Interest Driven Navigation in Visualization

Christopher G. Healey, *Senior Member*, *IEEE*, and Brent M. Dennis

Abstract—This paper describes a new method to explore and discover within a large data set. We apply techniques from preference elicitation to automatically identify data elements that are of potential interest to the viewer. These "elements of interest (EOI)" are bundled into spatially local clusters, and connected together to form a graph. The graph is used to build camera paths that allow viewers to "tour" areas of interest (AOI) within their data. It is also visualized to provide wayfinding cues. Our preference model uses Bayesian classification to tag elements in a data set as *interesting* or *not interesting* to the viewer. The model responds in real time, updating the elements of interest based on a viewer's actions. This allows us to track a viewer's interests as they change during exploration and analysis. Viewers can also interact directly with interest rules the preference model defines. We demonstrate our theoretical results by visualizing historical climatology data collected at locations throughout the world.

Index Terms—Bayesian network, classification, navigation, preferences, visualization.

1 Introduction

Scientific and information visualization convert large collections of strings and numbers into visual representations that allow users to discover patterns within their data. The focus of this paper is the visualization of large data sets containing n data elements and m data attributes. The size of these data sets normally exceeds the available screen resources, forcing much of the data set to lie offscreen. This leads to an important question for a data analyst: "How can I locate interesting data when most of the data is outside my current view?"

Various methods have been proposed to address this problem. Our particular solution applies an artificial intelligence technique known as *preference elicitation*. How can we order a person's preferences across a set of items? Techniques like Bayesian classification can be used to learn the person's preferences, both known and hidden [1]. We apply these theories to build rules that classify a subset of a data set's elements as "interesting" to the viewer. The elements are visualized as multidimensional glyphs, and presented using animated tours that focus on clusters of interesting elements within the data set.

Techniques for visualizing multidimensional elements and presenting them with camera animations have been discussed in previous work [2], [3]. Our focus in this paper is on how to construct rules that classify elements as interesting or not interesting, without requiring the viewer to explicitly describe these rules. The ability to automatically identify elements of interest offers a number of important advantages.

- Explicitly defining rules of interest is time consuming, particularly if these rules need to be updated every time a viewer's interests change.
- It may be difficult for a viewer to formulate an exact definition to describe why an element is interesting.
- A viewer may not know a priori what they will find interesting.

To our knowledge, this is the first attempt to automatically define and update in real-time data properties that are of interest to a viewer during visualization. Our contributions in this paper are as follows:

- A description of the area of preference elicitation and its relevance to data visualization.
- A Bayesian classifier capable of constructing user models to tag data elements as interesting or not interesting.
- A description of how a viewer's actions during visualization can be used as implicit cues to track and update the viewer's interests.
- A demonstration of integrating a preference-driven interest model into an existing framework for navigating and visualizing large, multidimensional data sets.

Although our goal is to automatically classify data elements for navigation, we believe this research has broad appeal for visualization environments that can benefit from understanding a viewer's interests. Our technique is applicable to any data set where combinations of attribute values can be used to determine a viewer's level of interest. It is designed to function efficiently in the presence of large numbers of multidimensional data elements, and will automatically adjust its user model in real time to track viewers' changing interests during their exploration and analysis.

2 Related Work

Information and scientific visualization algorithms have been proposed to visualize large, multidimensional information spaces. Artificial intelligence, information retrieval, and user interface approaches have been suggested to

- *C.G. Healey is with the Department of Computer Science, North Carolina State University, 890 Oval Drive #8206, Raleigh, NC 27695-8206. E-mail: healey@ncsu.edu.*
- *B.M. Dennis is with the Lincoln Laboratory, Massachusetts Institute of Technology, 244 Wood Street, Lexington, MA 02420-9108. E-mail: brent.dennis@gmail.com.*

Manuscript received 7 Jan. 2011; revised 3 Dec. 2011; accepted 22 Dec. 2011; published online 16 Jan. 2012.
Recommended for acceptance by H. Hauser.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-01-0006.
Digital Object Identifier no. 10.1109/TVCG.2012.23.

model a user's interests. Although these techniques offer important clues about how to track interests during visualization, none of the existing techniques fully satisfies our requirements.

2.1 Visualizing Large Data Sets

Various methods have been proposed to visualize large information spaces. Two topics that are relevant to our research are: 1) techniques for visualizing data sets with a large number of elements n, and 2) techniques for visualizing data sets with a high dimensionality m.

2.1.1 Overview+Detail and Focus+Context

Card et al. define information visualization as "the general application of assembling data objects into pictures, revealing hidden patterns" [4]. Two techniques from this field are closely related to our goal of visualizing large data sets: $overview{+}detail$ 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. For example, the treemap decomposes a data set D into rectangular regions that are hierarchically partitioned based on properties (or attributes) of the data within D [5]. The fisheye lens presents a low level-of-detail display of the entire data set, 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 [6]. The hyperbolic tree structures information in D as a tree embedded in the surface of a sphere [7]. 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 visualizes a hierarchical information space as a tree of semitransparent 3D cones, one for each category in the hierarchy [8]. Elements within a category are located around the base of the appropriate cone.

2.1.2 Multidimensional Visualization

Multidimensional visualization addresses the need to visualize data sets that contain multiple data attributes. One common and long-studied example is a cartographic map [9]. The basic concept of a map is well understood by most viewers. Visualization is most directly related to thematic maps—maps that focus on specific themes or properties of a geographic area. Examples include isarithmic (contour), proportional symbol, dot, or choropleth (color-coded) maps.

