Discovery Visualization and Visual Data Mining

William Ribarsky, Jochen Katz, Frank Jiang, and Aubrey Holland
Graphics, Visualization, and Usability Center
Georgia Institute of Technology

Abstract

This paper describes discovery visualization, a new visual data mining approach that has as a key element the heightened awareness of the user by the machine. Discovery visualization promotes the concept of continuous interaction with constant feedback between man and machine and constant unfolding of the data. It does this by providing a combination of automated response and user selection to achieve and sustain animated action while the user explores time-dependent data. The process begins by automatically generating an overview using a fast clustering approach, where the clusters are then followed as time-dependent features. Discovery visualization is applied to both test data and real application data. The results show that the method is accurate and scalable, and it offers a straightforward, error-based process for improvement of accuracy.

1 Introduction

To attack the problem of dealing with increasingly vast stores of information, we discuss in this paper a new approach to data exploration that requires the close coupling of man and machine. We call this approach *Discovery Visualization* to emphasize the importance of visual display and visual interaction and the fact that the purpose is to discover new relations, new features, new knowledge. A key element in discovery visualization is to heighten the awareness of the user by the machine so one has, for example, focus-based manipulation (i.e., based on where and how closely the user is looking) in addition to direct manipulation. This process makes no sense unless the machine is highly responsive. Further, we promote the concept of continuous interaction with constant feedback between man and machine and constant unfolding of the data. Finally, there must be a combination of automated response and user selection to achieve and sustain animated action, even in datasets of great or varying complexity. This means not only smooth depiction of time-dependent data but also depictions of motion parallax, changes in pattern and shape, and other behavior as one navigates through the data.

When one speaks of burrowing down into a data collection, extracting features and then unfolding those features to reveal inner details, one naturally thinks of data mining. Indeed, there are methods from this area that are applicable to the work described here. Ganti et al., [Gan98] recently showed how one could extract cluster features from arbitrary, non-spatial information collections. The clusters are characterized by a measure of their contents (in the form of a generalized distance metric) and their relations to other features. One can build on-the-fly hierarchies [Zha96] of these features, which are height-balanced and efficiently traversable. Scalability has been considered [Gan98, Est95; Ng94] by explicitly allowing for databases that must exist outside of main memory, developing clustering methods that are efficient and no more than O(n), and deriving incremental update methods for the clusters, their properties, and their hierarchical structures.

However, there are fundamental differences between the approach described here and data mining. First, discovery visualization is centered around the user, with responsiveness matched to maximizing her capabilities. Our previous work [Lin96, Lin99] shows that this requires considering, from the start, the time budget and efficient structure of the end-to-end system including data organization, data transfer, compact representations, and high efficiency in selecting and operating on data. In particular our approach is coupled closely to 4D (time-dependent) visual display and visual interaction, which requires close attention to the organization of data for graphical representation and fast, accurate selection via the visualization. In effect we do not seek to support a query mechanism of distinct call and response but rather an exploratory method of continuous adjustment and feedback. We believe this quite different approach holds great promise.

2 Related Work

In general 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. There is, however, relevant work on 2D systems. Schneiderman and his group investigate *dynamic query interfaces* (DQIs) [Tan97]. 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. In contrast to this work, 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 developed in this proposal but will not attempt an exhaustive analysis of this large field.

2.1 Clustering

Spatial clustering is the partitioning of a set of n data points into m subsets such that the sum of distances (or a similar metric) between each data point and the center of its cluster is minimized [Gro94, Hag94, Hof96]. Clustering achieves simplification by replacing all the points in a cluster with a single, average point at the center.

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 [Gro94, Hag94]. 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. 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. A faster method is 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 [Gro94]. 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. We show that this provides a faster path to optimal clustering than the arbitrary methods used in the basic approach.

More recently there has been some work on the visualization of cluster hierarchies [Hec98] that is similar to the methods developed here. In this previous work an eigenvector analysis finds principal components for determining subclusters. Our approach (see Sec. 3) is to divide a cluster region into sub-boxes for quick analysis of orientation and extent (leading to assignment of subclusters). We use this instead of principal components because it provides a fast analysis with strict time limits. Our approach also differs from that of Hec98 and others because it is scalable and because real-time constraints are applied throughout the analysis and display processes in support of discovery visualization.

