$
ray
$s_0$
d
$s_d$
Initial intensity at $s_0$
$I_0\ e^{-\int_{s_0}^{s_d}\tau(t)dt}$
Absorption along the path
from $s_0$ to $s_d$
ray
$s_0$
s
$s_d$
Absorption along the
path from s to $s_d$
Emission and absorption at s
$\int_0^d L(s)\tau(s)\ e^{-\int_s^d \tau(t)dt} ds$

Figure 3.9. Optical Model of Volume Rendering (The top image shows the absorption of light, and the bottom image shows the emission and self-absorption of light.)

The model is expressed as a summation of two terms as shown below [Engel 2006]:

$$I(s_d) = \underbrace{I_0 e^{-\int_{s_0}^{s_d}\tau(t)dt}}_{\text{Attenuation of incoming light}} + \underbrace{\int_{s_0}^{s_d} L(s)\tau(s)e^{-\int_{s}^{s_d}\tau(t)dt}ds}_{\text{Emitted light (including self attenuation)}},$$ Eq. 3.5

where $I_0$ is the initial intensity at $s_0$, $\tau$ is the absorption coefficient, and $L$ indicates the emission coefficient.

In the equation, the first term calculates the amount of incoming light that reaches the end of the volume. The second term adds the amount of light emitted at each point along the ray, taking into account the amount of attenuation from each point to the end of the ray as well. Practically, the integral is usually solved by an iterative computation procedure, which composites the pixel values either from the viewpoint to the volume referred as the front-to-back order or from the volume to the viewpoint referred as the back-to-front order.  Here, we choose the back-to-front approach [Engel 2006]:

$$C'_{dst} \leftarrow (1-\alpha_{src})C_{dst} + C_{src}\alpha_{src},$$
$$\alpha'_{dst} \leftarrow (1-\alpha_{src})\alpha_{dst} + \alpha_{src}.$$
Eq. 3.6

As shown in Figure 3.10, the new composited value $C'_i$ is computed from the color at current location $C_i$ blended with previous composited color $C'_{i-1}$ .

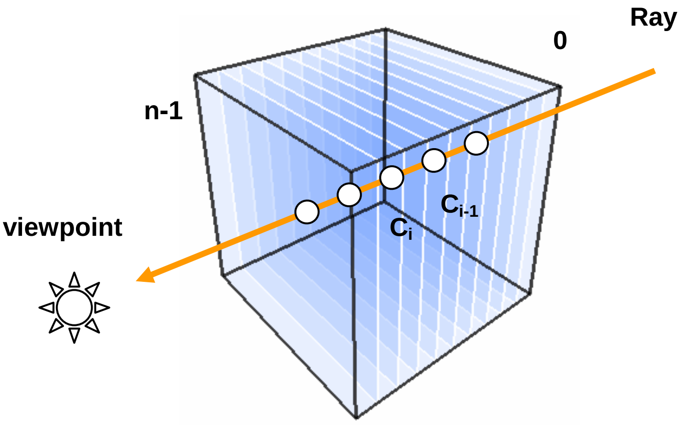


Figure 3.10.  Alpha Blending in Back-to-front Order

$$C'_i = \alpha_i C_i + (1-\alpha_i)C'_{i-1},$$
$$\alpha'_i = \alpha_i + (1-\alpha_i)\alpha'_{i-1}.$$
Eq. 3.7

To speed up the rendering process, we adopt the 3D texture-based volume rendering technique [Wilson 1994] with Graphics Processor Unit (GPU) hardware acceleration.  This technique consists of two basic steps: (1) constructing the 3D texture and (2) generating back-to-front ordered slices and texture-mapping the slices with the volume data using NVIDIA CG vertex and fragment shading program.

### 3.2.2. Decompression and 3D Textures

Since the compressed volumetric data are stored on the hard disk, as described in section 3.1.5, the initial step before the rendering process is to decompress the volumetric data and bring them back to the main memory. The 3D texture then is constructed and stored in GPU memory.

The decompression process entails the following procedure: (1) For the incoming streamed byte from the file, the bitwise AND operation with 10000000(128) is used to obtain the first bit of the byte and mask the rest bits of the byte. (2) If the first bit is set to 1, which means it is a non-repetitive value, the bitwise OR operation with 01111111(127) is then activated to retrieve the rest bits of the byte and to obtain the reflectivity value. (3) If, otherwise, the first bit is set to 0, indicating that the value is a repetitive, the algorithm then composites the next four bytes into an unsigned integer to acquire the repetitive number. (4) When the retrieved value and its repetitive number are obtained, they are then written into an array, allowing for the construction of the volume. By repeating these four steps until the last byte is reached, the data are decompressed and streamed into a large 1D array with the size defined by the multiplication of the volume length, the volume width and the volume height. The 1D array of voxel data are then transferred to a 3D texture and stored in the GPU.

### 3.2.3. Viewport-aligned Parallel Slices

In object-order volume rendering, a stack of 2D parallel polygonal slices are usually used to represent 3D discrete scalar field. Once the 3D texture is stored in GPU, it will be mapped into these semi-transparent slices to texture them. The slices are referred as proxy geometry, which only represent the shape of the data domain – the bounding box, instead of the shape of the real object. In our implementation, the 3D volume is represented as a unit cube, and is divided into viewport-aligned slices in the equal-distance fashion, the back-to-front order parallel to the image plane (Figure 3.11).

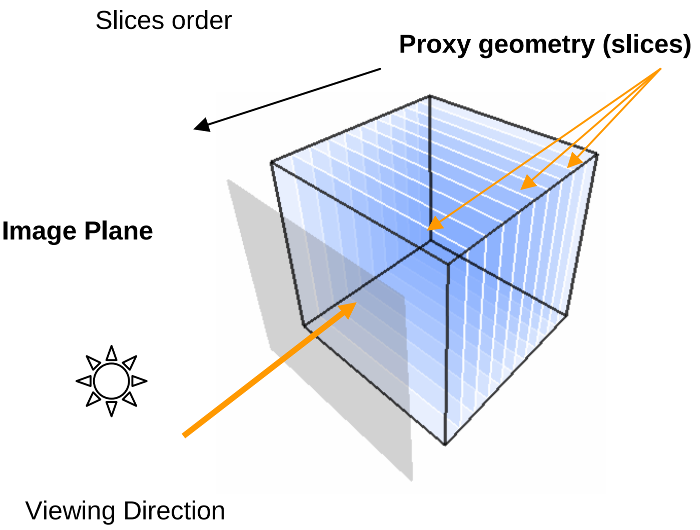


Figure 3.11. Viewport-aligned Parallel Slices

In order to keep a consistent sampling rate along all viewing rays casting from the image plane, orthogonal projection is adopted. To avoid re-computing the slices when the viewing direction changes, we keep the viewing direction and slicing planes fixed along the z axis, and apply the model-view transformation matrix to the volumetric object whenever a rotation or a translation transformation is involved. Initially, the number of slices is designed as an adjustable parameter on the interface with the default value of 256. Users can change the value for the purpose of adjusting the volume resolution for less or more details in the image. The detailed analysis and resulting images are given in Chapter 4.

### 3.2.4. Transfer Functions

After the stored 3D texture was applied to the proxy geometry, the scalar reflectivity value is mapped onto the slices by performing 3D texture lookup functions. At this stage, however, the optical properties of each sample, such as the color and opacity still remain undecided. Having considered the fact that the reflectivity is a scalar field, we choose a 1D look-up color table as the transfer function and store it as a 1D texture. By taking the 3D texture samples – the reflectivity data – as the look-up color table indices, and mapping them into the display attributes, the color and density of the volumetric object are identified. The look-up color table is similar to the one used in the NWS web page [NWS] and is widely used for radar reflectivity data (Figure 3.12).

