4-27-2010

Large-Scale 3D Visualization of Doppler Reflectivity Data

Peter Kristof
Computer Graphics Technology, pkristof@purdue.edu

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Peter Kristof

Entitled Large-Scale 3D Visualization of Doppler Reflectivity Data

For the degree of Master of Science

Is approved by the final examining committee:

Chair Dr. Bedrich Benes	Dr. Gary R. Bertoline
Dr. X. Carol Song	

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Dr. Bedrich Benes

Approved by: Dr. James L. Mohler (Head of the Graduate Program) 4/23/2010 (Date)

Graduate School Form 20
(Revised 1/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:
Large-Scale 3D Visualization of Doppler Reflectivity Data

For the degree of Master of Science

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Teaching, Research, and Outreach Policy on Research Misconduct (VIII.3.1)*, October 1, 2008.*

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Peter Kristof
Printed Name and Signature of Candidate

4/26/2010
Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/viii_3_1.html

LARGE-SCALE 3D VISUALIZATION OF DOPPLER REFLECTIVITY DATA

A Thesis

Submitted to the Faculty

of

Purdue University

by

Peter Kristof

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2010

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

I would like to thank Dr. Bedrich Benes for his close collaboration and support throughout my graduate studies and for advising me on the computer graphics methods.

My gratitude goes to my supervisors Dr. Carol X. Song and Lan Zhao at the Rosen Center for Advanced Computing (RCAC) for their trust, help and financial support through RCAC group. I am grateful to Dr. David Braun for his suggestions and help with the application deployment and for providing me with hardware resources at the Envision Center.

In addition, I would like to extend my thanks to Dr. Xavier Tricoche for the discussions on the volumetric visualization techniques and rendering performance optimizations.

I would like to thank Dr. Gary Bertoline for serving on my thesis committee.

I am also grateful to Dr. Jeff Trapp and other graduate students at the Earth and Atmospheric Science department for participating in my user-study and engaging in insightful discussions on how the 3D reflectivity visualization can be used for better weather analysis.

Last, but not the least, I want to thank my family for their love, understanding and strong support throughout my graduate studies at Purdue University.

PREFACE

The Rosen Center for Advanced Computing (RCAC) research group together with the Envision Center for Data Perceptualization at Purdue University is aiming at providing high-quality rendering interface to enable multiple users interactively access, analyze and visualize the reflectivity data from all the Doppler radars in 3D to study near real-time weather events. The interface will be widely accessible to users ranging from general public to educational and scientific research groups.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

2D	two-dimensional
3D	three-dimensional
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DVR	Direct Volume Rendering
fps	frames per second
GPGPU	General-Purpose computation on Graphics Processing Units
GPU	Graphics Processing Unit
LOD	Level-of-detail
LRU	Least recently used
NEXRAD	Next Generation Radar
NOAA	National Oceanic and Atmospheric Administration
NSSL	National Severe Storms Laboratory
NWS	National Weather Service
RGBA	Color space of red, green and blue with opacity alpha
RLE	Run-length encoding
TDWR	Terminal Doppler Weather Radar
WSR-88D	Weather Surveillance Radar, 1988, Doppler

GLOSSARY

base reflectivity	reflectivity value from a single elevation scan
brick	a voxel grid of constant size M^3 (usually with $M = 32$)
brick pool	memory pool for storing bricks
composite reflectivity	strongest reflectivity energy at all elevation scans
cone of silence	the area above the radar's maximum elevation where the radar cannot see
mosaicking	combining reflectivity data from two or more radars
node pool	memory pool for storing tree nodes
out-of-core rendering	visualization of data sets that are too big to fit to GPU
radial velocity	velocity of particles moving relative to the radar
reflectivity	echo intensity measured in dBZ (decibels) representing the amount and type of precipitation
resolution sample	a reflectivity sample storing the geographic distance to the nearest non-equal reflectivity sample
spectrum width	variance of the doppler signals (i.e. turbulence)
volume scan	collection of radar data, repeated at regular time intervals

ABSTRACT

Kristof, Peter. M.S., Purdue University, May 2010. Large-Scale 3D Visualization of Doppler Reflectivity Data. Major Professor: Bedrich Benes.

The super resolution NEXRAD Level II Doppler radar data provides critical information on reflectivity, wind velocity and spectrum width for the entire United States. The goal of this work is to develop a framework that enables multiple users to interactively access, analyze and visualize the Doppler reflectivity data in 3D to study near real-time weather events. To provide interactive high-quality volumetric weather visualization, we combined two approaches dealing with large-scale storage of global weather data and out-of-core volume rendering using CUDA ray casting. The results of our work show that the reflectivity data from multiple radars can be preprocessed into data format that is efficient for large-scale volumetric visualization of reflectivity data in near-real time and requires minimal run-time processing.

1. INTRODUCTION

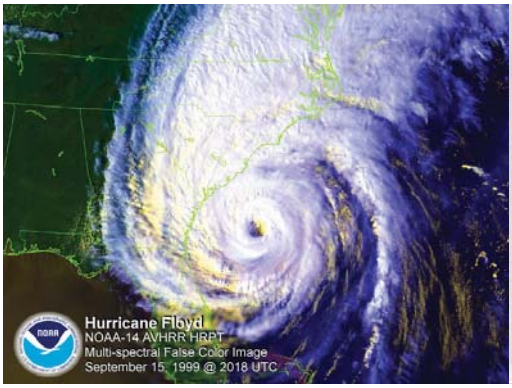
1.1 Severe Weather

Severe weather is a dangerous meteorological or hydro-meteorological phenomena that affects everyone on our planet. It often results in injuries, loss of human life and significant damage to property. Severe weather events impact our social life, economies, governments, wars - in fact, they affect the course of history itself. There are many different forms of severe weather, such as tornadoes, hurricanes, lightning, heavy precipitation and hailstorms. According to statistics (McNeill, n.d.) the United States face up to 1,000 tornadoes every year which result in an annual average of 1,500 injuries and 80 deaths. The strongest tornadoes can reach rotating winds of more than 250 mph and travel as far as 50 miles leaving behind evidence of its destructive force (Figure 1.1(b)). Figure 1.1(a) shows an example of a Mesocyclone tornado. Another 70 deaths p.a. and collosal property damage are caused by floods from thunderstorms and hurricanes. Hurricane Floyd (shown in Figure 1.1(c)) drenched the U.S. East Coast with 15-20 inches of rain in 1999 and resulted in property damage of estimated up to $6 billion Figure 1.1(d).

Weather forecasting can help reduce the pernicious effects of severe weather. If the area to be affected is provided with weather warnings and alerts ahead of time, people can take preemptive measures to protect their lives and properties. In order to provide such warnings, weather forecasters and researchers have to analyze the relevant atmospheric data (precipitation, temperature, wind speed, etc.) and make projections about the atmosphere's evolution. Nowadays, the weather forecasting takes advantage of modern computing and of weather visualization tools in order to achieve more accurate weather predictions. By providing better visualization tools,

(a) Mesocyclone tornado.

(b) Tornado aftermath.



(c) Hurricane Floyd.

(d) Aftermath of Hurricane Floyd.

Figure 1.1.: Examples of severe weather events and their effect on the environment (Images courtesy of NOAA).

which either show new information or assist in gaining new insights into the phenomena, we can enable researchers and forecasters to achieve a faster and more accurate weather analysis.

Figure 1.2.: NSSL's first Doppler Weather Radar located in Norman, Oklahoma. (Image courtesy of NOAA).

1.2 Background

1.2.1 Doppler Radar

The Weather Surveillance Radar-1988 Doppler (WSR-88D) is a pulsed Doppler radar used to detect meteorological and hydrological phenomena (Huber & Trapp, 2005). Figure 1.2 shows such a radar in the Oklahoma area. The Next Generation Radar (NEXRAD) network counts 159 radars at approximately $230km$ spacings across the continental United States (shown in Figure 1.3) and overseas areas. The purpose of the NEXRAD network is to provide three dimensional (3D) measurements of precipitation and winds at the highest temporal and spatial resolution to improve the forecasting ability of severe weather events (e.g. tornadoes, hurricanes and flash floods).

The output parameters of the radar data acquisition are discrete fields of: reflectivity (precipitation), radial velocity (wind) and Doppler spectrum width (turbulence). The radar data can be acquired from the National Weather Service (NWS) network in a compressed format, specifically in BZIP2. New data is fed to the NWS network at varying frequencies depending on the local weather conditions.

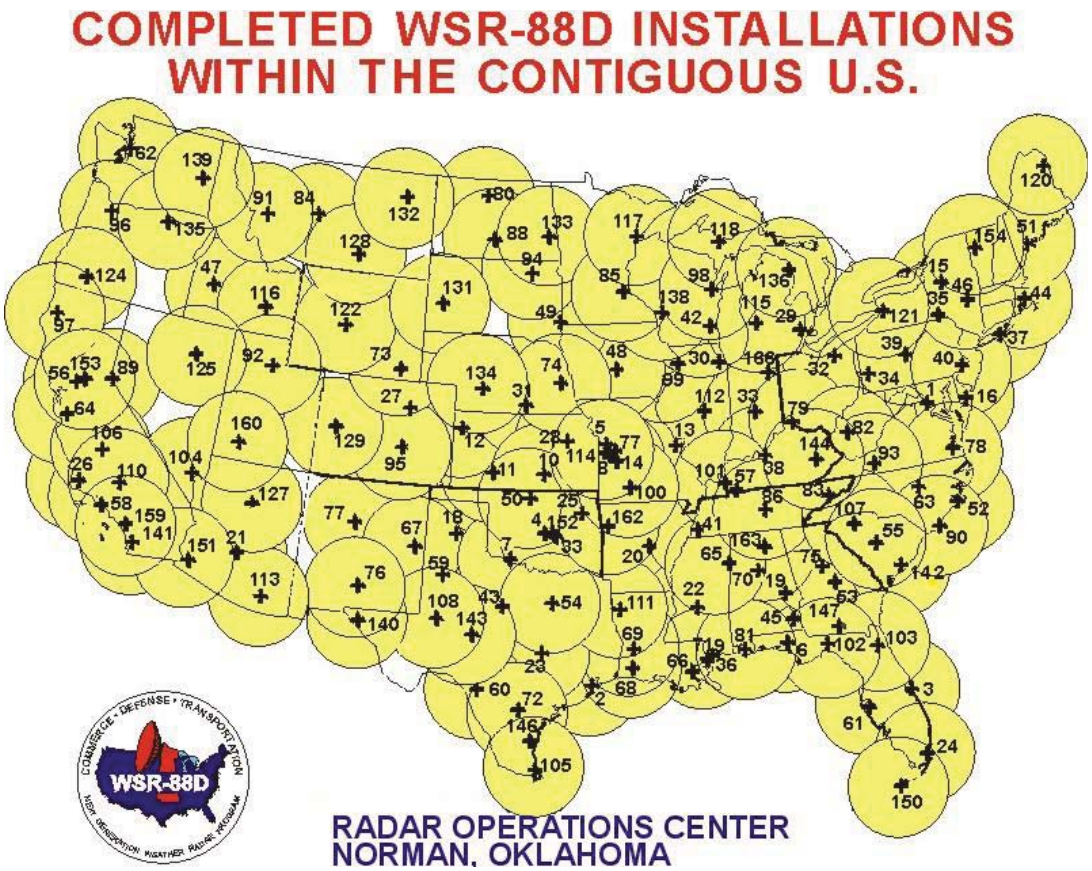


Figure 1.3.: Doppler radar sites in the continental United States (Image courtesy of NOAA).

