VISUAL QUERY OF TIME-DEPENDENT 3D WEATHER IN A GLOBAL GEOSPATIAL ENVIRONMENT

William Ribarsky, Nickolas Faust, Zachary Wartell, Christopher Shaw, and Justin Jang
GVU Center and Center for GIS and Spatial Analysis Technologies,

Georgia Institute of Technology

This Technical Report is a draft for a chapter in a pending book:

Mining Spatio-Temporal Information Systems, R. Ladner, K. Shaw, and Mahdi Abdelguerfi, Editors (Kluwer, Amsterdam, 2002).

Chapter 4

VISUAL QUERY OF TIME-DEPENDENT 3D WEATHER IN A GLOBAL GEOSPATIAL ENVIRONMENT

William Ribarsky, Nickolas Faust, Zachary Wartell, Christopher Shaw, and Justin Jang
GVU Center and Center for GIS and Spatial Analysis Technologies,

Georgia Institute of Technology

Abstract: A multi-key data organization is developed for handling a continuous stream of large scale, time-dependent, 3D weather data in a global environment. The structure supports inserting the data in real-time as they arrive or retrieving weather events at desired times and locations from archived weather histories. In either case data are organized for interactive visualization and visual query.

Key words: visualization, geospatial, weather, visual query, temporal database, real-time

1. INTRODUCTION

There are a burgeoning number of sources for weather data. These different sources often provide data with different formats, resolutions, and levels of confidence. Yet once these data are prepared for visualization they can be integrated, displayed together, compared and contrasted, and analyzed together. A main reason is that visualization ultimately requires mapping the data into a common space and structure. In this paper we will exploit this property by developing a data model that supports efficient and interactive visualization of integrated global data.

Our ultimate goal is to create a world geospatial digital library containing comprehensive geospatial data for any place on earth. The main method of accessing these data is via interactive visualization; thus we advocate a Visual Earth [Rib02] superceding the usual goal of a Digital Earth [Dig01]. The Visual Earth will definitely be a dynamic place. This is especially true for weather data, but also eventually for terrain, buildings, and other types of data. Since all these data are or will be time-dependent, the data model must be dynamic. Further to support visual queries and interactive visualization, the data model must produce multiresolution graphical representations appropriate for interactive rendering.

In this work we concentrate on 3D data, because these data have not been organized for exploration, query, and investigation as much as other data (e.g., satellite imagery). Indeed, even the National Weather Service is only now about to receive tools permitting interactive visual exploration of the full 3D structure of NEXRAD Doppler radar. Up to now forecasters have only viewed these data as 2D slices. In addition the 3D weather data often exhibit significant spatial non-uniformity, such as when there are overlapping Doppler radar sites or when data from different sources are combined. We have developed a novel rendering method for 3D time-dependent data that provides newly accurate depiction of the non-uniform 3D structure of the data and scales to regional and ultimately global coverage. Thus the structure must handle multiple overlapping radars, collections of radars that cover a state (Oklahoma has 11 NEXRAD radar sites), and even national or international collections (the U.S. has about 140 NEXRAD sites). We will discuss these new rendering methods and the scalable structure, and how they fit into the Visual Earth, in this paper

2. 4D DATA MODEL FOR THE VISUAL EARTH

Since we have 3D, time-dependent data, the data model must be able to handle histories. This requires a 4D model where time is a dimension on equal footing with the spatial dimensions. However, although placed on an equal footing in terms of efficiency of access and display, time requires different handling than the spatial dimensions since histories can stretch for years. We thus introduce a multi-key data organization that can handle a continuous stream of large-scale, time-dependent, 3D weather data. The structure supports inserting the data in real-time as they arrive or retrieving weather events at desired times and locations from archived weather histories that will ultimately contain years of data. The former case is important for timely responses to weather events, like those weather

forecasters must make, w hile the latter is important for analysis or general understanding of weather phenomena. For either case the data are organized for interactive visualization. Since it is for the Visual Earth, the interactive visualization environment supports simultaneous display of the weather data with high resolution terrain elevation and imagery data, clusters of 3D buildings, features such as roads or waterways, and a GIS data retrieved from a GIS database [Lin96, Fau00, Dav98, Dav99].

2.1 Relevant Work

Developing a fully realized tool of this nature requires capability in visualization, interactive 3D graphics, temporal and spatial databases, GIS and artificial intelligence. Below we describe some of the relevant work in these areas and then explore how the needs of inter active visualization queries compare and contrast with the goals of this work. While there are commercial and research groups working towards large scale, interactive, queryable, 3D visualization of geospatial data, much of the work appears in separate are as of expertise. A comprehensive solution does not exist, especially one that includes both terrain atmospheric data.

In the commercial GIS realm, Rigaux [Rig02] points out the many current GIS systems such as ArcInfo (ESRI), MGE and TiGRis (Intergraph) separate descriptive (or alphanumeric) data and spatial data management. Typically, a relational database management system (DBMS) is used for descriptive data while custom modules are built for handling spatial and temporal data. Rigaux discusses two such systems, Oracle8i and Postgress.

In the 3D graphics and visualization literature, a number of data structures and algorithms have been developed for visualizing time -varying volumetric data. Sutton et al. [Sut99] present the temporal branch -on-need tree (T -BON) that focuses on rendering isosurfaces from time varying volumetric data. Shen et al. [She99] present the time -space partitioning (TSP) tree for visualizing time varying volumetric data. These approaches support visual query on the level of consecutive , coherent time steps, as described further in Sec. 3, rather than temporal database queries of the sort made in a DBMS.

This line of research appears to have proceeded independently of the work on spatio -temporal databases within the database community. Examples of this latter work are in the proceedings of the Internal Workshop STDBM'99 (Spatio-Temporal Database Management); Advanced Database Systems by Zaniolo et al. [Zan97]; and Temporal Databases: Research and Practice [Etz98]. In this last volume, Jen sen et al. [Jen98] provide a "Consensus Glossary" of the concepts and terminology developed in the temporal database area over the past several decades. Much of these ideas

have been integrated into TSQL2, a standard temporal extension for the ubiquitous SQL. Similar overviews are found in texts such as Part II of Advanced Database Systems [Zan97].

While much of this literature does not produce real-time 3D visualizations, there are a few exceptions. Lutterman and Grauer [Lut99] add temporal components to the VRML scene graph. They illustrate interactive 3D VRML visualizations of ground water heads (a geological formation) and the evolution of a city. The user can move through either environment using standard VRML intersections plus each application has an interface for moving the state of the virtual world forward and backward in time. The temporal node types added to VRML were considered for integration into the VRML standard. However, the complexity and scale of the data is much less than the weather plus terrain data that we deal with here.

