Assisted Navigation for Large Information Spaces

Brent M. Dennis and Christopher G. Healey
Department of Computer Science, North Carolina State University

Abstract

This paper presents a new technique for visualizing large, complex collections of data. The *size* and *dimensionality* of these datasets make them challenging to display in an effective manner. The images must show the global structure of spatial relationships within the dataset, yet at the same time accurately represent the local detail of each data element being visualized. We propose combining ideas from information and scientific visualization together with a *navigation assistant*, a software system designed to help users identify and explore areas of interest within their data. The assistant locates data elements of potential importance to the user, clusters them into spatial regions, and builds underlying graph structures to connect the regions and the elements they contain. Graph traversal algorithms, constraint-based viewpoint construction, and intelligent camera planning techniques can then be used to design animated tours of these regions. In this way, the navigation assistant can help users to explore any of the areas of interest within their data. We conclude by demonstrating how our assistant is being used to visualize a multidimensional weather dataset.

CR Categories: G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms;* I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Display algorithms;* I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*

Keywords: camera planning, information visualization, multidimensional visualization, navigation, scientific visualization

1 INTRODUCTION

The analysis of large, complex information spaces is an important problem for researchers from numerous application domains. Advances in technology have allowed the construction of massive data collections in wide-ranging areas like environmental science, network security, e-commerce, and digital libraries. One promising approach is to construct visual representations that allow researchers to identify important properties and make new discoveries within their data. Unfortunately, the *size* and *dimensionality* of large datasets make them challenging to visualize. Techniques specifically designed for these types of datasets are needed to assist users in managing, viewing, and navigating their results [14].

In order to understand the properties of an information space, some formal definitions are presented. A dataset D is logically divided into a finite number of data elements e_i, $D = \{e_1, \ldots, e_n\}$, where n is the size of the dataset. D represents a set of data attributes $A = (A_1, \ldots, A_m)$, where m is the dimensionality of the dataset. Every data element $e_i = (a_{i,1}, \ldots, a_{i,m})$ encodes a value $a_{i,j}$ for every data attribute A_j in D.

Datasets are normally composed of large numbers of elements. As the size grows, visualizing D in its entirety becomes increasingly difficult. A number of novel approaches have been proposed to address this problem, for example, interactive navigation methods from scientific visualization, or hierarchical *focus+context* and *overview+detail* algorithms from information visualization [5, 9, 11, 13]. Although these techniques offer significant improvements, they cannot fully solve the problem of dataset size. Traditional local-detail displays that rely on interactive navigation provide a "window into the world" that hides off-screen information and forces users to maintain their sense of location and direction within the dataset (Fig. 1). Hierarchical displays can still be overwhelmed by datasets with a sufficiently large *n*.

The issue of dataset dimensionality further complicates the problem of display and analysis. Each additional data attribute to visualize produces increasingly complex images. Techniques are needed to ensure that the resulting displays support a viewer's exploration and analysis needs. Different methods have been developed to manage dimensionality, including data simplification, multidimensional glyphs, and perceptually controlled visualizations [2, 6, 7, 8, 15]. Although these techniques allow multiple data attributes to be shown in a single image, they do not support an arbitrarily large m (in fact, the visual system itself imposes a hard limit on the total amount of information it can process in a fixed period of time). As well, most multidimensional visualization algorithms ignore the issue of dataset size, assuming implicitly that some method exists to navigate the dataset, or to view it as a single on-screen image.

Multidimensional visualizations provide users with coherent representations of high-dimensional datasets. *overview+detail* and *focus+context* techniques help users study both the global structure and the local detail of their data. Unfortunately, neither method alone provides a complete solution to the problem of displaying multidimensional local detail *and* areas of potential interest in the global structure of the dataset. We hope to combine techniques from scientific and information visualization with a *navigation assistant*, a software system that allows users to identify, locate, track, and explore regions of interest within their data. A navigation assistant can help users identify "interesting" data, then structure those elements into global spatial patterns that highlight the locations of and relationships between different regions of interest. The assistant can also help users move to a region of interest and visit the elements that make up that region. This allows users to focus on exploring their data by reducing the burden of deciding where to search and what to look at.

All of the navigation operations remain under user control, guaranteeing full interactivity during the visualization session. A brief overview of how the navigation assistant operates is as follows:

1. The user specifies how to identify individual data elements of interest (EOIs).
2. The EOIs are spatially clustered into areas of interest (AOIs).
3. A graph is constructed to connect the EOIs within each AOI; this graph serves as a framework for navigating the AOI.
4. The AOIs are themselves connected into a high-level global network; this network allows navigation between different AOIs.
5. Multidimensional visualizations from the user's current viewpoint are combined with an inset display of the AOI network;

this offers a high level-of-detail local view together with the structure of areas of potential interest within the entire dataset.

6. Optimal view construction and camera planning techniques are used together with graph traversal algorithms to build tours of the areas of interest within the dataset.

The remainder of this paper describes the details of our navigation assistant, and presents results from its use in a practical visualization environment. Section 2 provides background information on the scientific and information visualization algorithms related to our work. Section 3 discusses how the EOIs are identified and segmented to form AOIs. It also explains how the local AOI graphs and global AOI network are built. In section 4, we describe the view construction and camera planning algorithms we apply to navigate the AOIs. Section 5 shows how our navigation assistant was used to explore an environmental dataset of North America. Finally, we present conclusions and future work in Section 6.

2 RELATED WORK

Our discussion of related work concentrates on two important topics: techniques from information visualization for displaying complex information spaces, and techniques from scientific visualization for displaying multidimensional data.