Specifically, the radar can operate in one of two modes - clear air mode with time interval of 10 minutes or precipitation mode at which the images are updated every four to six minutes. The clear air mode has an advantage of scanning the atmosphere for longer time periods and, thus, enable the radar to scan with increased sensitivity. The collection of radar data, repeated at regular time intervals, is then referred to as a *volume scan*.

1.2.2 Doppler Reflectivity

The Doppler radar obtains the reflectivity information based on the returned energy. The radar rotates around its axis and emits bursts of energy for each ray. After each burst it listens for any scattered energy that is returned back to the radar. The emitted energy is scattered in all directions if it strikes an object (rain drop, hail, bird, building, mountain, etc.). The reflectivity is expressed in dBZ (decibels of Z) and is computed based on the strength of the returned signal and the time delay since it was emitted. The logarithmic scale of dBZ values is related to the intensity of precipitation. Typically, the higher the dBZ, the stronger the rain rate with the light rain occurring when the dBZ value reaches 20. Figure 1.4 shows an example of reflectivity visualization during a severe weather event. A value of 60 to 65 dBZ is about the level where 3/4" hail can occur. However, values of 60 to 65 dBZ may not necessarily represent severe weather, because of various reasons such as situations when the radar is out of calibration.

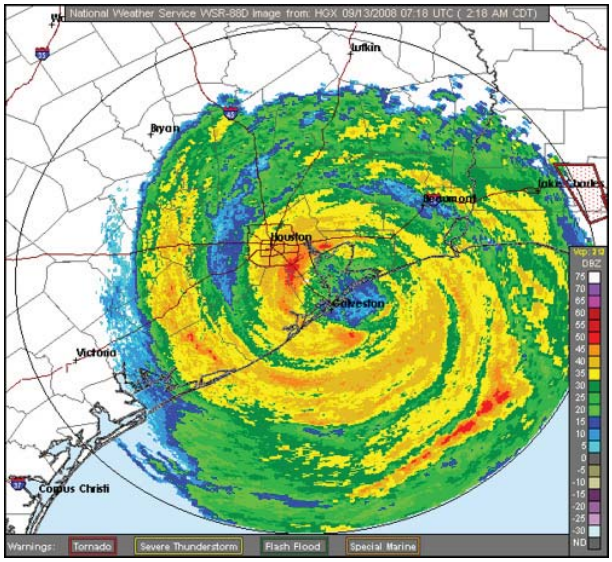


Figure 1.4.: Reflectivity visualization over the South Coast during the occurence of Hurricane Ike on 9/13/2008 (Image courtesy of NWS).

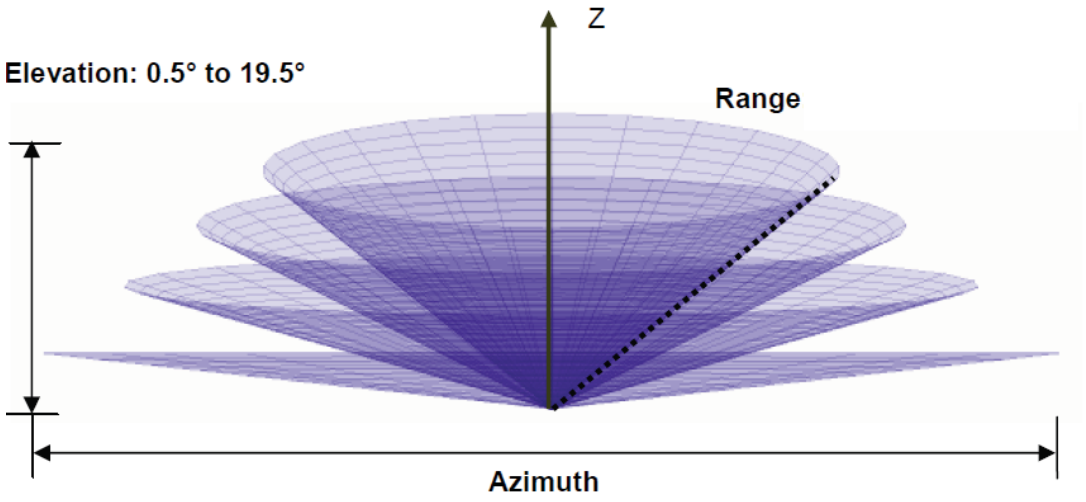


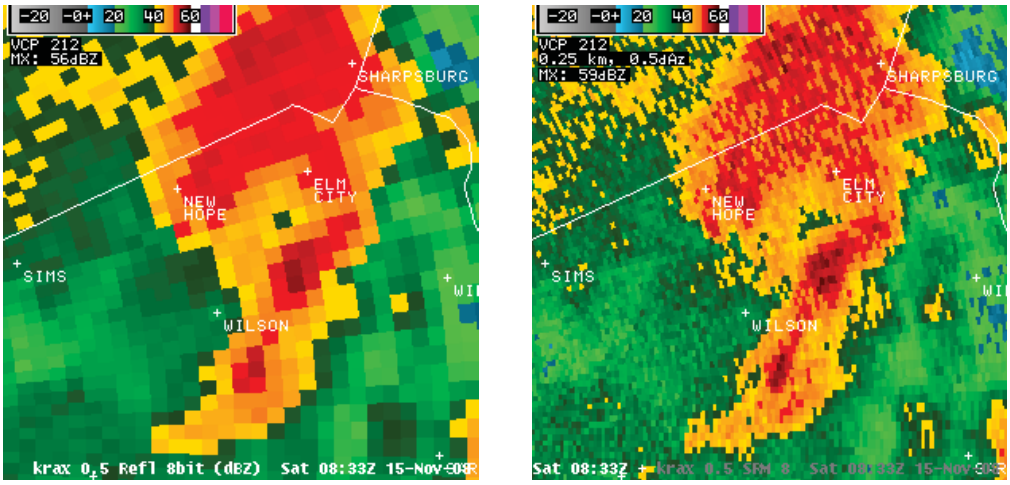
Figure 1.5.: WSR-88D volume scan structure (Image courtesy of Ru (2007)).

The 3D NEXRAD level II data is stored in the radar's spherical coordinates, consisting of elevation sweeps, azimuth rays and gates along each ray. This radar format is depicted in Figure 1.5. The number of sweeps per radar can be up to 14 with the elevation angle ranging from 0.5° to 19.5° depending on the local weather conditions. Basically, the more severe the weather the more elevations are used and the higher frequency the data is produced at. Some objects that do not reflect the energy sent from the radar very well (such as snow) the radar may switch to operate in clear air mode to scan with higher sensitivity.

The Legacy format of Doppler data has the following resolution setting. The range resolution is $1km$ for reflectivity and $0.25km$ for radial velocity and spectrum width. The radar's antenna continuously rotates over 360° in azimuth and the sampling angles are taken at elevation angles from 0.5° to 19.5°. The azimuthal resolution for all three parameters is one degree.

From 2008, the WSR-88D Doppler radars operate at higher super resolution at "split cuts" (scans at or below 1.5°) and thus the network provides higher 3D

spatial resolution than ever before. Specifically, the super resolution data provides increased reflectivity data resolution with reduced gate spacing from $1km$ to $0.25km$; increased azimuthal resolution of all three moments of data from 1° to 0.5°; and extended range of Doppler data from $230km$ to $300km$. The difference in resolution of reflectivity between the super-resolution and legacy format can be seen in Figure 1.6.



(a) Legacy resolution.

(b) Super resolution.

Figure 1.6.: Doppler reflectivity resolutions provided by WSR-88D radars severe weather events and their effect on the environment (Image courtesy of North Carolina State University).

1.3 Summary

Severe weather can have serious impact on our lives. One way to minimize its costs is by providing people with accurate weather predictions ahead of time. Interactive high-quality 3D weather visualization from multiple Doppler radars can help the forecasters to analyze the weather better and faster.

The rest of this thesis is organized as follows. Chapter 2 discusses the related work on doppler reflectivity processing, the methods for combining data from multiple radars, the visualization of the data and the recent work in the field of large-scale volumetric visualization.

In Chapter 3, we restate the problem and briefly outline our solution.

Chapter 4 contains explanation for the radar data preprocessing and transforming it to a global large-scale data structure that is effective for rendering.

Chapter 5 describes the visualization utilizing level-of-detail (LOD) techniques and Graphics Processing Units (GPUs).

In Chapters 6 and 7, we present results and additional techniques which can be implemented to either improve or extend the current visualization capabilities.

2. RELATED WORK

This chapter describes the work that has previously been done on the processing and visualization of Doppler reflectivity data. In addition, it provides details on the latest techniques for real-time large-scale volume visualizations using hierarchical data representation on GPUs.

