Time-Critical Visual Exploration of Scalably Large Data

William Ribarsky, Davis King, Ada Gavrilovska, and Rogier van de Pol
Graphics, Visualization, and Usability Center
Georgia Institute of Technology
{ribarsky, kingd, ada, rogier}@cc.gatech.edu

Abstract

This paper discusses visualization and analysis issues as datasets grow towards very large sizes, and it develops an approach to attack these issues. Datasets of this size become exploration-dominant since the scientists who create or collect them do not know, in detail, what's inside. Thus the methods developed here support exploratory visualization. To be fully successful these methods must be fast, so issues of time-criticality are addressed. Fast global overviews are prepared automatically based on an analysis of patterns in the data. From these, particular overviews can be generated followed by detailed subviews, where these last steps are controlled by user interaction. A particular approach is developed to recognize spatial clustering in 3D data, and this is applied to a variety of datasets. The performance of the approach as a function of dataset size is analyzed, and it is found that it holds promise for the exploration of large datasets. In addition an octree decomposition method is also developed as an adjunct to the clustering method. Both methods can be used to develop hierarchical structures for the datasets, which can be extended by user interaction. Information derived from the methods can be analyzed so that patterns in the datasets can be segmented according to shape, size, dynamic behavior, or content

I. Introduction

Society faces a data overload. With the advent of instrumentation to collect and digitize ever growing amounts of data, faster and faster computers for spewing out numbers, networking to permit pooling of information, and cheaper, faster, and bigger data storage capabilities, we have more data available than we ever dreamed possible. The result in most cases is that most of this data goes untouched, either archived and never seen again or simply thrown out. However, these unexplored data stores often contain useful information and even important revelations. One approach frequently taken is to glean some useful information by boiling the data down to a few essential details using pre-programmed, automated processes and then displaying only those details. But the catch is that for such larger data stores even the experts, the users who generated them, do not know their structure in detail. Thus there is little chance that a pre-programmed procedure, even one based on an expert user's insight, will discover new features and capture all key details. This problem will continue to grow as we move from terabyte to petabyte storage capacities (and needs) and as even desktop computers have processing speeds approaching GigaHertz (and beyond).

There are many examples of this data overload situation from a wide variety of areas. Earth observations and simulational data will soon be augmented at rates of terabytes per day. Global companies with millions of transactions per day need hands-on capabilities to track their inventory distribution and sales. Atomic-level physics simulations of macroscopic processes (e.g., material crack propagation) now exceed a billion particles. High resolution global terrain databases now reach hundreds of gigabytes and will soon be much larger with the advent of high resolution geographic imaging satellites (capable of collecting data at 1M resolution). Very large scale, unsteady, 3D computational fluid dynamics simulations approach a GB per time step and must be followed for thousands of time steps. Scanned 3D seismic data that must be studied in detail to find ever more precious oil reservoirs approach a terabyte per area explored.

This paper presents an approach to attack the problem of investigating very large scale data. Our premise is that as datasets grow in size they inevitably become *exploration-dominant*. The first question one asks about such large datasets is, "What's in there?" This is especially so if the dataset is time-dependent, has lots of variables, and/or contains complex patterns. For exploration-dominant data, the first goal is to present quick global overviews based on user-selected variables and ranges. Further refinement of the data view is based on user selection. The user may see a pattern that she wants to explore in greater detail; she may decide that the initial variable range needs adjustment or that different variables should be displayed; she may want to navigate through time or through space to see dynamic data evolve or to get a new or clearer view; she may want to apply specific

visualization and analysis tools to regions of data. For optimal exploration, the user should be able to accomplish any of these actions with high interactivity and thus the process becomes one of *time-critical visualization*.

We present here a framework for time-critical visualization directed by user explorations. This framework is applicable to large datasets, and we show how it can be extended to datasets of very large scale and beyond. For such datasets a hierarchy of scales is necessary; we discuss the order of these scales and the time budgets that must be met for each order and for various exploratory operations and interactions. Finally we present an implementation and results based on a "fast clustering" method we have devised. We also discuss other implementations.

The key new items in this paper are the casting of the visual exploration framework into questions of scales, time budgets, and user involvement. This leads to a consideration of the trade-offs between time cost and accuracy. Whereas, for example, there have been many approaches to clustering that concentrate on getting reasonably optimal clustering, our approach also concentrates on deriving results that are both fast and "good enough". Here "good enough" may be evaluated in terms of whether the results fit into a framework that supports continuous user involvement (e.g., where users can easily dig deeper or even unfold detail pre-attentively--without conscious selection or application of tools) and whether enough detail is presented at any scale to give hints to important features that merit deeper investigation. Additionally our approach yields a set of general procedures applicable to data of many different types.

II. An Approach for Exploring Scalably Large Data

It is useful to enumerate some capabilities that an ideal approach to the visual exploration of datasets of any size and general type might have.

- Strict upper bound on complexity
- Adjustable threshold proportional to scene complexity. This should:
  - have a screen-space threshold.
  - be either automated or user-controlled.
- View-dependent detail management
- Closely-spaced levels of detail (LODs)
- On-the-fly development of hierarchical structure (*visualization-driven* hierarchies)

A strict upper bound on complexity is mandated by the needs of interactive visualization. Obviously the first step in evaluating raw data will be of the order of the complexity of the raw data. However, thereafter the ideal algorithm would have a strict upper bound on complexity that would be different than the raw data complexity and would never be exceeded, even for very large datasets. Indeed for the original evaluation of very large datasets, one can imagine methods that sample the dataset and thus have complexity of the order of the number of samplings rather than of the order of the raw data.