Another approach is the use of multidimensional glyphs that modify their appearance to represent multiple attribute values. Guidelines built on properties of low-level visual perception are used to choose data-to-visual feature mappings that are effective and well suited to a user's analysis tasks. Original work in this area includes Chernoff faces and starplots [10], techniques that used facial characteristics and radial spokes, respectively, to visualize a data element with multiple attribute values. Laidlaw used painterly glyphs to visualize diffuse tensor scans of a mouse spinal cord [11]. Follow-on work used a similar approach for flow visualization [12]. Healey et al. conducted experiments to measure the capabilities of and interactions between basic visual properties of color, texture, and motion. These results are used to construct multidimensional glyphs, and more recently to design multidimensional brush strokes for nonphotorealistic visualizations [3], [13]. More abstract approaches also exist. Parallel coordinates are a well-known technique used to visualize the distributions of attribute values in a multidimensional data set [14]. This allows viewers to identify common trends, relationships, and outliers. Shneiderman proposed starfields and spotfire, techniques that array the data in a 2D scatterplot, then allow users to interactively filter the display based on ranges of attribute values [15].

2.2 Identifying User Interests

In the visualization area, a common approach to identifying a user's interests is to maintain a history of how previous users have visualized and analyzed a data set (e.g., as done in VisTrails [16]). More sophisticated approaches to automatically identify user interests have been presented, although not in the visualization area. Kelly and Teevan provide a comprehensive overview of recent work on inferring user preferences [17]. Much of this research focuses on document retrieval or web browsing. Kelly and Teevan use two axes to describe implicit user feedback: a *behavior* category—examining a web page, annotating a paper—and a *scope* category—for example, sentences, paragraphs, and pages represent three scopes for a document. They found that a majority of the existing techniques involve examination of small or medium-scope objects.

Other techniques have also been documented. Lam et al. propose a two-level approach to identify shifts in user interests [18]: a low-level machine learning algorithm for specific interests, and a higher level Bayesian analyzer for significant shifts in a general "interest profile." Kim and Chan build a user interest hierarchy, a continuum of general to specific interests based on words and phrases in web pages bookmarked by a user [19]. Goecks and Shavlik [20] and Claypool et al. [21] propose tracking a user's interactions with web pages to construct a user interest profile.

2.3 Navigation Assistant

Although $overview{+}detail$ and $focus{+}context$ algorithms offer important advantages, a sufficient increase in n can produce overviews that sample too sparsely to properly represent D, or increase the distance between overview and detail to a point where large distortions occur.

Rather than trying to build an overview, we construct a model of a user's preferences to identify interesting elements throughout D. We use an inset view and automated camera tours to help users navigate to areas of interest (AOIs). Data elements in the main view are visualized as multidimensional glyphs whose visual features are selected using guidelines from human perception.

2.3.1 Direct Interaction

Most visualization systems use direct interaction to examine elements of interest (e.g., picking or brushing). Automatically

identifying a viewer's interests complements these capabilities. The navigation assistant initially provided direct selection and a simple language to define rules that describe elements of interest. Unfortunately, this was not always efficient. For example

- It is difficult to locate elements of interest when only a small subset of the data is visible at any one time.
- It is time consuming to enter more than a few rules, or to update rules when new interests are found.
- It can be difficult to properly specify interests as a set of mathematical and Boolean operations on an attribute's values.

This motivated us to seek a method that "automatically" determines rules of interest based simply on what viewers select or where they look in the visualization.

The navigation assistant provides new interface operations to support interaction with the preference system. Users can reject recommendations about an element being interesting (or uninteresting) using a $\mathrm{keyboard} + \mathrm{mouse}$ click. A rule dialog is provided to allow users to enter known interests, and to modify interest rules the system builds (see Fig. 7a for an example). These interactions are fed back into the system, to track a user's interests and improve recommendations.

3 Preference Elicitation

Preference elicitation is used to construct an accurate user model u to assist a decision support system, in our case, a navigation and visualization system [22]. The outcomes O for a decision problem are defined by the assignment of values to a set of attribute variables, $X = (X_1, \ldots, X_m)$. Decisions require an ordering of the outcomes $o_i, o_j \in O$ to properly respect user preferences. $o_i \succeq o_j$ implies the user prefers outcome o_i to o_j.

In our system, $X_i = (x_{i,1}, \ldots, x_{i,m})$ is a specific set of attribute values and O is the set of all data elements $e_i \in D$. We seek a user model u that assigns an estimated interest to each e_i based on its attribute values X_i. We can then order $e_i \succeq e_j$ to show that e_i is potentially more interesting to a user than e_j.

3.1 Preference Queries

Constructing u requires ways to query a user's preferences. Queries can be explicit, where the user is asked to answer questions or evaluate examples, or implicit, where the user's actions are observed to extract information.

3.1.1 Example Queries

The simplest query example is an order query: "Do you prefer o_i or o_j?" or a rank query: "What is the rank of o_i?" Unfortunately, these queries may not be feasible when O is large. Another disadvantage of value queries is that they force a user to make statements about attribute values out of context. They also require detailed statements, even when a user's preference knowledge may be incomplete or generalized.

Presenting examples is an alternative way to obtain preference information. Various exampling mechanisms exist. One method is *tweaking*, which allows users to modify or "tweak" candidate solutions to narrow their search for an optimal solution [23]. A second approach is *candidate critiquing*, where a user describes the merits or flaws in an example [24], [25]. A third technique asks a user to order a set of examples, allowing for comparison and relative ranking.