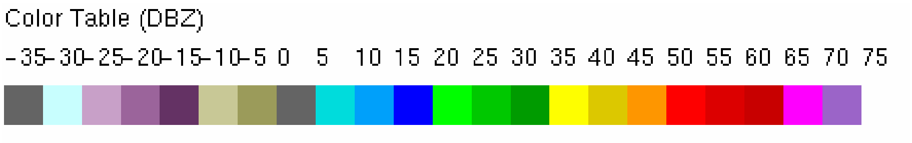


Figure 3.12. 1D Look-up Color Table

To support the interactive transfer function, we create an interface that acts as the transfer function window, allowing users to create and modify transfer functions to their interest. The created or modified transfer functions are then applied to map the data onto the appropriate color and opacity. Figure 3.13 shows the developed transfer function window for red, green, blue (RGB) and alpha channels representing the NWS look-up color table [NWS] in Figure 3.12.

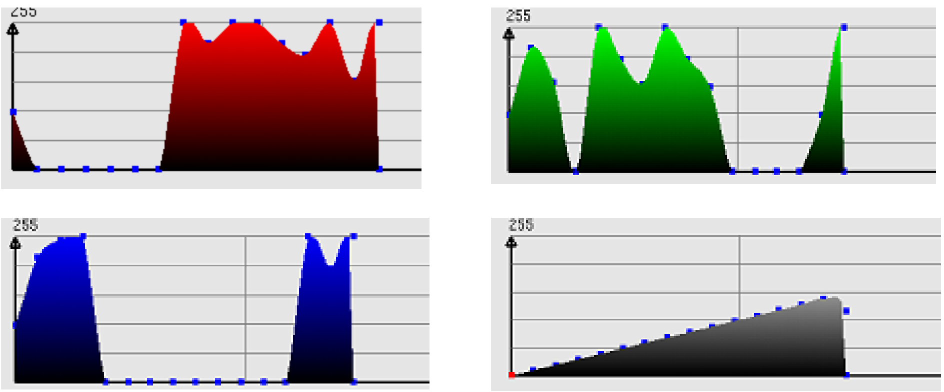


Figure 3.13. Transfer Function Window for RGB and Alpha Channels

To facilitate users' interaction with transfer functions, we employ the cubic Hermite spline (Cardinal Spline) to represent the color table. Mathematically, the cubic Hermite spline is a third-degree spline with each polynomial in Hermite Form. The cubic Hermite form is defined by the starting point $P_1$ at $t_1$ and the ending point $P_2$ at $t_2$, with the starting tangent vector $P_1'$ and the ending vector $P_2'$.

The computation of the interpolation at a specific time $t$ is given by the following equation:

$$P_t = (t^3, t^2, t, 1) M_H (P_1, P_2, P_1', P_2')^T, \qquad \text{Eq. 3.8}$$

where $M_H$ is the Hermite basis matrix as in

$$M_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \qquad \text{Eq. 3.9}$$

The utility of the transfer function is exemplified by a demonstration shown in Figure 3.14. When certain sample points on the spline curve of the red channel are modified by a user, the transfer function is recalculated, giving a new spline that fits the new dataset, all in the real time fashion.

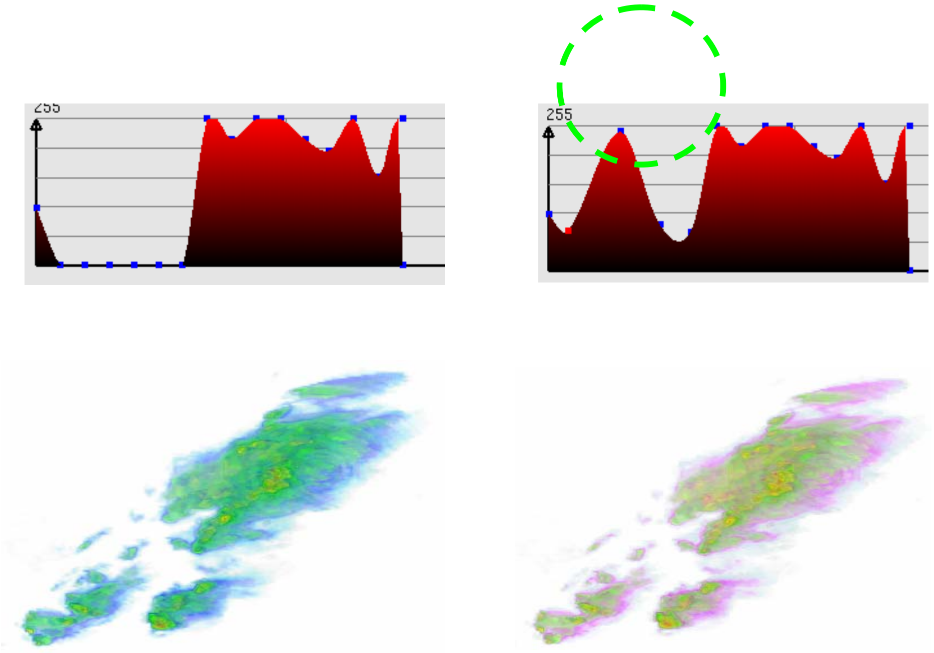


Figure 3.14. Example of the Red Channel Transfer Function

### 3.2.5. CG Shading and Animation

The shading and animation was performed using the NVIDIA CG shading programming language (Figure 3.15), which is very suitable for volumetric rendering the scene as well as accelerating the rendering speed.

In the vertex shader, the vertex processor converts a vertex position into the transformed position, computes the texture coordinates as outputs, and clips the volumetric object against the pre-defined planes. In the fragment shader, the fragment processor reads the textures and the fragment in, and maps the texture color and density onto the fragment.

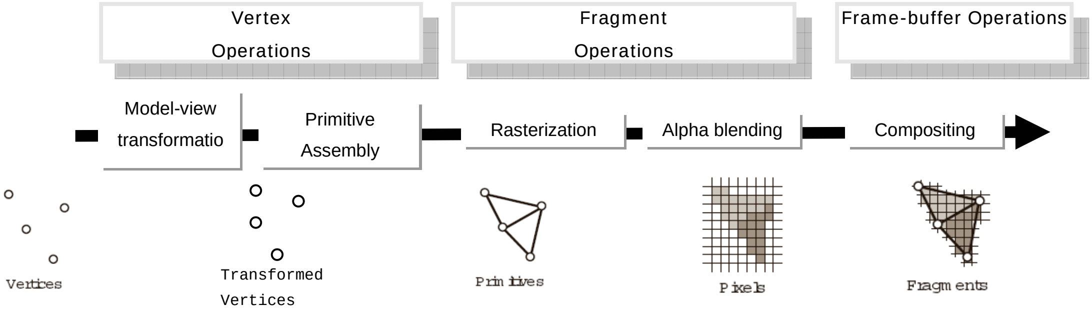
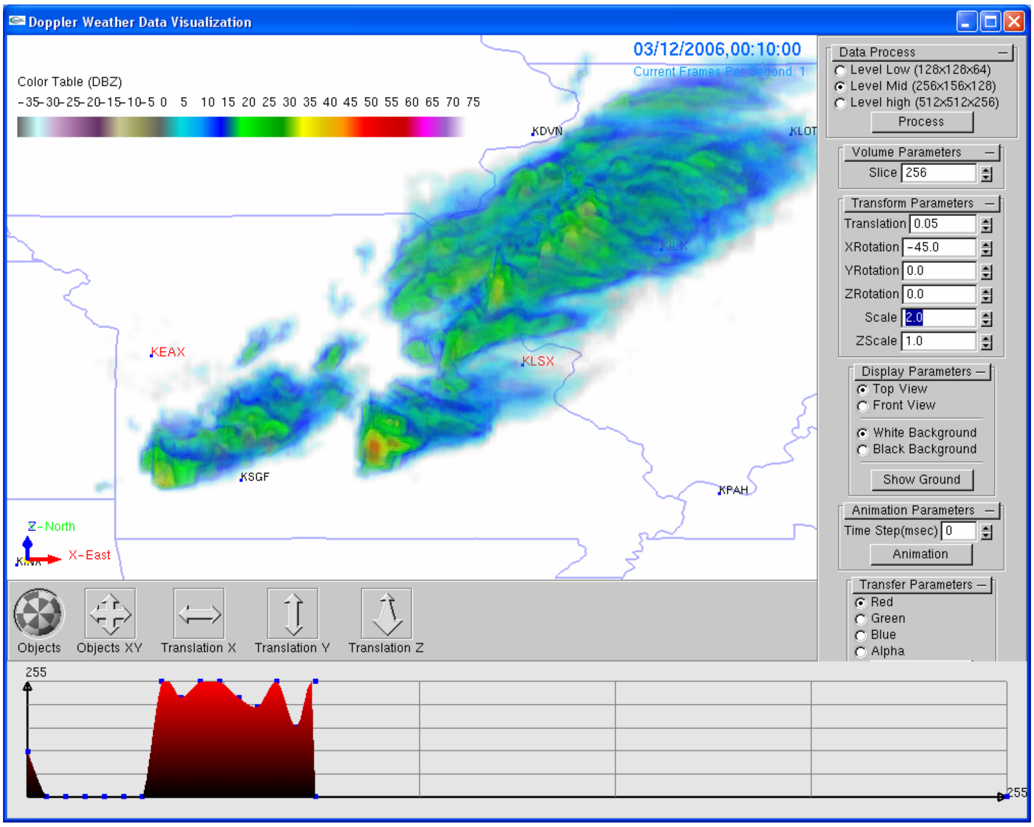