The notion of controlling complexity through frame-by-frame adjustment of a screen-space threshold comes from our work on terrain visualization [Lindstrom et.al., 1996; Koller et.al., 1995]. There we showed that such a mechanism could be used to relate perceptual fidelity to scene complexity. In particular this mechanism provides a direct way to choose on a frame-by-frame basis which details can be removed because they are sub-pixel. Such a threshold can be interactively adjustable so that when different details from different sources are fused in the rendered scene their complexities can be modulated based on relative importance. Closely related is the notion of view-dependent detail management in which the highest detail is rendered where the view is focused and where details not in the view are efficiently culled away.

Closely spaced LODs help optimize fast navigation. Distracting jumps and popping in detail are minimized, and the smooth unfolding of detail can significantly enhance the rate at which information is assimilated. We have shown a continuous LOD approach in our terrain work [Lindstrom et.al., 1996], where the system makes on-the-fly adjustments of details on a per pixel basis depending on visual importance. The result is smooth unfolding of detail during navigation. The challenge is to develop a comparable general set of methods for other types of data.

All these elements form a basis for effective and efficient user involvement. As we have discussed in the Introduction, user involvement is necessary for general and unexplored datasets of large to very large sizes, because there will be neither the time nor the capability for thorough and automatic explorations of their structures. However, our contention is that users will find it worthwhile to probe quickly presented global overviews (even if crude), especially if the exploratory system responds promptly to their requests for more detail. Indeed, a fully interactive approach opens the possibility of on-the-fly extension of hierarchical structure based on user focus or application of user-selected tools that operate by direct manipulation in the visualization environment. Fig. 1 shows a

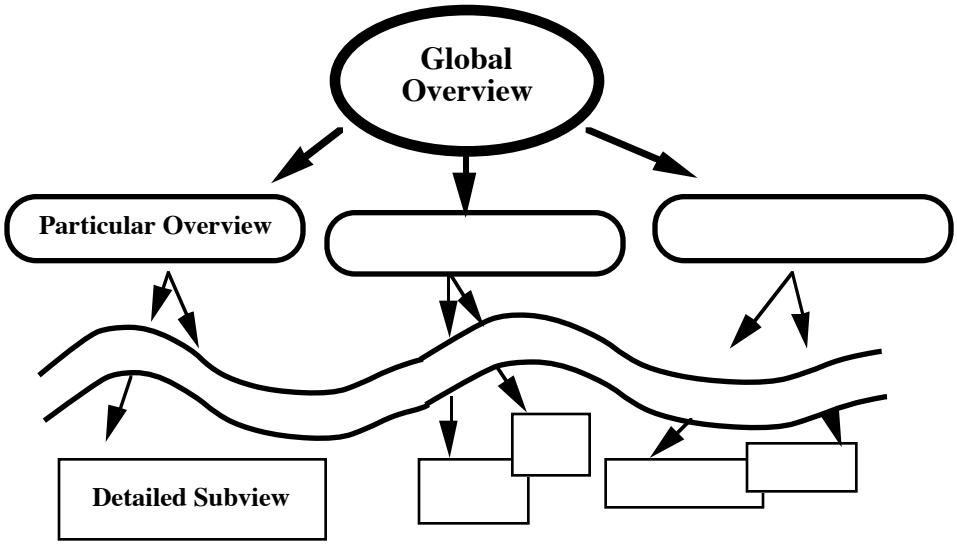

Fig. 1 A structure for exploratory visualization

structure for exploratory visualization based on the ideas of this section. Everything but the global overview might be developed based on user attention or selection. This user-in-the-loop approach need not be laborious; it should be semi-automated and only involve either simple or occasional user interactions. Also, the visualization and analysis methods in going from gross to highly detailed views could (and probably should) be different.

How can we reveal enough detail, even in a gross but fast global overview, to permit further exploration? And how can particular overviews derived from the global view, then its sub-views and so on, be fitted into a consistent structure for exploration? Our approach is to

Extract/Abstract/Model

at each level of exploration. To do this we need to develop a system that helps users

- Seek and formulate models
- Seek and formulate heuristics
- Develop more accurate models and then extract quantitative information

The idea is for the exploratory system to extract enough information so the user can derive preliminary abstractions that the she can follow and then establish initial heuristics for data behavior. The system should then provide tools for developing more accurate models and for quantitative analysis based on those models.

Our particular approach to extraction, abstraction, and modeling is to search for patterns in the data. If these are spatial data, the patterns will be spatial. However, the approach should support finding patterns in any collection of variables, whether spatial or not. The patterns might be clumping or clustering in the data, which might be defined in terms of the correlations of two or more variables. It is easy to define clumping or clustering in spatial data, but requires more care and thought for non-spatial data. Various techniques have been tried for non-spatial information spaces such as, for example, topical databases or Web spaces. In terms of visualizations, one question is how to define and display such clusters in the most visually-informative way [Wise et.al, 1995]. In spite of these questions, it is apparent that such an approach is general and should be applicable to data regardless of its type. After the patterns are found in the data, it should be possible to define simple models. For example, spatial clusterings should yield simple shape and volume information that can be depicted visually and also compiled into quantitative histories.