2.1 Reflectivity Processing

Doppler radar weather data is stored in the radar's spherical coordinates. This is, however, inconvenient for combining the data from multiple radars (Xiao, Liu, & Shi, 2008). In addition, effective volume visualizations have been proposed for data sets stored in rectilinear coordinates (Crassin, Neyret, Lefebvre, & Eisemann, 2009). Jang et al. (2002) transformed and stored the reflectivity data in a geographic space. Although the reflectivity data has rectilinear arrangement in the spherical coordinates, conversion to geographic coordinates results in an inconvenient conical structure, as shown in Figure 1.5. This is due to beam spreading, which makes the reflectivity samples dense at the base of the radar and more spread with an increasing range resulting in ever larger data voids. One would have to undersample the data substantially to avoid data voids in a uniform 3D grid.

The missing data in data voids between elevation scans can be generated using one of the interpolation schemes (Mohr & Vaughan, 1979; Jorgensen et al., 1983; Jay Miller et al., 1986; Askelson et al., 2000; Weygandt et al., 2002; Shapiro et al., 2003). Creation of 3D multiple-radar grids using several interpolation methods was examined by Trapp and Doswell (2000); Askelson et al. (2000); Zhang, Howard, and

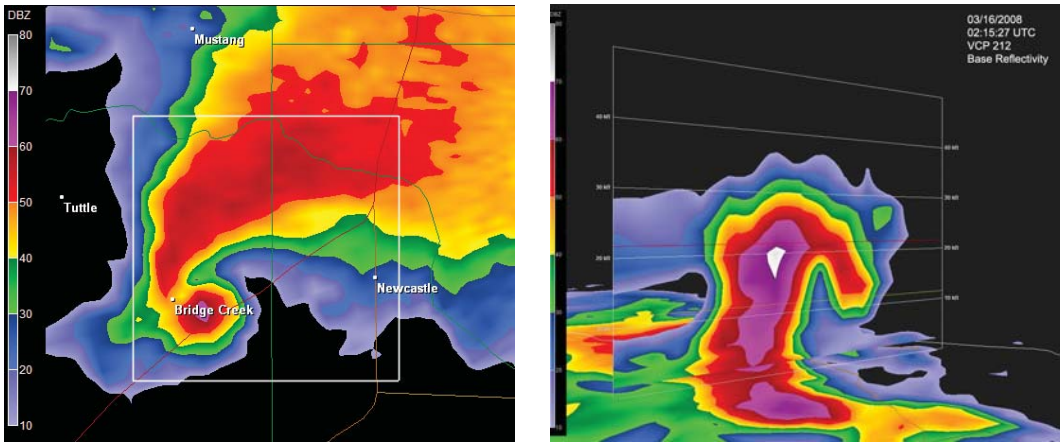
Gourley (2005); Xiao et al. (2008). Zhang et al. (2005) analyzed four interpolation schemes, including the nearest neighbor bin mapping technique and linear interpolation in either vertical, horizontal or in both directions. The authors summarized the choice of the interpolation scheme to be application-dependent and suggested vertical interpolation for convective storms and both vertical and horizontal interpolation with distance-weighted mosaicking scheme for general Cartesian grid mapping. In addition to these interpolation schemes, Xiao et al. (2008) evaluated a method using linear interpolation in all three dimensions, i.e. range, azimuth and elevation. The authors concluded the vertical interpolation with the nearest neighbor mapping on the range-azimuth plane results in the data most comparable to that of the raw data.

Furthermore, the structure of the Doppler radar scan has some other inherent problems such as beam height increasing due to the curvature of Earth and missing data acquisition below the lowest beam (e.g. at angle 0.5°) and above the highest beam (e.g. at angle 19.5°), which is referred to as the "cone of silence". These issues can be alleviated to some extent by way of combining the radar data from multiple sites of the NEXRAD network (Lakshmanan, Smith, Hondl, Stumpf, & Witt, 2006). In addition, the resolution can be potentially increased by using data from the FAA Terminal Doppler Weather Radar (TDWR) network, which covers areas around major airports (Vasiloff, 2001). There have been several methods proposed for creating mosaics from multiple radars, such as the nearest neighbor mapping, maximum value and by applying a distance weighted function (Zhang et al., 2005; Xiao et al., 2008). Xiao et al. (2008) suggested applying the distance weighted approach using an exponentially decaying function for the analysis of reflectivity data.

2.2 Reflectivity Visualization

The two-dimensional (2D) visualization has been used as the main means of rendering the Doppler reflectivity data. This is done by projecting a single reflectivity value onto a horizontal plane (as used in Figures 1.4, 1.6 and 2.1(a)). The projected reflectivity value can represent either a base reflectivity, which is a value from a single elevation scan, or a composite reflectivity accounting for the strongest reflectivity from any elevation angle at every range.

However, the weather data may contain additional information, such as formation of deep convection and 3D tornado structure, in the vertical domain that is of interest to researchers and forecasters in the context of weather analysis. The simplest way to visualize the vertical domain (i.e. altitude) is by sampling the reflectivity field with a single vertical plane. This is depicted in Figure 2.1(b). The volumetric



(a) 2D visualization.

(b) 2D visualization with a vertical slice.

Figure 2.1.: 2D reflectivity visualization using GRAnalyst2 (Image courtesy of Gibson (2010)).

visualization of Doppler radar data has predominantly been done only for a single radar in order to minimize the size of the input data. Second reason for this is to avoid computation of combining radar values from overlapping radars. Djurcilov

and Pang (1999) created a 3D visualization of Doppler reflectivity using isosurfacing. GRAnalyst2 software (Gibson, 2010) used the texture slicing method to visualize volume of a single radar station (shown in Figure 2.2).

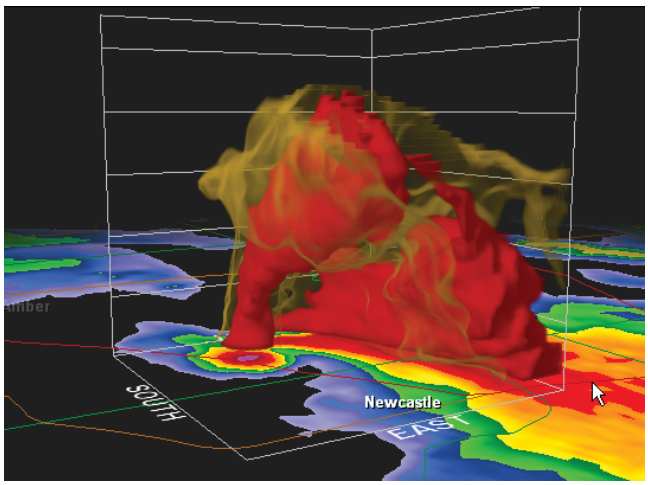
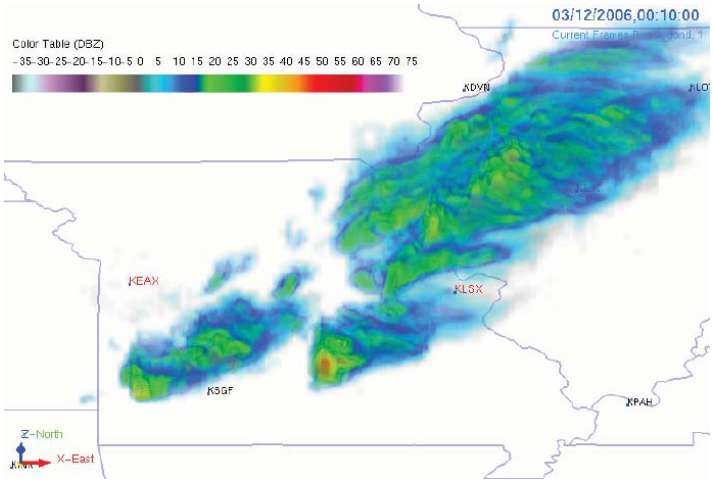


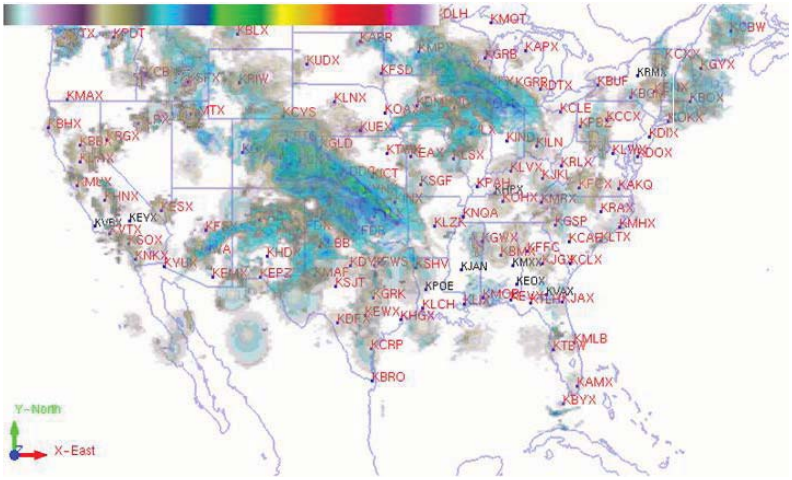
Figure 2.2.: 3D visualization of reflectivity data from a single radar using GRAnalyst2 (Image courtesy of Gibson (2010)).

Ru (2007) combined reflectivity data into a pre-defined 3D rectilinear grid and displayed it using the texture-slicing with a post-interpolative transfer function on the GPU. The 3D grid is created by way of merging the radar data from radar sites that are within the user selected area. Although, the visualization can be of acceptable quality in some cases, such as for visualization of a small area depicted Figure 2.3(a), in general, the approach suffers heavily in alias caused by undersampling, data resolution, and run-time performance in general. First, the 3D grid is merged at run-time and, thus, has to be recomputed for every new selection of area, in which the reflectivity is to be displayed, and as a result the computation can take up to tens of minutes on a single machine. Second, the visualization quality is heavily dependent on the grid's uniform resolution, which is limited by memory constraints and does not account for the vast empty regions. This is especially true for displaying mid to large areas, when the resolution of the grid is not fine enough, as shown in Figure 2.3(b). The resolution problem is further exacerbated on zoomed

views. Third, the post-interpolative classification for volume rendering requires a high sampling rate whenever there are high frequencies either in the actual data or in the transfer function. If the sampling rate is not high enough, distracting aliasing artifacts may appear. These are shown in Figure 5.3(a).