Card, Mackinlay, and Shneiderman define information visualization as "the general application of assembling data objects into pictures, revealing hidden patterns" [3]. Two techniques from this field are closely related to our goal of visualizing large, complex information spaces: *overview+detail* and *focus+context*. *overview+detail* methods present a global overview of an information space, together with ways to request increased levels of detail for subregions within the space. *focus+context* techniques display the global context of an information space, together with ways to interactively focus on a full-detail representation of specific locations in the space.

Different algorithms use different methods to represent the global structure and local detail within their displays. For example, the treemap [13] decomposes a dataset D into a rectangular image whose individual regions are hierarchically partitioned based on different properties (or attributes) of the data within D. Later revisions to the treemap allow users to select individual regions; this expands the region to fill the screen and show a higher level of detail, but at the expense of maintaining a view of the region's location and context within D. The fisheye lens [5] presents a low level-of-detail display of the entire dataset, together with an interactive lens that "zooms in" about its center, providing a higher level-of-detail display of the data directly beneath the lens. The hyperbolic tree [9] structures information in a dataset as a tree embedded in the surface of a sphere. A portion of the sphere facing outward uses hyperbolic geometric to form a lens, zooming the information being displayed as the sphere is rotated. A cone tree [11] visualizes a hierarchical information space as a tree of semi-transparent 3D cones, one for each category in the hierarchy. Elements within a category are located around the base of the appropriate cone.

overview+detail and *focus+context* techniques offer significant advantages for the visualization of large information spaces. In spite of this, we believe further improvements could increase our ability to manage the size and dimensionality that exist in many visualization domains. Potential problems with existing algorithms include:

1. Few techniques address the issue of visualizing multidimensional data elements; those that do (*e.g.,* treemaps) produce displays that may not be well-suited for the user's exploration and analysis needs.

2. Many techniques are not appropriate for spatial datasets that require specific locations for their elements (*i.e.,* the data cannot be arbitrarily repositioned to fit spatial structures required by the visualization algorithm).

3. Most *overview+detail* and *focus+context* techniques are still sensitive to dataset size; a sufficient increase in n can degrade their ability to display data in a coherent manner.

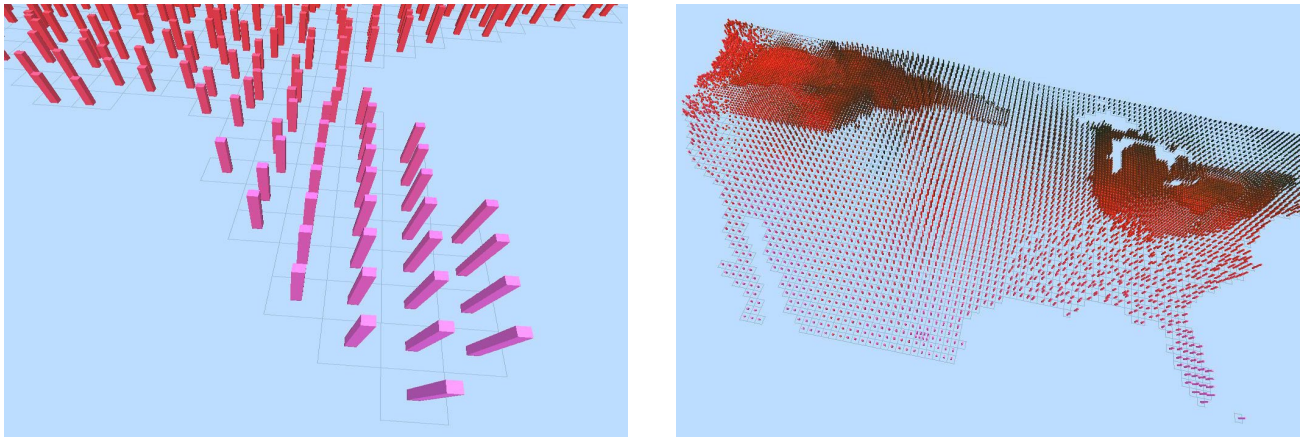
Certain methods from scientific visualization were designed to address the first two issues noted above: visualizing multidimensional data elements that are anchored at fixed spatial locations. One common technique is to use properties of different visual features like color, texture, and motion to represent the different attributes embedded in a dataset [2, 6, 15]. The resulting displays form visual patterns that are used to explore the underlying data (Figs. 1, 5a). More recent work has studied the perceptual properties of the different visual features [7, 8], in an attempt to produce displays where important information can be seen "at a glance." These *perception-based visualizations* are carefully designed to harness the strengths and avoid the limitations of the human visual system. This creates visualizations that can be analyzed very rapidly and accurately, often in a few tenths of a second or less. Unlike *overview+detail* and *focus+context* algorithms, these multidimensional techniques do not deal explicitly with the need to display both local detail and global structure simultaneously. In most cases, interactive camera navigation (*e.g.,* translation, rotation, and zoom) is used to change the viewer's location and focus on different subregions in the dataset.

Taken together, ideas from information and scientific visualization could be used to construct a system capable of representing both global structure and high-dimensional local detail. To address the final issue, datasets with very large n, we propose a navigation assistant, a software system designed to help users navigate within their data. Rather than trying to display the entire dataset D on-screen, we provide two separate views: a high level-of-detail local view of a subset of D, and a global overview showing a connected network of regions of interest within D. The navigation assistant constructs the global overview based on user-specified rules that identify "properties of interest." The global network allows users to situate themselves relative to regions of interest, while at the same time visualizing the multidimensional data elements within their field-of-view. As well, the graphs that make up the regions of interest can be used to construct animated tours to help viewers explore the areas of D most likely to contain important data.