2.2 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. However, if one had a highly interactive means to change the number of clusters and thus the amount of detail in the clustering, the lack of optimality in describing certain shapes would be less of an issue, since one could easily adjust detail for any feature of interest (assuming that one had an effective mode of interaction and display).

Of course the clusters themselves can be features, can be given an identity or meaning, and will have a set of properties that can be tracked. They then become a sort of “visual shorthand” for the data collection as a whole, except they will have (for a given error threshold) a complexity that is typically

much less than the complexity of the whole data collection. Simple feature extraction and feature tracking techniques have been developed by Silver and her colleagues [Sil95, Wal96]. 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. However, neither this nor other feature analysis approaches address what happens when datasets become very large, nor do they directly assess or constrain the complexity and time cost of their methods. Furthermore neither the simpler nor more exact feature analysis methods extend beyond direct spatial clustering to the distribution of other variables, including completely non-spatial distributions. Our clustering approach is generalizable.

3 Discovery Visualization Framework

We start by considering a framework for optimal interactivity for exploration. Successful navigation or other interaction in a variety of contexts requires a system responsiveness (time between user input and display of the result) of 0.1 second or less [Bry93]. To achieve this regime of time budgets for very large data, one must apply a sampling to reduce the number of elements from N to M where M<<N.

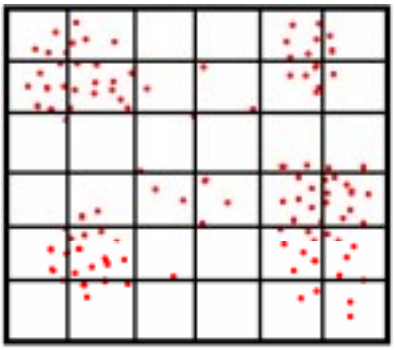


Fig.1 Applying bin sort to a data collection

3.1 Clustering Method

We give a brief description of the clustering method used in the discovery visualization approach. A fuller exposition is in Ref. Rib99. To identify 3D cluster centers, we define a spatial vector and average:

The sampling itself must be fast—we assume O(N). Obviously beyond some value of N even this will exceed the time budget, and we must resort to a method that does not touch every data element, such as random sampling.

The user now explores using the sample data. Our approach is to apply a fast clustering and feature extraction method that is O(M). As the user narrows her focus, homing in on details of interest, the analysis might be more complicated and more quantitatively accurate. One could find isosurfaces or volume visualizations, for example. At this point the user might do a subsampling at finer resolution or even look directly at a small subset of the original data.

Our discovery visualization approach thus starts with an initial bin sort, which requires two steps, both O(N). (See Fig. 1.) 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, the number of points, an error value, weighted averages for selected variables, and ranges for each variable. The bin sort is simple and highly parallelizable and, quite importantly, enables the uniform handling of data in any format (e.g., multiple meshes or observational and simulational data in completely different formats).

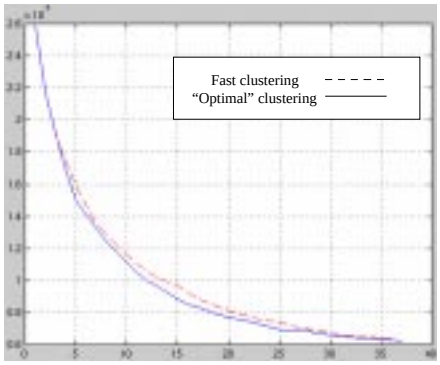


Fig. 2 Number of clusters (horizontal axis) versus total error.

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

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:

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

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

These definitions are enough to select "optimized" cluster centers based on either minimizing R for a given N_C (number of clusters) or choosing N_C such that $R/N \le \tau$ 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.