An ideal approach for visual exploration should also have optimal interactivity. At every level the system must be responsive enough to support and encourage exploration. Just what is optimal at all scales of exploration is unclear, although there is certainly a wealth of usability studies examining specific interaction regimes and performance in graphical user interfaces and virtual environments. To clarify the issues involved in optimal interactivity, we present the rough framework in Fig. 2. There should be some maximum amount of time to obtain a global overview, no matter what the scale of the dataset. It's not clear what that time should be, but if its too long, the user may be dissuaded from exploring the data. In fact, this is what often happens. For example, a colleague at Georgia Tech who generates sequences of computational fluid dynamics time steps, each step being quite large, typically does not look at them (except for automatically producing some

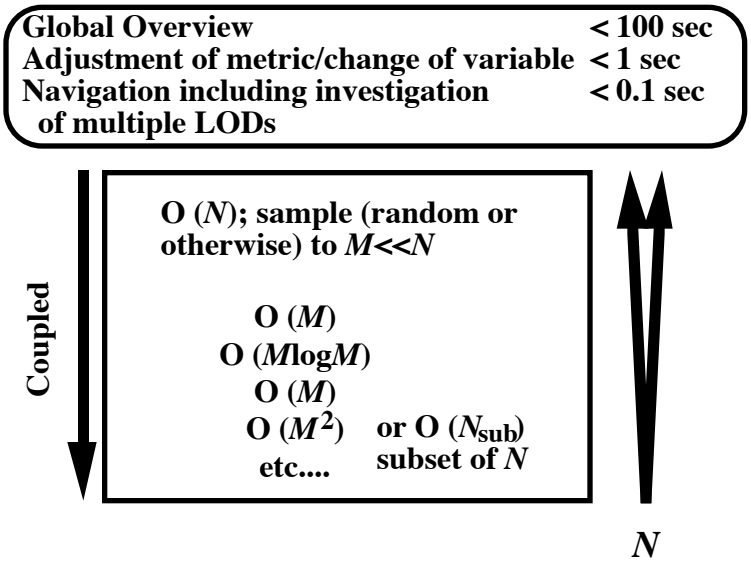

Fig. 2 A framework for optimal interactivity and exploration

general, averaged histories) because of the time barrier to producing individual visualizations. Let us say, then, that this initial time for producing the first view is of the order of 100 seconds--the point being that it should be of the order of a small number of minutes rather than an hour or more. Of course if one has a long sequence of time steps, each of which takes 100 seconds for initial display, one may well be reluctant to explore the data very deeply. In this case, time dependence may be the most important feature, and the system should be adjustable to allow display of even cruder features in less time if dynamic character can still be revealed. Alternatively, the entire collection of time steps could be considered as one huge dataset (with time as one of its dimensions) and then probed with pattern recognition tools. The initial visualization would have time as one of the axes, and there would be support for allowing the user to select subsets along the time axis for further exploration.

After the initial view, the time scale must change completely if one is to have effective exploratory visualization. For example, successful navigation in a variety of contexts requires system responsiveness (time between user input and display of the result of that input) of 0.1 seconds or less [Watson et.al., 1997; Bryson, 1993]. For optimal information collection, the system should also unfold detail (through multiple LODs) at that time scale. In addition, having seen data displayed for the first time, the user may well decide to adjust controls or ranges, or even change the variables being viewed. This process will require more computational effort than navigation; however it should still not take much time. Perhaps 1 second or less is a reasonable goal.

It is obviously not possible to achieve this time budget with methods that touch each data element in very large datasets. Our approach is indicated at the bottom of Fig. 2. Depending on the size of the dataset N, there is an initial sampling process of O (N). This can be a highly parallelized process (e.g., a bin sort method) but there still will be dataset sizes that do not fit the stated time budget, especially if the required time is only, say, 100 seconds. In this case random sampling would be the alternative where now the complexity is of O (S) where S is the number of samples. Not only is the random sampling also parallelizable, but in addition one can imagine how it could be extended to data structures that are collections of directories, distributed, and so on. We plan to investigate this further.

Whatever the initial step in our process, we end up with data at a sampled set of points M, whose properties are averaged over the properties of the original data elements associated with each sample point. Our position here is that this step should be fast and simple. In our present implementation we just perform a fast bin sort with no further analysis of, for example, variable distribution or correlation. After this step, further analyses are performed on the sampled points. To insure speed, the initial subsequent step might be O (M). As the user homes in on a specific part of the data, analyses might be more thorough (O (MlogM) or O (M^2), say). At some point the user might choose to look at the original data in some part of the visualization, or to look at a more detailed sampling of the original data. Note that we have provided the

option of retaining the links between each sample point and its original data, so such a choice is possible. But any display of the original data is based on user involvement. With the bin sort it is simple to select subsets of the original data for detailed viewing. It is apparent that this exploration process has an inherent hierarchy that should be exploited and may be extended by user involvement and selection.

To make these ideas more concrete we apply them to spatially coherent sets of data. Here the spatial coordinates are dominant, so one typically maps them onto the 3D coordinates of the visualization. Most scientific visualizations are of this type, and the data modeling is based on spatial clustering. One can then derive 3D spatial shapes from the clusterings, extract the shape information, and model features that can be followed, compared, and measured. The initial step in our approach is a spatial bin sort, which is applicable to data input in any structure. After that step the bins form their own structure which provides a consistent basis for subsequent steps. Before going on with the description of this approach, we will discuss relevant previous work.