3 BUILDING AREAS OF INTEREST

Every dataset contains a subset of data elements that the user considers "interesting." Users need methods to successfully locate these elements of interest (EOIs), then efficiently navigate between them. Typically, users can describe EOIs based on their attribute values. We allow users to create a set of explicit rules using standard mathematical and boolean operators. These rules are then applied to filter a dataset and identify its EOIs For example, during the visualization of a weather dataset, a user might enter a rule of the form *temperature* $>$ *avg*(*temperature*) + 2 $*$ *stdev*(*temperature*) (where *avg* represents the average temperature over all data elements, and *stdev* represents the standard deviation). If an element satisfies any rule, the navigation assistant considers it to be of potential interest to the user (Fig. 2a).

We assume the set of EOIs will be small relative to the size of the dataset. A large EOI set suggests that the user's rules are too broad, and thus do a poor job of filtering the data into interesting and uninteresting subsets. Although allowed, such a set of rules normally make the exploration task more difficult and time consuming. Our



(a) (b)

Figure 1: An example visualization of a weather dataset with $m = 4$ attributes $A = ($ *temperature, wind speed, cloud coverage, precipitation* $)$ represented with color (dark blues and greens for colder to bright reds and pinks for warmer), height (taller for stronger winds), density (denser for higher cloud coverage), and regularity (more irregular for heavier precipitation): (a) a close-up of Florida shows the details of individual data elements, but no global context; (b) a view of the United States shows interesting global structure, but makes it difficult to identify the attribute values assigned to individual data elements

technique for identifying EOIs works well, both for static and for dynamically changing datasets. Users can easily add, remove, or modify their rules to update their current interests during visualization. This allows them to quickly refocus as new or unexpected avenues of investigation unfold.

Once an initial set of EOIs has been identified, a navigation framework must be constructed. This process begins by clustering spatially neighbouring EOIs into areas of interest, or AOIs (Fig. 2b). The clustering algorithm works as follows:

- An element of interest e_i is selected to begin a new AOI.
- From the set of EOIs that do not yet belong to any cluster, e_j is selected such that it is closest to the convex hull of the new AOI.
- If the distance from e_j to the convex hull is below a user-defined threshold, e_j is added to the new AOI. Otherwise, the new AOI is closed and e_j becomes the initial member of the next AOI to be constructed.
- This process is repeated until every EOI belongs to an AOI.

The number and size of the AOIs that are formed can provide important insight into the nature of the elements of interest. A few large AOIs suggests that interesting elements are spatially coherent with one another, while a large number of very small AOIs suggests the interesting elements are randomly located within the dataset. A spatial partitioning of the dataset into areas with many AOIs, or with no AOIs, might also indicate important spatial dependencies inherent to the EOIs.

There are several parameters that the user can specify during clustering to control the structure of the AOIs. Specifically:

- *proximity:* the maximum allowable distance between a candidate EOI e_j and the convex hull of its AOI. This is used to tradeoff the physical size and density of each individual AOI against the total number of AOIs in the dataset.
- *area:* the maximum spatial area an AOI is allowed to occupy. This is used to control the physical size of the AOIs.
- *population:* the maximum number of EOIs that are allowed within a single AOI. This is used to control the logical size of the AOIs.

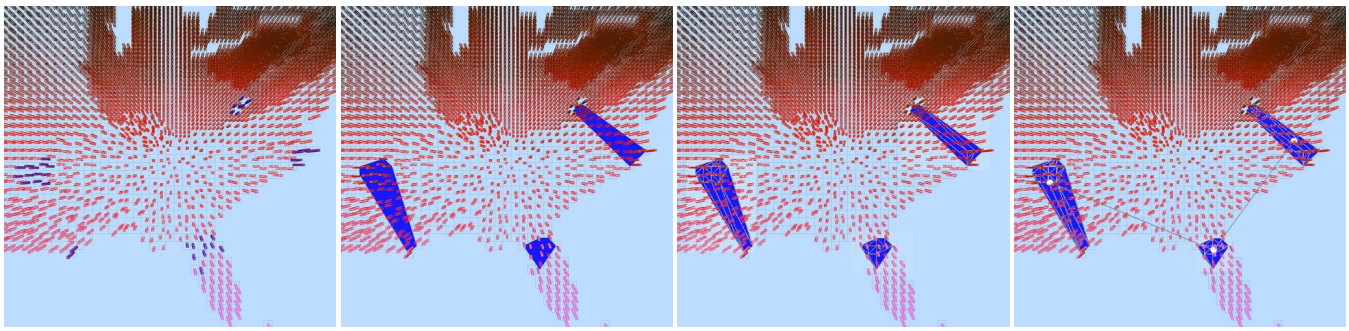
Although our clustering technique has worked well in practice, we are now studying more traditional hierarchical clustering algorithms (*e.g.*, agglomerative methods with various distance metrics) that may improve efficiency, flexibility, and robustness.

Once the AOIs are defined, a graph-based framework is built within each AOI. This framework is used to support efficient navigation, and to visualize the locations of and relationships between the EOIs that make up an AOI. We use a Delaunay triangulation of the EOI positions to achieve these goals (Fig. 2c). The maximum number of edges in a Delaunay triangulation is linear in the number of vertices ($3n - 6$ for n vertices), preventing dramatic increases in the graph size as an AOI grows. Given no more than $3n - 6$ edges, the average degree of each vertex is at most six. This guarantees a low-complexity branching factor that is independent of the number of vertices in the graph. Finally, the edge count and branching properties, together with the planar nature of a Delaunay triangulation, allow for the construction of polynomial time graph-traversal algorithms.