This formulation just gives spatial clustering. To include the effect of a variable, say V_2, we could define a new error R_V:

$$R_V = \sum_c \left(\frac{1}{k_c} \sum_{i \in c} S_{i,c}^2 \sum_{i \in c} D_{i,c}^2 \right)^{1/2} ,$$

$$S_{i,c}^2 = (V_{2i} - \overline{V_{2c}})^2 \quad (2)$$

R_V is invariant under spatial translation or addition of a constant to V_2 and tends to zero in a region where *either* V_2 is constant or the points are tightly clumped. Thus a cluster placed in such a region could encompass a wider spatial volume without raising its total error much. If one were exploring the distribution of V_2 on a uniform mesh, for example, this would be desirable since the underlying spatial distribution would be uninteresting. Yet Eqs. 2 retain a spatial dependence for regions where V_2 *does* vary. To bring out different characteristics for different types of data, one might use a different formulation for R_V. Alternatively, one could combine V_2 with $\mathbf{r}_i$ and use this extended vector in Eqs. 1. The problem with this approach is that it is not clear, in general, how to combine the different units associated with V and **r** (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]. However, whatever the form of R_V, we include time-dependence by treating *t* as a parameter and assuming all variables and coordinates depend on *t*. We then use spatial and time coherence to track features.

3.2 Clustering Algorithm

Using weighted bins and the error measures of Eqs. 1 and 2 (depending on whether the application data is totally spatial or has a strong variable dependence), we have developed a fast clustering approach to perform a rough analysis of cluster extent and orientation for each cluster. The steps are:

1 Determine cluster extent in x,y,z and define a box enclosing the cluster.
2 Subdivide the box into 27 sub-boxes; a central one and 26 nearest neighbors.
3 Compute the number of weighted samples, their centroids, and error in each sub-box.
4 Using a table of rules for the pattern of sub-boxes, define the orientation and shape of the cluster.
5 Use the orientation, shape, and relative errors to decide how to subdivide the cluster.
6 Locally iterate a fixed number of times to determine optimal position of the new clusters with respect to existing clusters.
7 Choose candidates for further subdivision (a priority list of candidates based on number of sample points and error) and repeat from step 1 for the new clusters and any clusters whose weighted averages (including number of points) have changed significantly.
8 Stop when the predetermined number of clusters has been reached or when the error threshold is reached.

The 27 sub-boxes permit an omnidirectional analysis of orientation and shape. By quickly applying a set of rules to the boxes with the highest weights (weighted centers and errors are used for each box), one can determine whether the cluster is flat, round, or long and thin. Further, the sub-boxes reveal orientation information. The algorithm subdivides (step 5) into 2 or 3 clusters, depending on how the sub-boxes with the highest weights are distributed. If we had, for example, 3 largest centers in a row either diagonally or along a box edge direction, we would subdivide the cluster along this axis.

Time tracking is implemented by starting with the clustering for the initial time step and then keeping an error close to the threshold τ for subsequent time steps. For coherent behavior over the time steps

the clusters will tend to move smoothly. When two clusters come together spatially, they will merge, whereas a cluster that spreads will tend to break into subclusters. If a variable is included in the error, its dynamic changes will also affect the clustering. Including time dependence entails adding the following steps:

1 If the total error is above the threshold plus a buffer value, divide the cluster with the largest error.
2 If the total error is below the threshold plus a buffer value, merge the clusters that will give the smallest combined error.
3. After iteration, repeat the procedure if the total error remains above or below the threshold (plus buffer).

Breaking clusters with the largest error but merging clusters with the smallest error helps keep the clustering near its optimum distribution for a given error threshold or number of clusters. The buffer values reduce the possibility of repeated jumping above and below the threshold. In addition we have imposed a limit to these jumps, should they occur. In practice this buffer method work well.