III. Relevant Previous Work

There is little work that has been done that specifically considers the question of speed versus accuracy in developing tools for exploration and analysis of large-scale data, especially 3D data. In addition we are not aware of any work that develops a framework for explicitly immersing the user in the visual exploration process. There is, however, work on 2D systems that is relevant. Schneiderman and his group investigate *dynamic query interfaces* (DQIs) [Tanin, Beigel, and Schneiderman, 1997]. DQIs are a database access mechanism that provides continuous feedback to the user during query formulation. They have proven quite successful for small databases but slow down considerably for larger ones. Tanin et. al. discuss the response times that are necessary to provide continuous response and how larger databases might be arranged to achieve these. Their response times are similar to the ones discussed in Sec. II above. We can think of our 4D (3D + time) visual explorations as dynamic queries so that the work of Schneiderman's group should be applicable. In contrast to the work on fast interaction, there is a significant amount of work on cluster analysis of spatial or other (multivariate) distributions and also relevant work on structure recognition, shape analysis and spatial feature extraction. In this section we will address the work most relevant to the methods described in this paper but will not attempt an exhaustive analysis of this large field.

Clustering

Spatial clustering is the partitioning of a set of n data points into m subsets such that the sum of a distance (or similar) metric between each data point and the center of its cluster is minimized [Gross, 1994; Hagen, 1994; Hoffman, 1996]. Clustering is used for simplification by replacing all the points in a cluster with a single, average point at the center. In analyzing multidimensional data sets, one can either simply combine nearby observations and compute averages, or one can perform clustering in a state-space to group regions of similar characteristics together. Spatial clustering can be thought of as an optimal sub-space decomposition where the metric is based on distance, weighted distance, or a related parameter.

The disadvantage of clustering is that it can be very computationally expensive. There are $k^n/k!$ ways to assign n points to k clusters, and choosing the optimal clustering among these is an NP-complete problem [Gross, 1994; Hagen, 1994]. There are a variety of ways to show it is NP-complete, depending on the exact formulation of the clustering problem; some proofs work by considering clustering as equivalent to a related geometric problem like covering points with a certain number of disks of a certain maximum size or to some graph problems that can be solved by clustering nodes on the graph [Johnson, 1982]. One reason for the complexity of the problem is the interaction between each decision -- changing the assignment of one point to a cluster will change the centers of both clusters and thus affect the merits of other decisions [see Gross, 1994]. The only way to ensure complete optimality is to consider all or at least a large subset of all possible point assignments.

Many sophisticated approximation techniques, such as simulated annealing and cluster-finding neural nets, are relatively good at escaping local minima, but they are generally too slow or have too high overhead for interactive use. Some faster methods are k-means clustering perhaps combined with algorithms based on Voronoi diagrams of cluster centers. K-means starts with a fixed number k of essentially arbitrary clusters, and it chooses one point at a time to move from cluster to cluster, improving the arrangement step by step until a local optimum or error bound is reached. Since a formula exists for the effect of each candidate move on the error function, the worth of a move can be evaluated in constant

time [Gross, 1994]. Each iteration therefore costs n * k steps to choose a good point to move, and that cost can be reduced greatly by using a smart data structure to keep track of which clusters {1..k} are near enough to bother checking [Faber, 1994]. Choosing the initial cluster center positions is a key to both fast and optimal k-means operation. In the basic algorithm these centers are allocated either on a regular grid or randomly. Our method described in the next section provides a fast clustering approach based on a preliminary analysis of where these centers should lie. It uses a divide-and-conquer approach where new centers are inserted based on the shape, orientation, and weights of existing clusters. This should provide a faster path to optimal clustering than the arbitrary methods used in the basic approach.

Most authors using k-means do not analyze the number of iterations it takes. Selim, however, proves that it converges in a finite number of iterations [Selim, 1984]. Tovey analyzes the convergence of hill-climbing problems in spaces similar to the space k-means must search [Tovey, 1985]. He finds a convergence in 3/2 e*n iterations for problems that involve searching the vertices of an n-dimensional cube. The k-means problem is similar, except that instead of having a space corresponding to the vertices of an n-dimensional cube (n binary digits), it has $k^n/k!$ possible states, equivalent to an n-dimensional lattice with k sites along each edge of the cube. As described below, we use a finite number of iterations to constrain the problem complexity. In this instance, however, it would be quite useful to have some estimate of the distance from convergence.

Feature Analysis

A straight cluster analysis may not bring out certain features very well. For example, there may be certain shapes (e.g., annular regions or long filaments in the data) that would not be well represented by collections of clusters unless the number were rather large. Although Edelsbrunner has shown in his work on alpha shapes that clusters of spheres can, quite generally, be used to describe any 3D shape to arbitrary accuracy [Edelsbrunner and Mucke, 1993], in practice this method can be quite time-consuming and not appropriate for exploration of very large datasets. Simpler feature extraction and feature tracking techniques have been developed by Silver and her colleagues [Silver, 1995; van Walsum et.al., 1996]. Silver points out the utility and importance of "object-oriented visualization" techniques for segmenting and analyzing complex datasets. van Walsum et.al. develop iconic feature extraction methods for identifying and following 3D features. This work is directly applicable to the clustering approach we offer here.

IV. Fast Clustering

As Hagen discusses [Hagen, 1994], many spatial clustering methods begin by considering the whole set of data points and define the initial cluster center using some sort of weighted mean. A subdivision strategy is then applied to create new cluster centers. This procedure is often via the k-means method. As described above, typically a Voronoi construction is made around all the centers to define the nearest cluster center for the data points. Then one usually iterates until each Voronoi cell center coincides with the weighted centroid of the points in the cell. The iteration produces changes in the Voronoi cell boundaries, sometimes significant ones.