Mennis et al. [Men00, Men02] bring in methods from artificial intelligence and human cognition to help organize spatio-temporal data. They describe the Pyramid Framework that "seeks to integrate principles of cognition into geographic database representation" [Men02]. The framework is composed of the Data Component and the Knowledge Component. The Data Component refers to traditional spatio-temporal GIS data consisting of location, time and theme (a GIS term describing the type of information being represented). The Knowledge Component contains information about higher-level semantic 'objects.' This component contains a taxonomy structure that group similar objects within a category along with a rule-base used to identify these objects from the lower-level Data Component. The Knowledge Component also has a partonomy structure that represents part-whole relationships between the high level objects. They implement this structure using a commercial OO database called Poet. They give an example where the Data Component consists of precipitation and temperature data, which is analyzed to produce a variety of weather storm objects which are the Knowledge Components. All this information is built within the Poet OODBMS and is queryable with the Object Query Language. No visualization component is directly provided for in this system. The Pyramid Framework appears to be quite general, and the Knowledge Component advances the notion of the 'weather events' described further below.

Our weather visualization tool needs to support both volumetric data representing the Doppler radar or other 3D weather information as well as discrete geometric object data that represents extracted data such as mesocyclones. The volumetric data case needs ideas similar to TSP trees [She99]. However, as pointed out by Plale [Pla01], the TSP structure is designed with the assumption that the TSP is constructed once as a

preprocessing step and then reused repeatedly for the interactive visualization without further modification. This does not suit our goals of having a database that is continually updated with new data.

The discrete geometric object data will require different techniques. The spatio-temporal database literature contains a fair amount of discussion on temporal spatial indexing of discrete objects. Nascimento et al. for instance compares a number of spatial database indices for spatio-temporal access of discretely moving points [Nas99]. They compare R-tree, the 3D R-tree, the 2+3 R-tree and the HR-tree. (These methods temporally extend the basic R-tree.) The R-tree and its pure spatial variants are a very common spatial data structure used in the spatial database community. It is designed specifically with disk storage and access time in mind and for data that is non-uniformly distributed.) Ideally, a system for weather visualization will maintain higher level constructs, such as a "tornado", that themselves include geometric extent. Being able to efficiently query and visualize these semantically higher-level geometric objects requires spatio-temporal indices suitable for interactive 3D visualization. This is the approach we take in this work.

2.2 The Dynamic Data Model

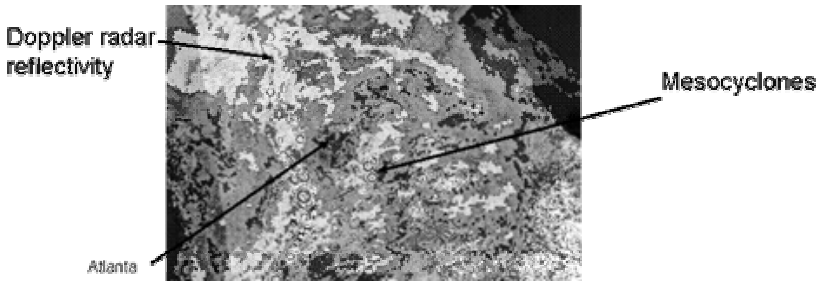

Figure 4-1. Doppler weather data and mesocyclones

To provide the appropriate 4D capabilities, the data model will use semantic features extracted from the data to organize and control access to the temporal hierarchy. This is a powerful approach based on content, and in this respect differs from the usual metadata approach that emphasizes data structure (formats, data types, data sizes, identifiers, etc.). In fact this approach will cut across the usual metadata structure since data of different types with different formats may be accessed simultaneously. (Access via traditional metadata can be provided as well.)

The temporal semantic features are in terms of events. Thus for the weather data there are weather events, which will be automatically generated

as the data are acquired. Examples of weather events are mesocyclones that track, over time, precipitation (amount and type), wind intensity, and wind shear in storm cells. (See Figure 1) Predicted trajectories over a limited period of time can be associated with each mesocyclone. Storm cells that are likely candidates for tornadoes are also identified. The mesocyclones describe the shapes, trajectories, and characteristics of extended storm fronts, as shown in Figure 1. These mesocyclone features, developed by researchers at the National Severe Storms Lab (NSSL) [Eil95], are generated on-the-fly and are part of the Level 2 Doppler radar dataset provided to weather forecasters and researchers. We are now working on features derived from 3D clustering of the Doppler radar data [Rib99] that will accompany the mesocyclones. These will give the overall shape and extent of a storm system and can be tracked in time. Average values for properties of the contained Doppler data will be attached to each cluster. We have developed a fast, multiresolution 3D clustering approach that will be used here [Rib99]. Other automated techniques will extract weather events from 2D data. Satellite imagery, for example, will be analyzed with a combination of image processing and computer vision techniques that we have applied successfully to identifying features in other types of geospatial imagery [Was01]. Features will be extracted by shape, color, and texture. The mappings (color, for example) to weather variables provided with the satellite images augmented by training using images of known weather events will permit automatic extraction of weather events (with their shapes, locations, and averaged properties) from the imagery. A goal and a challenge is to ensure that this feature extraction runs fast enough to fit into the Visual Earth library real-time insertion process and that the 2D weather events have multiresolution representations.

Whatever their source, the weather events are used to organize and annotate the temporal hierarchy. A dynamic time tree is associated with the weather (3D and 2D) spatial hierarchy. If there are no weather events for a particular time range for a region, there will be no nodes in the time tree at that level. Periodically, historical data are combined and older 3D data may be discarded. For example, data without storm events may be kept for a month and then discarded. During these periods of cleaning up, the weather event time tree is reorganized and re-balanced to maximize efficient traversal. An accumulation procedure is used to provide averaged values, ranges, deviations, etc. for weather events at higher levels of the time tree.

How should the dynamic time tree be integrated with the global geospatial structure? We take the position that the global structure is foremost. Since 3D data tend to be massive, it is most efficient to first traverse the global hierarchy to the region of interest (i.e., the region in view)

and then access the dynamic time tree for that region. The global hierarchy is a forest of quadtrees covering the earth; this structure has successfully been used to organize terrain, urban, and static 3D weather data [Fau00, Dav98, Dav99]. At a certain level in the hierarchy, there is the dynamic time tree and a quadtree-aligned volume tree. (See Figure 2.) The volume tree is described further in the next section. Traversal of the time tree finds the events of interest and brings forth the volume tree for each one. Each volume tree contains a sequence of time steps. The integration of time steps with view-dependent levels of detail (LODs) at this level ensures that one can produce interactive animations of the weather event behavior. It is possible to use spatial coherence between time steps to speed the selection of the appropriate LOD and volumetric representation [She99]. This structure shows the two levels at which time-dependence must be handled to have an efficient visual query system. At the 4D history level, there is a structure in terms of temporal events. At the level of the detailed 3D representation, there are sequences of coherent time steps.

