Hierarchical Storage and Visualization of Real-Time 3D Data

M. Robert Parry, Brendan Hannigan, William Ribarsky,
Christopher D. Shaw, and Nickolas Faust
GVU Center, Georgia Institute of Technology

ABSTRACT

In this paper "real-time 3D data" refers to volumetric data that are acquired and used as they are produced. Large scale, real-time data are difficult to store and analyze, either visually or by some other means, within the time frames required. Yet this is often quite important to do when decision-makers must receive and quickly act on new information. An example is weather forecasting, where forecasters must act on information received on severe storm development and movement. To meet the real-time requirements crude heuristics are often used to gather information from the original data. This is in spite of the fact that better and better real-time data are becoming available, the full use of which could significantly improve decisions. The work reported here addresses these issues by providing comprehensive data acquisition, analysis, and storage components with time budgets for the data management of each component. These components are put into a global geospatial hierarchical structure. The volumetric data are placed into this global structure, and it is shown how levels of detail can be derived and used within this structure. A volumetric visualization procedure is developed that conforms to the hierarchical structure and uses the levels of detail. These general methods are focused on the specific case of the VGIS global hierarchical structure and rendering system. The real-time data considered are from collections of time-dependent 3D Doppler radars although the methods described here apply more generally to time-dependent volumetric data. This paper reports on the design and construction of the above hierarchical structures and volumetric visualizations. It also reports results for the specific application of 3D Doppler radar displayed over phototextured terrain height fields. Results are presented for display of time-dependent fields as the user visually navigates and explores the geospatial database.
Keywords: hierarchical, geospatial, volumetric rendering, time-dependent data, large-scale, weather, decision support

1. Introduction

In the current information age everything is being digitized. This includes data acquired from sensors and measurement devices, which now are typically computer controlled and produce a digital stream of results. More and more information from these sources must be blended with information from other sources, such as simulations, or with archived data. Activities such as research and analysis of physical phenomena, decision support, or situation awareness can benefit significantly from this integrated information. Especially for decision support or situation awareness, it can be vital for the acquired data to be presented as soon as it is received. Weather forecasters, for example, may have only a few minutes to analyze the position, movement, and severity of a storm and issue a warning. In this case there is now a significant amount and variety of data that are available but not often used in a complete or integrated fashion because it has not been possible to acquire, organize, and display the data in the required amount of time. Thus, even though they have 3D weather data available, weather forecasters typically look at only 2D scans of weather patterns over fixed maps. In this paper we present a set of approaches that go well beyond that, integrating acquired 4D (3D + time) data with accurate terrain elevations and imagery, 3D urban areas, and other geospatial information in a uniform hierarchical structure. This hierarchical structure is useful for a variety of other decision support and situation awareness activities, besides weather forecasting.

In this paper the term "real-time 3D data" refers to volumetric data that are acquired and used as they are produced. These may include, for example, results from severe storm radar scans or simulations, pollution pattern measurements, rainfall patterns and resulting flooding, or other 3D data. In each case there is a time budget associated with the rate of acquisition and the activity involved. In general it is necessary to provide the data in usable form within the time frame that they are produced. For the purposes of this research, the usable form is in terms of an interactive, navigable visualization with appropriate analyses of the data features and behavior. However, since we are dealing with time series data, there will be both data that is viewed as it is acquired and data histories that are archived and viewed afterwards. The time budgets for these two activities may be significantly different. In particular the time frame for playback of already archived data is particular stringent, of the order of a fraction of a second.

The structure of this paper is as follows. We discuss first the mechanism for hierarchical storage of acquired, time-stamped 3D data. This must be done in a fashion that permits application to a global setting and integration with other geospatial

Visualization of Temporal and Spatial Data for Civilian and Defense Applications, Glenn O. Allgood,
Nickolas L. Faust, Editors, Proceedings of SPIE Vol. 4368 (2001) © 2001 SPIE · 0277-786X/01/$15.00

products such as terrain. As part of this discussion, we show how our fast clustering approach can be fitted into the framework and made hierarchical. The clustering provides a valuable automated overview of the data where patterns in space, time, and the dependent variable space can be extracted and tracked. We than discuss our approach for volumetric rendering of the time-dependent data. This approach uses the global hierarchical structure and provides a mechanism for determining rendering levels of detail. We end with a few results from these methods for the case of severe storm visualization.

2. Hierarchical Storage with Fast Retrieval of Real-Time 3D Data

In previous work we have built a global hierarchical structure [Fau00] that effectively handles terrain [Dav98] and buildings [Dav99], providing a paging and caching structure that permits handling of scalably large data. This structure also offers view-dependent detail management so that objects in view are rendered with controls on the amount of individual and overall detail. In this paper we will deal with the extension of this structure to handle time-dependent 3D data. As we shall show in this and the following sections, the extended structure not only effectively deals with the on-the-fly insertion and later retrieval of volumetric data, it also provides a framework for efficient distribution analysis (through fast clustering) and volume rendering.

We have built a structure for the dynamic acquisition, insertion, and use of global geospatial data. Fig. 1 shows the flowchart for this process. At the left the data are acquired in digital form and transmitted (usually via either the Internet or some special network connection) to the geospatial environment. The acquired data could be in a variety of forms. For example, it could be aerial images to be used as terrain phototextures, new terrain elevation information, ground-level imagery to be put in a 3D context, positions and movement information for groups of people or vehicles, or time-dependent volumetric data. Our basic premise is that all these data will be fitted into a global geospatial structure based on a forest of quadtrees [Dav98, Dav99, Fau00]. Even the volumetric data will fit efficiently into this structure because they describe processes in the atmosphere, which is a thin layer with respect to the earth's surface extent.