For our approach, key elements are how long the above steps take and how optimal the subdivision into new clusters can be given that it must occur in a restricted amount of time. Of course, the timing for each step depends on the number of data points, and steps such as the Voronoi construction can be time-consuming. (In the worst case it is $O(N^2)$.) As discussed in Sec. II, we get around this bottleneck by doing an initial bin sort and then forming clusters on the bins. The bin sort requires two steps, both $O(N)$. 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. For each bin, we keep a weighted centroid, weighted averages for selected variables, number of points, and (optionally) a list of data pointers. The optional list of data pointers can be turned on or off depending on whether its generation exceeds the time budget or space allocation. Using the list of pointers, other properties can be computed as needed. At this stage we don't perform further analysis (for example, to determine where we might need a finer set of bins to pick up detail). To an extent the clustering step in this section, or the octree decomposition described in the next section, carries out this analysis. Fig. 3 shows a progression through these steps. The initial feature extraction is easily based on the clustering step. Further analysis of the features (bottom of Fig. 3) can use the octree decomposition method described in Sec. V, further clustering, or another method.

As we have discussed in Sec. II, the bin sort results in a set of M sample points derived from the original N data elements. For very large datasets, $M << N$. The question arises as to the

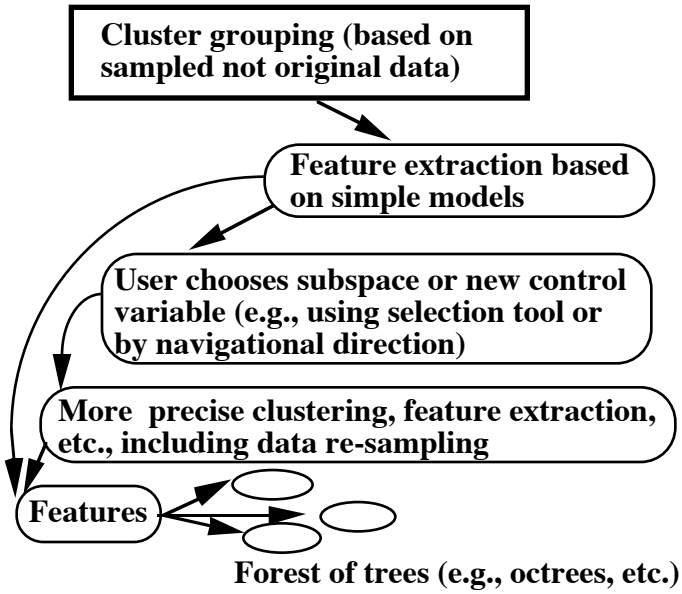

Fig. 3 Steps in the process of exploration and feature discovery

statistical error or uncertainty introduced by the sampling. For uniform sampling where every data element is touched, as described here, the optimal error should be of the order of the bin resolution. Thus spatial features smaller than the spatial bins would not be resolved. For random sampling a statistical analysis must be performed. For example, if random samples were dropped into a set of bins, the uncertainties in per bin averages of selected variables, fraction of total number of elements, etc., would depend on the total number of samples and the number in each bin. What uncertainty is "acceptable" is a topic that should be looked at further. Certainly if the user knows beforehand what size features are critical to analysis, sampling can be at a fine enough resolution to resolve these. (Here "size" is a general concept covering not only spatial extent and shape but also ranges in other variable spaces.) However, one should resist the temptation to trade too much interactivity for finer resolution. With high interactivity (including the ability to obtain finer detail quickly), the question becomes one of providing just enough detail in an initial view so that the user can ask for more where needed. Of course the cluster analysis and feature extraction described here are useful here because they bring out patterns in the data. The system could automatically analyze such structure and decide where to provide more detail based on some pre-determined criteria. But our focus is exploratory visualization, and here the user may not know what features are significant until she starts looking at the data. For exploratory visual analysis the user must be tightly in the loop.

Our fast clustering approach applies a rough analysis of cluster extent and orientation for each cluster. All analyses are performed on the weighted bins. The steps are:

- Determine cluster extent in x,y,z and define a box enclosing the cluster.
- Subdivide the box into 27 sub-boxes; a central one and 26 nearest neighbors.
- Compute the number of weighted samples and their centroids in each sub-box.
- Using a table of rules for the pattern of sub-boxes, define the orientation and shape of the cluster.
- Use the orientation and shape to decide how to subdivide the cluster. Collected information such as the number of weighted samples in the cluster, along with shape and extent information, can be used to determine *whether* to subdivide the cluster.
- Locally iterate a few times to determine optimal position of the new clusters with respect to existing clusters.
- Choose candidates for further subdivision (a priority list of candidates based on number of sample points and extent) and repeat first 4 steps for the new clusters and any clusters whose weighted averages (including number of points) have changed significantly. Otherwise skip to subdivision.
- Stop when the predetermined number of clusters has been reached or when another criterion has been met (e.g., when the total distance metric falls below a certain value or when certain thresholds are reached in terms of individual cluster extents or number of points).

The 27 sub-boxes permit an omnidirectional analysis of orientation and shape. By quickly applying a set of rules, one can determine whether the cluster is flat, round, or long and thin. Further one can find orientation information; in many cases this is clear enough to define a definite principal axis. The orientation information is quite useful in determining good positions for new clusters. We subdivide into either two or 3 clusters, depending on how the sub-boxes with the highest weights are distributed. Further the shape, orientation, and weighted averages provide a set of properties for each cluster that can be visualized or tracked for analysis.

The fast clustering approach is a variant of the k-means method discussed in Sec. III. However, we restrict the number of iterations in the optimization step because the method is only approximate anyway, especially during the intermediate cluster subdivision step. Often one uses a Voronoi decomposition to better define which sample points go with which cluster centers and to also determine what centers might be affected by the addition of a new center. Instead of doing this we use a measure of locality to determine the mostly strongly affected centers. It should also be possible to build an approximate, localized Voronoi construction that would be significantly faster than the full Voronoi construction and that could handle new centers. We plan to investigate this. But again one could delay global optimization to the final step of the cluster subdivision, as long as the centers are not too far away from their optimal positions, and still obtain an optimal clustering at that point.