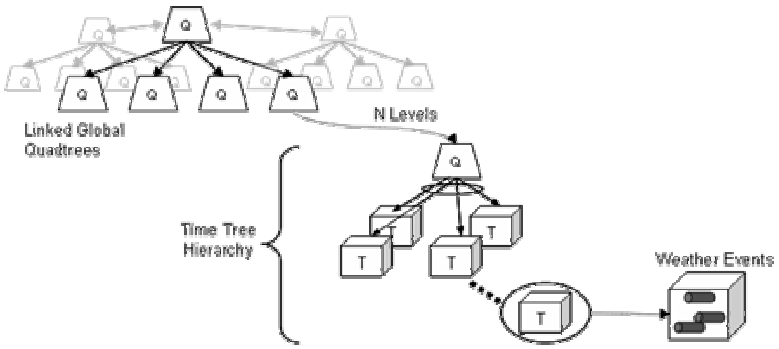

Figure 4-2. Data-adapted global quadtree for time-dependent volume data

The data model presented here enables new and enriched queries, such as:

- Show me storms containing tornadoes for this region over the summer of that year.
- Accumulate and display a time-ordered history of weather events for this time range and region.
- Show me severe storms above this level of hail and lightning for this time range and region.
- Show me storms with this range of rainfall that come from this direction for this region and time period.
- Show me storms in this region and time period that traverse terrain above this height (and similar queries using simple GIS capabilities embedded in the global geospatial structure).

These semantic queries lead to second, visual queries controlled by user navigation of and interaction with the weather (and terrain, if desired) visualization. One mode of visual query would just display the weather event features with further details, including full 3D information, generated by user interaction.

Compression and Analysis: This approach provides several levels of data compaction, which will be quite useful due to the variety of bandwidths that digital library users may employ. If just weather event features are transmitted, the amount of information transmitted may be a factor of a thousand or more less than the full 3D dataset. Since LODs will be available for these event features, this compression factor may even be greater for initial transmission and display. The 3D weather data itself is in a continuous resolution, view-dependent form [Jan02] so that only data for the part in view and at a selected resolution need be sent. Thus a feedback mechanism could be employed to trade lower resolution for a certain update rate, with higher detail being filled in when the viewpoint stops moving. Higher detail could also be provided for user-selected regions in the 3D space (details on demand). We are investigating the implementation of both these mechanism. The client interface that will be employed by users who access the digital library is set up so that the user interaction and rendering threads are separate from the scene update thread. In this way user interactivity is not impaired by slower retrieval and rendering of 3D detail, though scene information may be at low resolution or missing during quick movements of the viewpoint—but it will fill in later.

It is important to note that the data model retains the full resolution data (e.g, Level 2 Doppler radar data). Thus the full data are available for analyses, and the hierarchical geospatial structure described further below will permit quick access to these data. A very interesting question that we plan to pursue is how the LODs can be used for analysis. The LODs are set up for visualization and have error metrics appropriate to graphical detail. We will investigate how these metrics translate into errors for analyses such as rainfall density inferred by reflectivity signals, windfield structure over time, or flooding extent using terrain elevation data. Once these measures are in place our data model can support multiresolution analyses where fast overview calculations can be followed by more detailed and accurate calculations based on user selection.

2.3 System Organization

Figure 3 shows the high level system organization. Circles represent some type of general c omputation process while the cylinders represent

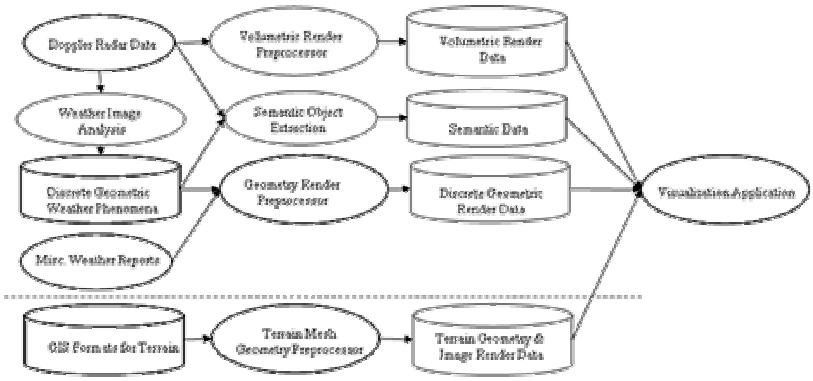

Figure 4-3. High level dataflow diagram of system.

permanent storage. This diagram emphasizes the need for preprocessing geometric data into a form that can be quickly rendered by the visualization application. This rendered geometric data is too large to store in primary memory. To maintain the most interactive frame rates possible requires spatial p artitioning and indexing and pre -computed level of detail information for the rendered geometry, as described further below. The volumetric data is analyzed by three processes. The Volumetric Render Preprocessor computes spatial and temporal decomposition and LOD information needed for interactive display. This data is stored as the Volumetric Render Data. The above -described weather event data are geometric in nature and will have some visual geometric representations. Again to allow for interactive 3D dis play some preprocessing steps may be necessary to compute spatio -temporal decompositions and LODs for this data. A third major process extracts higher level conceptual or semantic data. This data is analogous to the “Knowl edge Component” in Mennis et al.’s Pyramid Framework, which conceptually subsumes the weather events. This higher level data may group related Doppler radar data, discrete geometric weather phenomena, 3D lightning fields, satellite weather imagery, and weather simulations into a single weather event. For instance, a weather event might be ‘Atlanta Georgia Storm of July, 1998’. Finally, the visualization system of course has the underlying terrain geometry and imagery data. Terrain is visualized simultaneou sly with the above data; the

terrain process dataflow is shown at the bottom of Figure 3 and is discussed further in the next section.