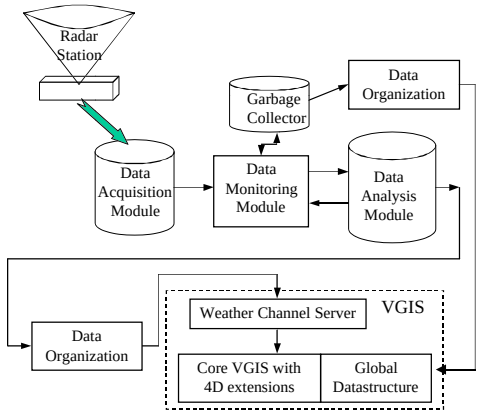


Fig. 1 Flow chart for acquisition, insertion, and display of real-time geospatial data on terrain.

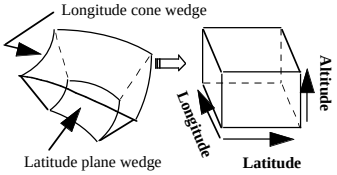


Fig. 3 Relation between bin in Cartesian space and scaled bin in lat/lon/altitude space.

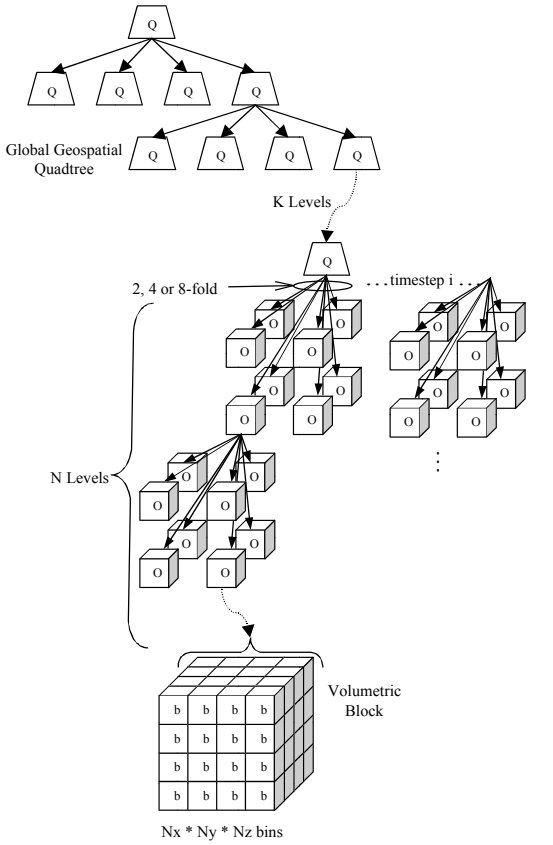


Fig. 2 Geospatial hierarchy for time-dependent volumetric data.

For each data type there is a data acquisition module (Fig. 1) that organizes the data for insertion into the global hierarchical structure. The acquisition process includes some pre-analysis steps that can function both for subsequent efficient data access and detail management and for improved understanding of the data. For example, we can insert fast clustering algorithms [Rib99] at this stage to identify overall patterns in the acquired data and to track features at different levels of detail. The acquisition processes are monitored by a data monitoring module, whose job is to watch for overflows where the incoming data flow is greater than the rate at which the system can process and insert the data. For this case we have developed a method where the data are partially organized with respect to the global hierarchy but not inserted into the global data structure. This process makes the acquired data available in a more efficient form; if there is an overflow, a separate thread completes organization of the data and inserts them into the global data structure during moments when the system is not acquiring or visualizing data. Other methods could be included here to insure that the acquisition and insertion process is completely scalable. The purpose of the process in Fig. 1 is to produce pre-analysis results, raw acquired data, and multiple LODs based on the raw data as a time-stamped stream of objects for real-time display in the VGIS global geospatial system [Lin96, Dav98, Dav99, Fau00]. At the point of display (or other analysis) these results are integrated with other data in the global structure such as terrain, buildings, maps, and so on.

2.1 Real-time 3D Data Organization

We now focus on the organization and insertion of dynamic volumetric data in the geospatial structure. We choose an approach that is consistent with the handling of terrain data [Fau00, Dav98] and static 3D objects such as buildings [Dav99]. In all cases we follow the global quadtree to a selected level and then switch to a mode dependent on the type of data (e.g., volumetric, terrain, or 3D objects) for handling the highest levels of detail. For volumetric data, we choose the organization shown in Fig. 2.

The quadnode is divided into N_x x N_y x N_z 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) and provides an initial organization for detail management and for view frustum culling. Note that the bins are not rectilinear in Cartesian space (see Fig. 3), a factor that may affect the volume rendering algorithm, for example. In general, the bin widths in each direction are non-uniform (e.g., each of the bins in the, say, N_z direction may have a different width). This gives useful flexibility in distributing bins such as, for example, when atmospheric measurements are concentrated near the ground with a fall-off in number at higher altitudes. A sub-case of this distribution, of course, is to have uniform bin widths in each direction. The bin sort is also the first step in our fast clustering algorithm [Rib99], which tracks space and time patterns in data distributions. In the next section we discuss how to insert the fast clustering results into the data structure, thus producing a hierarchical clustering approach where view-dependence can be used to choose what clusters to display.

For time-dependent data, the sequence of time steps are stored at the quadnode, as indicated in Fig. 2. This procedure provides quick access to data in sufficiently large chunks so that animations of time sequences can be efficiently displayed. Several quadnodes might contribute to a display frame, depending on the volumetric data extent and the viewpoint position.

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. 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 $N_x = 2^m$, $N_y = 2^n$, $N_z = 2^p$ 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 data distribution.