Cluster Results

We have applied the above approach to sets of test data so that we have control over the size and distribution of the datasets. We are also now applying it to data from our global atmospheric models. The test datasets are 3D spatial distributions in varying patterns and of varying sizes. We clustered based just on the spatial distribution of the test data; there were no variables associated with each point and thus no analysis of variable distribution. For our data we wanted to see how well our clusters reproduced the distribution and clumping in the data and how fast they ran.

The number of data points in the datasets ranged from 5K to 200K. We chose one of the datasets for extensive timing tests. In Table 1 below we compare results with and without the bin sort for number of points up to 50,000. In the bin sorted set we first determined the x,y,z extent of the dataset and then divided these extents into n_x,n_y,n_z equal partitions. In this case the partitions are 10x10x10, so the total number of bins is 1,000. For each set of points, we found around 50-60 cluster centers. (The number varies slightly depending on the distribution of data points.) All calculations were performed on an SGI O_2 with R10000 CPU. Initialization is the time to read in the dataset and find its x,y,z extent, and set up the bins (in the bin sort case). "Iteration" is the time to perform the cluster subdivision and reorganization calculations until the target number of clusters is reached. Table 1 reflects that the bin sort is a small part of the total time as the number of points grows above 25K; further analysis shows that the data read takes most of the time. As we might expect, in the bin sort case (Table 1a) the cluster calculation times do not grow with number of points. However, the clustering without the bin sort grows in a super-linear fashion and is 40 times slower than the sorted case at 50K points. Although the timings in Table 1b can undoubtedly be lowered by optimization of the code, it is clear that one cannot do cluster analysis of the original data points if one has a small time budget.

Table 1a With bin sort

Size:	5,000	10,000	25,000	50,000
Initialization time:	0.198s	0.391s	0.990s	2.048s
Iteration time:	0.311s	0.263s	0.256s	0.361s
Total time:	0.508s	0.654s	1.246s	2.410s

Table 1b Without bin sort

Size:	5,000	10,000	25,000	50,000
Initialization time:	0.196s	0.376s	0.912s	1.808s
Iteration time:	3.913s	8.705s	36.815s	93.650s
Total time:	4.110s	9.081s	37.727s	95.458s

To further investigate the effects of dataset size, we looked at bin sorted datasets with up to 200K points (Table 2). It is clear that the effects of the initial data read begin to dominate while the clustering continues to remain constant. At 200K points the data read time is nearly 20 times that of the clustering, dominating both the clustering and the bin sort. Thus if we had a time budget of, say, 100 sec. (see Fig. 2) for the initial global overview of the data, we might be restricted to datasets no larger than around 2M points. Of course, these results clearly suggest where big improvements in speed can be made. If the data reading (and perhaps the bin sort, too) were parallelized, significantly larger clusters could be rendered within the time budget. Ultimately, however, touching each data point no longer works and one must convert to a statistical sampling approach. This analysis gives information about where that point might occur.

Table 2 Bin sorted clusters as a function of size

Size:	50,000	100,000	200,000
Initialization time:	2.048s	4.21s	8.31s
Iteration time:	0.361s	0.39s	0.40s
Total time:	2.41s	4.60s	8.71s

The question remains as to how good our fast clustering method is. We have not done a thorough analysis, but our results indicate that the method is reasonably good in defining appropriate clusterings. Certainly it appears good enough to give a quick overview of data patterns and structure with enough information that the user can build a more detailed exploration. For example, Fig. 4 shows both the points from one dataset and the cluster centers derived from them. This is for the case where a bin sort was applied first. The cluster centers are represented by spheres whose sizes are proportional to the number of points contained. We chose this simple representation for this first work but have additional information about the shape and orientation of the clusters that we could also represent. This information will be the basis of the feature extraction and feature-following methods that we will pursue in the future. The "arched" spatial structure of the dataset in Fig. 5, with number of points varied as needed, forms the basis of the analyses presented in Tables 1-3. We have tried datasets with other spatial structures and find that clustering also captures these structures reasonably well. For comparison we have also run the arched dataset without the bin sort as shown in Fig. 6. The clustering results with and without bin sort show the same arched structure and clumping of cluster centers near the top of the arch, though they differ in some details. This general agreement is good news because it supports using the much faster bin sort method. We are presently working to further minimize the differences between the two methods.

V. Octree Hierarchical Construction

The above clusters can be put in a loose hierarchical structure (e.g., clusters unfold into sub-clusters and so on), but it is often useful to make a more formal hierarchical construction with a well-defined structure. Among the simplest 3D spatial hierarchies are octrees, which have been applied to a variety of topics including visualization of 3D scalar fields and development of multiresolution grids for 3D simulations. Here we use them for exploratory visualization because it is relatively easy both to adapt the level of detail to local features and to control the level of detail dynamically.

In our approach we initially subdivide each octcell until its data satisfies some constraint, such as the number of points in the cell reaches a predetermined minimum, a measure of the variance in a control variable falls below a certain value, or the depth of the octree at that cell exceeds a predetermined level. The subdivision can be controlled interactively by adjusting any of these controls. The selection could occur either by an active choice or by a procedure that responds to the motion of the viewpoint through the data. In practice we apply a combination of these constraints; for example, we use a variance threshold in the control variable and also don't allow the octree to exceed a certain depth. Of course, the octree approach has the usual problem of possibly dividing clumpings of data along arbitrary cell boundaries. This may be acceptable when the approach is used as an exploratory technique to be followed by more precise analyses. And the octree provides an efficient hierarchical data structure that can be used for

further data analysis or can even be a building block for other techniques.