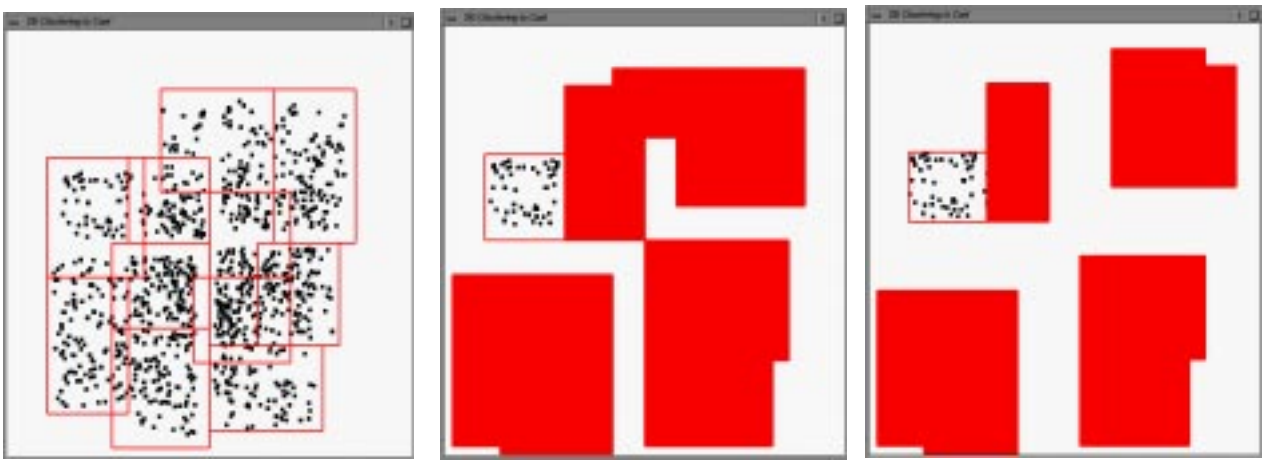
3.3 Timing and Accuracy

To test the timing and accuracy of the fast clustering in our discovery visualization framework, we have analyzed 3D test data, where the clustering was based just on the spatial distribution of the data and there was no time-dependence. The number of data points in the datasets ranged from 5K to 200K, and for each set of data points we found 50-60 clusters and partitioned the dataset into 10x10x10 or 20x20x20 bins. For larger datasets, the total time did not depend much on the number of bins since the time to touch and sort each data point dominated. As an example, for 100,000 points using an SGI R10000 processor, the initialization time (to set up bins and sort data) and total time (including clustering) were 1.64 sec and 1.69 sec for ASCII data and 0.22 and 0.27 sec for binary data. In contrast the total times for clustering without the bin sort were over 60 times greater, showing the effectiveness of our approach in producing fast clustering. Since the bin sort is easily parallelizable, there could be a parallel approach that would significantly lower timings. However for a finite number of processors, there will still be an upper limit above which a statistical sampling approach must be used. Note that after the bin sort, the clustering and feature tracking do not depend on the original size of the dataset and will be quite fas t.

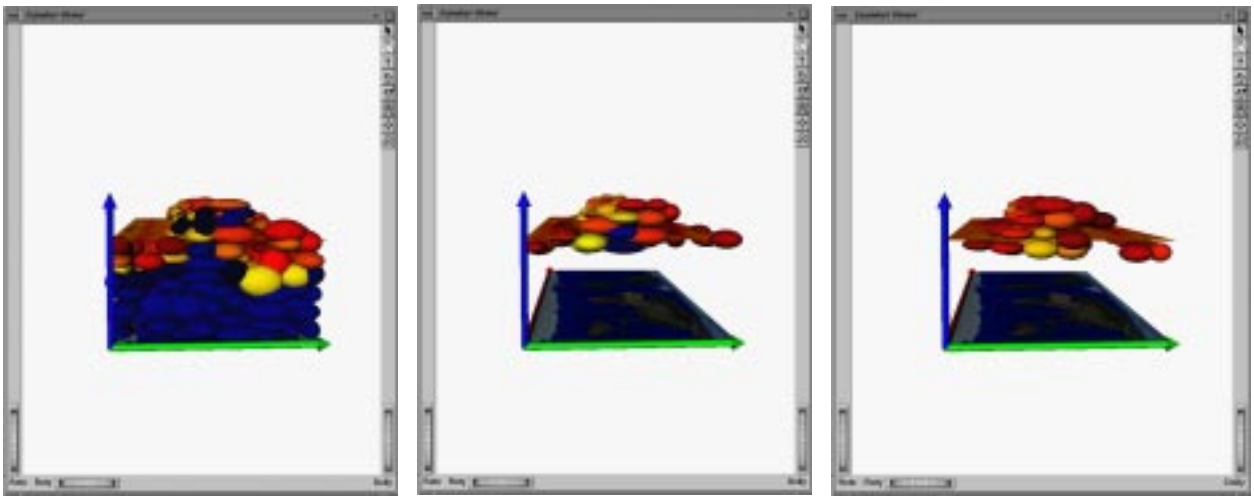
Fig. 2 compares, for one of our 3D test datasets, fast clustering to “optimal” clustering, showing behavior as the number of clusters changes. Optimal clustering is determined by finding cluster centers through several cycles of random placement and then iterating until centers do not move any more and errors are constant. The cluster arrangement with the lowest error for each number of clusters is chosen as the optimal clustering. We see that the shapes for fast and optimal clustering are nearly the same and, for these data, fast and optimal clustering are quite close. This offers a criterion for quality of fast clustering. The shape of error versus number of clusters should be close to the shape in Fig.2. (Tests with other types of data all give similar shapes.) In addition the error for a given clustering should not be “too far” from optimal. The latter is harder to determine for general cases.