Properties at parent nodes are derived from weighted averages of child properties. The parent also carries the following attributes: (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 M 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 reasonably effective handling of volumetric data in the lon/lat/altitude dimensions. Note that the bin structure and volume hierarchy are static in 3D space. We expect to efficiently apply this structure even to distributions of volumetric elements that move in 3D space as long as the range of local spatial densities (and the overall volume of the data) does not change too much over time. Being able to construct the volume hierarchy just once for a given quadnode and then use it for all time steps provides significant cost savings.

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 level of detail process is the transition from the bins to the underlying raw data. Because the volume hierarchy permits fast traversal, we expect this choice to be efficient even for sparse data with holes and high density clumps. Application to 3D Doppler data, which is quite non-uniform, supports this expectation [Par01].

2.2 Real-time Retrieval and Use

It is not enough to have a sufficiently fast mechanism for organizing and storing acquired data. One must also have a mechanism for retrieving data for use in the time frame appropriate for the selected application. The application considered here involves interactive visualization (more specifically exploratory visualization). There may be differnt time frames for interactive, on-the-fly visualization, as the volumetric time steps are acquired, versus for playback of histories, after the data are archived. 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.

To support these interactive visualization requirements, we further organize the volumetric structure as indicated at the bottom of Fig. 2. (Note that this structure is likely to be useful for a variety of applications, not just interactive visualization.) The tree structure goes down to a certain level, after which the bins are arranged in volumetric blocks. This is a more efficient mechanism than the tree for providing data for detailed rendering and display. The block will be either a 3D array of bins or a list of filled bins, depending on whether the data distribution is dense or sparse. 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 will address this issue more fully in the next section.

2.3 Scalability

To insure scalability, insertion into or access from 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. The volumetric blocks allow a paging and caching procedure similar to that used for terrain [Fau00, Dav98]. For the terrain case, blocks in the range of thousand elements could be efficiently paged and set in a priority queue for rendering, with old pages being discarded. We use similar size blocks for the volumetric data. As with terrain [Dav98], we expect to develop a node skipping procedure, based on the predicted trajectory of the viewpoint, for efficient retrieval of volume blocks during fast navigation.

If the application is 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 shown in Fig. 2 and 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. The rather regular layout of bins at the block level offers further opportunities for detail management. 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.

2.4 Flexibility

The structure 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. The same structure can be used throughout for dynamic data as long as the data does not change dramatically over time in density or in altitude range. Because of complete coverage by the global geospatial quadtree, spread in lon/lat extent or movement across 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, regardless of their detailed data element positions or connectivity. The only thing that will change is which bins and thus which nodes in the volumetric structure are filled. In the future we will automate this tuning so that the system investigates a collection of acquired data and sets up the volumetric structure, relieving the user of this chore.

3. Rendering of Time-Dependent 3D Data

For concreteness, we will concentrate in what follows on displaying results for time-critical decision-making. However, this work applies equally well to any situation where an analysis is required within a prescribed time budget. Also, although we discuss the hierarchical volumetric structure above in the context of the global geospatial hierarchy, it can equally be used for volumetric data in other contexts.

To provide real-time 3D data in the right form for analysis and time-critical decision-making, a system will need a variety of interactive visualization tools. Among these will be tools for automatically finding and displaying the spatial distribution of the data as it develops in time, tools for finding and tracking features due to important physical processes in the data, and tools for accurate rendering of the detailed 3D structure of the data. The spatial distribution and feature tracking tools give an automatic overview of the data with important aspects displayed quickly and clearly. The results from these tools can also direct the rendering of the 3D data so that visual details can be provided where they are most telling, reducing both clutter and rendering times. Both of these capabilities are important in situations where fast but accurate decisions must be made. Further it is important that all three tools operate within a time budget (determined by the rate and frequency of data acquisition and the requirements for making decisions) and that all their results be capable of simultaneous display since this will provide valuable correlative information not available in separate displays.

All tools operate within the hierarchical 4D structure described above. It is imperative that they do so both for scalability, in terms of being able to handle even large scale data, and for efficiency. The hierarchy provides a means for quickly deciding which data are needed and for retrieving them. It also provides a spatial relational structure even for completely unstructured data, thus permitting quick determination of what is in the neighborhood of any data element (e.g., clusters, features, other data elements). In an environment with heterogeneous data, one is often faced with datasets that have completely different structures and overlapping regions that must be viewed simultaneously. The above hierarchy permits arrangement of these data into a common spatial structure.

In the following we will discuss our extended fast clustering approach and new time-dependent volume rendering scheme for heterogeneous data. In the results section below we will discuss physical feature tracking with examples from a specific application.

3.1 Hierarchical Fast Clustering

Since we are dealing with real-time streams of data whose detailed structure is unknown, we need an automated method without much preprocessing to determine overall data distribution and feature structure. This method can then be used for high-level analysis of the time-dependent data and for rendering. To automatically find the time-dependent spatial distribution we have extended our fast clustering approach.

Previously we have presented a fast clustering technique [Rib99] that works on general 4D data and permits description of the dataset in terms of clusters that can then be tracked in time as features. The clustering is both over the spatial distribution of the data and any variables that it may depend on. Total cluster error and individual cluster errors, calculated for each set of clusters, can be used to determine the relative quality of clusterings and to choose levels of detail for view-dependent display. In the present work we will extend this work by offering a hierarchical clustering in terms of the structure presented in the last section.