(a) A small region.



(b) The whole U.S. area.

Figure 2.3.: Volumetric reflectivity visualization using the technique by Ru (2007) at different area scales.

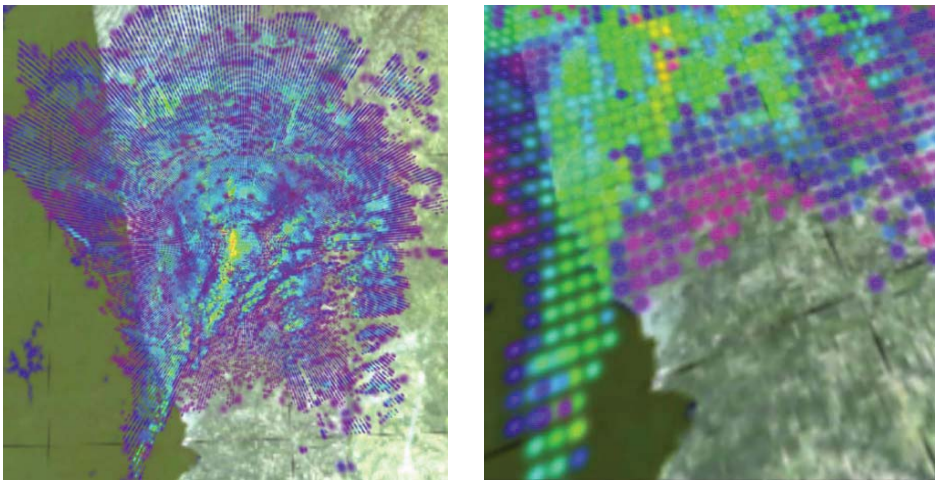
Lakshmanan et al. (2006), similarly to the work of Ru (2007), employed a merging technique by way of using intelligent agents, which act as autonomous particles

within the system and are advected by storm motion estimates. These agents are created at radar gates and are used for more physically accurate computations of reflectivity values in the final 3D grid. But again, the merging is done at run-time and it requires substantial computational resources to provide data in a reasonable time.

To render the reflectivity volumes from all the radars interactively, using a 3D grid representation is not efficient, considering that the weather data is usually sparse and large. A grid of reflectivity values covering the whole continental U.S. at the resolution equal to the gate size (i.e. $250m$) of super-resolution scans by Doppler radars would require tens of gigabytes of data. Moreover, even this resolution may not be sufficient to capture all the details in the measured data set because of the conical structure of volume scans, in which the reflectivity samples can be as close as few meters.

To provide interactive exploration of the radar data, Jang et al. (2002) applied a LOD technique by way of using multi-level hierarchical data structure. The data structure matches the geographic nature of the thin coverage of reflectivity data around the globe. The authors employed elliptical point-splatting to do volumetric visualization, which is shown in Figure 2.4(a). However, the visualization suffers from gaps between the reflectivity samples at the highest resolution (see Figure 2.4(b)). Ribarsky, Faust, Wartell, Shaw, and Jang (2002) further developed this method to create a framework capable of capturing multiple time steps and other types of data, such as satellite imagery, within the same data structure.

In climate research, there are several visualizations which are commonly used, such as IDV, GrADs, Vis5D+, Cave5D, GMT, ODV, NCAR Graphics, GMT, Avizo, GRAnalyst2, etc. An overview of these tools can be found in Ru (2007); Nocke, Sterzel, Bttinger, and Wrobel (2008).



(a) Visualization over North Georgia. (b) Close-up view.

Figure 2.4.: 3D visualization of reflectivity field using elliptical point splatting (Image courtesy of Jang et al. (2002)).

2.3 Large-scale Volumetric Visualization

The direct volume rendering (DVR) was described by Kajiya and Von Herzen (1984) more than 25 years ago. Since then, the volumetric rendering has been a subject of active research in computer graphics and various methods have been introduced. The DVR methods fall into two categories: object-order methods (splatting (Westover, 1990), shear-warp (Lacroute & Levoy, 1994), 3D texture slicing (Wilson, VanGelder, & Wilhelms, 1994)) and image-order methods (e.g. ray-casting (Levoy, 1988)). Too much work has been done on developing real-time algorithms from these base methods to be covered in this thesis. We refer the reader to a recent overview of real-time volumetric visualization techniques by Hadwiger, Kniss, Rezk-salama, Weiskopf, and Engel (2006).

Interactive methods for large-scale volume visualizations have become possible with the introduction of programmable graphics hardware. The first implementations of GPU ray casting (Kruger & Westermann, 2003; Roettger, Guthe, Weiskopf, Ertl, &

Strasser, 2003) were published in 2003. Kaehler, Abel, and Hege (2007) applied GPU ray casting to do near-interactive visualization of medium-scale datasets by using ray casting on the GPU. With further advancement in the computer graphics hardware and application of LOD techniques and multi-resolution octrees or N^3 trees it was shown that the volume rendering of large data sets can now be done in real-time on current GPUs (Gobbetti, Marton, & Guitian, 2008; Lux & Fröhlich, 2009; Crassin et al., 2009). Crassin et al. (2009) presented an efficient streaming of data to the GPU with a low computational demand on the CPU. The streaming is guided by the information computed during ray casting. The authors used the KD-restart algorithm (Foley & Sugerman, 2005), which starts at root node for each node lookup, for octree traversal.

Traditional stack-based tree traversal algorithms adapt poorly to GPUs due to the stack's memory requirements for each thread (Foley & Sugerman, 2005). Instead, the KD-restart can be performed very fast and the hardware texture cache sufficiently hides the penalty for repetitive node lookups in the memory (Crassin et al., 2009). In addition to the KD-restart algorithm, Foley and Sugerman (2005) introduced another stack-less traversal algorithm for kd-trees on GPUs, called the KD-backtrack. The KD-backtrack method requires an additional pointer to a parent at each node and allows for a faster traversal at the cost of increased memory requirements. More efficient kd-tree traversal algorithms for ray casting on GPUs were presented by Horn, Sugerman, Houston, and Hanrahan (2007). In contrast to Foley and Sugerman (2005), they proposed a Short-Stack based algorithm, which gives a 1.5 to 3 times increase in ray throughput at an additional small memory requirement. They also described two additional algorithms: Packets and Push-Down, both of which are slightly slower than the Short-Stack algorithm.

3. PROBLEM STATEMENT AND PROPOSED SOLUTION OUTLINE

3.1 Statement of the Research Problem

Developing an interactive 3D visualization of Doppler reflectivity data from all the radar sites and for multiple users presents the following problems:

1. Displaying the 3D reflectivity data from multiple radar sites simultaneously is problematic (Ru, 2007). This is because the WSR-88D Doppler radars at different locations operate at dissimilar paces and the radar data is generated asynchronously.

2. The radar's native spherical coordinates are impractical for visualization of multiple radar sites. The data should be pre-processed and converted into a data structure that supports effective 3D visualization. This requires heavy computational load, which if done on run-time, substantially limits the interactivity and number of users that can be supported at once.

3. The conical arrangement (shown in Figure 1.5) of the measured radar data makes the data very dense close to the radar and sparse at the end of rays in the geographic coordinates. It is non-trivial to create an adaptive data structure which is capable of capturing the high resolutions and which is effective for the purposes of the visualization at the same time.

4. Visualization of large volumetric data remains a challenging problem in many scientific applications.

In view of these technical difficulties, the purpose of our study is to provide a high-quality 3D visualization which enables multiple users to interactively access, analyze and visualize the reflectivity data from all the Doppler radars in 3D in order to study near real-time weather events.

3.2 Outline of the Solution

The proposed high-quality 3D visualization of Doppler reflectivity data from multiple radars can be displayed at interactive frame rates by storing the data in a global multi-resolution hierarchical data structure (Jang et al., 2002) and by applying efficient GPU rendering (Crassin et al., 2009) that utilizes LOD support of the data structure.

The advantage of using a hierarchical data structure is that it can adapt to the input data and focus processing and high-resolution sampling merely on the non-empty regions. Nodes, storing a small constant-sized 3D grids will be used for storing the volume data so as to utilize efficient hardware-based data filtering during rendering. Even though this may result in some oversampling of the input data due to the grid representation being the lowest building block, any empty cells will be filled out using the vertical interpolation with the nearest neighbor mapping for the range-azimuthal plane (Xiao et al., 2008). Furthermore, the overlapping data from multiple radars will be combined together using the distance weighted function (Xiao et al., 2008).

The new radar data will be pre-processed into the hierarchical data structure at regular time intervals. Then multiple users can be provided with data in near real-time by directly streaming the pre-generated data from the data repository *to the clients*. Last but not least, the heavy pre-processing can be easily parallelized thanks to the multi-level layout of the hierarchical data structure (Jang et al., 2002).

4. DATA PREPROCESSING

This chapter describes the algorithm used for preprocessing of the input reflectivity data into a global data structure, which is then used for interactive volumetric visualization. The data structure is based on a hierarchical data structure for global 3D atmospheric data (Ribarsky et al., 2002). We introduce a resolution estimation method for adaptive sampling of the input data in order to retain the high frequencies in the input data while keeping the size of the stored data to a minimum.

4.1 Data Structure

The radar reflectivity data is stored in a hierarchical data structure in Geographic coordinates to provide LOD functionality. Considering the fact that the Doppler radars scan as far as 230km at the highest elevation of 19.5°, the altitude is sampled merely up to approximately $75km$. This makes the data coverage very thin in the altitude dimension around the globe. Therefore, sampling such 3D space in a uniform manner would result in oversampling the altitude dimension to match desired resolution in the remaining dimensions, namely longitude and latitutde. In addition, the reflectivity data is sparse and uniform storage would result in an unnecessarily high memory footprint. Therefore, we chose to store the atmospheric data in a multi-level hierarchical data structure similar to the one presented by Jang et al. (2002).

In the authors' approach, the top level structure is a forest of quadtrees covering the entire earth. However, in the present study, we process the reflectivity data from Doppler radars across the continental United States covering smaller geographic

area and, thus, we do not use the forest of quadtrees. The whole hierarchy is depicted in Figure 4.1 below and has the following multi-level layout.