Figure 3.15. CG Vertex and Fragment Programs



Data Process
○ Level Low (128x128x64)
● Level Mid (256x156x128)
○ Level high (512x512x256)

Volume Parameters
Slice 256

Transform Parameters
Translation 0.05
XRotation -45.0
YRotation 0.0
ZRotation 0.0
Scale 2.0
ZScale 1.0

Display Parameters
● Top View
○ Front View
● White Background
○ Black Background

Animation Parameters
Time Step(msec) 0

Transfer Parameters
● Red
○ Green
○ Blue
○ Alpha

Figure 3.16. Volumetric Rendering and Graphic User Interface

Our program also includes the Graphic User Interface (GUI) to reflect functionalities that allow users to interact with the data and to choose the representation from different perspectives. An actual image of the created user interface is shown in Figure 3.16.

To visualize the weather data over a period of time, it is more desirable to present them as a sequence and viewed as an animation. For this purpose, we stored 3D textures on the disk based on an interval of five minutes. The program is designed to orderly read in the sequential texture files on a continuous basis and to conduct the rendering job, where the rendering time relies on both the CPU and GPU speed. Detailed discussions will be shown in Chapter 4. An adjustable parameter is also provided to moderate the interval between two frames through the user interface. As a supplement of the thesis, a CD is provided to show a movie about the 24-hour supercell storms which were observed on March 12, 2006, from the three locations (KEAX, KILX and KSLX) in the Midwest of the United States.

# CHAPTER 4. RESULTS

In this chapter, some rendering results are first presented to verify our methodology, and a number of experiments are described to show the costs and performance of computation and rendering in the system. The datasets used in our system for testing were downloaded from NWS FTP server, which contain the data from scanning 24-hour supercell storms on Match 12, 2006, in the Midwest region of the United States. All simulations have been carried out on a Windows desktop equipped with a 3.20GHz Pentium 4 processor, 2.0 GB of main memory, and an NVDIA Quadro FX 3500 graphics card with 256 MB of video memory. An exception is in section 4.3.4 where we aim to compare results between different computer hardware. The resulting image resolution is set to 796x532 for all the experiments.

## 4.1. Visualization Results

The fist example includes four rendering images at different timestamps (Figure 4.1) selected from the animation. The images are rendered when the viewing direction is tilted approximately 20 degrees relative to Z axis. The complete 24-hours animation can be obtained on the attached CD. From the animation, we could see the changing weather pattern on that particular day.

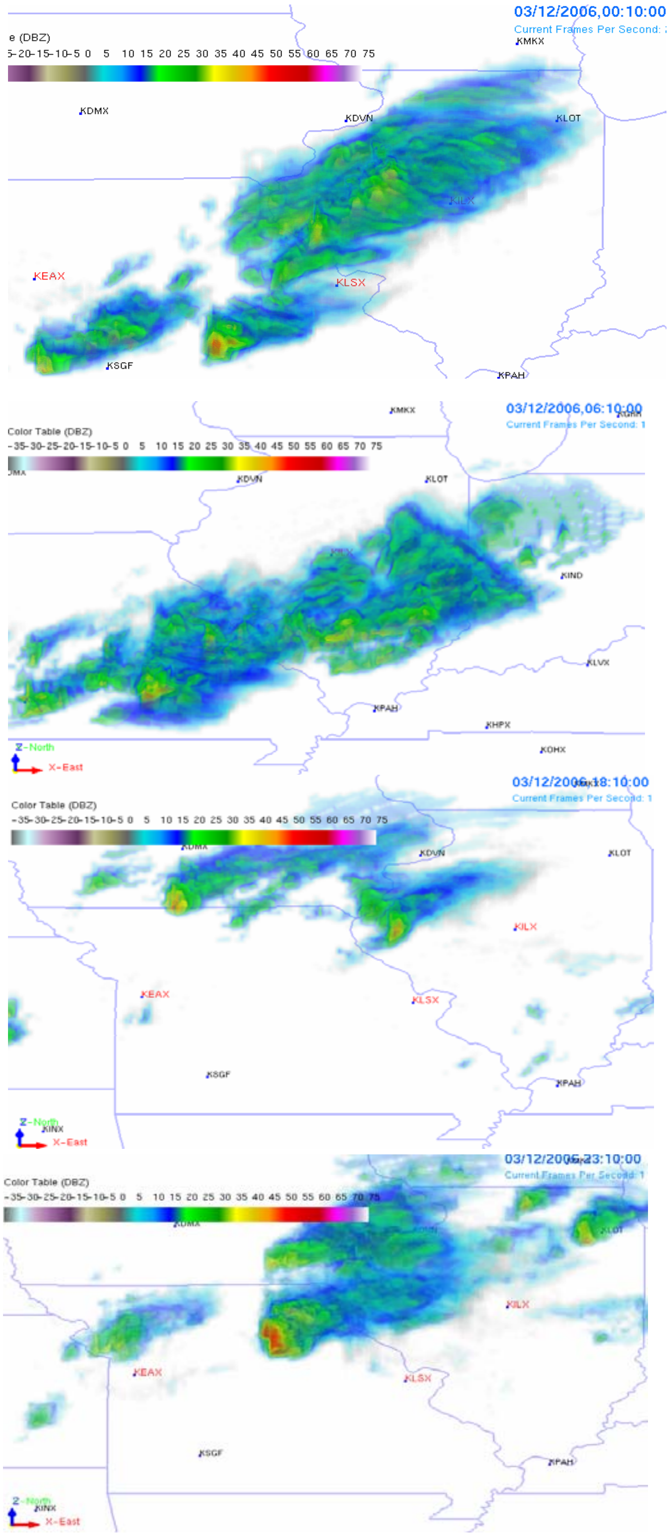


Figure 4.1. Images Rendered from Different Timestamps – top left: 00:10:00; top right: 06:10:00; bottom left: 18:10:00; bottom right: 23:10:00

The second example in Figure 4.2 shows that the different observations of the data by applying different iso-values.