4 Application to Large-Scale Data

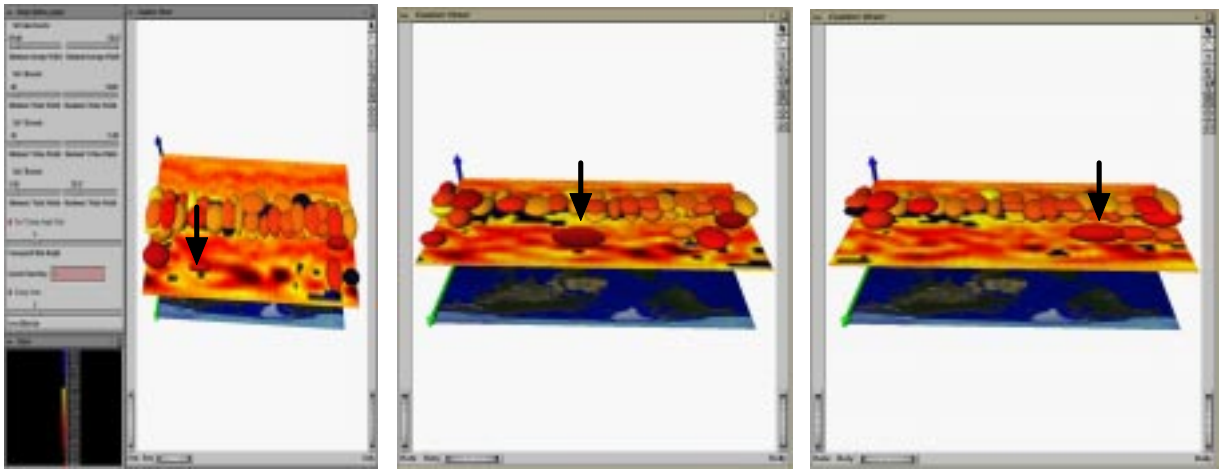
We have found that the discovery visualization as implemented in the clustering algorithm provides effective visual data mining. We considered the movement of large collections of independently moving objects in 2D, such as vehicles moving on a terrain. This is an important application, for example, where one may need to quickly track the distribution and formations of large numbers of vehicles in a large-scale battlefield simulation. Here the features might be battalions at one level and platoons at a lower level. Discovering how these collections form and how they move is as important as seeing the movements of individual vehicles. Figs. 3 show clustering applied automatically to moving groups of objects over a 2D surface and then represented by simple boxes. The clusters describe well the underlying dynamic structure and are computed quite fast—at least 20 frames per second for over a thousand objects even on a small SGI or PC. One can interactively change the threshold to get larger or smaller clusters. In the first frame of Figs. 3 all cluster boxes are made transparent to show the data distribution. One can see and then follow the underlying data by selecting a cluster, as shown in the last two frames. One could also do this selection automatically, based on distance of the eyepoint from the cluster. This application shows two main aspects of visual data mining, being able to see features in context and being able to selectively and directly unfold them to reveal inner detail.



Figs. 3 Fast clustering applied to movement of collections of objects on a 2D surface.



Figs. 4 Global atmospheric model at time steps 0, 50, and 17 0.



Figs. 5 Global atmospheric model showing movement cluster in successive time steps.