The file format of the discrete geometric weather phenomena is basically predetermined by software produced by weather ex perts such as researchers at NSSL. The storage format for the volumetric weather render data, the semantic data, and the discrete geometric render data, however, is part of the visualization application, as described further below. The semantic data may be best managed by an object oriented database. This is the tactic take by Mennis et al. who use the Poet database. Several Open Source databases are available such as the object relation DB, PostgreSQL [Pos02]. The semantic data can reference the render dat a through either files names, integer id's, or perhaps the Render Data can be stored directly as BLOB (Binary Large Object) using the DBMS. This would have some benefits such as leveraging the client-server capability of the DBMS Server, but we are not aware of any true comparisons of this approach with using separate files for storing Render Data. In the present implementation we reference via file name.

3. SCALABLE, HIERARCHICAL 3D DATA STRUCTURE

3.1 The Data Structure

In this section we specify the details of a structure based on the data model described in the last section. This structure is global in scale and will accept different 3D data formats. . We choose an approach that is customized for volumetric data but is still consistent with the handling of terrain data [Fau00, Dav98] and static 3D objects such as buildings [Dav99] on a global scale. In all cases we follow a linked global quadtree structure (actually a linked "forest" of quadtrees that provides access to all parts of the Earth) to a selected level and then switch to a mode customized for the data type (e.g., volumetric, terrain, or 3D objects) for handling the highest LODs.

Our premise is that the linked global quadtrees provide an efficient and scalable structure even for volumetric data; this is supported by the performance results below. The problem for the volume structure then reduces to choosing a hierarchy that fits into the global quadtree at an appropriate level. We choose the organization shown in Figure 4. Time step sequences are stored a t the quadnode level, as discussed above, so this level is chosen to provide sufficient amounts of data for efficiency in both detail management and time sequencing for animation while not providing too

much data to impede efficient access and paging. At this point an additional time structure could also be inserted [She99] to provide further efficiency in rendering through temporal coherence. Several quadnodes might contribute to a display frame, depending on the extent of the volumetric data and the position of the viewpoint.

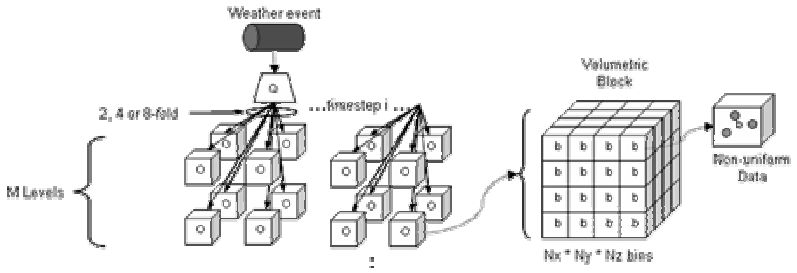

Figure 4-4. Weather Event Index

For volumetric data the quadnode is divided into Nx x Ny x Nz bins where x,y,z are the longitude, latitude, and altitude directions, respectively. The bin sort is fast (O(n) where n is the number of volumetric data elements). This is a key step because the bins provide a structure that is quickly aggregated into a hierarchy for detail management and for view frustum culling. However, the data element positions are retained in the bins for full resolution rendering (and analysis), if desired. The hierarchy provides significant savings in memory space and retrieval cost since only data element coordinates for viewable bins at the appropriate LOD are retrieved. Note that the bins are not rectilinear in Cartesian space, a factor that may affect some analysis or volume rendering algorithms. In general, the bin widths in each direction are non-uniform (e.g., each of the bins in the, say, Nz direction may have a different width). This gives useful flexibility in distributing bins, for example, when atmospheric measurements are concentrated near the ground with a fall-off in number at higher altitudes. A sub-case, of course, is to have uniform bin widths in each direction.

Each of the dimensions N in the x,y,z directions is a power of 2. This permits straightforward construction of a volume hierarchy that is binary in each direction. Our tests show that this restriction does not impose an undue limitation, at least for the types of atmospheric data we are likely to encounter. The number of children at a given node will be 2, 4, or 8. If all dimensions are equal, the hierarchy is an octree. Typically the average number of children is between 4 and 8. We restrict the hierarchy to the following construction. (Others are possible.) Suppose that Nx = 2m, Ny = 2n, Nz = 2p where m > n > p. Then there will be p 8-fold levels (i.e., each

parent at that level has 8 children), n-p 4-fold levels, and m-n 2-fold levels. If two out of three exponents are equal, there will be only 8-fold and 4-fold levels. The placement of 2, 4, and 8-fold levels within the hierarchy will depend on the distribution for the specific type of volumetric data.

Properties at parent nodes are derived from weighted averages of child properties. The parent also carries the following weighting factors: (1) the total raw volumetric data elements contained in the children; (2) the total filled bins contained in the children; (3) the total bins contained in the children. The quadnode level is chosen such that there are between 1-10 K bins (i.e., leaf nodes in the volume hierarchy). This gives reasonable balance between the costs of traversing global quadtree and volume hierarchies while enabling effective handling of volumetric data in the lon/lat/altitude dimensions. Note that the bin structure and volume hierarchy are static in space. We can efficiently apply this structure even to distributions of volumetric elements that move in space as long as the range of local spatial densities (and the volume of the data) does not change much over time.

The bin sizes are chosen such that there are at most a few volumetric elements in each bin. The reason for this choice is that we want a smooth transition between rendering of bin-based levels of detail and rendering of the raw data. The final step in the LOD process is the transition from the bins to the underlying raw data. Because the volume hierarchy permits fast traversal, this choice is efficient even for sparse data with holes and high density clumps, as shown in the application below.

Data Retrieval and Use: One must have a mechanism for retrieving data for use in the time frame appropriate for the selected application. The application presented here involves interactive visualization (more specifically exploratory visualization). The time frames for interactive, on-the-fly visualization, as the volumetric time steps are acquired, and for playback of histories, after the data are archived, may be different. In particular the time frame for playback should be at least 10 frames per second to insure continuous animation. In either case the time between user interaction and system response should be 0.1 second or less to insure good user performance. We shall discuss below the acquisition and display time frames for weather data.

To support these interactive visualization requirements, we further organize the volumetric structure as indicated at the bottom of Figure 4. The volume tree structure goes down to a certain level after which the bins are arranged in volumetric blocks. The block can be either a 3D array of bins or a list of filled bins, depending on whether the data distribution is dense or sparse. We have found in our tests so far that a block containing one bin gives good results. (In other words, the volume tree goes all the way to

single bin leaf nodes.) However, the multi-bin block structure is available if it should prove efficient for future data distributions. Ultimately, we expect that the data distribution will inform the visualization technique used. It may be sufficient to use traditional (continuous field) volume visualization techniques for dense data, but different techniques may be better for sparse data. We address this issue further below. This structure is set up to handle simultaneous, overlapping datasets. These might include, for example, different types of acquired weather data along with data from weather simulations, all of which might have different spatial distributions. The application below demonstrates the ability to handle overlapping datasets.