3.1.2 Implicit Queries

Implicit preference identification avoids direct requests to a user. Instead, the user's interactions with the system are studied to infer preferences. Most research on implicit preferences involves web browsing [17], [21], [26]. A user's actions are tracked—reading web pages, bookmarking pages, following links, and so on—as well as the time spent for each action.

Implicit queries have a number of drawbacks. Implicit information is uncertain. Observed behavior may not be an appropriate source from which to infer preferences [27] (e.g., if a user views a web page for a sustained period, is he interested in the page, or is he away from his workstation?) Interactive behavior may not be consistent between users. Finally, implicit collection is driven by the user and not by the system. This means the system has no way to request a specific piece of information.

3.2 Preference Elicitation and Classification

Although preferences provided a theoretical framework for our research, the solution we implemented uses a standard classification approach: a Bayesian network is constructed from a training set to generate a continuous range of outcomes representing estimated user interests over each data element. The outcomes—and by extension the data elements—are thresholded into two categories: interesting and not interesting.

Using classification algorithms to generate preferences is not uncommon. Approaches like Bayesian networks are appropriate because they can satisfy a preference algorithm's requirements: support for efficient incremental training input, useful estimates based on a small initial set of example classifications, and relative comparisons both within and between categories. Although other methods (e.g., support vector machines) were considered, Bayesian networks best matched the inputs and uncertainties we need to support.

Preference elicitation *does* contribute important techniques to query a user's interests. Thus, our system can be viewed as a preference-based approach to collect clues about a user's interests, a Bayesian classifier that uses those clues to tag each data element as interesting or not interesting, and association rule mining to compress the set of interesting attributes into a manageable collection of interest rules.

3.3 Navigation Assistant

We seek a user model u that orders elements such that $e_i \succeq e_j$ implies the user is more interested in e_i than e_j. Information about a user's preferences is collected in different ways. Users can define rules to identify known elements of interest. They can also select elements in the visualization that they find interesting. Users perform example ordering to create a starting set of preferences, and example tweaking to refine interest rules the system

discovers. Finally, implicit tracking allows us to infer preferences based on where the user looks in D.

4 User Models

We construct a user model to learn a viewer's preferences, then apply that knowledge to identify interesting data elements. We want a model that provides both qualitative and quantitative information: whether an element is interesting, and if it is, the viewer's level of interest.

Bayesian classifiers are a simple, yet effective tool for performing classification [28], [29]. Bayesian classifiers are a form of supervised learning. They take as input a training set that contains pairings between a collection of attribute values and a class value. Bayesian classifiers can be quickly and easily retrained, so they are appropriate for an interactive environment.

Bayesian classifiers use probabilistic models to equate preference values to the likelihood that a viewer will select an item from a set of data. Qualitatively, Bayesian classifiers assign a class value to a set of attribute values. Quantitatively, they provide a probability distribution describing the confidence of their class assignment.

4.1 Bayesian Probability

Bayesian probability combines some prior probability P for a hypothesis H with new data D to determine a new posterior probability for H. The prior probability is often called the *prior*, and the posterior probability the *posterior*. Specifically, the posterior is the likelihood of H multiplied by the prior

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)}. \tag{1}$$

The posterior probability $P(H \mid D)$ equals the likelihood of seeing data D given hypothesis H is true, $P(D \mid H)$, times the prior probability $P(H)$. $P(D)$ is the probability of seeing data D over all possible hypotheses. Since $P(D)$ is identical for all H, it is often considered a normalizing constant, and removed from the formula.

4.2 Bayesian Networks

A Bayesian network is a directed acyclic graph where nodes represent variables and directed edges represent parent-child dependencies. Each node X_i maintains a probability function that takes as input the node's parent variables, and produces as output a probability for each of X_i's possible values. In other words, each node represents a posterior probability distribution $P(X_i \mid \pi(X_i))$, where $\pi(X_i)$ represents the parents of X_i.

The probability of a particular set of values $(x_1, \ldots, x_m)$ for random variables $X_1, \ldots, X_m$ is computed as a joint distribution

$$P(x_1, \ldots, x_m) = \prod_{i=1}^{m} P(X_i = x_i \mid \pi(X_i)). \tag{2}$$

4.3 Bayesian Classification

Bayesian analysis can be used to convert a training set into an underlying probability distribution function modeled as a Bayesian network. Given a data set D with m data attributes $(A_1, \ldots, A_m)$, suppose a subset of the data elements in D are assigned a classification C (e.g., $C = \{\text{interesting}, \text{not interesting}\}$). We want to use this training set to tag unclassified data elements $e = (a_1, \ldots, a_m)$ with classification $c \in C$. Bayes rule (1) allows us to do this

$$P(c \mid a_1, \ldots, a_m) = \frac{P(a_1, \ldots, a_m \mid c)P(c)}{P(a_1, \ldots, a_m)}. \tag{3}$$

The likelihood $P(c)$ comes directly from the training set. As previously noted, the marginal probability $P(a_1, \ldots, a_m)$ is independent of C, and can therefore be removed. This leaves only the posterior $P(a_1, \ldots, a_m \mid c)$ to be derived.