For a different application, we ran the discovery visualization method on some real data obtained from a global atmospheric simulation. This is a time-dependent simulation of N_2O comprising several months of time steps [Jea95]. Since the N_2O concentration is an important analysis variable, we ran the clustering using Eqs. 2. The simulation produces time steps marked 15 minutes apart so a massive amount of data is collected when the simulation is run for several months. These data are a good candidate for discovery visualization because the developers of the global atmospheric model have found the data too large and complicated to produce overviews for the complete set of time steps. Because of limited disk space, we restricted our data collection to one time step every four hours and accumulated time steps for September, October, and November. Still this amounted to over 550 time steps. The collection of a complete set of cluster information, however, took only 40 minutes for all 550 time steps on an R10000. (The clustering algorithm was not optimized; we expect the total time can be significantly lower.). Figs. 4 show the cluster results at time steps 0, 50, and 170. At the bottom is a map projection of the earth oriented so that the latitude axis is along the horizontal direction. (The vertical axis is altitude; the top of the clustering is at about 30 miles.) The plane in the middle shows color contour slices of N_2O concentration and can be set interactively. The coloring goes from blue (high concentration) to dark red (low concentration). The clustering in Figs. 4 shows a characteristic peaked N_2O distribution at the equator, but in addition it shows how this distribution shifts rapidly from the North Pole towards the South Pole (right to left in the figures) and changes shape. Since the N_2O distribution is similar to that of ozone, the cluster visualization shows how the southern ozone hole fills in after the southern winter. The first frame of Figs. 4 also shows the complete distribution of N_2O. The uninteresting lower altitude (blue) clusters are easily clipped away in subsequent frames.

Figs. 5 show the cluster visualization over time from a different orientation. The longitude axis is now along the horizontal direction, and the view is from the North Pole. By interactively rotating the view from north to south while stepping through time, one can easily see that the flow of N_2O in the upper atmosphere is along the longitudinal direction near the poles (i.e., circulating around the poles) and is significantly greater around the North Pole. The black arrows in Figs. 5 show the progression of a particular cluster feature over successive time steps at the North Pole. The N_2O contour slice reinforces this flow pattern. As a variation of the unfolding detail capability discussed for the 2D data, we have implemented a transparent 3D box whose shape and position can be set by direct manipulation. Clusters within the box can be shown in greater detail or with another variable (e.g., the windfield vectors). Quite importantly for visual data mining there is high interactivity—the animation proceeds smoothly on an SGI IR and produces several time steps per second even on an O2.

5 Hierarchical Structure

A complete visual data mining approach needs a structure that will support a highly interactive exploration and discovery process for data of any scale. The structure must also support fast queries and collection of data, where necessary. An appropriate hierarchical structure can fulfill these needs. However, the hierarchical structure must be designed in a way appropriate to visual data mining, which means supporting rapid display and providing the data in the appropriate context. (For example, a query does not just return a piece of data but rather returns that data so it can be displayed in relation to other data.)

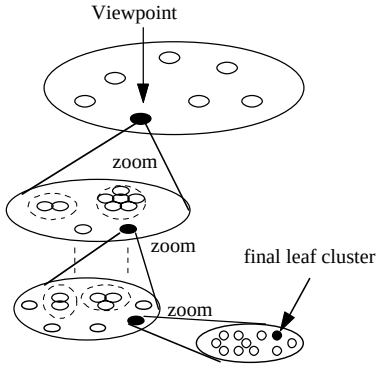
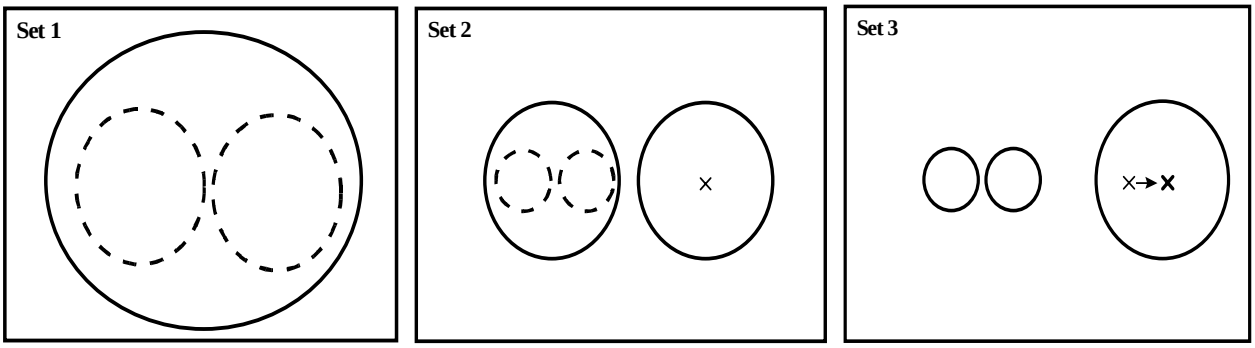