We give a brief description of our clustering method. A fuller exposition is in [Rib99]. To identify 3D cluster centers, we define a spatial vector and average:

$$\mathbf{r}_i = [x_i, y_i, z_i], \quad \bar{\mathbf{r}}_c = \frac{1}{k_c} \sum_{i \in c} \mathbf{r}_i \ ,$$

where c denotes the cluster and k_c is the number of data points in the cluster. We then define a distance metric D for each data point in terms of the spatial vector, and a total error R, which is the sum over all clusters of the (unnormalized) root mean square deviations:

$$D_{i,c} = [(\mathbf{r}_i - \bar{\mathbf{r}}_c) \cdot (\mathbf{r}_i - \bar{\mathbf{r}}_c)]^{1/2}$$

$$R = \sum_c \left(\sum_{i \in c} D_{i,c}^2 \right)^{1/2} = \sum_c \left(\sum_{i \in c} \mathbf{r}_{i,c}^2 - \bar{\mathbf{r}}_c^2 \right)^{1/2} \quad . \quad (1)$$

These definitions are enough to optimize a set of clusters based on either minimizing R for a given N_C (number of clusters) or choosing N_C such that $R / N \leq \tau$ where τ is some threshold value and N is the total number of points in the dataset. For the former, one would specify N_C, and R would decrease with increasing N_C. For the latter, N_C would not be fixed but would depend on the value of τ. In either case, the data points closest to a given cluster center (i.e., for which D_C is smallest) would belong to that cluster.

The clustering of Eq. 1 only describes spatial distributions. How do we handle dependence on one or more variables? After considering several alternatives [Rib99], we have concluded that the best choice is to extend to a higher dimension vector

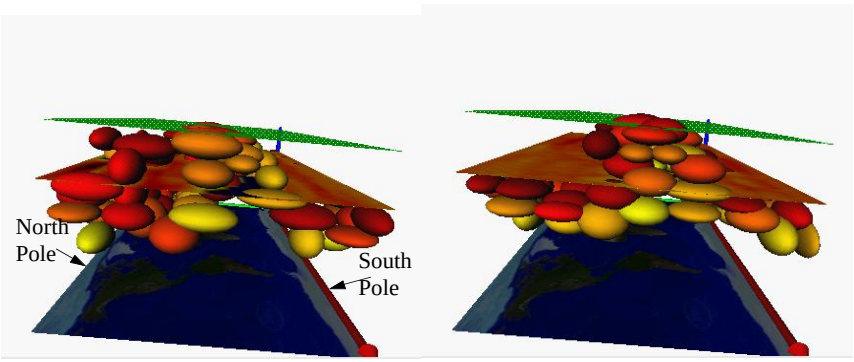
$$\mathbf{p} = [\alpha(x,y,z), vt, \beta V, \gamma \mathbf{F}, \ldots] \quad (2)$$

where *t* is time, V is a scalar attribute, **F** is a vector attribute, and so on. In a given dataset these attributes might be temperature, pressure, flow velocity, stress tensor, and so on. Note that time is now included in a fundamental way rather than being considered as an attribute to which time coherence is applied from time step to time step (as is usually done). One can now cluster, if desired, over a range of time. Since these scalars, vectors, etc., in general have different units that must be converted to some common basis for clustering and display, there are conversion factors, etc. The generalization of Eq. 1 is then

$$R_{\mathbf{p}} = \sum_c \left(\sum_{i \in c} \mathbf{p}_{i,c}^2 - \bar{\mathbf{p}}_c^2 \right)^{1/2} \quad (3)$$

One problem with this approach is that it is not clear, in general, how to assign the different conversion factors (i.e., combine the different units) in Eq. 2 (although one could certainly do this for specific variables and applications). This is related to a problem from data mining where one tries to find a mapping into a coordinate space for a set of disparate variables [Fal95]. The problem is that this general mapping often does not produce clusters of good quality [Gan98]. We are working on developing a general and automatic way of assigning the conversion factors. In the meantime we have found, if only one or two attributes are included in the clustering, that it is often straightforward to assign conversion factors. We have also found that a powerful extension is to permit the user to set these factors interactively, thus emphasizing or reducing the importance of each.

We can now develop an algorithm starting with one big cluster and subdividing and rearranging until a requested total error or number of clusters is reached. (See [Rib99] for details.) A key step is the first one. A bin sort is applied to the data, which requires two steps, both O(N). For spatial clustering, the first step determines the extent of the dataset in x,y,z. The second step uses these extents to form the bins and sort the data. Each bin contains a weighted centroid, weighted averages for selected variables, number of points, ranges for each variable, and values for the error metric. The bin sort is simple and highly parallelizable and, quite importantly, enables the uniform handling of data in any structure. Most significantly, the bin sort immediately reduces the order of the data for all subsequent steps from N to M (the number of bins) where M<<N if N is very large. For extremely large data one could even randomize the sort so that not all N elements need be touched.



Figs. 4 Clusters for global atmospheric model at time step 1 and time step 170.

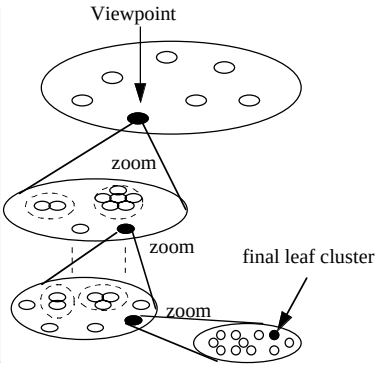


Fig. 5 View-dependent tracking of a cluster down to a leaf node.

The clustering has been used on a variety of test and real data with good results. In Figs. 4, for example, are data from a global atmospheric simulation of N_2O. The automatic clustering correctly follows the shape and movement of the peak at the top of the atmosphere.