We apply the product rule $P(A, B) = P(A \mid B)P(B) = P(B \mid A)P(A)$

$$P(a_1, \ldots, a_m, c) = P(a_1 \mid a_2, \ldots, c)P(a_2, \ldots, a_m, c) \tag{4}$$

$$= P(a_1, \ldots, a_m \mid c)P(c). \tag{5}$$

Equating (4) and (5), then using the product rule to replace $P(a_2, \ldots, a_m, c)$ with $P(a_2, \ldots, a_m \mid c)P(c)$ gives

$$\begin{aligned} P(a_1, \ldots, a_m \mid c) &= \frac{P(a_1 \mid a_2, \ldots c)P(a_2, \ldots, a_m, c)}{P(c)} \\ &= P(a_1 \mid a_2, \ldots, c)P(a_2, \ldots, a_m \mid c). \end{aligned} \tag{6}$$

By recursively reducing the final term in a similar way, we can rewrite (6) as

$$\begin{aligned} &P(a_1, \ldots, a_m \mid c) \\ &\quad = P(a_1 \mid a_2, \ldots, c)P(a_2, \ldots, a_m \mid c) \\ &\quad = P(a_1 \mid a_2, \ldots, c) \cdots P(a_{m+1} \mid a_m, c)P(a_m \mid c). \end{aligned} \tag{7}$$

If the attribute values are independent of one another, this reduces to

$$P(a_1, \ldots, a_m \mid c) = \prod_{i=1}^{m} P(A_i = a_i \mid c). \tag{8}$$

To handle attribute dependencies, the product is rewritten as

$$P(a_1, \ldots, a_m \mid c) = \prod_{i=1}^{m} P(A_i = a_i \mid \pi(A_i), c). \tag{9}$$

This is the same joint distribution shown in (2) for Bayesian networks.

4.4 Learning Bayesian Structure

Given a training set, we want to "learn" a Bayesian network that fits the training set. This provides the probabilities needed to solve the joint distribution in (9).

One possibility is a naive Bayesian classifier that assumes attribute independence. The computation needed to build a naive classifier is small, and it often performs as well as more sophisticated techniques [29].

The classifier's performance can be further improved by *boosting*. Boosting is a form of ensemble learning, where a collection of learning models are combined to form conclusions from existing knowledge [30]. Boosting creates different models by varying the weights assigned to elements in the training set. For Bayesian classifiers,

increasing the weight of a training element is equivalent to increasing its frequency in the training set.

Boosted Bayesian classifiers create a new model M^{t+1} based on the accuracy of the previous model M^t. Elements misclassified in M^t have their weights increased, while elements correctly classified have their weights decreased. For example, AdaBoost, a well-known boosting algorithm, runs as follows [31]:

1. Given a base network G and training set D of size n.
2. Let w_i^t be the weight of e_i in the training set at iteration t. Initialize the weights $w_i^1 = \frac{1}{n}$, $i = 1, \ldots, n$.
3. For $t = 1, \ldots, T_n$,
   a. Define error $\epsilon^t = \sum_{i=1}^{n} w_i^t m_t(e_i)$, $m_t = 1$ if M^t's classification of e_i is incorrect, $m_t = 0$ otherwise.
   b. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
   c. Set $w_i^{t+1} = w_i^t \beta^{m_t(e_i)}$.
   d. Normalize w_i^{t+1}.
4. Output ensemble structure $M = (M_1, \ldots, M_{T_n})$.

4.5 Navigation Assistant

We implemented a boosted Bayesian network classifier (BBNC) to classify data elements as interesting or not interesting [32]. BBNCs have been shown to be computationally efficient and accurate [1], [33], [34], [35]. BBNCs can also be run iteratively, accepting new training examples and updating the network as a user interacts with a visualization. Once interesting elements are identified, association rule mining is used to build rules that distinguish the elements within the data set. The rules summarize what we believe are the subset of elements that inspire the viewer's interest.

5 Navigation Framework

We decided to modify and extend an existing navigation assistant that clusters elements of interest and visualizes them as multidimensional glyphs. A brief overview of the system is provided here. Interested readers are directed to [2] for more detail.

In the current system elements of interest must be manually identified by the viewer using mathematical expressions and boolean operators. These elements are spatially clustered into local regions called areas of interest (AOIs). A Delaunay triangulation of the elements of interest in each AOI is reduced to form a local graph cycle that visits each element exactly once. Next, a complete graph of the AOI's centroids is built, with edges weighted by their Euclidean distance. A minimum spanning tree of the graph is constructed to produce a minimum-length tree that visits each AOI. Both local and global graph structures are displayed as an inset within the visualization, providing wayfinding cues to direct viewers to offscreen regions of interesting elements (Fig. 5).

The graph framework also provides a fundamental data structure for constructing automated animations. Graph traversal algorithms are used to build camera paths to view elements of interest within an AOI. Visibility algorithms position the camera to provide unoccluded views of each element of interest as the camera moves along the animation path.

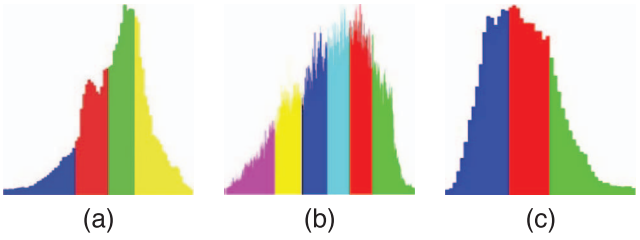

Fig. 1. Discretizing continuous attributes, unique colors identify each bin. (a) *Cloud coverage*, $n = 59$ unique values. (b) *Temperature*, $n = 311$. (c) *Wind speed*, $n = 51$.