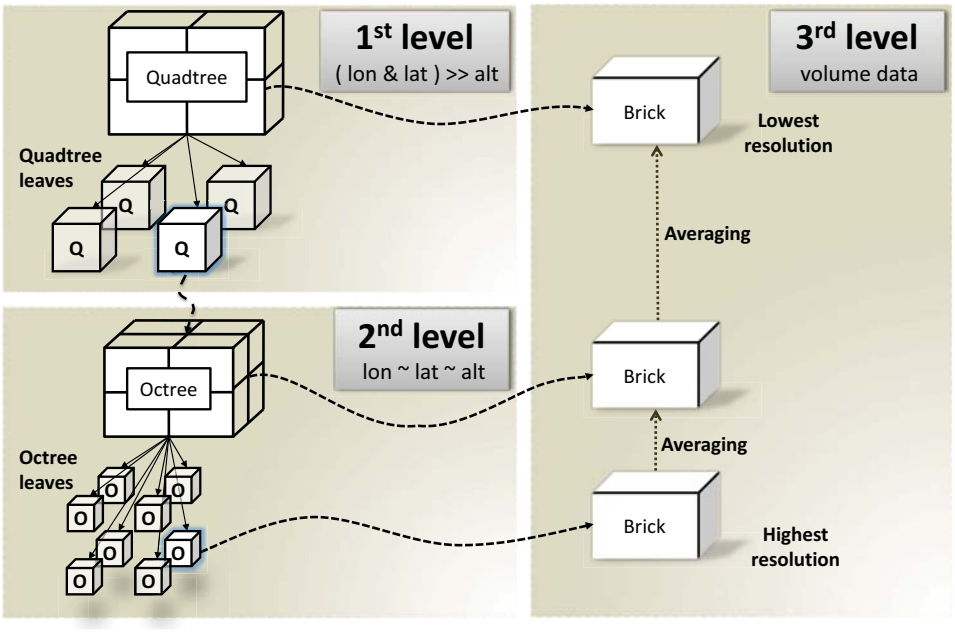


Figure 4.1.: Multi-resolution hierarchical data structure for reflectivity data.

The first level is formed by a lat/lon quadtree, which is refined until the lat/lon dimensions are of the same magnitude as the altitude dimension. This way each non-empty quadtree leaf essentially represents an almost cubical volume space and is very suitable for uniform subdivision in all three axes. In particular, the second level is represented by an octree and is further subdivided until a desired data resolution is met. Each Octree leaf stores a pointer to a *brick*, which is a small 3D grid of predefined size M^3 (generally $M = 32$). The bricks in the octree leaves represent the highest resolution of the stored volume data. The idea behind using small constant sized 3D grids is to allow for fast grid-based ray casting over these small volumes and utilize hardware accelerated data interpolation (Crassin et al., 2009). In view of the fact that we want to be able to sample various resolutions during the visualization, all the parent nodes in the octree and quadtree have a

brick associated with them as well. These bricks represent lower resolutions of the volume data and are built per node by averaging the bricks of the node's children.

The desired resolution for an octree leaf is such that the brick cell contains at most one reflectivity sample and it is estimated by analyzing the data resolution locally around each reflectivity sample. In particular, the resolution is computed based on the geographic distance to the nearest neighbor that has a different reflectivity value than the source sample. To avoid finding the neighbors and computing distances for every resolution query, we generate a *resolution sample* for every non-zero reflectivity sample during the initial stage of the preprocessing.

Similarly to Crassin et al. (2009), we store the tree nodes in a 3D texture, referred to as the *node pool*. The data in the nodes are repartitioned to take only 64 bits so as to lower the memory requirements and improve the data coherence, which in turn helps the texture caching. The bit structure of a node is explained in Table 4.1. Each node either contain a brick or a single data value and stores a pointer only to the first child. The remaining children are stored right after the first child, so that no more pointers are necessary.

Total bits	Description
29	pointer to a first child
1	whether node is a leaf
1	Indicates whether the node is refined to maximum, or the original volume still contains more data: - For Quadtree node, Octree node/leaf: whether the brick has been loaded - For Quadtree leaf: whether the octree has been loaded
1	Stores whether the content is a single reflectivity value or described by a brick
18	Brick index or single reflectivity value
7	Minimum reflectivity value in the subtree of the node
7	Maximum reflectivity value in the subtree of the node

Table 4.1: The bit representation of a tree node.

Bit encoding Limitations

4.2 Building

An overview of the whole preprocessing step is shown in Figure 4.2.

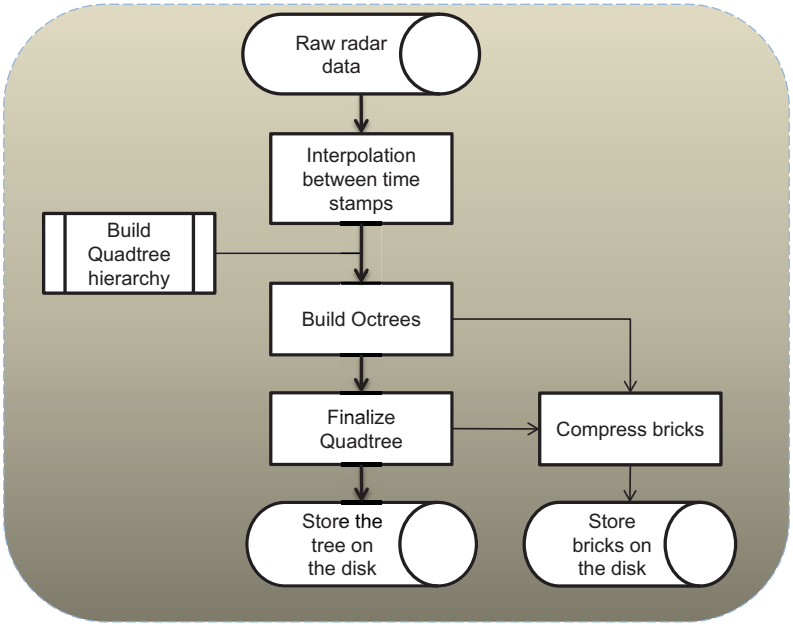


Figure 4.2.: Data processing overview.

First, the raw radar data has to be synchronized, because the radars operate at different time intervals and different scanning speeds. Thus, each radar is interpolated in time to provide data at the same time stamp. Then, we generate a file containing the resolution samples for each radar.

Next, the data structure described in Section 4.1 is built in three stages as depicted in Figure 4.2. In the first stage, the quadtree is built over all the radar sites and each quadtree leaf then stores a reference to radars that are within the radar radius from the leaf. In the second stage, the octrees and their bricks are built. In particular, an octree is built using all the resolution samples that are within its

bounding box. The octree is subdivided until the maximum data resolution for a brick has been met. The data resolution is evaluated by finding the resolution sample with shortest distance. Then a brick is built for the leaf by computing reflectivity values for each cell from all the contributing radars. After all the leaf bricks are computed, the bricks for parent octree nodes are built by averaging the children bricks in a bottom-up fashion. This is then repeated for the quadtree nodes after the all the octree nodes have been built.

The most computationally demanding part of the tree building process is building the leaves' bricks. For each brick, we have to compute reflectivity values from contributing radars for M^3 (i.e. for $M = 32$ that is 32768) brick cells. The computation per brick cell include converting the cell's geographic position to the radar's spherical coordinates (computed by using several computationally expensive trigonometry and squared root functions) and interpolating the neighboring values using the vertical interpolation with nearest neighbor mapping in the azimuth-range plane as suggested by Xiao et al. (2008). Contributions from multiple radars are resolved using the distance weighted function (Xiao et al., 2008). Postponing the interpolation up to the latest stage results in higher data accuracy then in the technique by Ru (2007), who first mapped the reflectivity samples to a grid introducing a numerical error, because the reflectivity samples may not be completely aligned with the center of a grid cell and then interpolated missing values from these grid cells.

Last, before the bricks are stored to a hard-disk, they are compressed using the Run-length encoding (RLE). This compression technique has proved to be very efficient for compressing 3D memory blocks of reflectivity data (Ru, 2007).

4.3 Summary

In this chapter, we described the multi-level tree-like data structure to store the reflectivity data from multiple radars. The leaves of the tree structure contain bricks, which provide the highest resolution of the input data. The nodes, in contrast, store bricks that represent lower resolutions of the data, so as to provide multiple resolutions.

The building process of the data structure is split into three stages, namely construction of the quadtree hierarchy, complete build of all the octrees and finalization of the quadtree, which includes linking of the octree data generated in the second stage. After the data structure is built, the brick data is compressed using RLE and stored on a disk.

Next chapter presents a way how this data structure can be utilized for effective volume visualization.

5. VISUALIZATION

This chapter describes our method for large-scale interactive 3D visualization of reflectivity data using the multi-resolution hierarchical data structure introduced in Chapter 4. The key to a successful interactive large-scale visualization is using LOD techniques to limit the loaded data to fit the memory constraints, also referred to as *out-of-core rendering*. The data management is guided by the Least recently used (LRU) algorithm on the CPU. The LRU table is updated with the tree nodes' usage information collected during GPU ray casting. The whole rendering pipeline is described in Figure 5.1.

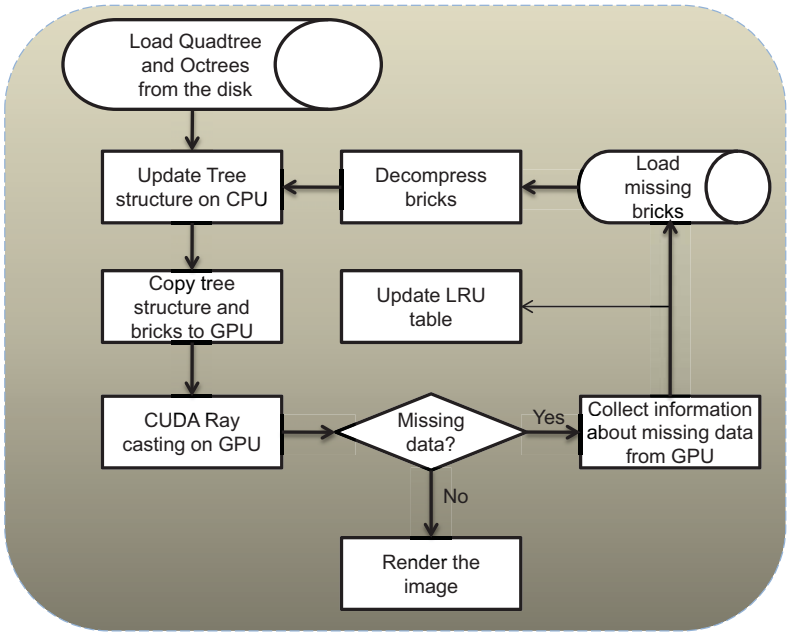


Figure 5.1.: Visualization overview.