Building Hierarchies. If we have a very large dataset, the original bin sort may well not resolve the details in some region of interest discovered after interactively exploring the original clustering. In that case, we must subdivide the bins in the region of interest. Alternatively, we may want to impose a spatial detail management structure on the clusters themselves, which may be too numerous to handle, especially in a display with lots of other detail (e.g., terrain features). To make these and other operations efficient, we need to build a hierarchy. To do this we note that the bin sort step here is more or less the same as the bin sort for the dynamic volumetric data in the last section. We can build on this similarity and use the hierarchical structure assigned previously. The cluster bins now conform to the volume tree structure (binary along each of the principal axes) and are leaf nodes. It is straightforward to subdivide the bins using the same structure. One simply generates new leaf nodes in the tree.
The initial, automatic clustering builds a tree to a certain depth. After that deepening of the tree is based on either direct or "focus-based" user manipulation. (Focused-based user manipulation enacts deepening of an in-view portion of the clustering—and tree—depending on where the user is looking closely.) This whole process is shown schematically in Fig. 5. The hierarchy will permit navigable visualizations where the user can zoom in, see detail in context, or back up to gain an overview. How do we rationalize the cluster hierarchy with the volumetric structure of the last section? We can do this by assigning a branching factor of N_c clusters to each node of the volumetric tree. A cluster is in a given node if its center is in the volume extent of that node. If N_c clusters are in a node and a subdivision would add one or more, a set of child nodes are generated and the clusters distributed among the children. The total number of clusters and certain averages are updated for the parent node. So far we have chosen $N_c = 1$, which makes it especially straightforward to follow the clustering in the volumetric hierarchy. Now the clusters within the view frustum can be quickly found by determining the nodes within the view frustum. (Note that this includes neighboring nodes that may be just outside the view frustum because clusters could extend outside their resident nodes.) The appropriate level of detail in terms of the number of clusters displayed for a given viewpoint can be found by traversing down the tree for parent nodes. A screen-space error for the spatial extent of the cluster is used, along with the cluster error itself (via Eq. 2), to approximately determine the "best" display of clusters that meets the screen space error criterion while minimizing the total cluster error. The screen-space error criterion is applied in a manner similar to that for terrain [Lin96] and buildings [Dav99]. We are testing other values of the branching factor and also investigating the balance of the selected volumetric tree structure. For non-uniform spatial distribution of data, the tree can become unbalanced.

3.2 Hierarchical Volume Rendering

Although there has been separate work on rendering unstructured 3D data [Wil98] and on rendering time-dependent volumes [Ma98], there has been little work that considers these aspects together. Furthermore, we wish to consider data that may be sparse or clumpy with holes or regions of null readings. For these types of data, one must rethink the usual premise for volume rendering. The usual premise is that the volume represents a continuous, semi-transparent medium, but that is not true for sparse or clumpy data. For these data some regions may be reasonably continuous and others may not. We must have a rationale for handling both types of regions. One conclusion we have reached is that intrinsic properties of the data production come into play, such as what the values at a sample data element tell us about its surrounding region. These issues of confidence levels and uncertainty are hidden in the usual volumetric data, where there is typically a uniform distribution of samples that one assumes fairly represents the underlying continuous field, but they must be considered explicitly here.

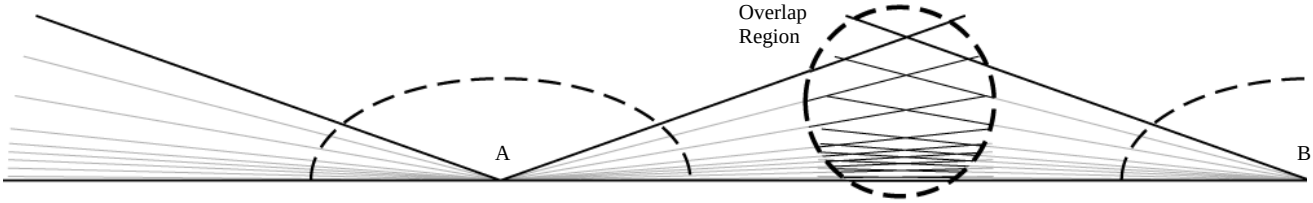


Fig 6. Cut-away side view of 2 NEXRAD Doppler weather stations. Each radar station collects data at 9 sweeps (represented by grey lines) at increasing angles to the horizontal. The entire extent of radar sweeps from station "A" is shown from the left to the center right. The dashed half-ellipse indicates the subset of station A's readings that sample sufficiently to be considered continuous. The overlap region highlighted by the dashed circle to the center right may also have sufficient sampling density to be considered continuous.

An example will provide clarity. Consider the highly non-uniform collection pattern for 3D Doppler radars shown in Fig. 6. Data are collected in consecutive cone-shaped sweeps at increasing angles to the horizontal. In each sweep readings are collected along radial lines spaced about 1^o apart in the azimuthal direction. The collection gates are evenly spaced along each radial. Although the radar is sampling continuous fields such as wind velocity and reflectivity (correlated with rainfall amount), a continuous representation of these 3D fields cannot be made everywhere. For example, the regions inside the dotted ellipses/circles, close to the radar source or where two radars overlap significantly, could be represented as continuous 3D volumes. But other regions, such as between the sweeps to the far left of Fig. 6, cannot. We need a process for both characterizing and visualizing all these regions. To do this we must consider what the measured samples in a neighborhood tell us about the field value at a nearby point. If we use an interpolation method to calculate this value, it will come with a confidence level affected by the interpolation method, the distance from sampled points, the nature of the collection process, and other factors. If the confidence level is too low, it may be inappropriate to assign a value to this point or, indeed, to other points in its region.

Footprint Calculation and Accumulation. We have chosen to represent these confidence regions as 3D footprints surrounding each sample. In general these footprints may change depending on field value, time-varying external conditions (such as different atmospheric conditions affecting the meaning of the Doppler radar readings), or other factors. A cutoff below a certain confidence level could produce, for example, layered volumes along the sweep arms that only overlap in certain areas (Fig. 7). In other non-uniform data distributions there might be holes or concavities. It will be important to distinguish these shapes as due to limits in what the data tell us about the underlying field rather than as intrinsic structures in the field itself.