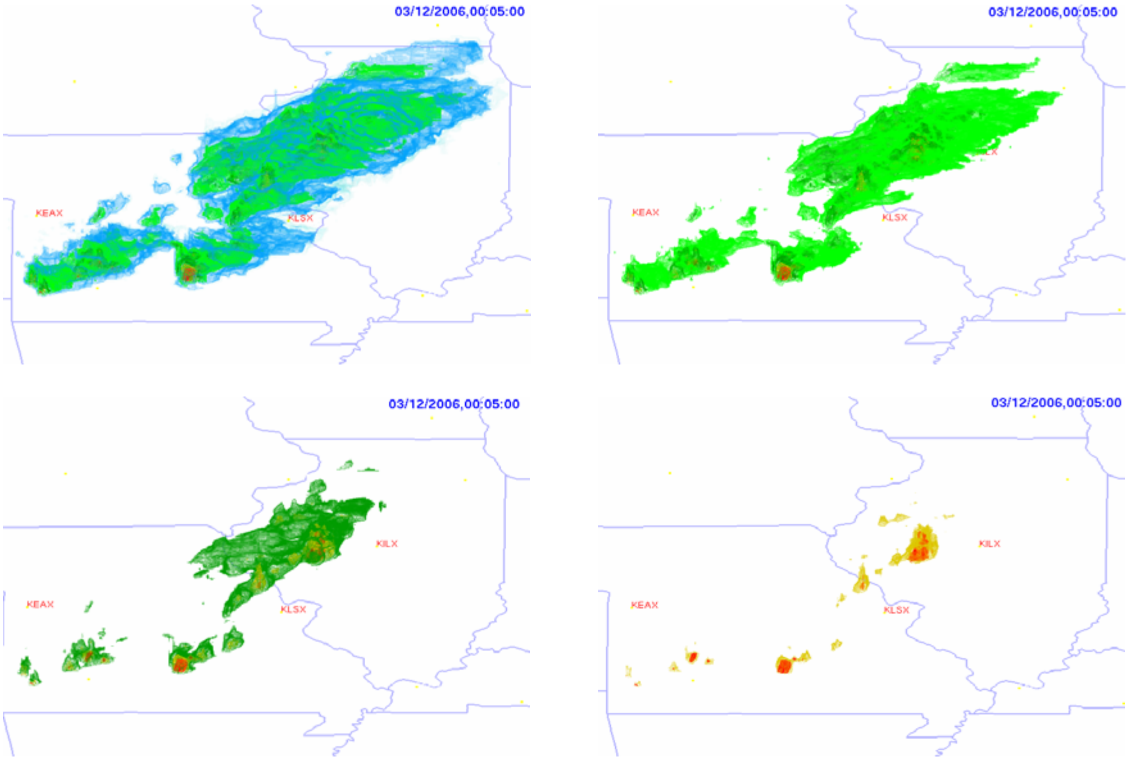


Figure 4.2. Images Rendered by Applying Different Iso-values

## 4.2. Quality of Visualization

### 4.2.1. Effect of Volume Resolutions on Visualization Results

Our first experiment was done for the purpose of comparing the visualization results using different volume resolutions. In this experiment, the effect of volume resolution was exemplified by visualizing datasets with three different levels of detail. Datasets from the first group have the lowest resolution of 128x128x64, the volume size of the second group is set to 256x256x128, and the third group has the highest resolution of 512x512x256. The results of visualization show clear evidence of the effect of the volume resolution, illustrated in Figure 4.3. In particular, the reduction in volume resolution results in blurriness and fuzziness in the image, while the shape of the object is still preserved. However, when the resolution of the volume is increased, the generated image appears to capture finer details of data.

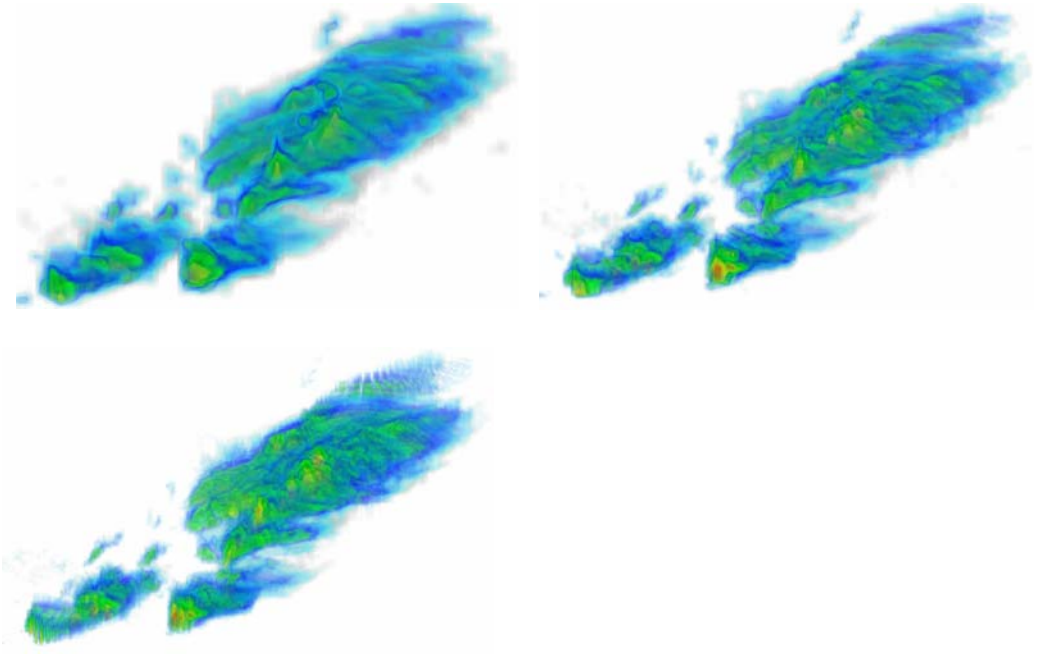


Figure 4.3. Images Rendered from Different Resolution Datasets – Resolution: top left: 128x128x64; top right: 256x256x128; bottom left: 512x512x256

### 4.2.2. Effects of Sampling Rates on Visualization Results

In the second set of experiments, the effect of the sampling rate – decided by the number of slices of sampling planes - on quality of the rendering was studied. Specifically, the weather datasets were rendered with different sampling rates whereas the volume data resolution remains unchanged. In this experiment, the volume data resolution is set to a fixed size of 256x256x128, and the number of sampling planes varies from 64, 128, 256 and 512. The images pictured in Figure 4.4 clearly show that applying fewer slices of the proxy geometry serves to reduce quality of the image. The image quality is improved when the number of slices is increased. The color difference is caused by the color blending using different number of slices.

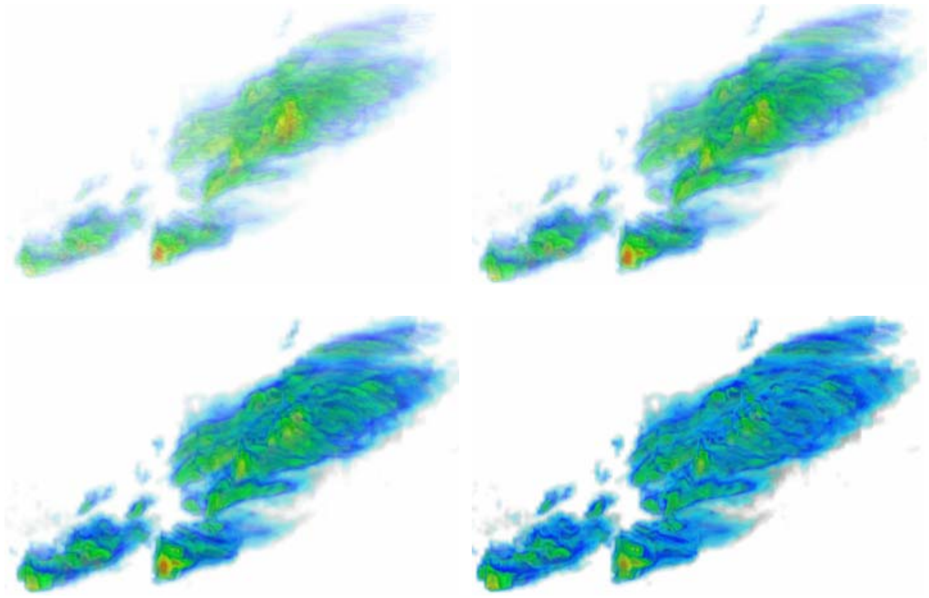


Figure 4.4. Images Rendered from Different Sampling Rates – Resolution: top left: 64 slices; top right: 128 slices; bottom left: 256 slices; bottom right: 512 slices