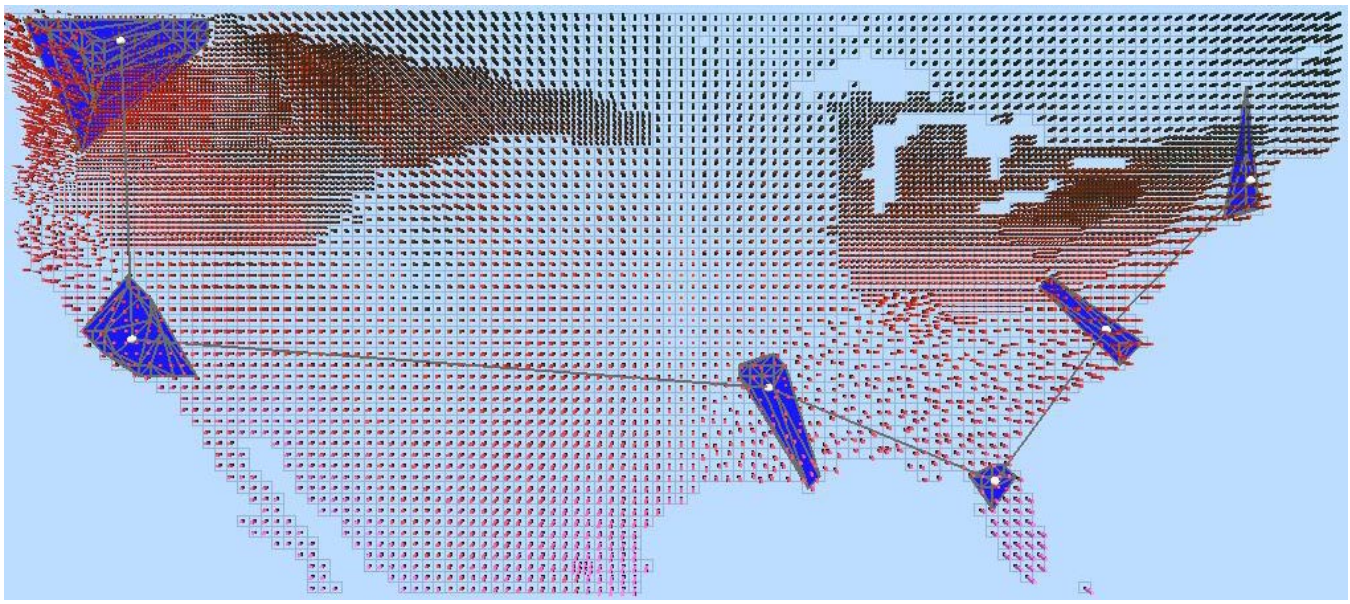
The last step in our AOI construction involves linking the AOIs together. The centroids of each AOI are connected via a minimum spanning tree built with edge weights set to edge lengths (*i.e.*, the Euclidean distances between the AOIs). This produces a low-complexity graph that connects the AOIs (Fig. 2d). Since the graph minimizes the sum of its edge weights, it guarantees efficient paths between the AOIs. This acts as our global network, visually representing both the locations of and connectivity between the AOIs. Just as the Delaunay triangulations are used to navigate locally within an AOI, the minimum spanning tree is used to navigate globally from one AOI to another.

4 ASSISTED NAVIGATION

With the graph framework in place, users have the ability to manage their visualization sessions, both by locating areas of potential



(a) (b) (c) (d)



(e)

Figure 2: Areas of interest constructed in an example weather dataset: (a) individual elements of interest (shown as shaded solids with highlighted borders) are selected via user specified rules; (b) the elements are spatially clustered into areas of interest; (c) elements within each area are connected with a Delaunay triangulation; (d) areas of interest are connected with a minimum spanning tree; (e) a global view of six areas of interest within the continental United States

interest, and by asking the navigation assistant to help them to explore within those areas For example, a user might ask the assistant to move to a particular AOI and present a tour that visualizes its EOIs. Another example might move a user from the closest EOI to the furthest EOI within the same AOI, and visualize any intermediate EOIs the camera passes close to during the traversal. The assistant handles these requests by constructing an animated camera path based an AOI's underlying Delaunay triangulation. The camera path visualizes the elements of interest as follows:

1. Use graph traversal algorithms to generate the sequence of EOIs to visit, in order.
2. Use an optimal viewpoint algorithm to generate the camera coordinates from which to visualize each EOI in the sequence.
3. Generate camera motion parameters to animate smoothly between the EOI view positions.
4. Begin the tour, allowing the user to stop at any point and reacquire control of the camera.

4.1 Graph Traversal

Each navigation request applies a graph traversal algorithm to an AOI's Delaunay triangulation to generate a sequence of EOIs to visualize. Two different algorithms are currently being used: a shortest path technique to move between two EOIs, and an approximation of a Hamiltonian cycle to tour within an AOI. The properties of the Delaunay triangulation support efficient traversal algorithms. A shortest path can be computed in time $O(E)$, where E is the number of edges in the graph. An approximation of a Hamiltonian cycle can be computed in $O(E\,lgE)$.

Dijkstra's algorithm is used to construct a shortest path from a starting EOI e_0 to a target e_j. A brief description of the algorithm is provided here; interested readers are directed to [10] for a more complete explanation and formal proof. The algorithm begins by identifying edges of the form (e_0, e_i) to define the set of elements connected to e_0. Each e_i has its path length set to the weight of the edge (e_0, e_i), and its predecessor set to e_0. These paths are incrementally extended from each e_i; every new element encountered

has its length and predecessor set in a similar fashion. This process continues until all EOIs are visited. At this point, the predecessors define the shortest path between e_0 and any target EOI e_j.