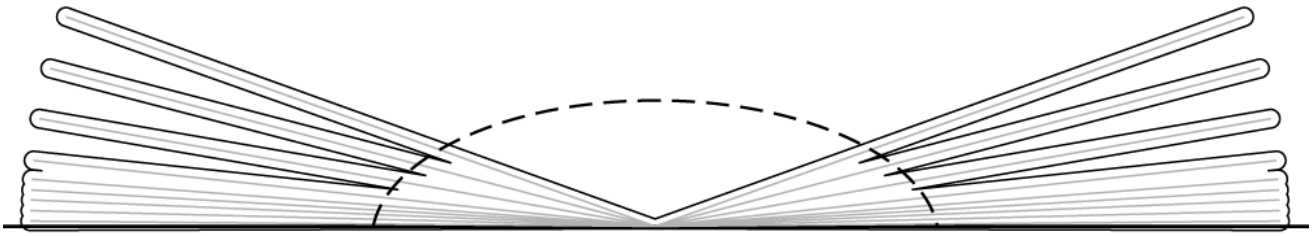


Fig. 7. A cut-away side view of a single NEXRAD Radar station, showing the 9 cone-shaped Radar sweeps (grey lines) with surrounding footprint extents (black lines). The dashed half-ellipse shows the approximate zone of completely overlapping footprints.

These footprints are similar to the splats used to resample unstructured data and derive topological structure [Sch97] or to produce volume rendering representations [Wes90, Swa97]. We shall build on these similarities. Note that the footprint defines a spatial region but does not, in general, provide an interpolated value. An additional method may be needed for that. In what follows, however, we will assume that the footprint is also used (as is often the case) to provide interpolated values. The hierarchical representation described in the previous section provides its own topology, which we will use. As discussed in the previous section, each leaf node contains a small number of (unstructured) sample points. The field values of the contained samples are averaged and a footprint that inscribes the footprints of the contained samples is established. This is done recursively up the hierarchy. For simplicity we assume that the footprints at all levels can be represented by ellipsoids.

The procedure for assigning child footprints and accumulating them into parent nodes is as follows. The leaf nodes in the hierarchy contain, on average, 2-3 unstructured volume elements. We can do a principal component analysis to find the optimal footprint axes for this collection of elements [Gro94]. To do this we calculate

$$\mathbf{r}_i = [x_i, y_i, z_i], \quad m = \frac{1}{M}\sum_{i=1}^{M} \mathbf{r}_i$$

where $\boldsymbol{m}$ is the mean vector of the collection of M elements. For the difference vector $\mathbf{d}_j = \mathbf{r}_j - \boldsymbol{m}$, we can calculate the covariance matrix C in 3D space

$$C_{u,v} = \frac{1}{M}\sum_{i=1}^{M} d_{i,u} d_{i,v}^{T}$$

where *u*,*v* refer to components along the x,y,z axes. We can diagonalize this real, symmetric matrix, and the rotated axes will be the optimal axes for the ellipsoid. For simplicity we assume that all data element footprints are the same. Different footprints can be accommodated by a straightforward generalization of the procedure described below. The specific procedure is

- For 1 element, just use that element's footprint.
- For 2 elements, use the axis joining the centers as one principal axis. After transformation to align along this axis, the resultant C will have a 2x2 non-diagonal sub-matrix that can be solved by an additional rotation as described below.
- For 3 elements, choose one principal axis as perpendicular to the plane defined by the elements. Again, after transformation the resultant C will have a 2x2 non-diagonal sub-matrix that can be solved by an additional rotation.
- For 4 or more elements, find the matrix C and apply the Jacobi transformation [Numerical]. The resultant eigenvectors are the principal axes.

Since there are usually 3 or fewer elements, this procedure is reasonably fast.

The additional rotation of the 2x2 sub-matrix can be found by examining the transformed matrix

$$R^{T} \bullet \begin{bmatrix} a & b \\ b & d \end{bmatrix} \bullet R, \quad R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Setting the off-diagonal element of the transformed matrix to zero and solving, the result is

$$\cot(2\theta) = \frac{a-d}{2b}.$$

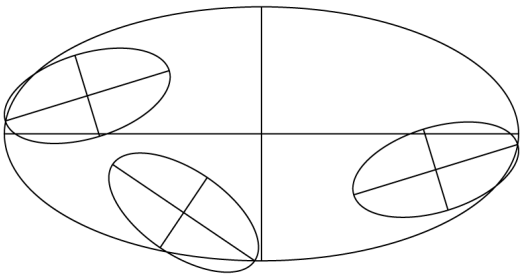


Fig. 8 Hierarchical ellipsoidal footprints. The large ellipsoidal footprint contains 3 sub-footprints whose major-axis endpoints intersect with the enclosing footprint's boundary.

The orientation of the ellipsoid can be further improved by projecting the axes of the individual element ellipsoids onto the plane defined in the 2 or 3 element cases. An additional point can be placed along the largest of these projections for each element, *m* and the resulting 2x2 sub-matrix recalculated, and new rotated axes found. A similar procedure can be used to set up the C matrix for 4 or more elements and, indeed, can be extended to collections of elements with non-equal footprints. The final step is to calculate the point along each new principal axis where the confidence level falls below the prescribed cutoff. This is done approximately by projecting the cutoff extents along each of the individual element axes (see Fig. 8) onto the new ellipsoid principal axes and then using the maximum values as the new cutoffs.