Individual data elements are represented as geometric "tower" glyphs that can vary their hue, luminance, height, density, and regularity of placement to represent multiple attribute values. Previous research in our laboratory has shown that the low-level human visual system processes these features very rapidly and accurately. We have conducted numerous controlled experiments to define the information-carrying capacity of each feature, both in isolation, and in combination with other features [3], [36], [37], [38]. The result is a multidimensional visualization system that helps viewers to locate and examine data elements within the data set that are likely to be of interest.

A critical disadvantage of the existing system is that elements of interest must be defined by the viewer. As previously noted, this can be time consuming, imprecise, and is often ineffective at locating all the elements a viewer may want to study.

5.1 Discretizing Attributes

In order to reduce the size of the outcome space O, we discretize continuous attributes A_i into a set of ranges. We believe that most users will not have strong preferences for a specific attribute value, but instead are interested in ranges of values. If specific values are important, a user can identify those by providing explicit interest rules.

There are several approaches for discretizing a continuous space, for example, intervals that have equal width or equal frequency. Discretization can also be viewed as clustering the attribute values [39]. We built a hybrid k-means clustering algorithm to discretize A_i into k bins $X_i = (x_{i,1}, \ldots, x_{i,k})$.

1. Set $k = log_2(n_i)$, where n_i is the number of unique attribute values in A_i.
2. Run m trials of k-means clustering using random seeds as starting points and record the cluster centers $C = (c_{1,1}, \ldots, c_{1,k}, \ldots, c_{m,1}, \ldots, c_{m,k})$.
3. Run hierarchical clustering on C to collapse neighboring centers to a common position.
4. Run a final k-means clustering using C as the starting points. Assign the cluster boundaries to X.

Our algorithm is designed to preserve locally dense regions, and to assign outliers to their own clusters. Attribute values $a_i \in A_i$ map to an interval $x_{i,j} = [x_{i,j,lo}, x_{i,j,hi}]$, $x_{i,j,lo} \leq a_i \leq x_{i,j,hi}$ (Fig. 1). Data elements $e_i \in D$ replace their attribute values $a_{i,j}$ with the index of the bin containing $a_{i,j}$, producing a discretized data set D'.

TABLE 1
Example BBNC: (a) Training Set to Capture Preferences $A_1 = 1$ and $A_4 = 3$; (b) BBNC Classification c and Average Confidence p for Four Sets of $e_i \in D'$

t_i	c
$(1,1,2,1)$	1
$(1,1,3,1)$	1
$(1,2,1,2)$	1
$(1,2,2,1)$	1
$(2,2,3,3)$	1
$(2,3,2,3)$	1
$(3,2,3,3)$	1
$(3,3,2,3)$	1
$(2,1,2,1)$	0
$(2,2,2,2)$	0
$(2,3,3,1)$	0
$(3,2,1,1)$	0
$(3,2,2,2)$	0
$(3,2,3,1)$	0
$(3,3,2,2)$	0

(a)

e_i	c	p
$(1,*,*,*)$	1	0.94
$(*,*,*,3)$	1	0.94
$(1,*,*,3)$	1	1.00
$(\{2,3\},*,*,\{2,3\})$	0	0.75

(b)

5.2 Bayesian User Model

The navigation assistant uses a boosted Bayesian network classifier to tag data elements as interesting or not interesting. As users interact with the visualization, they generate new training examples t_i that are added to the training set T. The BBNC is then retrained, and the elements of interest are updated.

To demonstrate our BBNC's performance, consider T with 15 elements $t_i \in D'$, $D' = (A_1, A_2, A_3, A_4)$, $A_i = \{1,2,3\}\ \forall i$ (Table 1a). The user is interested in two properties: $A_1 = 1$ and $A_4 = 3$. The first four t_i in T satisfy the first property, the next four satisfy the second property, and the last seven satisfy neither. Table 1b shows the BBNC classification for all 81 elements in D'. Every element of interest is properly identified with average confidences ranging from 0.94 to 1.0. Every element that is not interesting is also properly classified with an average confidence of 0.75.

5.2.1 Boosting

Unfortunately, the standard boosting process cannot be integrated directly into the navigation assistant. The main problem is that a general BBNC views all $t_i \in T$ as correct and equally important to the learning process. There may be uncertainty about the correctness of the examples we collect. This is especially true for information that comes from implicit actions. t_i with low certainty need to have a weaker influence during boosting.

An obvious approach is to set the initial boosting weight w_i^1 to t_i's certainty. This will not work, however, since later boosting iterations may increase w_i^1 to a much larger value to compensate for misclassifying t_i.

To solve this problem, we assign two weights to each t_i: the boosting weight w_i^1 and a certainty weight u_i. w_i^1 is modified to guide subsequent user models toward a correct classification of t_i, exactly as before. u_i remains constant for each boosting iteration to reflect the influence t_i should have over the entire ensemble learning process.

Consider the same D' from the previous example (Table 1) but with a preference of $A_1 = 1$ and a new training set containing errors and certainty weights (Table 2a): $(1,1,1,2)$, $(1,2,1,2)$, and $(1,3,2,1)$ should be labeled $c = 1$, while $(2,3,3,2)$, $(2,3,1,1)$, $(2,3,1,3)$, $(3,2,3,2)$, and $(3,2,2,1)$ should be labeled $c = 0$.