## 4.3. Costs and Performance

### 4.3.1. Effects of Volume Resolution on Costs of Data Processing

Although it has been shown in section 4.2.1 that the choice of lower volume resolution will cause some loss of details in the resulting image, it, however, has some benefits both during the data processing stage and the rendering stage. In the data processing stage, each volume dataset is generated from six raw radar datasets (see section 3.1.4) and stored on the hard disk at one timestamp. For visualizing the weather for an entire day (24 hours), more than 280 texture files are generated based on the interval time of five minutes.

The processing time required under different resolution is given in Table 4.1. For the lowest resolution tested (128x128x64), the average processing time is about six seconds for generating each dataset of volumetric data. The approximate total time of computation and integration for generating all the texture files ranges from about half an hour (30 minutes) with the lowest resolution to one hour with the highest resolution (512x512x256) . In other words, the data processing with the highest resolution costs twice as much time as with the lowest resolution (Figure 4.5).

Table 4.1. Costs of Processing Radar Data of Different Resolution

| Resolution | 128x128x64 | 256x256x128 | 512x512x256 |
|---|---|---|---|
| Average processing time (secs) | 6 | 8 | 12 |
| Total processing time (mins) | 30 | 45 | 57 |

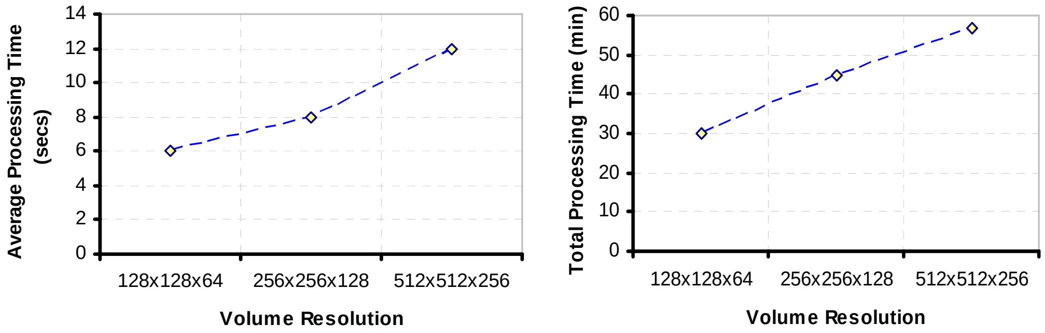


Figure 4.5. Volume Resolution versus Processing Time

### 4.3.2. Effects of Volume Resolution on Costs of Data Rendering

The reduction of the dataset resolution also could speed processing in the rendering state. To demonstrate this relationship, we measured the time it takes to complete the rendering procedure under different volume resolutions. The total processing time is made up of three individual steps: $t1$- reading the volumetric dataset from the hard disk with decompression, $t2$ - the time of reconstructing and updating the 3D texture in GPU, and $t3$ - the actual rendering time for one frame.

The results, which are shown in Table 4.2 and Figure 4.6, were obtained using a constant sampling rate – 256 slices. It can be seen from the table that the total time of the rendering stage increases with higher resolution.

Table 4.2. Costs of Rendering Radar data of Different Resolution

| Resolution | 128x128x64 | | 256x256x128 | | 512x512x256 | |
|---|---|---|---|---|---|---|
| Average time of reading texture (msecs) | <1 | - | 8 | 5.19% | 100 | 28.01% |
| Average time of updating texture (msecs) | 3.8 | 3.02% | 24 | 15.58% | 131 | 36.69% |
| Average time of actual rendering (msecs) | 122 | 96.98% | 122 | 79.22% | 126 | 35.29% |
| Total time(msecs) | 126 | | 154 | | 357 | |

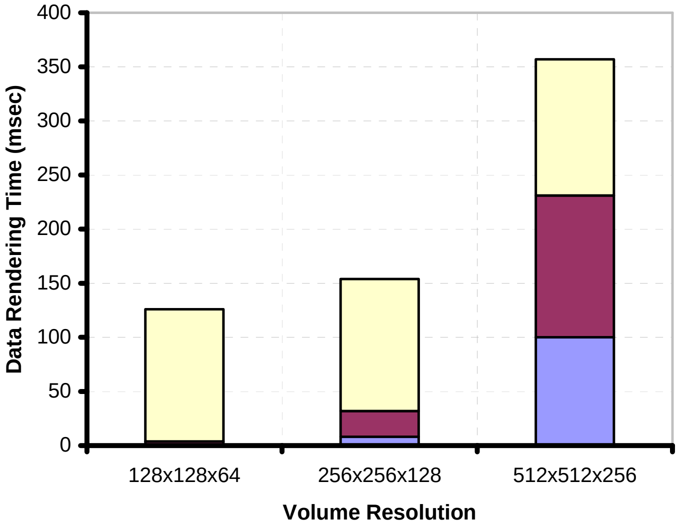


Figure 4.6.  Volume Resolution versus Data Rendering Time

Another important observation from the Table 4.2 is that the actual rendering – rendering on GPU - computation accounts for the majority of time during the rendering stage, which is largely independent of the volume resolution.  The pie chart in Figure 4.7 gives a graphical representation of the time distribution at the rendering state.  In other words, reducing the volume resolution facilitates faster processes of reading and updating texture, which, however, does not help much to speed the actual rendering process.  This phenomenon is controlled by both the size of the datasets and number of sampling planes.  In a word, the resolution determines texture reading and updating time as well as the data processing time, and the number of slices dictates the actual rendering time.

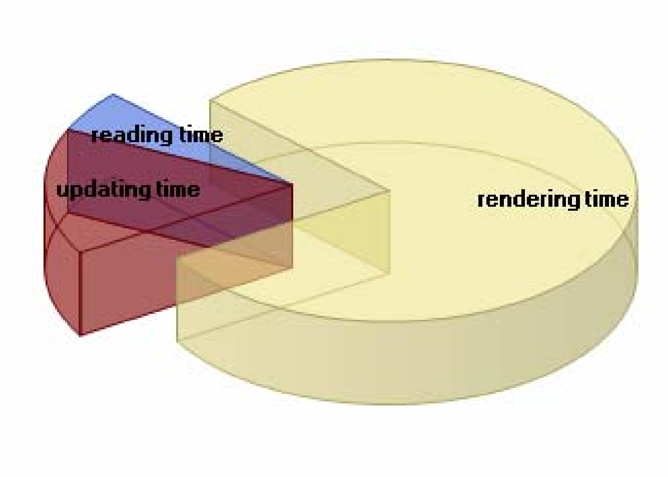


Figure 4.7. Proportion of Time Spent on the Rendering Stage (for resolution of 256x256x128)

### 4.3.3. Performance of Combining Different Resolutions and Sampling Rates

To further verify the argument in section 4.3.2, an additional experiment was conducted. In this experiment, we measure the frame rate – the number of frames per second - under various combinations of the resolution and the sampling rate to evaluate the performance. The frame rate $fps$ is defined in Equation 4.1:

$$fps = \frac{1}{t_1 + t_2 + t_3}. \qquad \text{Eq. 4.1}$$

$t_1, t_2, t_3$ are defined same as in section 4.3.2. The resulting data are shown in Table 4.3, and its graphical representation is shown in Figure 4.8.