Fig. 6 View-dependent tracking of a cluster

The discovery visualization hierarchy is built (at least in the initial, automatic phase) as a height-balanced feature tree [Gan98], with the clusters defining the features. To obtain an efficiently balanced tree we apply a branching factor B such that each non-leaf node will have at most B children. If a cluster is broken into subclusters during application of the error threshold and this increases the number of children beyond B, a new leaf node is formed. To preserve the clusters as persistent features, the leaf node is formed from the

subclusters of the broken cluster (rather than by reorganizing all the clusters in the previous level.) The initial, automatic clustering builds a tree with a certain predetermined number of leaf clusters. After that deepening of the tree is based on either direct or focus-based user manipulation. This whole process is shown schematically in Fig.6. The hierarchy permits navigable visualizations where the user can zoom in, see detail in context, or back up to gain an overview. We are using a branching factor of B=3, but more study is needed to find which branching factors and automatic hierarchy depths are most efficient. Although these will depend to some extent on the nature of the data, we hope to find general criteria that may be used as guides, as well as specific criteria tied to particular data types.



Figs. 7 Sets resulting from cluster breaking as error is lowered.

In order to provide context, the hierarchy needs capabilities not found in the usual data mining. These include the ability to present for visualization all features at a selected error threshold, either in overview or for a selected region of the data. As the user changes the error threshold or moves the selected region, these features must be quickly updated. Thus the hierarchy is not just a structure for quickly passing attention to the appropriate leaf nodes, as it is in query systems. It also must provide enough information at intermediate levels to carry out feature tracking and visualization. Consider the clustering scenario depicted in Figs. 7. The clustering breaks apart from set 1 to set 2 to set 3 as the error is lowered. The right cluster remains in sets 2 and 3. However, in general the center and other properties of the right cluster may change slightly due to exchange of elements between clusters during the iteration process. The system must pick up all this information during traversal of the hierarchy. This is done by enhancing the information stored at each node. Each node has the list n, n+k, C_n, i, C_{n+i},.... Here n is the beginning cluster set and n+k the ending cluster set for this node (e.g., 2 and 3 for the right cluster in Figs. 7). C_n denotes properties for the cluster including center, error value, number of points contained, maximum dimensions, and average values and rms extents for any variables of interest; and n+i is a set for which the property list C_{n+i} changes (e.g., due to exchange of elements during iteration). Note that there may be only a few such changes. Separately there is a list of all cluster sets containing the number of clusters in the set, total error, and a pointer to the topmost node in the set. When the user chooses an error threshold or target number of clusters, the system finds the appropriate pointer from this list. It then searches only nodes at that level or below and only for ranges of sets {n,n+k} that contain the target cluster set.

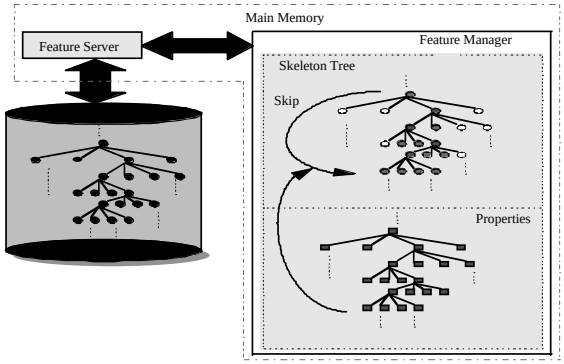


Fig. 8 Paging structure for feature-based hierarchy.

down to a leaf node.

For complete scalability, we also need a paging mechanism. This will be tied to the hierarchy as shown in Fig. 8, which depicts a tree with B=4. The main idea is that only the top structure of the tree plus sections being explored at the moment are in main memory. The rest of the tree plus the data are

out of core. All that is kept in memory is a "skeleton tree" with a sufficient set of linking properties so that the manager can retrieve the next section of the tree or associated data. The manager then watches the user-controlled visual exploration process and decides what should be read in next. Once the appropriate nodes are located, the associated cluster data and sample bin data is read in, which may cause the discarding of older data. We have applied a similar model with good success to the specific case of large scale terrain navigation [Lin99, Dav98], where the terrain data can be 20 GB or more.