In order to tour the EOIs in an AOI, we need to construct a spanning cycle, a path that starts and ends at a common element e_0 and visits every $e_i \neq e_0$ exactly once. Finding a Hamiltonian cycle, a minimum cost spanning cycle (where cost is the sum of the edge weights in the cycle), is NP hard. We instead use the algorithm of Rosenkrantz et al. [12] to build an approximation of the Hamiltonian cycle for a Delaunay triangulation. Given a minimum spanning tree T of a Delaunay triangulation (computed in $O(E\,lgE)$ time using Kruskal's algorithm [10]), we create T' by doubling every edge in T. Since T' is a connected Euler graph, an Euler cycle w can be constructed to visit each e_i at least once. It can be shown that the cost of w is no more than two times the cost of the Hamiltonian cycle. To avoid visiting elements more than once, we assign an arbitrary direction to w and begin a tour at e_0, marking elements as visited when they are first seen. Any element along w that was already visited is skipped. Since w is built from a Delaunay triangulation, all edge weights must obey the triangle inequality, so skipping edges will never increase the cost of the cycle. Therefore, the final tour must also have a cost of no more than twice the Hamiltonian cycle.

4.2 Optimal Views

Once the sequence of EOIs is known, an appropriate camera coordinate (*i.e.*, position and viewing direction) must be selected for each EOI. One simple choice would be to ask the user to define a preferred camera position, a default location in three-space centered about each element to be visualized. Unfortunately, in certain cases an element may not be visible from its preferred camera position. This is especially true when elements are visualized using 3D glyphs that vary in height. In these situations, a different technique must be applied to guarantee an "optimal" camera position that is not occluded by other elements in the scene.

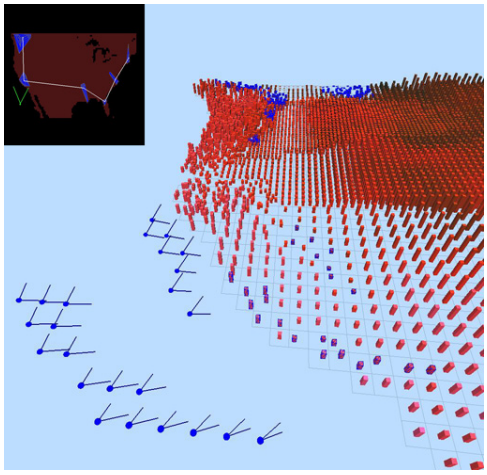


Figure 3: An example of optimal viewpoints, shown as locations in three-space together with the camera's look-at and up vectors at each location, for 24 EOIs in an AOI in California

We use a partial constraint solver to choose each element's camera coordinates. Our solver is loosely based on the work of Drucker and Zeltzer [4] and Bares and Lester [1]. Three separate constraints are specified to control the visualization of an element:

- *occlusion:* the ability to "see" the EOI from the camera's view position,
- *view angle:* the preferred range of relative orientations between the camera and the EOI, and
- *view distance:* the preferred range of distances between the camera and the EOI.

A spherical coordinate system centered about the EOI is used to identify regions in space that satisfy each constraint. For view angle, this is a simple decomposition of the upper hemisphere into allowable and restricted areas. For occlusion, a camera is placed at the EOI's location, and visible object boundaries are projected onto the hemisphere around the element. The hemispheres are then unioned together, to search for regions that satisfy both constraints. If such regions exist, an optimal camera coordinate within one of the regions is selected (the optimal position is chosen based on a number of criteria, for example, it must lie in the preferred range of distances from the EOI, it should minimize variations between the camera coordinates of neighbouring EOIs in the tour, and so on).

If no non-empty regions exist after intersection, we must relax our constraints to find an acceptable camera coordinate. Currently we always choose to relax the view angle constraint, since we deem this less important that the occlusion constraint (*i.e.*, we prefer to be able to see an element, rather than guaranteeing the camera never moves outside a fixed range of viewing angles). A camera coordinate is selected from the first acceptable region we find, with the additional requirement that this coordinate minimize the amount of constraint relaxation it requires.

Our constraint-based solver works well in practice, and has proven to be reasonably efficient, allowing us to recompute camera coordinates in real-time for our dynamic visualization environments. Because it is easy to add, remove, or change the constraints our system uses, we have the added advantage of a simple, flexible method to control how optimal views are constructed.

4.3 Camera Motion

After the navigation assistant constructs an ordered set of camera coordinates for each EOI, it must build a camera path through these locations. Spline curves are used to move between the EOIs's optimal viewpoints. The camera's view direction is defined by its position on a segment, and by the segment's two endpoints. The spline curve S_i with endpoints e_i and e_{i+1} is parameterized such that the camera looks at e_i on the range $[0, \frac{1}{3})$, and at e_{i+1} on the range $[\frac{2}{3}, 1]$. The camera pans smoothly between e_i and e_{i+1} on the range $[\frac{1}{3}, \frac{2}{3})$.

As the camera path for each curve is built, the assistant tries to guarantee shots that are fluid and free from distortion. Previous work has investigated techniques for visually coherent camera motion [1, 4]. Although we have found no fully automated or complete set of rules to guarantee perfect camera placement, these studies suggest a number of common guidelines, including: (1) maintaining alignment between the camera and the world up vector; (2) focusing the camera's view on the subject of a shot, and (3) constructing camera motion that does not collide with objects in the scene. Each of these issues was addressed when we built our camera paths.