Table 4.3. Frame Rates measured with Different Resolution and Number of Slices

| Slices \ Resolution | 128x128x64 | 256x256x128 | 512x512x256 |
|---|---|---|---|
| 64 | 20 | 18 | 4 |
| 128 | 15 | 12 | 4 |
| 256 | 8 | 8 | 3.5 |
| 512 | 5 | 5 | 2.5 |
| 1024 | 3 | 3 | 1.5 |

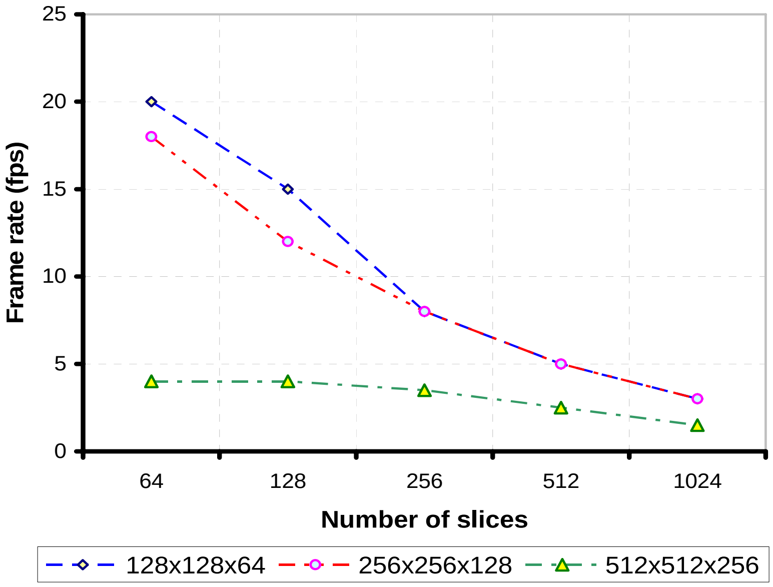


Figure 4.8. Number of Frames per Second (FPS) versus Number of Slices

The plot indicates two important features:

(1) With the resolution increased, the frame rate drops accordingly for most test cases. The trend is due to the fact that the higher resolution requires longer texture reading and updating time ($t_1 + t_2$), thereby decreasing the number of frames processed per unit time. Another observation is that, the impact of resolution on frame rates is more dramatic at a lower sampling rate. Comparatively, at a higher sampling rate, the change of the frame rate is less dependent on the volume resolution. The reason is that at the low sampling rate,

the proportion of the time spent on the actual rendering $t_3$ over the total rendering time is less than the proportion at a high sampling rate. An increase of resolution which primarily increases $t_1 + t_2$, therefore exerts a greater influence on the frame rate.

(2) With an increase in the number of slices, the frame rate declines accordingly. This is because the sampling rate dictates the actual rendering time $t_3$, with a higher sampling rate leading to the reduced frame rate. It is also noticed that, the effect of sampling rate on the frame rate is more substantial when a lower resolution is used. With the highest resolution (512x512x256), the frame rate is almost independent of the sampling rate. The reason of this observation is that with the lower resolution, the time cost of texture reading and updating texture ($t_1 + t_2$) is relatively insignificant as compared to on the actual rendering time $t_3$. Since $t_3$ is dictated by the sampling rate, the majority of the computation resource is recruited for the actual rendering at a higher sampling rate. At low resolution, the magnitude of $t_1 + t_2$ is too small to effectively dilute the effect of sampling rate, making it the single most important rate-controlling parameter for the visualization process.

### 4.3.4. Testing with Different Hardware

We also have tested the program on five different machines. The results are shown in Figure 4.9. With better hardware such as CPU and GPU, the rendering speed is significantly improved. For example, the computer with a Dual-Core AMD Opteron 2.2GHz processor with 2GB RAM and an NVIDIA GeForce 8800GTX graphics card with 768MB memory can render around 45 frames per second with the resolution of 128x128x64 of and 256 slices.

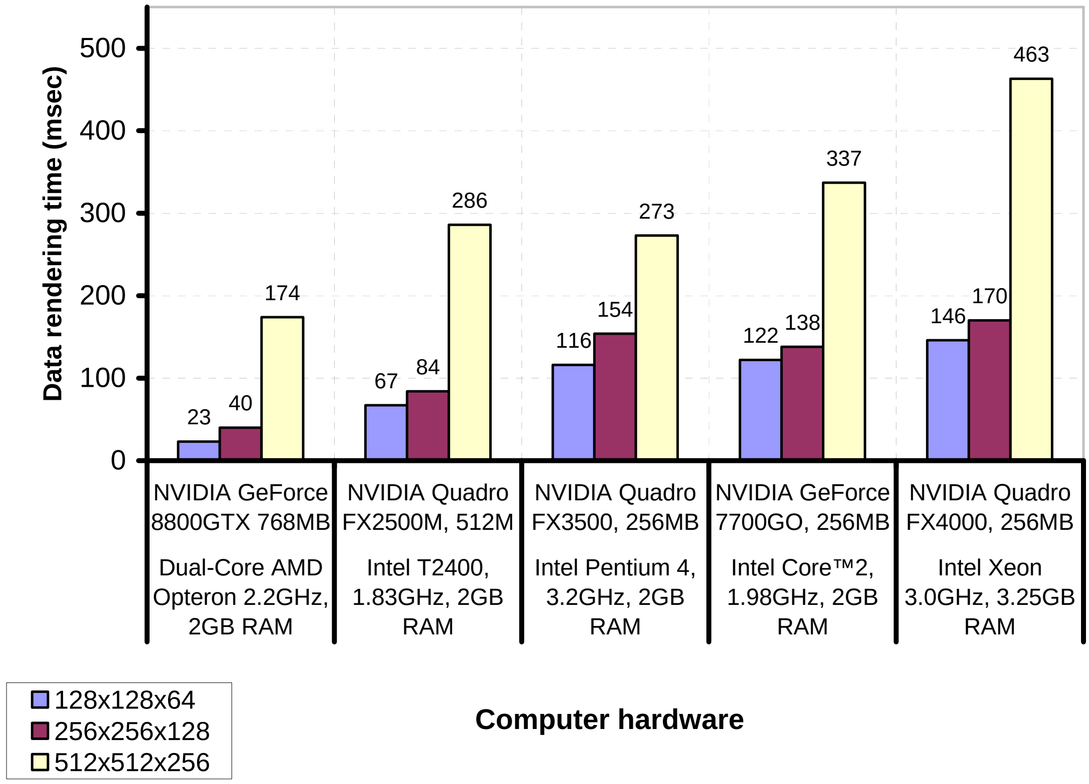


Figure 4.9. Rendering Time on Different Machines

### 4.3.5. Effects of the 3D Texture Compression Algorithm

The last evaluation was conducted to show the benefits of the texture compression work. Table 4.4 shows that the disk space needed for one dataset is dramatically decreased by utilizing the RLE compression algorithm. For example, for a 256x256x128 dataset, the space on the disk required is $2^8 \times 2^8 \times 2^7 = 2^{23} = 8M$ bytes. With the compression, the size of dataset is shrunk to 54K, which is about 0.66% of the size of an uncompressed texture.