TABLE 2
Managing Uncertainty u: (a) Training Set to Capture Preference $A_1 = 1$; (b) Elements of Interest (EOI) for T_n Boosts, Confidence p for Classes $c = 0$ and $c = 1$, Overall Accuracy a, $w_i^1 = \frac{1}{n}$; (c) $w_i^1 = u$; (d) Both u and $w_i^1 = \frac{1}{n}$

t_i	c	u
$(1,1,1,2)$	0	0.1
$(1,2,1,2)$	0	0.1
$(1,3,2,1)$	0	0.1
$(2,2,2,2)$	0	1.0
$(3,3,3,3)$	0	1.0
$(2,3,1,1)$	0	1.0
$(3,3,1,2)$	0	1.0
$(2,1,1,1)$	0	0.1
$(3,1,1,3)$	0	0.1
$(2,2,1,2)$	0	0.1
$(3,2,3,1)$	0	0.1
$(3,1,2,2)$	0	0.1
$(1,1,1,1)$	1	1.0
$(1,2,1,3)$	1	1.0
$(1,2,2,1)$	1	1.0
$(1,3,2,2)$	1	1.0
$(2,3,3,2)$	1	0.1
$(2,3,1,1)$	1	0.1
$(3,3,1,3)$	1	0.1
$(3,2,3,2)$	1	0.1
$(3,2,2,1)$	1	0.1

(a)

$w_i^1 = \frac{1}{n}$

T_n	EOIs	p for $c=0$	p for $c=1$	a
1	44	0.59	0.66	0.67
10	47	0.55	0.59	0.63
20	46	0.55	0.59	0.63

(b)

$w_i^1 = u$

T_n	EOIs	p for $c=0$	p for $c=1$	a
1	35	0.86	0.61	0.96
10	35	0.64	0.57	0.96
20	35	0.64	0.57	0.96

(c)

u and $w_i^1 = \frac{1}{n}$

T_n	EOIs	p for $c=0$	p for $c=1$	a
1	32	0.88	0.63	0.96
10	36	0.89	0.64	0.96
20	36	0.89	0.65	0.96

(d)

Table 2 shows the performance of using: no certainty u (Table 2b); u as the initial boosting weight w_i^1 (Table 2c), and u independent of w_i^1 (Table 2d).

When u is ignored, the BBNC misclassifies a large number of elements. After $T_n = 10$ boosts over half of D' is classified as interesting, with low average confidence $p = 0.57$ and accuracy $a = 0.63$. When $w_i^1 = u$, there is an immediate improvement in BBNC accuracy, with almost every element the user prefers being captured ($a = 0.96$). Notice, however, that the confidence in elements classified as interesting declines significantly in later boost iterations. Erroneous examples influence the model more during later iterations as the BBNC tries to "correct" them. Finally, when u and w_i^1 are managed independently, the model's accuracy is as good as for $w_i^1 = u$. This approach also maintains strong confidence for interesting elements with $A_1 = 1$.

Unfortunately, all three schemes incorrectly classify some elements with $A_1 \neq 1$ as interesting. Although this is undesirable, we believe it is better to have false positives—suggesting elements that are not interesting—rather than false negatives—missing interesting elements entirely—as long as the number of false positives is relatively small.

5.3 User Input

Visualization is normally interactive. A user can change how the data is visualized, he can navigate to new locations in the data set, he can look at the same data elements from different locations, and so on. User intervention can also be used to guide and correct the assistant. Our assistant must transform existing visualization actions into inputs that are compatible with our BBNC preference model. Several explicit and implicit methods are provided to support this.

5.3.1 Updating the Preference Model

The training set T is the bridge that transforms user input into preference values. T must be updated in a meaningful way when new user input is received. Preference statements have a subject, a classification, and a certainty. For example, when a user clicks on a data element e_i to mark it as interesting, the subject is e_i, the classification is positive, and the certainty is high. This action is converted into $t_i = (e_i, c_i, u_i)$ made up of the discretized form of e_i, classification c_i, and certainty u_i.

Next, the assistant must integrate t_i into T. A simple approach is to append t_i to T. This means T can contain multiple t_i with the same e_i and c_i, but possibly different u_i. Since u_i defines the frequency of e_i in T, we can sum common t_i to produce an overall certainty, for example, $\Sigma\{(e_i, 0, 0.5), (e_i, 0, 0.75), (e_i, 0, 0.4)\} \Longrightarrow (e_i, 0, 1.65)$.

Unfortunately, this basic scheme leads to the problem of overclassification. All of e_i's attribute values have the same u_i, even when the user is only interested in a subset of these values. Consider $t_i = (1, 2, 1, 2)$ in Table 2a. t_i was added because of a user interest in $A_1 = 1$, but $A_2 = 2$, $A_3 = 1$, and $A_4 = 3$ will also be considered as interesting as $A_1 = 1$.

If an attribute value is of interest—for example, $A_1 = 1$ in Table 2a—it should occur in many t_i. The other attributes A_2, A_3, and A_4 will have values with relatively uniform distributions over their domains. Instead of summing to update certainty u_i, we calculate a likelihood that at least one attribute value in e_i is correct.

1. For $t_i = (e_i, c_i, u_i)$, search T for an entry t_j that matches e_i and c_i.
2. If no entry is found, add t_i to T.
3. Otherwise update $u_j = 1.0 - (1.0 - u_j) * (1.0 - u_i)$.

5.4 Explicit Input

Explicit feedback from a user is the most reliable source of information, but it comes at the cost of forcing the user to temporarily stop exploring to specify preferences. We try to keep our explicit input methods simple, both in terms of the interface for the methods, and in the type of preference information they request.