Scalability: To insure scalability, insertion in or access to the dynamic data structure cannot depend on the overall size of the structure. Further there must be a mechanism to extract data in constant size chunks and in constant time no matter how large the data structure becomes. The volumetric blocks are the key to meeting this requirement. Finally, an efficient out-of-core mechanism is required. Here the volumetric blocks allow a paging and caching procedure similar to that used for terrain [Fau00, Rib02, Dav98]. For the terrain case, pages in the size range of thousand elements were shown to be efficiently paged and set in a priority queue for rendering, with old pages being discarded. We use similar size pages for the volumetric data. The results section shows how the scalability requirement is met for a specific application.

For interactive visualization and exploration, data must be provided in an appropriate range of LODs so that the amount of detail displayed does not exceed an upper limit, no matter how much data are in view. The structures described above have this capability. Lower resolution LODs can be provided by intermediate nodes in the geospatial quadtree or the volume hierarchy, using the appropriate average values stored at the nodes. At the highest resolution the bins disappear to reveal a representation of individual data elements. (One can imagine, for example, a smooth transparency transition where a bin disappears and the data elements inside appear as a user flies closer.) Extensions of view-dependent techniques for visual detail management [Lin96, Hop98] can be used here. We will discuss the effect of applying LODs in the results section below.

Out-of-Core Paging: We cannot usably apply scalability unless we have an out-of-core strategy. Ours is related to that of Cox and Ellsworth [Cox97] and has been shown to be effective for large scale terrain paging [Dav98]. For volumetric data there is a volume paging thread with a server and a manager. The server loads the volume tree skeleton (without data) for quadnodes in view and the manager decides which LOD should be loaded. Traversal of the skeleton takes into account user viewpoint, direction, and speed to minimize bad pages and to permit predictive paging. If, for

example, the user is moving fast, pages may be skipped because they will be out of view by the time they are loaded and rendered. Only at this point are pages retrieved. The page contains a volume block whose size has been determined, through testing, to provide a good balance between latency in page I/O and number of pages needed for a view. This procedure has proved efficient and flexible. Also, since the p aging thread is separate from the interaction and rendering threads, page latency does not affect interactivity.

Flexibility: The framework we have presented is quite flexible. It depends only on general properties of the data such as their spatial range, the total number of elements, and the average density range. It does not depend on the details of the volume data distribution or its geometry or topology. However, the data organization within the framework can change (e.g., locations where the volume tree sprouts with branches containing data), even from one time step to the next, permitting significant customization for efficient use. Because of complete coverage by the global geospatial quadtree, spread in lat/lon extent or movement of a 3D field acros s the earth's surface is not a problem. The bin sizes, volumetric tree structure, and block sizes can be tuned for a collection of datasets. One can then acquire and insert several datasets for simultaneous display or analysis, even if they are significant ly different in their detailed data element positions or connectivity. The only thing that will change is which bins and thus which nodes and branches in the volumetric structure are filled and linked. Our results so far indicate that we will be able to au tomate the tuning of the data organization and bin structure so that the system investigates a collection of acquired data and automatically sets up the volumetric structure.

3.2 Results for Acquiring and Visualizing Time-Dependent Data

We concentrate on the i nsertion of 3D Doppler radar data and weather simulations in the same geospatial framework. These data have significantly different organizations. Some work has been done on the visualization of 3D Doppler data [Dju99, Jia01], but this work does not consid er LODs or their management. In the present application we visualize weather, terrain, and groups of buildings from the same general geospatial structure, which has not been done before. The overall size of the data in the structure can be quite large. The NEXRAD tri -agency radar provides 3 pieces of information—reflectivity, velocity and spectrum width—at 9 to 14 elevation scans every five minutes. This amounts to a volume of up to 50 MB of raw data every 5 minutes. Typically there are multiple radars with overlapping coverage. The central Oklahoma site, for example, has 11 radars and the

Peachtree City site near Atlanta (used in this paper) has 3 radars. Thus there can be up to 55 MB of data streaming in every 5 minutes and nearly 7 GB of weather data archived for a site over a 10 hour period when storm development might be tracked. In addition there is a large amount of geospatial data. For Georgia there is 30 M resolution terrain elevation for the state and phototextured imagery at varying resolution (high resolution areas up to 1 meter resolution in downtown Atlanta; 1 foot resolution at Georgia Tech). In addition the database contains hundreds of 3D buildings for downtown Atlanta and Georgia Tech. There is also 30 M resolution terrain elevation and photo texture data for Oklahoma and environs. With 1 M resolution terrain data now available for several of these locales, the terrain database size will swell to several hundred gigabytes. We will explore the scalability and efficiency of the volumetric data structure next.

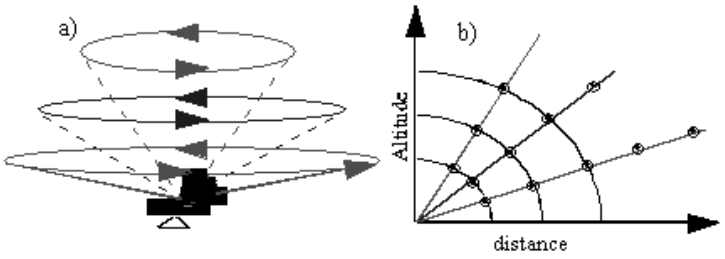

Figure 4-5. Radar scan pattern

Doppler Data: The 3D volume is built up from a series of cone-shaped radial sweeps as depicted in Figure 5. The datasets we consider here are composed of 9 sweeps per time step, separated by angles from 1o at the first sweep to 5o at the ninth sweep. The number of sweeps and the incremental angle between sweeps can be adjusted, though this is not usually done in the middle of data collection over the time span of a storm. In addition each sweep has a time stamp that can be used to animate the volume collection. In a sweep readings are collected along radial lines spaced about 1o apart in the azimuthal direction. Data are then collected at evenly-spaced gates along the radial line (Figure 5). However, different sweeps have different numbers of gates; in general lower sweeps have more gates extending farther from the radar. The total number of gates in 9 sweeps is approximately 1 M. But there are usually only about 100 K valid readings for a given variable over a set of sweeps. The positions of valid readings change from time step to time step.

It may be supposed that it would be more efficient to formulate a data structure and rendering algorithm that specifically takes account of the non-uniform, curvilinear nature of the sweeps. This might be true if we were

handling just one radar. But we are handling several radars with overlapping sweeps; the spatial lay out of these radars vary from site to site. In addition we are adding weather simulations, which tend to have more uniform data patterns, and will add other types of data, such as 3D lightning fields. The present volumetric data structure fulfills this need for flexibility.