Table 4.4. Costs of Disk Storage of the 3D Textures

| Resolution<br>Size | 128x128x64 | 256x256x128 | 512x512x256 |
|---|---|---|---|
| Size before Compression | 1M | 8M | 64M |
| Size after Compression | 10.2K | 54.0K | 308.4K |
| Compression Rate | 99.1% | 99.34% | 99.53% |

## 4.4. Summary

From the experimental results, we could conclude that both volume resolution and number of sampling planes have great influence on image quality and program performance. Typically, the resolution determines texture reading and updating time as well as the data processing time on CPU, and the number of slices regulates the actual rendering time on GPU. Practically, we need to take both effects of sampling rate and resolution into consideration to find a best-fit setting to ensure a good visualization.

# CHAPTER 5. CONCLUSIONS AND FUTURE WORK

## 5.1. Summary

The project presents a visualization tool that displays and manipulates sequences of the Level II Doppler radar data in 3D volumes from multiple sites. A case study is provided to record the supercell storms on March 12, 2006, in the Midwest of the United States. We propose a feasible integration method to process data from multiple sites and a fast approach to render the data in real time. Our methods have the following advantages:

- The data are integrated from three radar sites in an entire volume, with which integrating data from more sites will not suffer from limited memory on GPU. Additionally, with one integrated dataset, the shading code is easier to be implemented. The rendering speed is also improved because of reduced computations on GPU.
- The radar data are displayed in 3D and the rendering speed is fast by utilizing the graphics hardware.
- 3D textures are stored as compressed files that not only save around 99% of originally required disk space but also speed up the rendering procedures.

The resulting images and animation demonstrate how fast and robust our techniques are both, making it suitable for rendering the Doppler radar data from multiple sites in production of weather visualization.

## 5.2. Limitation

Currently the field studied and visualized is limited to reflectivity. Visualization on other important fields such as velocity and spectrum width still remains unresolved. Although the rendering process could be done in real-time, the data processing is done beforehand.

The 3D texture-based volumetric rendering technique with the hardware acceleration is efficient at this stage, however it still will become computationally expensive when large-scale volume-data are involved.

The system was performed and tested on a standalone windows machine, and the interactive visualization is available for a single user at a time. It could not handle a large amount of jobs submitted; hence it is not feasible for multiple users to visualize the data at the same time.

### 5.3. Future Work

Weather data comprises a large number of volumetric scalar fields, vector fields and tensor fields. For a long time, studies have focused on methods and techniques for visualization of a single field. Displaying multiple fields simultaneously remains as an important and challenging task for scientific researchers [Johnson 2004]. An extension to this work is to construct and visualize other meteorological fields and derived quantities such as velocity and spectrum width to graphically represent wind and turbulence, which will supplement knowledge on the data to meet more needs of possible users.

We would like to improve our visualization algorithm by using the adaptive volumetric rendering technique. A hierarchical volume will be constructed, and the hierarchy will be traversed to find the best-fit volume according to some level of quality and speed. We also could combine lighting effects to illuminate the volume samples.

Another interesting path to explore is to integrate the visualization into TeraGrid. Since real-time Doppler data are also available at Purdue University in TeraGrid, direct data extraction can be conducted by connecting to Storage Resource Broker (SRB) distributed file systems. The visualization will be displayed on a website through Virtual Network Computing graphical sharing system (VNC). Then the visualization tool would be available to ordinary people as well as researchers for the purpose ranging from education to scientific research. With this interface such as a web portal, users can have access to the remote Doppler data and interactively visualize them online in a near-real-time fashion.

One more challenging but very important enhancement will be refining the system that is able to handle both computing and rendering jobs in a distributed way. The benefits include: accelerating the processing and rendering speed; providing scalability to the system; guaranteeing accessibility for more users. Furthermore, we could pre-process and produce datasets readable for other simulation systems such as ParaView [ParaView].

# LIST OF REFERENCES

[CAVE] http://www.evl.uic.edu/pape/CAVE/.

[CAVE5D] http://www.ccpo.odu.edu/~cave5d/.

[Chen 2001A] Chen, X., Barron, J. L., Mercer, R. E., Joe, P., "3D Least Squares Velocity from 3D Doppler radial Velocity," *Visual Interface* June 2001, 56-63.

[Chen 2001B] Chen, X., Barron, J. L., Mercer, R. E., Joe, P., "3D Regularized Velocity from 3D Doppler radial Velocity," *IEEE International Conference on Image Processing 2001*, Volume 3, 664-667.

[DisasterReport] http://www.wind.ttu.edu/Research/DebrisImpact/Reports/DDS.pdf.

[Djurcilov 1999] Djurcilov, S., Pang, A., "Visualizing Gridded Datasets with Large Number of Missing Values," *In proceedings of the conference on Visualization '99*, 405-408.

[Dobashi 2000] Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., Nishita, T., "A Simple, Efficient Method for Realistic Animation of Clouds," *Proc. SIGGRAPH2000*, 2000-7, pp. 19-28.

[Dobashi2001] Dobashi, Y., Nishita, T., Miyazaki, R., Yoshida, S., "Modeling and Dynamics of Clouds Using a Coupled Map Lattice," *Proc. Siggraph 2001 Technical Sketches*, pp. 229, Los Angeles (USA), August 2001.

[Drebin 1988] Drebin, R. A., Carpenter, L., Hanrahan, P., "Volume Rendering," *Proceedings of SIGGRAPH '88, Computer Graphics,* August (22), 65-74.

[Ebert 1990] Ebert, D. S., Parent, R. E., "Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques," *Computer Graphics*, 24(4), 357-366.

[Engel 2001] Engel, K., Kraus, M., Ertl, T., “High-quality pre-integrated volume rendering using hardware-accelerated pixel shading,” *Proceedings of the ACM SIGGRAPH/ENROGRAPHICS Workshop on Graphics Hardware 2001,* 9-16.

[Engel 2006] Engel, K., Hadwiger, M., Kniss, J. M., Rezk-Salama, C., Weiskopf, D., “*Real-Time Volume Graphics,*” Wellesley, MA: A K Peters, Ltd.

[Gao 2001] Gao, J., Xue, M., Shapiro, A., Xu, Q., Droegemeier, K. K., "Three-Dimensional Simple Adjoint Velocity Retrievals from Single-Doppler Radar," *Journal of Atmospheric and Oceanic Technology*, Vol. 18, 26-38.

[Harasti 2006] Harasti, P. R., Lee W. C., Bell, M. M., “Real-time implementation of VORTRAC at the National Hurricane Center,” *ERAD 2006 Proceedings,* P11A.6.

[Harris 2001] Harris, Mark J., “Real-Time Cloud Rendering,” *Computer Graphics Forum (Eurographics 2001 Proceedings)*, 20(3), 76-84.

[Harris 2002] Harris, Mark J., “Real-Time Cloud Rendering for Games,” *Proceedings of Game Developers Conference 2002*, March 2002.

[Harris 2003] Harris, Mark J., Baxter III, William V., Scheuermann, T., Lastra, A., “Simulation of Cloud Dynamics on Graphics Hardware,” *Proceedings of Graphics Hardware 2003*.

[Hibbard 1990] Hibbard, B., Santeck, D., "The Vis-5D System for Easy Interactive Visualization," *Proceedings of IEEE Visualization October 1990*, 28-35.

[Hibbard 1998] Hibbard, B., "VisAD: Connecting People to Computations and People to People," *ACM SIGGRAPH Computer Graphics*, 32(3):10-12.

[Huber 2007] Huber, M., Trapp, J., "A Review of NEXRAD Level II: Data, Distribution, and Applications," *Journal of Terrestrial Observation*, in press.

[Iourcha 1997] Iourcha, K. I., Nayak, K. S., Hong, Z., “System and method for fixed-rate block-based image compression with inferred pixel values,” US Patent 5956431, 1997.

[IDV] http://www.unidata.ucar.edu/software/idv/.

[Jang 2002] Jang, J., J, Ribarsky, W., Shaw, C., Faust, N, “View-Dependent

Multiresolution Splatting of Non-Uniform Data," *In proceedings of the symposium on Data Visualization 2002*, 2002.