To create high-quality interactive visualization we have decided to employ volume ray casting. Our visualization approach is based on the recent technique presented by Crassin et al. (2009) using large-scale GPU ray-guided ray casting. The next section shall briefly explain the volume rendering and will provide details on ray casting in the hierarchical data structure described in Section 4.1.

5.1 Direct volume rendering

The DVR seeks to visually extract information from the 3D discrete data, where each sample has a potential to contribute to the final image. The volume data is considered to consist of density particles, in which the light gets absorbed and emitted depending on the assigned optical properties. In the simplest case the optical properties consist of color C and opacity α. This emission-absorption model is expressed by a volume rendering integral (Hadwiger et al., 2006), which has the following form:

$$I(D) = I_0 e^{-\int_{s_0}^{D} \kappa(t)\mathrm{d}t} + \int_{s_0}^{D} q(s) e^{-\int_{s}^{D} \kappa(t)\mathrm{d}t} \mathrm{d}s \tag{5.1}$$

where I_0 is the initial intensity at s_0, κ is the absorption coefficient and q is the source term describing emission. The first term represents the attenuation of the incoming light while the second interprets the emitted light (including self attenuation). The discretized volume-rendering integral is computed by an iterative method with a front-to-back (from the eye point to the volume) compositing scheme (Hadwiger et al., 2006), which is generally used for the ray casting method:

$$C'_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst}) C_{src} \tag{5.2a}$$

$$\alpha_{dst} \leftarrow \alpha_{dst} + (1 - \alpha_{dst}) \alpha_{src} \tag{5.2b}$$

where the $_{dst}$ is the value at current location and $_{src}$ is the previous composited value.

5.2 Large-scale ray casting

Ray casting is an image-based DVR method (Levoy, 1988). The heart of this method lies in casting independent rays into the volume from each pixel in the image and accumulating color and opacity. Its main advantage lies in the ability to treat rays independently. Many times, large parts of a volume data set may not even contribute to the final image due to either being completely transparent or not visible (Hadwiger et al., 2006). In view of the fact that each ray can be computed independently with regard to the other rays it is possible to employ optimization strategies, such as adaptive sampling, empty space skipping and early ray termination. Another advantage of this method is that it allows for precise floating-based blending operations to create high-quality volume composites. In addition, if need be, ray casting is flexible enough to be extended for the visualization of other phenomena, such as scattering.

The algorithm for large-scale ray casting on GPU is performed in a loop until the ray leaves the volume. First, each ray is initialized and then the volume integration is done in the loop consisting of following steps:

Traverse the tree The tree hierarchy is traversed until the node providing desired resolution for the current position p is found.

Convert ray into the node's space Ray's position p is converted into position p_B, which is relative to the node's brick space, so that $p_B \in [0,1]^3$.

Do volume ray casting in the brick The ray is casted through the $[0,1]^3$ volume of the brick until it leaves its space. The color and opacities are collected along the integrated ray inside the brick based on the used transfer function classification.

Update the ray position The integrated distance in the brick is converted to the tree root's space and the ray's position p is updated. This position then serves as the input position to the next iteration.

Check for ray termination The ray is terminated when it leaves the tree's volume. Also to avoid negligible computations, the ray is terminated when the accumulated opacity reaches a satisfactory value, such as $\alpha = 0.95$, for which any further volume integration would have a minimal effect on the final color, because the assigned color is already almost opaque at 0.95.

5.2.1 Tree traversal

Similarly to Crassin et al. (2009), the ray traversal of the hierarchical data structure is done from the tree root by *kd-restart* algorithm (Foley & Sugerman, 2005). The tree traversal is computationally efficient because the point coordinate p can be directly used to locate it within a node. Let $p \in [0, 1]^3$ be the point's local coordinates in the quadtree's bounding box, c be the pointer to the first child of the root and assuming the tree hierarchy is stored in a 3D texture. The offset to a child, to which the p falls, is $(int)(p * 2)$ (read integer part of multiplication $p * 2$) for p_x and p_y coordinates within a quadtree node and p_x, p_y and p_z within an octree node. Then, pointer to the child is simply $c + (int)(p * 2)$.

The descent is iterated until either a leaf or a node with the desired resolution is reached. The criterion for the resolution is that one voxel projects to at most one pixel. If the node represents a single color the volume integral is computed analytically for the volume of the node. Otherwise, the node has a brick associated with it and standard ray marching is applied until we leave the node. However, it should be noted that the ray direction d changes as we descend within the quadtree because it is only x and y dimensions that are subdivided. In an octree, where all the dimensions are subdivided at the same time, the d is constant. The integrated

distance, which is expressed in the node's local space $[0, 1]^3$, for the ray is transformed into the quadtree's root local volume space and the p is moved according to that distance along the ray. The new p then constitutes an input to the next descent.

5.2.2 Data Management

What we said about ray casting in the previous section should be done a bit differently to allow for out-of-core rendering.

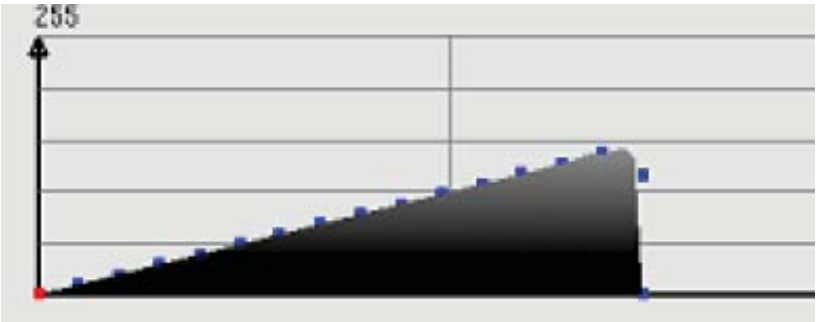
Specifically, during a ray casting pass we collect information on which nodes have been used or need to be loaded. In our case, the tree hierarchy is small enough to be kept constant during rendering and therefore the whole tree hierarchy is loaded at the beginning. However, the bricks, which store the actual volume data, can take up to tens of gigabytes in uncompressed format and have to be loaded only when they are required. For this purpose, we keep an array of flags with a flag for each node. The flag can have three states. First, the node was not reached during ray casting. Second, the node was reached but the brick is missing from the working set on the GPU. When this happens during ray casting we move the position p on the ray out of the node and continue with the rendering. Third, the node was reached and the brick is available. After the ray casting is finished, we copy the 2D array of node flags to the CPU and update the LRU table accordingly. If the node was visited, its priority is increased in the LRU table. After that, the missing bricks are loaded and if the brick pool is full we remove the bricks with lowest priorities from the LRU table.

It should also be noted that both the *node pool*, which contains the all the nodes of the tree, and the *brick pool* on the GPU are transformed into 3D layout and stored stored in 3D textures on the GPU. This is to improve the locality of the data on the GPU and, thus, make the caching more effective.

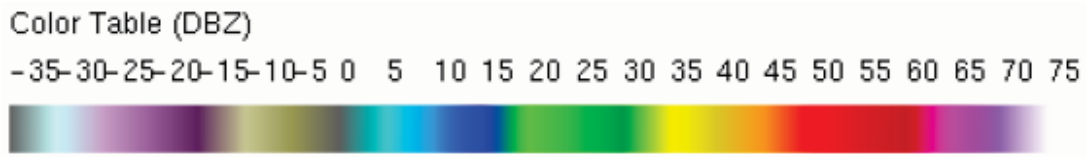
5.3 Transfer function

Mapping of volume data to optical properties (such as color and opacity) is expressed by a transfer function, which is also referred to as a color map or color table. Essentially, it guides the volumetric visualization to hide unimportant features or highlight the data of interest in the final image. Two of the inherent problems of volume rendering are visual clutter and data occlusion, both of which can be addressed to some extent by defining an appropriate opacity function. In the former method, Ru (2007) developed a transfer function interface (shown in Figure 5.2(a)), where the function is defined by modifying the control points of the cubic Hermite spline.

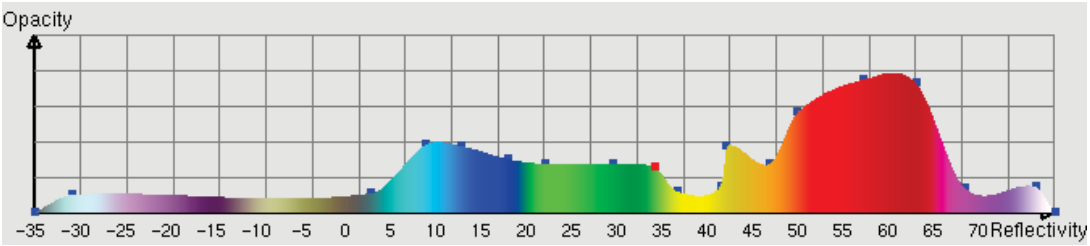
The cubic spline approach removes the burdensome task of specifying every point on the x-axis and allows for easy definition of complex high-frequency transfer functions. To make the interface even more user-friendly, we have introduced two changes. First, the axes had a proper name assigned and had the grey background of the opacity function changed to the color of the remaining RGB functions (to see the end result we refer to Figure 5.2(c)). These changes make the modification of the opacity function much more intuitive and interpretive. This is especially true for a first time user. Secondly, we added an interface for storing and retrieving predefined functions. These functions may be used to speed-up analysis of the volume data. The visualization is also dependent on the actual volume data and, thus, it is difficult to generalize the perfect transfer function. However, the desired function may be tuned from a predefined function relating to what the user seeks to achieve faster.