A similar process is followed for accumulating child footprints as one goes up the hierarchy. Here there can be up to 8 child footprints. We are investigating simplified procedures for collecting 6-8 footprints since these will tend to be spread uniformly over the parent volume.

Volume Rendering. Now that we have a hierarchical structure for the footprints, we must set up the volume rendering procedure. We could just use the footprints as a sampling strategy and employ any volume rendering procedure, including ray-casting. However, there is an apparent advantage to using splatting [West, Swan], where the accumulated footprints are projected on the image plane. A main advantage over ray-casting, is that splatting is of order k^2n^3, where k is the number of samplings per splat and n^3 is the number of samples in the volume, whereas ray-casting is at least of order k^2n^3. The difference is due to the projection of the splat before sampling and compositing. We thus use splatting in our volume rendering procedure.

There are disadvantages to splatting, namely, in the appearance of the volumetric image under interaction (e.g., rotation) or animation. To minimize these problems, the voxel-by-voxel composition has to be done in the right order and care has to be taken in applying transfer functions and shading to the splats. However, methods to correct some of these disadvantages have been developed [Swa97]. In addition efficiency of the splatting procedure can diminish if perspective projection is used since

one can no longer sample a generic splat for all pixels in the image space, as can be done for orthographic projections. Our hierarchical structure provides a significant advantage here. In addition the shading and coloring of the splats are important for amorphous and time-varying volumes without definite boundaries. The Doppler radar data and other weather or atmospheric data will be like this. The shading brings out shape information and coloring/opacity helps identify and classify time-varying regions. A carefully constructed transfer function can significantly improve coloring/opacity and classification. The construction of the transfer function is significantly aided by the cluster analysis described above.

4. Results

We present here a few results for acquisition and display of 3D Doppler radar weather data. The results presented here are for a series of severe storms that hit North Georgia and passed over metro Atlanta on March 19, 1996. Volume scans for a couple of hundred time steps were collected over a period of several hours. At certain times, regions of the storm had tornado signatures. The National Weather Service facility in Georgia has 3 Doppler radars covering contiguous areas in North Georgia, Northeastern Alabama, and Northwestern South Carolina. Thus a storm's path and development can be followed over several hundred miles.

The data presented here are for only one of the 3 Doppler radars. There are 9 sweeps making up a volume in each time step. Acquired along with the volumetric data are on-the-fly analyses called mesocyclone cells. These are areas of intense wind and rain activity, and are defined by position, extent, and height. Several radar scans can contribute to a mesocyclone. Certain mesocyclones with high wind shear between neighboring cells indicate tornadic activity. Figs 9 a,b,c show frames from an animation of mesocyclones passing over high resolution terrain imagery in North Georgia. The radial pattern in Fig. 9b shows the extent of the 3D Doppler radar scans for the radar at Peachtree City, just south of Atlanta. One can fly in for a close-up view of the weather and see its detailed relation to the terrain, as depicted in Fig. 10.

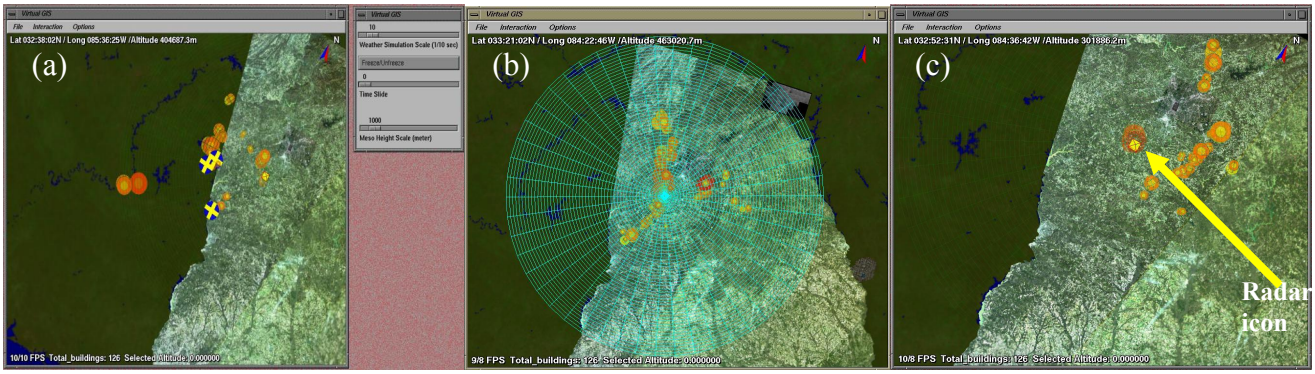


Fig. 9 Overview in orbital mode of storm progression. Animation control is at the top right of Fig. 2a.

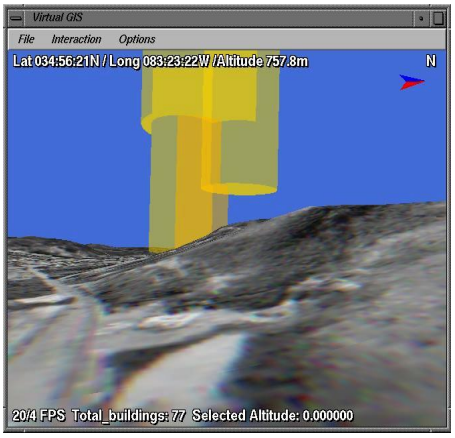


Fig. 10 Close-up view of mesocyclones near North Georgia mountains.