5.4.1 Preference Statement Interface

A user can define known preferences by entering them explicitly using a preference statement interface. The attributes' discretized ranges are presented, allowing the user to select the subset of ranges that match the known preference.

5.4.2 Broad Critiquing

The navigation assistant highlights elements of interest in the visualization. A user can critique the model by explicitly adding or removing interesting elements.

An element e_i incorrectly tagged as uninteresting can be added with a $\mathrm{keyboard} + \mathrm{mouse}$ selection. e_i is added to a list of elements explicitly labeled as interesting by the user. Any $e_j \in D', e_j = e_i$ is marked as an element of interest, and T is updated

1. For $t_j \in T$ with $e_j = e_i$, $c_j = 0$, remove t_j from T.
2. If $t_i = (e_i, 1, u) \in T$, set $u = 1$, otherwise add $(e_i, 1, 1)$ to T.

If a user clicks on an interesting e_i to tag it as uninteresting, the same process is applied with the class c set to 0. The user does not need to explain which properties of e_i make it interesting or uninteresting. The Bayesian user model analyzes the elements to determine what caused the user's actions.

5.4.3 Fine Critiquing

Selecting glyphs is a simple way to indicate preferences, but it offers no information about which attribute values are interesting to a user. The navigation assistant provides an interface to allow the user to enter this information. In order to avoid an overly complex interface, our fine critiquing dialog asks only two questions (Fig. 7a).

1. Is data element e_i interesting?
2. Which attribute values of e_i make it interesting?

For example, suppose a user confirms e_i is interesting, and specifies $Y \subseteq X$ as the attributes of interest. In order to update T with respect to Y, we must allow for incomplete training examples. These examples influence preferences toward a subset of attribute values. For example, $(*, *, 1, *)$ specifies information about the third attribute only. All e_i with $a_{i,3} = 1$ will have their preference value influenced by this example.

The values y for the attributes Y must be added to T. The obvious solution of using $t_i = (y, 1, 1)$ is insufficient. We want t_i to have a stronger influence than normal examples t_j, since t_i represents a strong statement by the user toward y_i. It would take multiple fully specified t_j to produce the same influence. To support this, we specify a certainty $u \geq 1$.

A second issue is whether the user's preference toward the values in y are independent of one another, that is, should the assistant strengthen preferences toward e_i containing at least one attribute value in y, or only toward e_i that contain all the attribute values in y? Since our interface provides no mechanism to define this, we generate all possible subsets of $y \in Y$ and add each of them as incomplete training examples. The assistant assigns t_i with fewer undefined attribute values a larger u_i. Although there are $2^{|Y|} - 1$ total subsets, we assume most fine critiques will involve only a few attributes, keeping the number of subsets manageable.

Consider $e_i = (3, 1, 4, 4, 5)$, and the user indicates he is interested in the values of the first three attributes. The following incomplete t_i will be added to T with $c_i = 1$

$$u_i = 3: \quad (3,1,4,*,*),$$
$$u_i = 2: \quad (3,1,*,*,*), (3,*,4,*,*), (*,1,4,*,*),$$
$$u_i = 1: \quad (3,*,*,*,*), (*,1,*,*,*), (*,*,4,*,*).$$

The same process is applied for values marked as uninteresting, but with $c_i = 0$. We also derive information from $Z = X - Y$, the attribute values the user ignored, by assuming he is indifferent to these values. Incomplete t_i based on Z are added as both interesting and uninteresting. Since it is possible that $|Z| \gg |Y|$, we only add incomplete t_i with one attribute value set. In the above example, we would add $t_i = ((*,*,*,4,*),\{0,1\},1)$ and $t_i = ((*,*,*,*,5),\{0,1\},1)$ to T.

5.5 Implicit Input

Implicit input occurs alongside a user's normal interaction with the visualization. This type of input contains a degree of uncertainty. Because of this, only cumulative implicit feedback can significantly influence the user model. Noisy input will have little or no effect.

5.5.1 Broad Focus Events

The navigation assistant tracks which parts of the data set move through the user's focus. The intuition is that an element's onscreen presence offers an indication of its importance.

We measure a *degree of focus* to compute an element's *wear weight*. The degree of focus depends on three viewing parameters.

1. **Camera distance.** Elements closer to the camera present more detail, and are assumed to be of more interest to a user.
2. **Center distance.** Elements near the center of the window are centered in the user's view, and are assumed to be more interesting.
3. **Viewing time.** Elements that are in view for a sustained, continuous period of time are assumed to be of more interest to the user. We use continuous rather than cumulative time to manage common elements that are frequently in view, but only for short durations.

OpenGL's projection operators are used to define a frustum that contains a subset of the elements in view. Elements in a circular region V inscribed within the frustum are selected for consideration. The radius of V is one-half the length of the shorter edge of the window. For each element $e_i \in V$ a wear weight wt is computed

$$wt = t_v\,(k_1 + k_2(1-z) + k_3\,(1-d)), \tag{10}$$

where t_v is the viewing time, z is the camera distance, and d is the center distance. Constants k_1, k_2, and k_3 weight the contributions of z and d to wt. The constants depend on sampling frequency: higher sampling rates produce smaller k_i.

Fig. 2 shows examples of wt for different camera positions. Notice how the wear weight increases for a sustained view over time (Fig. 2b) versus a single snapshot (Fig. 2a). Once wt for e_i exceeds a minimum threshold, the assistant adds $t_i = (e_i, 1, \beta)$ to T.