For a camera path along S_i, if the angle between the camera up vector and the world up vector becomes too large, additional control points are inserted to force the camera to maintain a consistent orientation. This prevents intermediate shots that look directly down onto the data elements. The allowable difference between the two vectors is a function of the camera's position along S_i and the optimal views defined at S_i's endpoints. The camera must conform to the coordinates specified at the endpoints, regardless of how poorly the up vectors diverge. Because of this, the up vector requirement is also considered during optimal view selection; when a choice of view angles is available, views with a lower angle are preferred over higher, look-down views.

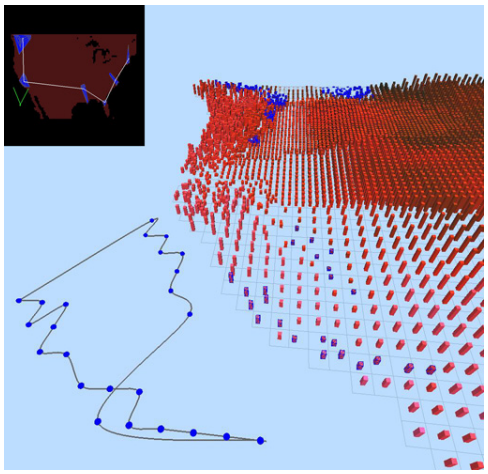


Figure 4: An example of a camera path through a set of camera coordinates for 24 EOIs in an AOI in California

Each camera path is built to focus on an EOI e_i for $\frac{1}{3}t_{i-1} + \frac{1}{3}t_i$, where t_i is the time needed to traverse a curve segment S_i ($\frac{1}{3}t_{i-1}$ as the camera approaches e_i, and $\frac{1}{3}t_i$ as it moves away). Although e_i may not be fully visible for this entire period, the existence of an optimal view guarantees the user will be able to see e_i as the camera passes near it. Any occlusion that might exist will smoothly disappear (or reappear) as the camera moves through its tour.

We check for collision with elements along the camera path for each curve segment when it is built. Any collision forces us to raise the height of the camera at that point along the path to avoid the element in question. We identify all potential collisions prior to building any part of the path. This allows us to construct a path that rises monotonically to its highest point, then descends back to e_{i+1} as necessary. Although this may violate the user's preferred viewing angle, the assistant treats this requirement as less important than the need to avoid passing through elements during a tour.

4.4 Moving Between Areas of Interest

Just as users want to tour within an AOI, they may also ask to move between AOIs. We can apply exactly the same algorithms to address these navigation requests.

The simplest way to move a user to a target area of interest a_j is via a straight-line camera translation to the center of a_j. We do, in fact, support this type of traversal. However, we have found an equally useful request takes the form: "Move me to a_j, but show me all the other AOIs that are nearby during that move." We use the global AOI network to build this type of tour as follows:

1. Locate the center of the AOI a_0 closest to the user's current view position v_{cur}.
2. Identify the shortest path through the minimum spanning tree from a_0 to a_j using Dijkstra's algorithm.
3. Construct optimal views for the center of each AOI along the path $(v_{cur}, a_0, \ldots, a_j)$ using our optimal view algorithm.
4. Build camera paths through the optimal views that focus on each AOI in turn using our camera planning algorithms.
5. Execute the tour by animating the camera along the camera paths.

The tour we produce moves the user from v_{cur} to a_j, while at the same time focusing on AOIs along the navigation path.

5 VISUALIZING WEATHER DATA

In order to test our navigation assistant in a real-world context, we turned to a collection of monthly environmental and weather conditions. This dataset contains mean monthly surface climate readings in $\frac{1}{2}^\circ$ latitude and longitude steps for the years 1961 to 1990 (*e.g.*, readings for January averaged over the years 1961-1990, readings for February averaged over 1961-1990, and so on). We chose to visualize four weather conditions: mean temperature *(temp)*, wind speed *(wind)*, cloud coverage *(cloud)*, and precipitation *(precip)*. Each data element e_i contains $m = 7$ values: the four attributes listed above, plus latitude, longitude, and month.

We used three-dimensional perceptual texture elements (or pexels) to visualize data in our high-detail local view. Our pexels look like rectangular solids sitting up off the surface of an underlying map (Fig. 5a). Colour represents *temp:* dark blues and greens for low temperatures to bright reds and pinks for high temperatures. Height represents *wind:* taller pexels for stronger winds. Density represents *cloud:* higher density (*i.e.*, more pexels packed into a unit area of screen space) for denser cloud coverage. Finally, regularity represents *precip:* in areas with little or no rainfall, pexels are positioned in a regular, grid-like fashion; in areas with high rainfall, pexels are allowed to walk randomly from these anchor points.

Fig. 5 shows the dataset with six AOIs built using the following rules (the AOIs can be seen in the global overview in Fig. 5b; see also Fig. 2e):

- *precip* $>$ *avg*(*precip*) $+$ *stdev*(*precip*) && *cloud* $<$ *avg*(*cloud*) $-$ $\frac{1}{2}$ $*$ *stdev*(*cloud*)
- *precip* $>$ *avg*(*precip*) $+$ *stdev*(*precip*) && *wind* $>$ *avg*(*wind*) $+$ *stdev*(*wind*)
- *precip* $>$ *avg*(*precip*) $+$ *stdev*(*precip*) && *temp* $<$ *avg*(*temp*)

These rules focus on areas of high precipitation that correlate with secondary weather conditions that are interesting (*e.g.*, high wind speeds) or unexpected (*e.g.*, low cloud coverage). Intuitively, the three rules correspond to regions of: (1) high rainfall and low cloud coverage; (2) high rainfall and high winds, and (3) high rainfall and low temperatures.