(a) Transfer function interface using cubic Hermite splines for opacity (Image courtesy of Ru (2007)).



(b) Reflectivity to RGB color mapping (Image courtesy of Ru (2007)).



(c) Our transfer function GUI interface. It enables to set opacity value on the vertical axis for each reflectivity value on the horizontal axis. The inside area of the function is colored according to the set colors for each reflectivity value.

Figure 5.2.: RGBA transfer function for the volume rendering of reflectivity data.

5.4 Pre-integrated classification

The transfer function may contain high frequencies, for which a high sampling rate would be necessary so as to avoid visual artifacts (shown in Figure 5.3). Engel, Kraus, and Ertl (2001) presented a way to account for high frequencies in a transfer function without increasing the volume sampling rate. The authors introduce a preprocessing step for texture-slicing based volume rendering. In particular, this is

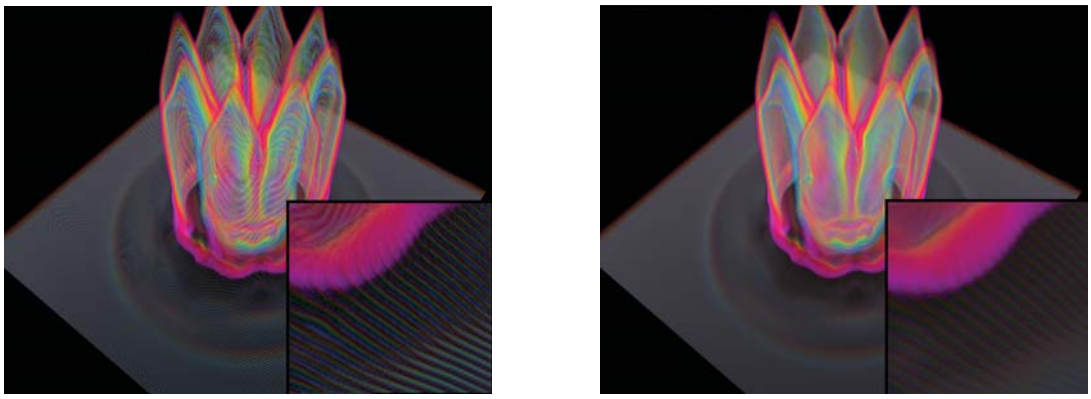
achieved by way of computing all possible combinations of the transfer function integrations between two samples for the used transfer function. The result is then stored in a texture and is subsequently sampled by the ray caster in a rendering pass. In this manner, the ray integration accounts for higher frequencies that would otherwise be missed or aliased due to a low sampling rate of a transfer function. Thus the benefit of pre-integrated classification is precise integration even during undersampling of the volume. Moreover, it is also much better to use for the empty space skipping algorithm, described in Section 5.5.

The volume ray integration is done by way of using a pre-integrated classification. The only drawback of the pre-integrated classification is that the sampling step size has to be known for computing the pre-integrated table. Therefore, to allow for adaptive sampling it is necessary to create pre-integrated tables for each sampling step size size.

Comparison of the traditional post-interpolative (Hadwiger et al. (2006)), which was also used in the previous method of Ru (2007), and the applied pre-integrated classification can be seen in Figure (5.3). Further improvement could be achieved by stochastic jittering (Hadwiger et al., 2006) in order to hide the slice artifacts due to resulting from the synchronized ray stepping for all rays. I found the slicing artifacts to be minimal after applying the pre-integrated classification.

5.5 Performance strategies

By using hierarchical data structure we are implicitly skipping the empty space, because the data structure is not refined in the empty regions. We can skip the regions with data that is of no interest to us as well. Such data is defined by setting its opacity to zero in the transfer function. Thus, we can use the transfer function to speed up rendering by skipping the subtrees that do not contribute to the final visualization. For this purpose, we added min/max reflectivity variables to the node



(a) Post-interpolative classification.

(b) Pre-integrated classification.

Figure 5.3.: Volume visualization of ionization front data set using different transfer function techniques. The ray stepping was set to 200 steps, which are not enough for the post-classification method in this dataset and result in strong aliasing artifacts, known as the wood-grain artifacts.

structure (shown in Figure 4.1) accounting for the lowest and maximum reflectivity within the subtree. Then, during the tree traversal, we sample the transfer function using the min/max values of a node and if the integral is zero we can safely skip the area. Note that this algorithm strongly benefits from using the pre-integrated tables as we can evaluate the integral exactly with one single texture fetch.

6. RESULTS

The present chapter shows the timings of the data structure build and the visualization of reflectivity data using ray casting described in the previous chapter. The tests were performed on a laptop computer with an Intel Core i7 Q820 processor, the NVIDIA GeForce 280m GTX graphics card, 4GB of system memory, Intel X25-M G2 SSD and the Windows 7 OS.

6.1 Preprocessing performance

The data structure build was tested on two data sets. The first data set consists of reflectivity data from 12 radar sites during the Hurricane Ike event in September 9^{th} 2008, when it entered the area of Galveston, Texas. Specifically, two radar sites KHGX and KLCH in the area near Galveston and 10 other sites across the continental U.S were selected.The second data set consists of data from 116 sites across the continental U.S. scanned at 12:50pm UTC on April, 24^{th}, 2010.

The memory and performance requirements of processing the two data sets are summarized in Results:Preprocessing. The processing of 116 radar sites took almost 70 minutes, from which the 43 minutes were spent on building bricks from radar data for each octree leaf and 20 minutes were taken by creating resolution samples for each radar.

On average each brick takes around 10 ms to compute the reflectivity data for all the 32^3 cells of the brick. This includes conversion from geographic to radar's spherical coordinates and interpolation from eight reflectivity samples. For the Hurricane Ike data set, 218 octrees were built with an average of $2s$ computational time required

Task	Whole U.S.	Hurricane Ike
Number of sites	116	12
Generate resolution samples	$20m$	$3.5m$
Build Quadtree Hierarchy	$3.4s$	$0.2s$
Build Octree hierarchy	$3.3m$	$0.9m$
Build bricks for Octree leaves	$43m$	$7.4m$
Finalize Quadtree	$27.5s$	$2.3s$
Total time	$69m$	$12m$
Maximum resolution [meters]	3	2
Number of built octrees	892	218
Number of bricks in octree leaves	$264,985$	$40,089$
Number of all bricks	$341,161$	$52,102$
Node pool size	4.8 MB	0.8 MB
Compressed brick pool size	2.65 GB	0.6 GB
Uncompressed brick pool size	10.35 GB	1.62 GB

Table 6.1: The preprocessing details of constructing the data structure from 116 sites at 12:50pm (GMT), 4/24/2010 and 12 sites at 7:10am (GMT), 9/13/2008, during the Hurricane Ike event in Texas.

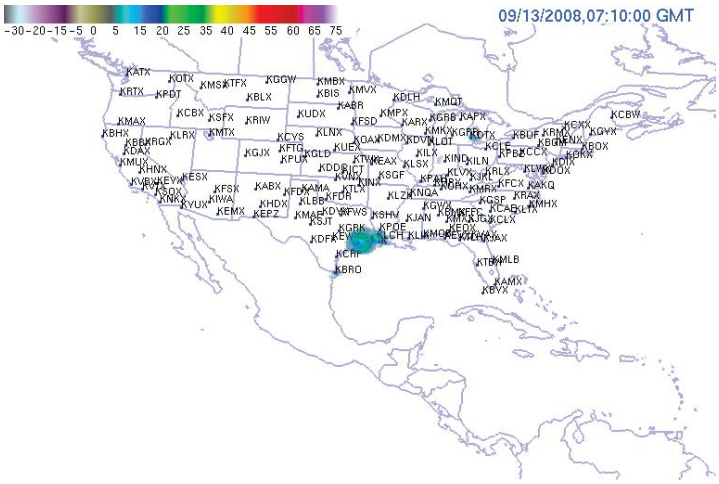
for each octree. It can be seen the most performance demanding part is the octree construction, consisting of building the octree hierarchy and its bricks, and computation of resolution markers. Both of these tasks can be easily parallelized and we believe the whole preprocessing time can be then done within few minutes and allow for periodic construction of updated radar data every ten minutes in order to provide the preprocessed data in near-real time. Because our data structure samples data adaptively at the appropriate resolution, the brick compression is only three to four which is much lower than the 100 times compression of Ru (2007).

6.2 Visualization

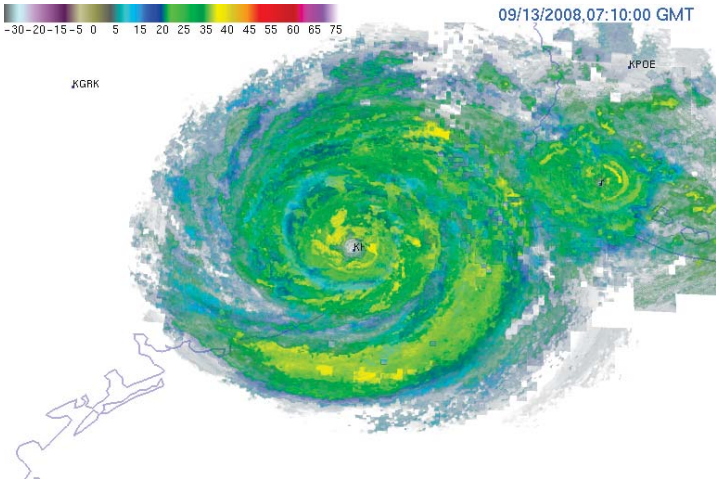
The visualization has been implemented to the extent it supports the data sets of size that fits into the system memory. Therefore, we tested the visualization performance only over the data set of Hurricane Ike described in the previous section. The visualization has been tested by zooming in from the view of whole United States to a close up view at the center of the KHGX radar site, where the severe weather event takes place, in Galveston. The generated images are shown in Figure 6.1 and the rendering times are summarized in Table 6.2. The main bottleneck of the visualization is due to loading bricks. Although the compressed bricks have been pre-cached to the system memory it still takes up to almost $2s$ depending on the amount of bricks that need to be loaded. Another important factor is also whether the bricks are loaded from the same tree file or not, because the bricks are compressed per file. To improve the interactivity of the visualization during the camera manipulation, the resolution of the output image is decreased four times in width and height.