6 Conclusions and Future Work

We have shown how our discovery visualization approach can be used for visual data mining. Our approach permits automatic, fast generation of overviews in terms of clusters, which can then be followed as time-dependent features. The clustering can be in terms of spatial distribution or any variable in the data. Users can interactively select the number of clusters or the total cluster error. Furthermore, they can select regions of the data for more detailed clustering and feature tracking.

The discovery visualization results can be given a hierarchical structure that can be generated during the initial automatic phase. The user can then explore this structure with high interactivity by changing error thresholds or choosing cluster regions to dig deeper. In addition the user can deepen the hierarchy by selecting features or regions of the data space.

By applying discovery visualization to both test data and real application data, we show that the clustering methods used are fast and scalable. Further we show that the automatically generated, time-dependent overviews have sufficient accuracy for their task and a straightforward, error-based process for improvement.

The hierarchical structure developed in this paper does not include time-dependence, although there is a mechanism for creating and tracking time-dependent clusters. What is needed is a process that permits the dynamical development over time of the hierarchical structure and the associated list of cluster sets. This will take advantage of the fact that often much of the cluster structure will not change, except for small modifications of cluster center positions and other properties, from time step to time step. More generally, one can treat time as another dimension on the same footing as the spatial dimensions. We will be looking at these issues. In addition, the clustering methods presented here can be extended to non-spatial data. We will look first at data with a limited number of important variables, since mappings to a coordinate space and subsequent visual representations will be easier to handle.

References

Bry93 S. Bryson. Implementing virtual reality. *SIGGRAPH 1993* Course #43 Notes, pp. 1.1.1-1.6.6, 16.1-16.12.

Dav98 Douglass Davis, William Ribarsky, T.Y Jiang, and Nickolas Faust. Intent, Perception, and Out-of-Core Visualization Applied to Terrain. Report GIT-GVU-98-12, pp. 455-458, *IEEE Visualization '98.*

Est95 Martin Ester, Hans-Peter Kriegel, and Xiawei Xu. Knowledge discovery in large spatial databases: Focusing Techniques for Efficient Class Identification. *Advances in Spatial Databases. 4th International Symposium, SSD '95,* pp. 67-82.

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.

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).

Gro94 M. Gross. Subspace Methods for the Visualization of Multidimensional Data Sets. Scientific Visualization, pp. 172-185 (Academic Press, New York, 1994).

Hag94 H. Hagen. Visualization of Large Data Sets. Scientific Visualization, pp. 186-198 (Academic Press, New York, 1994).

Hec98 B. Heckel and N. Hamann. Visualization of Cluster Hierarchies. *Proc. of SPIE*, vol. 3298, pp. 162-171 (1998).

Hof96 T Hofmann, J. Puzicha, and J. Buhmann. "Unsupervised Segmentation of Textured Images by Pairwise Data Clustering. *International Proceedings of IEEE*

International Conference on Image Processing Vol. III, pp. 137-140.

Jea95 Yves Jean, William Ribarsky, Thomas Kindler, Weiming Gu, Gregory Eisenhauer, Karsten Schwan, and Fred Alyea. An Integrated Approach for Steering, Visualization, and Analysis of Atmospheric Simulations. Report GIT-GVU-95-15, *Proceedings IEEE Visualization '95*, pp. 383-387.

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.

Lin99 Peter Lindstrom, 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*.

Ng94 Raymond Ng and Jiawei Han. Efficient and Effective Clustering Methods for Spatial Data Mining. *Proc. of VLDB*, 1994.

Rib99 William Ribarsky, Jochen Katz, Frank Jiang, and Aubrey Holland. Fast Clustering and Feature Tracking for Exploration. Report GIT-GVU-99-21.

Sil95 D. Silver. Object-Oriented Visualization. *IEEE Computer Graphics and Applications*, 15, 3 pp. 54-64.

Tan97 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.

Wal96 T. van Walsum, F. Post, D. Silver, and F. Post. Feature Extraction and Iconic Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2,2 pp. 111-119.

Zha96 T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An Efficient Data Clustering Method for Large Databases. *Proc. SIGmod 96*, pp. 103-114.