Fig. 5b visualizes data elements at the start of a tour through an AOI located in the Pacific Northwest (the EOIs were selected because of high *precip* and low *cloud*, and high *precip* and high *wind*). This camera shot focuses on EOIs near the Washington-British Columbia border. The user has asked to highlight the EOIs with blue borders, and to display the camera path as a grey curve in the local view. The global overview in the upper-left corner of the screen visualizes the locations of all the AOIs in the dataset. It also displays the user's current position and view direction, shown as a green view frustum. The view frustum allows users to determine their location and orientation. This is particularly helpful when the navigation assistant is animating the camera. Ten additional camera views from the tour are shown in sequence in Figs. 5c-l. Each view focuses on a new element of interest in the AOI.

Although not shown in this example, users can also request tours between EOIs within an AOI, or between the AOIs themselves. These tours are similar to the one shown in Fig. 5: a path through a set of optimal views is constructed, allowing the assistant to navigate smoothly from the user's current position to the final EOI or AOI, viewing each area of interest as it pass near the camera.

6 CONCLUSIONS

This paper describes a new method for visualizing large, multidimensional datasets. Our technique combines the local details of

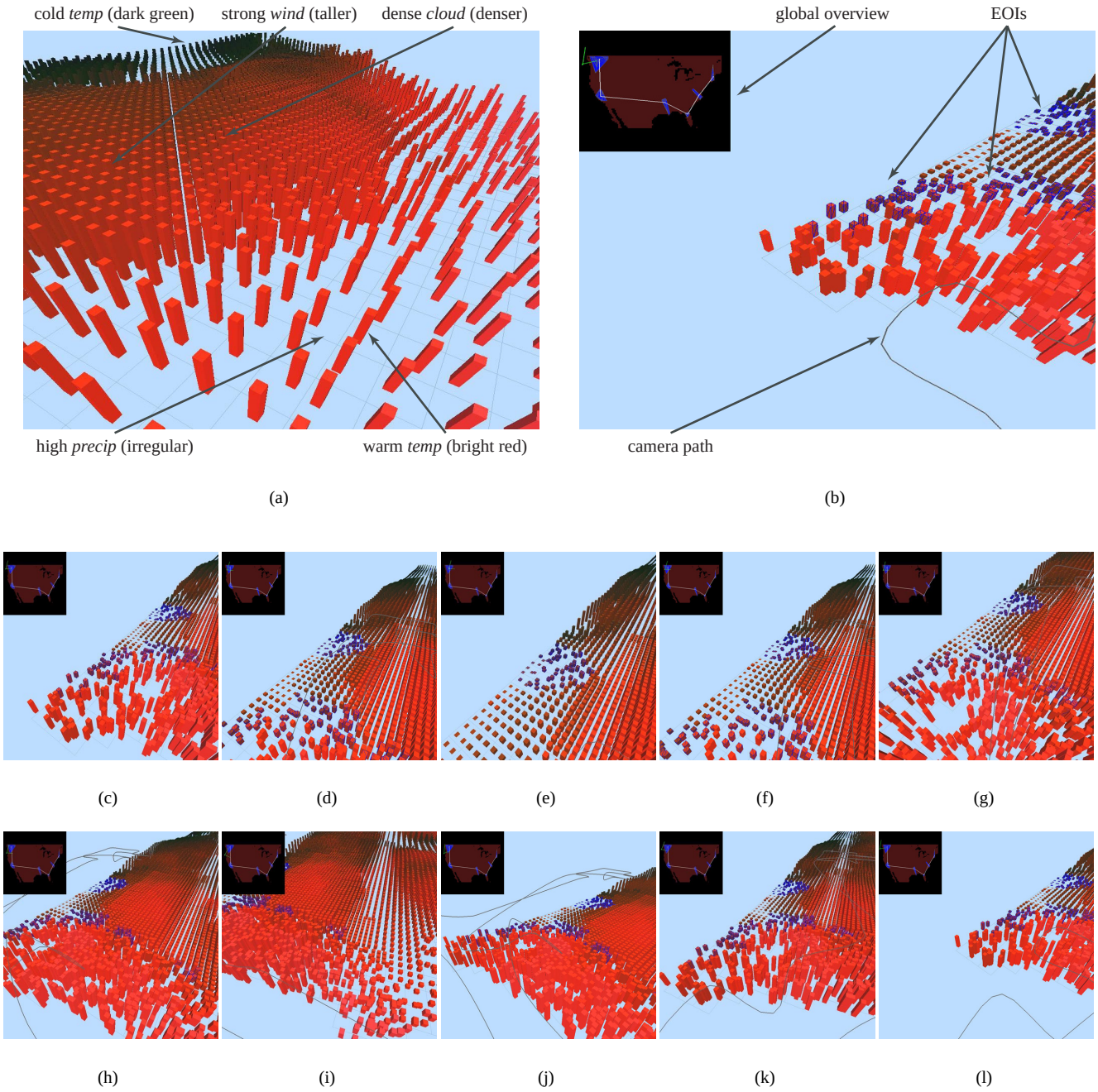


Figure 5: Visualizations and a tour from the weather dataset: (a) visualizing data in the northeastern United States to demonstrate different weather conditions and the visual features they form; (b) the start of a tour of an AOI in the Pacific Northwest, showing EOIs highlighted with blue borders, the camera path rendered as a grey curve, and the global overview in the upper-left corner with the user's current location and view direction displayed as a green arc; (c-l) an ordered sequence of ten camera shots along the tour

individual data elements together with a global overview of the locations and structure of areas of potential interest within the dataset. A navigation assistant identifies these areas of interest (AOIs), then offers users different ways to explore them. Graphs based on Delaunay triangulations are used to connect elements within an AOI. A minimum spanning tree is built to link the AOIs together. Visual tours can then be constructed with graph traversal algorithms, optimal viewpoint construction, and intelligent camera planning.