	Average	Maximum
Ray casting	$0.06s$	$0.12s$
Brick decompression	$0.6s$	$1.7s$
Node pool copy to GPU	$0.02s$	$0.02s$
Brick pool copy to GPU	$0.12s$	$0.12s$

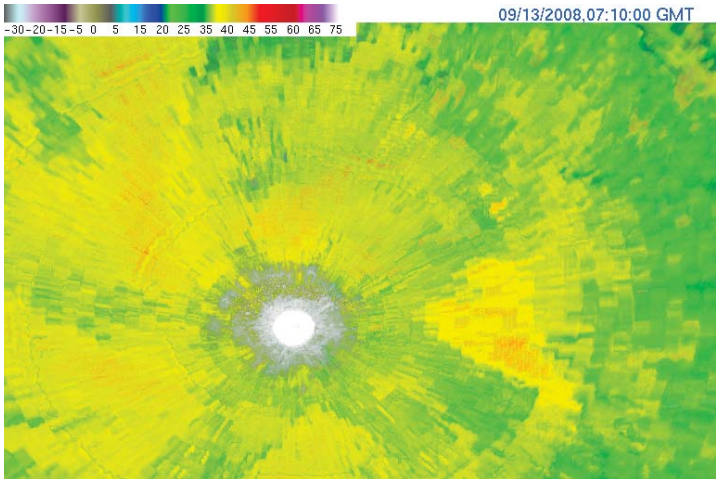
Table 6.2: Visualization times.



(a) Top view.



(b) Medium view.



(c) Zoomed in view.

Figure 6.1.: The large-scale visualization of Hurricane Ike over Galveston on 9/13/2008.

7. CONCLUSIONS

In this thesis, we have presented a new approach for large-scale volumetric visualization of reflectivity data from multiple Doppler radars. We reached our main goal that is to preprocess the data in a way that promotes effective and high quality large-scale volumetric visualization with minimal run-time data processing so as to support for multiple users. Although the implementation is not yet optimized it already provides acceptable interactivity. But there are still several ways how to improve the visualization performance and quality further. Specifically, the rendering algorithm should provide a fall back to a lower resolution brick that is available in the working set on GPU while the higher resolution brick is loaded. The contribution from bricks along the ray have to be filtered to provide smooth transitions between different resolution levels or noticeable artifacts occur when there is a change of resolution.

The transfer function user interface we developed provides a good way to define a transfer function. However, the visualization can still result in visual clutter which can be difficult to remove. This could be alleviated by allowing volume clipping feature using clipping primitives, such as plane or box, which would allow for further removal of occluding data.

Using the resolution samples and interpolation within each radar's coordinates we managed to capture even the highest frequencies in the data set. The data resolution and coverage could be further improved by including data from TDWR radars at airports near major cities in the U.S.

Last, to provide even better tool for weather analysis, the visualization should combine and display other types of data, such as wind velocity, spectrum width, temperature field and warnings.

LIST OF REFERENCES

LIST OF REFERENCES

Askelson, M. A., Aubagnac, J.-P., & Straka, J. M. (2000). An adaptation of the barnes filter applied to the objective analysis of radar data. *Monthly Weather Review*, *128*(9), 3050-3082.

Crassin, C., Neyret, F., Lefebvre, S., & Eisemann, E. (2009, feb). Gigavoxels : Ray-guided streaming for efficient and detailed voxel rendering. In *Acm siggraph symposium on interactive 3d graphics and games (i3d).* Boston, MA, Etats-Unis: ACM Press. Available from `http://artis.imag.fr/Publications/2009/CNLE09` (to appear)

Djurcilov, S., & Pang, A. (1999). Visualizing gridded datasets with large number of missing values (case study). In *Vis '99: Proceedings of the conference on visualization '99* (pp. 405–408). Los Alamitos, CA, USA: IEEE Computer Society Press.

Engel, K., Kraus, M., & Ertl, T. (2001). High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Hwws '01: Proceedings of the acm siggraph/eurographics workshop on graphics hardware* (pp. 9–16). New York, NY, USA: ACM.

Foley, T., & Sugerman, J. (2005). Kd-tree acceleration structures for a gpu raytracer. In *Hwws '05: Proceedings of the acm siggraph/eurographics conference on graphics hardware* (pp. 15–22). New York, NY, USA: ACM.

Gibson, M. S. (2010). *Gr2 analyst web pages.* Available from `http://www.grlevelx.com/`

Gobbetti, E., Marton, F., & Guitian, J. A. I. (2008). A single-pass gpu ray casting framework for interactive out-of-core rendering of massive volumetric datasets. *Vis. Comput.*, *24*(7), 797–806.

Hadwiger, M., Kniss, J. M., Rezk-salama, C., Weiskopf, D., & Engel, K. (2006). *Real-time volume graphics.* Natick, MA, USA: A. K. Peters, Ltd.

Horn, D. R., Sugerman, J., Houston, M., & Hanrahan, P. (2007). Interactive k-d tree gpu raytracing. In *I3d '07: Proceedings of the 2007 symposium on interactive 3d graphics and games.* New York, NY, USA: ACM.

Huber, M., & Trapp, J. (2005). A review of nexrad level ii: Data, distribution, and applications. *Journal of Terrestrial Observation.*

Jang, J., Ribarsky, W., Sha, C. D., & Faust, N. (2002). View-dependent multiresolution splatting of non-uniform data. In *Vissym '02: Proceedings of the symposium on data visualisation* (pp. 125–ff). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Jay Miller, L., Mohr, C. G., & Weinheimer, A. J. (1986). The simple rectification to cartesian space of folded radial velocities from doppler radar sampling. *Journal of Atmospheric and Oceanic Technology*, *3*(1), 162-174.

Jorgensen, D. P., Hildebrand, P. H., & Frush, C. L. (1983). Feasibility test of an airborne pulse-doppler meteorological radar. *Journal of Climate and Applied Meteorology*, *22*(5), 744-757.

Kaehler, R., Abel, T., & Hege, H. C. (2007, September). Simultaneous gpu-assisted ray casting of unstructured point sets and volumetric grid data. In (pp. 49–56).

Kajiya, J. T., & Von Herzen, B. P. (1984). Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, *18*(3), 165–174.

Kruger, J., & Westermann, R. (2003). Acceleration techniques for gpu-based volume rendering. In *Vis '03: Proceedings of the 14th ieee visualization 2003 (vis'03)* (p. 38). Washington, DC, USA: IEEE Computer Society.

Lacroute, P., & Levoy, M. (1994). Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Siggraph '94: Proceedings of the 21st annual conference on computer graphics and interactive techniques* (pp. 451–458). New York, NY, USA: ACM.

Lakshmanan, V., Smith, T., Hondl, K., Stumpf, G. J., & Witt, A. (2006). A real-time, three-dimensional, rapidly updating, heterogeneous radar merger technique for reflectivity, velocity, and derived products. *Weather and Forecasting*, 802–823.

Levoy, M. (1988). Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, *8*(3), 29–37.

Lux, C., & Fröhlich, B. (2009). Gpu-based ray casting of multiple multi-resolution volume datasets. In *Isvc '09: Proceedings of the 5th international symposium on advances in visual computing* (pp. 104–116). Berlin, Heidelberg: Springer-Verlag.

McNeill, A. (n.d.). *Annual disaster/death statistics for us storms.* Available from `http://www.depts.ttu.edu/weweb/Research/DebrisImpact/Reports/DDS.pdf`

Mohr, C. G., & Vaughan, R. L. (1979). An economical procedure for cartesian interpolation and display of reflectivity factor data in three-dimensional space. *Journal of Applied Meteorology*, *18*(5), 661-670.

Nocke, T., Sterzel, T., Bttinger, M., & Wrobel, M. (2008, November). Visualization of climate and climate change data: An overview. *Digital Earth Summit on Geoinformatics 2008: Tools for Global Change Research (ISDE'08)*, 226–232.

Ribarsky, W., Faust, N., Wartell, Z., Shaw, C., & Jang, J. (2002). Visual query of time-dependent 3d weather in a global geospatial environment.

Roettger, S., Guthe, S., Weiskopf, D., Ertl, T., & Strasser, W. (2003). Smart hardware-accelerated volume rendering. In *Vissym '03: Proceedings of the symposium on data visualisation 2003* (pp. 231–238). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Ru, Y. (2007). *Volumetric visualization of nexrad level ii doppler weather data from multiple sites*. Unpublished master's thesis, Purdue University, West Lafayette, Indiana, USA.

Shapiro, A., Robinson, P., Wurman, J., & Gao, J. (2003). Single-doppler velocity retrieval with rapid-scan radar data. *Journal of Atmospheric and Oceanic Technology*, *20*(12), 1758-1775.

Trapp, R. J., & Doswell, C. A. (2000). Radar data objective analysis. *Journal of Atmospheric and Oceanic Technology*, *17*(2), 105-120.

Vasiloff, S. V. (2001). Improving tornado warnings with the federal aviation administration's terminal doppler weather radar. *Bulletin of the American Meteorological Society*, *82*(5), 861-874.

Westover, L. (1990). Footprint evaluation for volume rendering. In *Siggraph '90: Proceedings of the 17th annual conference on computer graphics and interactive techniques* (pp. 367–376). New York, NY, USA: ACM.

Weygandt, S. S., Shapiro, A., & Droegemeier, K. K. (2002). Retrieval of model initial fields from single-doppler observations of a supercell thunderstorm. part i: Single-doppler velocity retrieval. *Monthly Weather Review*, *130*(3), 433-453.

Wilson, O., VanGelder, A., & Wilhelms, J. (1994). *Direct volume rendering via 3d textures* (Tech. Rep.). Santa Cruz, CA, USA.

Xiao, Y., Liu, L., & Shi, Y. (2008). Study of methods for three-dimensional multiple-radar reflectivity mosaics. *SCI Meteorological Journal*.

Zhang, J., Howard, K., & Gourley, J. J. (2005). Constructing three-dimensional multiple-radar reflectivity mosaics: Examples of convective storms and stratiform rain echoes. *Journal of Atmospheric and Oceanic Technology*, *22*.