Upon analysis, a good bin size that on average contains only a few data elements and also is commensurate with the quadnode areas turns out to be 350 m x 350 m x 170 m. Bins of this size that contain filled gates have from 1 to 5 of such gates with an average of around 2. This holds over the complete time sequence and is typical for a large severe storm. The above bin proportions are attractive for rendering in that all dimensions are of the same order. Having several million bins in the volumetric tree provides a good balance between the quadtree and volumetric tree portions of the data organization. This can be achieved for the present bin sizes by choosing quadnodes down 10 levels of the geospatial quadtree. The footprint for each of these nodes is 50 KM x 50 KM, and the overall volume is 2500 KM^2 x 14 Km. About 40 nodes are needed to cover the areal extent of 1 Doppler radar.

At each of the quadnodes there are 28 x 28 x 27 (256 x 256 x 128) bins. This translates to a tree with four 8 -fold levels and one 4 -fold level. We choose to place the 4 -fold level at the top of the tree, meaning that the first subdivision is in the lon/lat directions only. After that, the tree behaves like an octree.. All these results give specific details to the structure in Figure 4.

Weather Simulation: A typical mesoscale weather simulation, such as MM5, has grid points at 4 Km intervals. High resolution simulations for local areas, which use MM5 as a starting point, have grid spacings of 1 KM or greater. We as sume here a high resolution simulation on a regularly spaced grid of 1 KM x 1 KM x 0.5 KM over the volume of the Doppler radars. A good bin size for this structure is a 4 x 4 x 4 block of the Doppler radar bins (1400 M x 1400 M x 680 M). These bins will contain 1 simulation element in most cases but 2 -4 elements in a few cases. The bins will fit into the same volume tree structure as above, and the volumetric tree for this case is 4 levels deep. A significant advantage of this approach is that now both radar and weather simulation data are available in the same framework and can be retrieved simultaneously for detailed rendering or analysis. In fact the nodes can be traversed efficiently (as shown below) accessing either or both radar and simulation data.

Performance Results:. We are now ready to do some performance tests. We loaded 100 time steps of 3D Doppler radar data for 1, 2, 3, or 4 radars. The 2 radars were separated by 80.5 Km and the 3 and 4 radar cases were on the vertices of an equilateral triang le and square, respectively, each with sides of 80.5 Km. For these data a single bin structure and volume tree

template can be used, but the actual tree will change from time step to time step. At acquisition the bins are filled and nodes and links are cre ated for filled blocks in the tree structure. A Doppler radar volume is collected at the NEXRAD site every 5 -7 minutes. A time budget of a minute or less for sorting, inserting, and displaying the radar data would be sufficient for real - time performance, given that there are other analyses that may be performed. Among these is an analysis finding the locations, sizes, and intensities of mesocyclones, which under certain conditions can indicate tornadoes.

Table 4-1. Times for filling the bins and linking the volumetric tree.

No. of Radars	Vol. Trees	Blocks	Bins	Data Pts	Creation Time (s)
1	67	85k	.736M	.761M	1.1-7.0
2	70	89k	.755M	1.52M	1.1-8.0
3	98	136k	1.15M	2.28M	2.1-17.2
4	109	173k	1.48M	3.04M	3.2-23.5

The timing tests were run on an SGI Origin 200 server with 4 R12000 processors. The server has RAID disks and a Gigabit network connection to a local network containing several PC -based and UNIX graphics workstations. As s hown in Table 1, sorting and insertion (for 1 radar) takes only between 1.1 and 7 sec per time step for the valid data elements. The range of creation times is due to the varying number of valid data elements over the time sequence. These data were for a s evere storm with heavy precipitation, so the upper limit of valid data points is near the maximum expected. At 80.5 Km there is significant overlap between the radars so that the timing for 2 radars is nearly the same as that for one. For a case where the radars are far enough apart so that their extents just touch, the times increase nearly linearly between 1 —4 radars. Thus timings for multiple radars will be between these two extremes. The results show that insertion times are well within our time budget even for several radars. Even when a high resolution simulation is inserted over the range of several radars, the total time will be within budget. For example, the high resolution weather simulation described above will have number of data points and bins a factor of 50 or more less than those in Table 1 for the areal extents used and will not affect the total timings significantly. Note that much of this radar and simulation insertion process can be done in parallel, if desired, greatly reducing total times.

Even for the case of extensive storm activity, only a portion of the radar data gates contain valid data, which are all that need be retrieved. Sometimes analysts want to distinguish positions of all gates including those giving null readings. This is easily done, however, by referring to a mask block structure

where all elements are null and then comparing with the filled tree for a given time step.

Table 4-2. Times for accessing data at leaf nodes within a selected volume.

Longitude Size	Latitude Size	Altitude	Ground Area	Bins Accessed	Worst Case Access Time (s)
444km	444km	14000m	197136 km2	741k	0.183
222km	222km	7000m	49284 km2	109k	0.028
106km	106km	3500m	11236 km2	49k	0.021
53km	53km	3500m	2809 km2	37k	0.019
26.5km	26.5km	3500m	702 km2	15k	0.0048
13.3km	13.3km	3500m	177 km2	6k	0.0022
6.65km	6.65km	3500m	44.2 km2	2k	0.00091
3.33km	3.33km	3500m	11.1 km2	626	0.00040
1.67km	1.67km	3500m	2.78 km2	225	0.00033
0.835km	0.835km	3500m	0.697 km2	75	0.00021

Retrieval Times. To test retrieval time performance, Table 2 shows the times to access all of the data at all leaf nodes that are within a particular volume of the atmosphere. Ground area for this accessed volume ran ges from almost 800,000 square KM down to less than one square KM. To determine the worst case access time, we located the volume to be accessed at numerous places within the radar volume.

As discussed above, an appropriate retrieval time for continuous animation of dynamic data is 0.1 sec or less; in fact, if we allot equal amounts of time to retrieval and rendering of the data, the retrieval time should be no more than 0.05 sec. We see that this criterion is not met for the largest volume in Table 2 (next page) . However, the rendering algorithm uses LODs and a view -dependent procedure, which will reduce the amount of detail retrieved based on the screen space error metric used. The worst case view will be where the volume data fills the screen and all data elements are within the view frustum. Using a conservative LOD calculation for this case, we find that a screen space error of 2 -3 pixels should suffice. With LODs, the retrieval times for larger volumes than those shown in Table 2 level off. Also, a typi cal view will not be worst case since one may have only part of the data in the view frustum (as when moving in for a close -up) or the data may be farther away from the viewpoint.