Data is visualized using a combination of assisted navigation and traditional interactive camera placement. Since the AOI graphs can be dynamically recomputed, the rules that define elements of interest can be easily changed. This allows users to try various approaches to search for unexpected results in their data, and to study the relationships that exist between the attribute values stored in each data element. We showed how our navigation assistant is being used to visualize multidimensional weather data. We are not limited to this particular problem domain, however. The navigation assistant was designed to be application independent, and can be applied to a wide range of practical visualization environments.

Although we have not completed controlled validation experiments to measure the performance of our system against existing visualization techniques, anecdotal feedback from our users has highlighted a number of potential advantages and limitations. In particular:

+ The global overview is very useful, both for navigating within the data, and for rapidly identifying the locations of EOIs in the dataset as a whole.

+ The ability to add, remove, and modify rules of interest allows "what if" analyses to be performed by studying how different rules generate different EOIs in the global overview.

+ The navigation assistant can be used to quickly relocate to an off-screen AOI, and to tour EOIs within an AOI.

– Explicitly specifying the rules of interest can be time consuming, moreover, it is sometimes difficult to define rules that capture exactly the combinations of attributes that identify a particular set of EOIs.

– More choices are needed in the types of tours the navigation assistant provides.

We are pursuing a number of new ideas to expand and improve our navigation framework. One project is studying the use of *implicit indicators* to identify elements of interest. This will allow users to select EOIs directly from the high-detail local view. Learn-by-example algorithms (*e.g.*, data mining classification) can then be applied to build rules that define the specific combinations of attribute values that make these elements different from others in the dataset. Users will be able to modify the implicit rules exactly as before, allowing them to manage any suggestions made by the navigation assistant. A second project is focusing on new types of tours that may provide additional insight into the makeup of a dataset. One example is a tour that follows a path around the boundary of an AOI while looking in on the EOIs; this could help to better define the shape of the AOI and the relative positions of the EOIs. We are studying guidelines from computer animation and virtual cinematography to construct tours of this type. Finally, we are designing a set of formal validation studies that compare the performance of our navigation assistant to existing visualization techniques. Our plan is to implement several *focus+context* and *overview+detail* algorithms, extend them to support multidimensional glyphs, and then measure their performance for a set of representative visualization tasks. The navigation assistant will be tested on the same set of tasks. The results will be used to identify the strengths and limitations of each technique.

In summary, our navigation assistant combines ideas from scientific and information visualization, graph theory, and camera planning to support effective exploration and analysis of large, complex, multidimensional datasets. The result is a technique that offers a number of unique and useful improvements over existing visualization systems, for a wide range of problem domains.

References

[1] BARES, W. H., AND LESTER, J. C. Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In *Proceedings Intelligent User Interfaces '99* (Redondo Beach, California, 1999), pp. 119–126.

[2] BERGMAN, L. D., ROGOWITZ, B. E., AND TREINISH, L. A. A rule-based tool for assisting colormap selection. In *Proceedings Visualization '95* (Atlanta, Georgia, 1995), pp. 118–125.

[3] CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1999.

[4] DRUCKER, S. M., AND ZELTZER, D. Intelligent camera control in a virtual environment. In *Proceedings Graphics Interface '94* (Banff, Canada, 1994), pp. 190–199.

[5] FURNAS, G. W. Generalized fisheye views. In *Proceedings CHI '86* (Boston, Massachusetts, 1986), pp. 16–34.

[6] GRINSTEIN, G., PICKETT, R., AND WILLIAMS, M. EXVIS: An exploratory data visualization environment. In *Proceedings Graphics Interface '89* (London, Canada, 1989), pp. 254–261.

[7] HEALEY, C. G., AND ENNS, J. T. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 145–167.

[8] HEALEY, C. G., ST. AMANT, R., AND CHANG, J. Assisted visualization of e-commerce auction agents. In *Proceedings Graphics Interface 2001* (Ottawa, Canada, 2001), pp. 201–208.

[9] LAMPING, J., AND RAO, R. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing 7*, 1 (1996), 33–55.

[10] PREPARATA, F. P., AND SHAMOS, M. I. *Computational Geometry*. Springer-Verlag, New York, New York, 1985.

[11] ROBERTSON, G. G., MACKINLAY, J. D., AND CARD, S. K. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings CHI '91* (New Orleans, Louisiana, 1991), pp. 189–194.

[12] ROSENKRANTZ, D. J., STEARNS, R. E., AND LEWIS, P. M. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing 6* (1977), 563–581.

[13] SHNEIDERMAN, B. Tree visualization with tree-maps: A 2D space filling approach. *Transactions on Graphics 11*, 1 (1992), 92–99.

[14] SMITH, P. H., AND VAN ROSENDALE, J. Data and visualization corridors report on the 1998 CVD workshop series (sponsored by DOE and NSF). Tech. Rep. CACR-164, Center for Advanced Computing Research, California Institute of Technology, 1998.

[15] WARE, C., AND KNIGHT, W. Using visual texture for information display. *ACM Transactions on Graphics 14*, 1 (1995), 3–20.