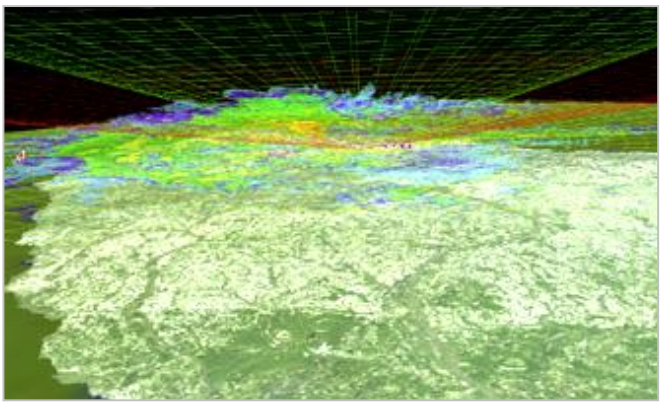


Fig. 11 View of raw 3D volume data over terrain in fly mode.

Fig. 11 shows a view of the raw 3D volume data from which the mesocyclones are derived. In this case the data are retrieved from the hierarchical data structure and displayed as a set of images texture mapped onto cones representing the individual sweeps of the Doppler radar. It is of interest to display these data simultaneously with the mesocyclone features because the relation between these structures has not been looked at in detail before. One can fly in for close views of these relations. Finally Fig. 12 shows a volume rendering of the 3D Doppler data using the method described in Sec. 3 above. One can now see 3D details that were not apparent in other methods of rendering. The results in Figs 9-11 depict Doppler radar over 3D terrain, which has not been done before. Previously no terrain elevation information has been available along with Doppler radar data. These visualizations can be important in determining the effect of terrain features on weather patterns, for quickly identifying locales that might be affected by severe storms (maps can also be overlaid), and for determining flooding or other results of heavy rainfall or severe winds.

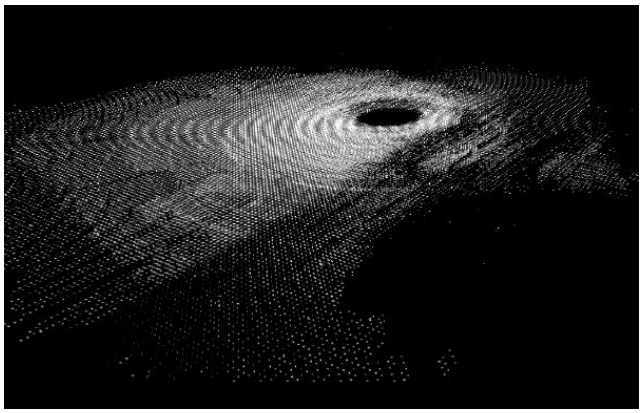


Fig. 12 Volume rendering of 3D Doppler radar data.

Acknowledgments

This work was performed in part under a grant from the NSF Large Scientific and Software Data Visualization program and under a MURI grant through ARO.

References

Dav98 D. Davis, T.Y Jiang, W. Ribarsky, and N. Faust. Intent, Perception, and Out-of-Core Visualization Applied to Terrain. Rep. GIT-GVU-98-12, pp. 455-458, *IEEE Vis. '98*.

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

Fal95 C. Faloutsos and K.I. Lin. Fastmap: A Fast Algorithm for Indexing, Datamining, and Visualization of Traditional and Multimedia Databases. *Proceedings of SIGmod 95*, pp. 163-174.

Fau00 N. Faust, W. Ribarsky, T.Y. Jiang, and T. Wasilewski. Real-Time Global Data Model for the Digital Earth. *Proceedings of the INTERNATIONAL CONFERENCE ON DISCRETE GLOBAL GRIDS (2000).* An earlier version is in Report GIT-GVU-97-07.

Gan98 V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French. Clustering Large Datasets in Arbitrary Metric Spaces. Technical Report , U. of Wisconsin-Madison, 1998 (www.cs.wisc.edu/~vganti/birchfm.ps).

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

Gro94 M. Gross. Subspace Methods for the Visualization of Multidimensional Data Sets. Scientific Visualization, Advances and Challenges, pp. 173-186 (Academic Press, London, 1994).

Lin96 Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Real-Time, Continuous Level of Detail Rendering of Height Fields. Report GIT-GVU-96-02, *Computer Graphics (SIGGRAPH 96)*, pp. 109-118.

Ma98 Ma, Kwan-Liu, Diann Smith, Ming-Yun Shih, and Han-Wei Shen. Efficient Encoding and Rendering of Time-Varying Volume Data. ICASE Technical Report #98-22, NASA/CR-1998-208424, Hampton Va, 1998.

Par01 Robert M. Parry, William Ribarsky, Chris Shaw, Tony Wasilewski, Nickolas Faust, T.Y. Jiang, and Beth Plale. Acquisition, Management, and Use of Large-Scale, Dynamic Geospatial Data. Submitted to *IEEE Visualization '01*.

Rib99 William Ribarsky, Jochen Katz, T.Y. Jiang, and Aubrey Holland. Discovery Visualization Using Fast Clustering. IEEE Computer Graphics & Applications 19(5), pp. 32-39 (1999).

Sch97 Will Schroeder, Ken Martin, and Bill Lorenson. The Visualization Toolkit, 2nd Edition. Pp. 394-397 (Prentice Hall, Upper Saddle River NJ, 1997).

Swa97 J.E. Swan, K. Mueller, T. Moller, N. Shareef, R. Crawfis, and R. Yagel. An Anti-Aliasing Technique for Splatting. . *Proc. IEEE Visualization '97*, pp. 197-204 (1997

Wes90 L.A. Westover. Footprint Evaluation for Volume Rendering. *Computer Graphics* (proceedings of SIGGRAPH) 24(4), pp 367-376 (1990).

Wil98 P.L. Williams, N.L. Max, and C.M Stein. A High Accuracy Volume Renderer for Unstructured Data. *IEEE Transactions on Visualization and Graphics*, Vol. 4 (1), pp. 37-54 (1998).