The chief difficulty in creating octrees for large datasets is that building a tree by recursive subdivision can be rather slow, since each point must be touched at each level (unless a cell reaches a threshold and thus does not divide into children at the next level) . As a trade-off we apply the bin sort described in the last section (in this case to 2^{3n} bins where there are n+1 octree levels) and initialize the corresponding octree. The depths for the initial visualization can then be set lower in less interesting regions by setting a 'deleted' flag on particular octree branches. When the user selects individual regions to deepen, the cost of expansion will be less than the cost of expanding the whole tree because only a fraction of the points will need to be resorted.

To choose which bins to merge into the next higher level of the octree , we first move up the tree to compute summary statistics at every node. Then we move back down the tree, deleting the children of any node that fails to satisfy constraints such as those mentioned at the beginning of this section. The 'deleted' flags speed the response to a user's request to show more detail in a region. If the user selects an area with some deleted bins, the system can simply change flags back to 'undeleted' to refine the visualization. During this process a hierarchical pointer structure is built connecting parent cells to children and providing pointers to original data values at the lowest levels of the octree. The pointers to data values permit analyses based on subcells of the octree to be carried out, where the hierarchical structure permits the correct data values to be located quite quickly. Of course, as the dataset grows in size, retaining these pointers to original data may produce an unbearable overhead. Thus we have provided the option of turning off these pointers; the user would access the original data for further analyses only after choosing a small region of the hierarchy.

One disadvantage of starting with a tree expanded to a constant depth is that it may waste a lot of memory on sparse branches. Furthermore, allocating large blocks of memory for each bin can be the slowest part of using the octree. We have found that we can reduce those costs to a barely noticeable level by allocating only a small amount of initial memory to each bin, and then allocating additional memory as needed on a bin-by-bin basis. For example, each bin might start with memory proportional to the average number of points per bin in the entire dataset. As the bin sort progresses, some bins will remain almost empty, and the bins that start to fill can request additional memory.

Octree Results

To test out our approach we have applied our octree method to some of the test data described in the last section. Fig. 7 reveals the hierarchical subdivision for a dataset with three main clumpings of data. A simple metric based on the density of data points was used to decide where to merge octcells. One can easily add a metric based on the distribution within the cell. Fig. 8 reveals more detail in the data clumpings by coloring cells where the color represents density of contained points. Gold cells have the highest density of points, and some cells are made semi-transparent to reveal the internal structure. Results for the test datasets show that the octree distributions match the fast cluster distributions. Since the octrees tend to be slower, one might employ the fast clusters for initial overviews. However, the octrees provide a data structure that can even be used with the clusters and offer an orderly LOD structure when more detail is desired. This structure can be traversed quickly with delete and undelete flags set to switch detail off and on.

VI. Exploratory Visualization Applied to Terrain

A significant motivation for this work is our ongoing work on real-time terrain visualization [Lindstrom et. al., 1996; Koller et.al., 1995; Lindstrom et.al, 1997]. In this paper we recast the terrain visualization as a problem in exploratory visualization of very large databases. It then becomes a working demonstration of how one can start with a (global) overview and unfold ever greater detail, in this case continuously and just by looking closer at regions of interest.

VGIS (Virtual Geographic Information System) has recently been totally revamped into a global visual simulation system [Lindstrom et. al., 97]. It support accurate depiction of terrain elevation and imagery, moving vehicles, buildings and other static objects, and features such as ground cover, trees, and roads. VGIS permits interactive display of geo-specific global terrain down to millimeter resolution at interactive rates. It provides a geographic-based data structure where elevation and image data can be added to existing datasets and where nested high resolution insets can be placed accurately in lower resolution background data. VGIS supports fine-grained multithreading for maximum CPU utilization and rendering speed, navigation within multiple independent windows, and detailed control of visual detail allowing either manual or automatic

mechanisms for balancing display accuracy and rendering speed.

VGIS conforms to several of the criteria established in Sec. II. The continuous LOD terrain algorithm [Lindstrom et.al., 1996] produces an adjustable threshold proportional to scene complexity. It is view-dependent and has a screen-space threshold. The latter means that one has a direct means of relating scene (and thus time) complexity to image quality. It has an optimized data paging and caching mechanism whereby data is brought in (in fixed size blocks) and is swapped with data blocks no longer in use. On the other hand, the VGIS hierarchical structure exploits the special characteristics of terrain or other height field data. It builds its quadtrees from the bottom up in contrast to the top-down approach used in the fast clustering methods. Highest resolution patches for any given area form leaf nodes to which lower resolution nodes are linked until finally they connect to the lowest resolution global overview. All data is pre-processed so that it can be inserted into the linked hierarchical structure for fast retrieval. Although this precludes the fast overview discussed above, it is appropriate for terrain data where one usually has a good general idea of which high resolution areas are of interest. Furthermore, we have recently developed new capabilities in terrain dataset building that permit high resolution inserts to be added to existing datasets in relatively short time, no matter what the size of the existing dataset. Thus one could explore more flexibly by adding (or removing) higher resolution patches to a global view. In fact an expanded version of VGIS that we are planning will have both the top-down (for visualizing certain 3D fields of data) and bottom-up (for terrain) approaches depending on the type of data.