4. INTERACTIVE, ACCURATE VISUALIZATION OF NON-UNIFORM DATA

The above hierarchic al structure provides efficiencies in rendering volumetric data. Adjusting an error metric derived from the hierarchy (e.g., in terms of the RMS deviation of scalar values contained in a node from their average value) permits one to select from the multipl e resolutions contained in the hierarchy. One could use a larger error for initial interactive visualization of the volume and then, when pausing for more detailed inspection, smaller errors for progressive refinement of the volumetric rendering. Such a pr ocedure could also be employed for compression and then progressive transmission of the volumetric data. Similar approaches have been developed by others [Lau91]. The hierarchy also permits quick culling, as the viewpoint changes, of volume elements that a re not in the view frustum.

Interactive navigation of large scale data requires additional efficiencies. Here some data elements may be close to the viewpoint, some at midview, and others far away from the viewpoint. The rendering system should take into account the projection of these elements onto the screen and thus their perceivability to the user. Several neighboring elements that are small or far away may fall within a single pixel. Such elements should be rendered at lower resolution, resulting in an image that displays several LODs at once. To be effective such a set of LODs should be updated every time the viewpoint changes, which can be every frame for continuous navigation. To achieve these goals, we have extended hierarchical splatting [Jan02] by using a variation of the view-dependent approach that has been applied by us [Lin96] and others [Hop98] to surface polygonal data.

Splat Structure: Several approaches have been applied to splat footprint construction and compositing. Polygon and texture -based methods [Mer01, Lau91, Rot00] using 2D or 3D textures can take advantage of graphics hardware to composite and render overlapping polygons. These approaches work well for regular data. However, for non-uniformly distributed data, one must either constrain to uniform bounding boxes for the splats (Meredith et. al. use cubic boxes [Mer01]) or face significant complications in correctly projecting, sampling, and rendering non -uniform splats. Using regular bounding boxes amounts to an approximation that s mooths out non - uniformities. We would like an approach that permits retention of these details, although uniform splats may also be used.

We have chosen to construct splats that are not necessarily uniform in their orientation or aspect ratios. However the y all fit into rectilinear bounding boxes. The approach of Zwicker et. al. [Zwi01] permitting efficient perspective projection and correct anti -aliasing for arbitrary

elliptical kernels can thus be applied. Other approaches, including texture -based methods , can be applied as well. Using splats of non -uniform orientation and aspect ratio requires more computation than uniform splats. However, the view -dependent hierarchical approach helps keep the splat construction and rendering efficient. Interactive techn iques, such as magic lenses, can also be applied.

The issue for a non -uniform distribution is illustrated with the following example. A single elliptical splat, even if optimally oriented, cannot provide uniform coverage for all the neighbors of the central data point in the wedge-shaped cell of the Doppler radar. For example, if it provides correct coverage for the top neighbors, it overlaps the bottom neighbors too much. On the other hand, multiple elliptical splats can be oriented and sized to provide better coverage over the cell. We have developed and are implementing a procedure, based on the Voronoi cell describing the region around a point, to place and orient splats to give more uniform coverage in non -uniform cells. Closest splats are combined to g ive LODs. At the moment we use elliptical, textured splats centered at the sample points with appropriate scale and orientation. The splat construction and compositing could be replaced with that of Zwicker et. al. [Zwi01], if desired.

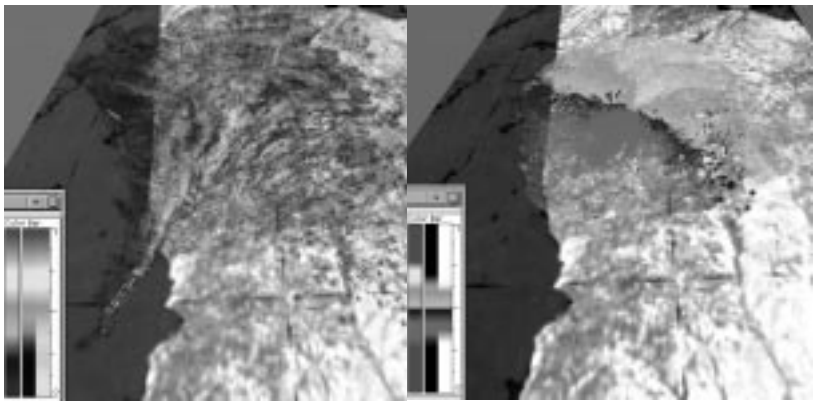

Figure 4-6. (left) Image of Doppler radar reflectivities over North Georgia. Heavy rainfall is evident along a front from the southwest towards the center (yellow and red splats). Doppler velocities are shown in the right image for the same time step. Because the winds are heading northeast, the northeast half show positive velocities (away from the radar) and the southwest half show negative velocities (toward the radar).

Results: Figure 6 (left) shows a volume rendering of the reflectivity readings from a NEXRAD Doppler Radar covering North Georgia for a severe storm that passed over the Atlanta area in late March 1996. The red

and yellow band running North-South through the central region is an area of very heavy rainfall. The color bars to the bottom left show the transfer function from reflectivity to RGBA values. The left bar is RGB and the right two bars show RGBA blended with black and white. Figure 6 (right) shows the velocity readings of the same storm at the same time. The major feature running from Northwest to Southeast is the line perpendicular to the wind direction for the entire storm. Because Doppler radar gives velocity only along each radial line, there will be a sign change in velocity values where the radar is perpendicular to the wind direction. In Figure 6 right, the storm is heading Northeast. We have set the color ramp to change abruptly from cyan (positive) to blue (negative). Further result are in Jang et. al. [Jan02].

ACKNOWLEDGMENTS

This work was performed in part under a grant from the NSF Large Scientific and Software Data Visualization program. In addition this work was supported by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research through the Army Research Office under Grant DAAD19-00-1-0352.

REFERENCES

Cox97 M. Cox and D. Ellsworth. Application-Controlled Demand Paging for Out-of-Core Visualization. Proceedings, IEEE Visualization '97, pp. 235-244 (1997).

Dav98 D. Davis, T.Y Jiang, W. Ribarsky, and N. Faust. Intent, Perception, and Out-of-Core Visualization Applied to Terrain. pp. 455-458, IEEE Visualization '98.

Dav99 D. Davis, W. Ribarsky, T Jiang, N. Faust.. Real-Time Visualization of Scalably Large Collections of Heterogeneous Objects. IEEE Visualization '99, pp. 437-440.