[JavaNEXRADViewer] http://www.ncdc.noaa.gov/oa/radar/jnx/index.php.

[Jiang 2001] Jiang, T., Wasilewski, T., Faust, N., Hannigan, B., Parry, M., "Acquisition and Display of Real-Time Atmospheric Data on Terrain," *Eurographics-IEEE Visualization Symposium 2001*, 15-24.

[Jonson 2004] Johnson. C., "Top Scientific Visualization Research Problems," *IEEE Computer Graphics and Applications*, vol. 24, no. 4, pp. 13-17, Jul/Aug, 2004.

[Kajiya 1984] Kajiya, J. T. and Von Herzen, B. P., "Ray Tracing Volume Densities," *Computer Graphics, 18, 3 (July 1984)*, 165-174.

[Kelleher 2007] Kelleher, K., Droegemeier, K. K., et al., "Project CRAFT: Technical Aspects of a Real Time Delivery System for NEXRAD Level II Data via the Internet," *American Meteorological Society*, 88, 1045-1057.

[Kniss 2003] Kniss, J., Kindlmann, G., Hansen, C., Shirley, P., McPherson, A., "A Model for Volume Lighting and Modeling," *IEEE Transactions on Visualization and Computer Graphics,* 9 (2), 109-116.

[Lacroute 1994] Lacroute, P. and Levoy, M., "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," *Proc. SIGGRAPH '94*, Orlando, Florida, July, 1994, 451-458.

[Laroche 1994] Laroche, S., Zawadzki, I., "A Variational Analysis Method for Retrieval of Three-dimensional Wind Field from Single-Doppler Radar Data," *Journal of Atmospheric and Oceanic Technology*, Volume 3, 77-86.

[Levoy 1988] Levoy, M., "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications,* Vol. 8, No. 3, May, 29-37.

[Lichtenbelt 1998] Lichtenbelt, B., Crane, R., Naqvi, S., "Introduction to Volume Rendering," Upper Saddle River, NJ: *Prentice Hall PTR*.

[Lindstrom 1996] Lindstrom, P., Ribarsky, W., Hodges, L. F., Faust, N., Turner, G. A., "Real-Time, Continuous Level of Detail Rendering of Height Fields," *Proceeding of SIGGRAPH '96, Computer Graphics*, 109-118.

[Lynch 1985] Lynch, T. J., "*Data Compression: Techniques and Applications,*" Belmont, CA: Lifetime Learning Publications.

[Nishita 1996] Nishita, T., Dobashi, Y., Nakamae ,E., “Display of Clouds Taking Into Account Multiple Anisotropic Scattering and Skylight,” *Computer Graphics Proceedings, Annual Conference Series: SIGGRAPH '96*, 379-386.

[NWS] http://www.nws.noaa.gov/radar_tab.php.

[OGC] http://www.opengeospatial.org/.

[OpenDX] http://www.research.ibm.com/dx/; http://www.opendx.org/.

[ParaView] http://www.paraview.org.

[Priegnitz 1995] Priegnitz, D. L., "IRAS: Software to display and analyze WSR-88D radar data," *Eleventh International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, 197-199.

[Qiu 1992] Qiu, C., Xu, Q., "A Simple Adjoint Method of Wind Analysis for Single-Doppler Data," *Journal of Atmospheric and Oceanic Technology*, Vol. 9, 588-598.

[Qiu 2001] Qiu, W., Mercer, R. E., Barron, J. L., "3D Storm Tracking in 3D Doppler Precipitation Reflectivity Datasets," *Irish Machine Vision and Image Processing Conference*, Volume 3, 77-86.

[Rana 2004] Rana, M. A., Sunar, M. S., Nor Hayat, M. N., "Framework for Real Time Cloud Rendering," *International Conference on Computer Graphics, Imaging and Visualization (CGIV'04),* 56-61.

[Riley 2003] Riley, K., Ebert, D., Hansen, C., Levit, J., "Visually Accurate Multi-Field Weather Visualization," *Proceedings of IEEE Visualization 2003*, 279-286.

[Riley 2004a] Riley, K., Ebert, D., Hansen, C., Levit, J., “A System for Realistic Weather Rendering," *Proceedings of the 84th American Meteorological Society Annual Meeting, January*.

[Riley 2004b] Riley, K., Ebert, D. S., Kraus, M., Tessendorf, J., and Hansen, C., "Efficient Rendering of Atmospheric Phenomena," *Proceedings Eurographics Symposium on Rendering 2004*, 375-386.

[Riley 2006] Riley, K., Song, Y., Kraus, M., Levit, J., and Ebert, D., "Visualization of Structured Nonuniform Grids," *IEEE Computer Graphics and Applications*, Vol. 26, No. 1, 24-33.

[Schpok 2003] Schpok, J., Simons, J., Ebert, D. S., Hansen, C., “A Real-Time Cloud Modeling, Rendering, and Animation System,” *Symposium on Computer Animation,* 2003.

[Schroeder 2003] Schroeder, W., Martin, K., Lorensen, B., “*The Visualization Toolkit: An Object Oriented Approach to 3D Graphics 34d Edition*”, Kitware, Inc.

[Song 2006] Song, Y., Yi, J., Svakhine, N., et al. "An Atmospheric Visual Analysis and Exploration System," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 5, 2006.

[Storer 1988] Storer, J. A., "*Data Compression: Methods and Theory*," Computer Science Press, Rockville MD.

[Treinish 1998] Treinish, L., "Task-specific Visualization Design: a Case Study in Operational Weather Forecasting," *Proceedings of IEEE Visualization 1998*, 405-409.

[TeraGrid] www.teragrid.org.

[Treinish 2000] Treinish, L., Praino, A., "Multi-resolution visualization Techniques for Nested Weather Models," *Proceedings of IEEE Visualization 2000*, 513-516.

[Treinish 2002] Treinish, L., "Interactive, Web-based Three-dimensional Visualizations of Operational Mesoscale Weather Models," *Proceedings of 18th International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology, American Meteorological Society*, J159-161.

[Treinish 2004] Treinish, L., Praino, A., "Customization of a MesoScale Numerical Weather Prediction System for Transportation Applications," *Proceedings of 20th International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology, American Meteorological Society*, J159-161.

[TRMM] http://trmm-fc.gsfc.nasa.gov/trmm_gv/software/rsl/index.html

[Ueng 2005] Ueng, S., Wang, S., “Interpolation and Visualization for Advected Scalar Fields,” *IEEE Visualization 2005 – (VIS’05)*, 78-88.

[Vis5D+] http://vis5d.sourceforge.net.

[Wang 2004] Wang, N., “Realistic and Fast Cloud Rendering,” *Journal of graphics tools*, 9(3):21-40.

[Watson 1995] Watson, R. J., Bebbington, D. H. O., “Combining ground based meteorological radar data from multiple overlapping sites,” *Geoscience and Remote Sensing Symposium, 1995. IGARSS '95. 'Quantitative Remote Sensing for Science and Applications', International*, Vol.3, 1660-1662.

[Westover 1990] Westover, L. A., “Footprint Evaluation for Volume Rendering,” *Proceedings of SIGGRAPH ‘90*, Vol. 24(4), 367-376.

[Wilson 1994] Wilson, O., VanGelder, A., Wilhelms, J., “Direct Volume Rendering Via 3d Textures,” *University of California at Santa Cruz, Santa Cruz, CA*, 1994.

[Zhou 2005] Zhou, Y., Stull, R., “Single-Doppler Radar Wind-Field Retrieval Experimentation on a Qualified Velocity-Azimuth Processing Technique,” *Ninth Symposium on Integrated Observing and Assimilation Systems for the Atmosphere, Oceans, and Land Surface, 2005*.

# LIST OF REFERENCES