In Figs. 9a and 9b, we show navigation of a terrain dataset where one moves from a view of the Southeast U.S. to a closer view of Georgia around the Atlanta area. The terrain dataset being navigated is nearly 3 GB in size. It starts with an overview of the world at 8 Km resolution, has the U.S. at 1 Km resolution, Georgia at 100 M resolution, central Atlanta at 1 M resolution, and the Georgia Tech campus at 0.5 M resolution. Fig. 10 shows a 3D view of downtown Atlanta that appears as one moves closer. Thus a user can continuously navigate (except for a few momentary pauses to page in high resolution data) from a view of the whole earth to discover features that are just a few feet across. Detail unfolds and features reveal themselves as the user moves closer to them. This is an example of what we wish to achieve with our visual exploration methods.

VII. Conclusions and Future Work

In this paper we have considered the visual exploration of scalably large datasets and developed some new methods for this process. Since user involvement is essential in this process, we looked carefully at issues of time criticality. We have thus cast the visual exploration framework into questions of scales, time budgets, and user involvement. In particular we have shown that one can get useful global overviews from resampled data at a small fraction of the cost for considering the whole dataset. Nevertheless one can still delve down to get greater detail in any part of the data that evokes interest and can, if desired, even analyze original data. In addition an octree decomposition method is also developed as an adjunct to the clustering method. The data structure derived for the octree method is of use for both methods considered here. Both methods can be used to develop hierarchical structures for the datasets that can be extended by user interaction. Information derived from the methods can be analyzed so that patterns in the datasets can be segmented according to shape, size, dynamic behavior, or content. Finally we recast our terrain visualization approach as a problem in exploratory visualization. This highlights key general issues for developing tools to explore very large datasets and shows what can be done with a particular, optimized application.

This research suggest much future work. Data from the clustering and octree methods can be used to establish 3D features that can then be categorized, tracked in time (or in response to the changing of any other variable), and analyzed. New methods for unfolding the detailed structure of selected parts of the dataset via user interaction should be pursued. These methods will lead to on-the-fly building of a database hierarchy. Based on the techniques presented here, new, more accurate fast clustering methods with larger numbers of clusters and support for sub-clustering could be developed. The work described in this paper suggests a separation of data I/O and initial sorting procedures to data-resident processors (perhaps distributed or running in parallel) followed by transport of results to a visual interface for analysis and steering of the exploration process. We are pursuing several of these possibilities.

VIII. References

1. Bryson, S. (1993). Implementing virtual reality. *SIGGRAPH 1993* Course #43 Notes, pp. 1.1.1-1.6.6, 16.1-16.12.
2. Edelsbrunner, H. and E. Mucke (1994). Three-dimensional Alpha Shapes. *ACM Trans. Graphics* 13, pp. 43-72.
3. Faber, V (1994). Clustering and the Continuous K-means Algorithm. *Los Alamos Science* 22.
4. Gross, M (1994). Subspace Methods for the Visualization of Multidimensional Data Sets. Scientific Visualization, pp. 172-185 (Academic Press, New York, 1994).
5. Hagen, H (1994). Visualization of Large Data Sets. Scientific Visualization, pp. 186-198 (Academic Press, New York, 1994).
6. Hofmann, T., J. Puzicha, and J. Buhmann (1996). "Unsupervised Segmentation of Textured Images by Pairwise Data Clustering. *International Proceedings of IEEE International Conference on Image Processing* Vol. III, pp. 137-140.
7. Johnson, David S. (1982). The NP-Completeness Column: An Ongoing Guide. *Journal of Algorithms* 3, pp. 182-195.
8. Koller, David, Peter Lindstrom, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner (1995). Virtual GIS: A Real-Time 3D Geographic Information System. Report GIT-GVU-95-14, Proceedings *IEEE Visualization '95*, pp. 94-100.
9. Lindstrom, Peter, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner (1996). Real-Time, Continuous Level of Detail Rendering of Height Fields. Report GIT-GVU-96-02, *Computer Graphics (SIGGRAPH 96)*, pp. 109-118 (1996).
10. Lindstrom, Peter, David Koller, William Ribarsky, Larry Hodges, and Nick Faust (1997). An Integrated Global GIS and Visual Simulation System. Report GIT-GVU-97-07, submitted to *Transactions on Visualization and Computer Graphics*.
11. Selim, Shokri and M.A. Ismail (1984). K-means-type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6 1, pp. 81-73.
12. Silver, D. (1995). Object-Oriented Visualization. *IEEE Computer Graphics and Applications*, 15, 3 pp. 54-64.
13. Tanin, Egemen, Richard Beigel, and Ben Schneiderman (1997). Design and Evaluation of Incremental Data Structures and Algorithms for Dynamic Query Interfaces. Proceedings *IEEE InfoVis '97*, pp. 81-86.
14. Tovey, Craig (1985). Hill Climbing with Multiple Local Optima. *SIAM Journal of Algorithms and Discrete Methods* 63, pp. 384-393.
15. van Walsum, T., F. Post, D. Silver, and F. Post (1996). Feature Extraction and Iconic Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2,2 pp. 111-119.
16. Watson, Ben, Neff Walker, William Ribarsky, and Victoria Spaulding (1997). The Effects of Variation of Frame Rate And Latency on Performance in Virtual Environments. Report GIT-GVU-96-33, to be published in *Transactions on Computer-Human Interactions.*
17. Wise, J., J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow (1995). Visualizing the non-Visual: Spatial Analysis and Interaction with Information from Text Documents. Proceedings *IEEE InfoVis '95*, pp. 51-58.