Dig01 See www.digitalearth.gov

Dju99 S. Djurcilov and A. Pang. Visualizing gridded datasets with large number of missing values. Proceedings IEEE Visualization '99, pp. 405-408.

Eil95 M.D. Eilts, J.T. Johnson, E. DeWayne Mitchell, Sarah Sanger, Greg Stumpf, Arthur Witt, Kurt Hondl, and Kevin Thomas. Warning Decision Support System. 11th Inter. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, AMS, pp. 62-67 (1995).

Etz98 Opher Etzion, Sushil Jajodia, Suryananrayana Spripada (Eds.). Tempoal Databases: Research and Practice. Springer-Verlag, Berlin, Heidelberg, New York, 1998.

Fau00 N. Faust, W. Ribarsky, T.Y. Jiang, and T. Wasilewski. Real-Time Global Data Model for the Digital Earth. Rep. GIT-GVU-01-10, Proceedings of the INTERNATIONAL CONFERENCE ON DISCRETE GLOBAL GRIDS (2000).

Hop98 Hoppe, H. Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering. Proc. IEEE Visualization '98, pp. 35-42 (1998).

Jan02 Justin Jang, William Ribarsky, Chris Shaw, and Nickolas Faust. View -Dependent Multiresolution Splatting of Non -Uniform Data,. To be published, Eurographics -IEEE Visualization Symposium 2002.

Jen98 C. S. Jensen, C. E. Dyreson, M. B őhlen, J. Clifford, R. Elmasri, S. K. Gadia, F. Grandi, P. Hayes, S. Jajodia, W. Käfer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peressi, B. Pernici, J. F. Roddick, N. L. Sarda, M. R. Scalas, A. Segev, R. T. Snodgrass, M. D. Soo, A. Tansel, P. Tiberio, and G. Wiederhold, The Consensus Glossary of Temporal Database Concepts – February 1998 Version. In Temporal Databases – Research and Practice, Editors O. Etzion, S. Jajodia, and S. Sripada, Springer-Verlag Berlin Heidelberg, 1998, pp. 367-405.

Jia01 Tian -yue Jiang, William Ribarsky, Tony Wasilewski, Nickolas Faust, Brendan Hannigan, and Mitchell Parry. Acquisition and Display of Real -Time Atmospheric Data on Terrain. Rep. GIT-GVU-01-12, pp. 15-24, EG-IEEE VisSym 01.

Lau91 D. Laur and P. Hanrahan. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. Proc. SIGGRAPH '91, pp. 285-288 (1991).

Lin96 Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N., and Turne r, G.A. Real-Time, Continuous Level of Detail Rendering of Height Fields. Proc. SIGGRAPH '96, Computer Graphics, pp. 109-118 (1996).

Lut99 Hartmut Luttermann and Manfred Grauer. Using Interactive, Temporal Visualizations for WWW -based Presentation and Exp loration of Spatio -Temporal Data. In Spatio -temporal Database Management : International Workshop STDBM'99, September 10 -11, 1999, Editors Michael H Bohlen,, Christian S. Jensen, Michel O Scholl, pp. 100- 118.

Men00 Jeremy L. Mennis, Donna J. Peuquet, Dia nsheng Guo. A Semantic GIS Database Model for the Exploration of Spatio -Temporal Environmental Data. 4th International Conference on Integrating GIS and Environmental Modeling (GIS/EM4): Problems, Prospects and Research Needs, September 2 - 8, 2000.

Men02 Jeremy L. Mennis, Donna J. Peuquet, Luijian Qian. A Conceptual Framework for Incorporating Cognitive Principles into Geographic Database Representation. Forthcoming in International Journal of Geographical Information Science.

Mer01 J. Meredith and K.L. Ma. Multiresolution View -Dependent Splat -based Volume Rendering of Large Irregular Data. Proc. IEEE 2001 Symp. On Parallel and Large -Data Visualization and Graphics, pp. 93-155 (2001).

Nas99 Mario A. Nascimento, Jefferson R. O. Silva, Yannis Theodoridle. Evaluation of Access Structures for Discretely Moving Points. In Proceedings of the International Workshop STDBM'99, September 1999, editors Michael H. Bohlen, Christian S. Jensen, Michel O. Scholl, pp. 171-188.

Pla01 Beth Plale. Performance Impact of Str eaming Doppler Radar Data on a Geospatial Visualization System, College of Computing, Georgia Institute of Technology, Technical Report GIT-CC-01-07.

Pos02 http://www.postgresql.org

Rib99 William Ribarsky, Jochen Katz, T.Y. Jiang, and Aubrey Holland. Disco very Visualization Using Fast Clustering. Report GIT-GVU-99-14, IEEE Computer Graphics & Applications, 19(5), pp. 32-39 (1999).

Rib02 William Ribarsky, Christopher Shaw, Zachary Wartell, and Nickolas Faust. Building the Visual Earth. To be published, SPIE 16th International Conference on Aerospace/Defense Sensing, Simulation, and Controls (2002).

Rig02 Philippe Rigaux, Michel Scholl, Agnes Voisard. Spatial Databases with Applications to GIS. Morgan Kaufmann Publishers, Inc., San Francisco, California, 2002.

Rot00 S. Rottger, M. Kraus, and T. Ertl. Hardware -Accelerated Volume and Isosurface Rendering Based on Cell Projection. Proc. IEEE Visualization 2000, pp. 109-116 (2000).

She99 Han -Wei Shen and Ling -Jan Chiang and Kwan -Liu Ma. A Fast Volume Rendering Algorithm for Time -Varying Fields Using a Time -Space Partitioning (TSP) Tree, IEEE Visualization '99, pp. 371-378 (1999).

Sut99 P. Sutton and C.D. Hansen. Isosurface Extraction in Time -varying Fields Using a Temporal Branch-on-Need Tree (T-BON). IEEE Visualization 1999, pp. 147-153 (1999).

Was01 Tony Wasilewski, Matthew Grimes, Nickolas Faust, and William Ribarsky. Semiautomatic Landscape Feature Extraction and Modeling. Vol. 4368A, SPIE 15th Annual Conference on Aerosense (2001).

Zan97 Carlo Zahniolo, Stefa nd Ceri, Christos Faloutsos, Richard T. Snodgrasss, V.S. Subbrahmanian, Roberto Zicart. Part II Temporal Databases in Advanced Database Systems. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997.

Zwi01 M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA Volume Splatting. Proc. -IEEE Visualization ’01, pp. 29-36 (2001).