β is a constant used for all implicit input events. It is small enough so it does not immediately impact the preference model, but large enough so that multiple implicit events for e_i will direct the model to classify e_i correctly. For example, with $\beta = 0.2$, suppose a user views e_i long enough for its wt to exceed the interest threshold. If this happens three times during a visualization session, three occurrences of $(e_i, 1, 0.2)$ are added to T, producing a certainty $u = 0.49$ for classification $c = 1$ (see the formula for u at the end of Section 5.3.1).

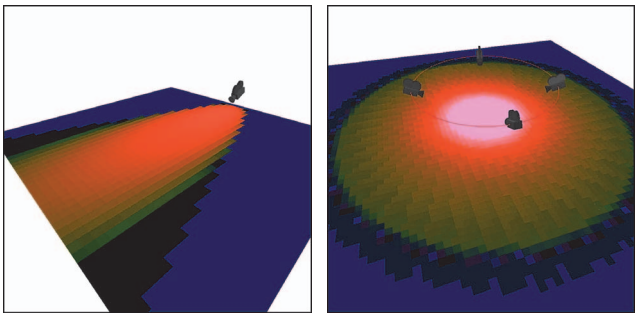

Fig. 2. Wear weight examples, low to high colored dark blue to bright pink. (a) A straight camera shot. (b) Effects of time as a camera rotates about a common view position.

5.5.2 Fine Focus Events

Our multidimensional visualizations use geometric glyphs that vary their visual properties to represent multiple attribute values. A user can infer approximate values based on the perception of a glyph's appearance. A user can also select an element e_i to show e_i's attribute values in an information balloon. We assume that when an element is examined at this level of detail, the user can accurately determine whether the element is interesting. The navigation assistant uses this action as an opportunity to update its user preference model.

The assistant's response depends on whether e_i is already an element of interest. If it is, and if the user does not explicitly remove e_i from the set of interesting elements after examination, the assistant reinforces the model's classification of e_i by adding $t_i = (e_i, 1, \beta)$ to T. If e_i is not interesting, the assistant requires additional evidence to decide how to proceed. If some other action within a short time window suggests e_i is interesting, the assistant strengthens e_i's preference by adding $t_i = (e_i, 1, \beta)$ to T. Otherwise the assistant weakens e_i's preference by adding $t_i = (e_i, 0, \beta)$.

5.6 Interest Rules

Interest rules are defined as a nested list of discrete attribute value ranges. This is a common format for data mining algorithms. Rules are generated using association rule mining on the current set of elements of interest [40], [41]. First, collections of attribute values that occur together with a significant frequency—frequent item sets—are identified. Normally, there are more frequent item sets than a user would want to see. We score interest rules based on their importance and relevance. Rules are presented in descending sorted order by score, prioritizing the rules the user is most likely to find interesting.

The support σ of a frequent item set ϕ is the number of e_i that contain the item set. We calculate support over the set

of elements of interest, $\sigma_{\phi,E}$, and for the entire data set D', $\sigma_{\phi,D'}$. If $\sigma_{\phi,E}$ is below a minimum threshold, the interest rule is discarded. Otherwise, a combination of the support values is used to score the rule. If $\sigma_{\phi,E}$ and $\sigma_{\phi,D'}$ are both high, it is unlikely that ϕ is a property of interest. If $\sigma_{\phi,E}$ is high and $\sigma_{\phi,D'}$ is low, however, then ϕ is useful for distinguishing elements of interest within D'. Based on this intuition, the score w_ϕ for each ϕ is

$$w_\phi = \frac{(\sigma_{\phi,E})^2}{\sigma_{\phi,D'}}. \quad (11)$$

To further reduce the number of frequent item sets, we merge item sets together if the following criteria are met.

1. Both item sets have a minimum $\sigma_{\phi,E}$.
2. Both item sets contain the same nonempty attribute ranges.
3. The difference between nonempty attribute ranges is no more than 1.

5.7 Initial View Selection

When visualization begins, the assistant initializes the user model with general preference information using a set of intelligently chosen views. The goal is to quickly eliminate large numbers of e_i as being potentially interesting. To do this, the assistant needs to determine preferences for prominent yet distinctive features in D'.

Using k-means, we cluster D' to form groups of e_i with similar attribute values. Each cluster C_i is mined to collect its descriptive features f_i. Each f_i is assigned a weight w_i based on its support in C_i and D', $w_i = (\sigma_{f_i,C_i})^2/\sigma_{f_i,D'}$. f_i that are common in C_i, but uncommon throughout D' have higher w_i. f_i with the largest w_i is selected as the representative feature for C_i.

For each C_i, a view is constructed that contains a large neighborhood of elements with attribute values f_i. The assistant positions the OpenGL camera to visualize the neighborhood. A hill-climbing search is used to move the camera back and forth along a parabolic arc, bringing elements into view that contain f_i, or removing visible elements that do not contain f_i. The view with the highest support σ_{f_i} for f_i is selected.

The views for each C_i are presented in an array layout (Fig. 3). The user indicates interest in a view by selecting it with the mouse. For selected views, incomplete training examples $t_i = (f_i, 1, 2.0 * \sigma_{f_i})$ are added to T. For unselected views, we assume the user is indifferent to its f_i. We, therefore, add $t_i = (f_i, \{0, 1\}, 2.0 * \sigma_{f_i})$ to mark the feature as equally interesting and uninteresting.

6 Visualizing Climatology Data

We investigated the practical abilities of our navigation assistant using a large climatology data set collected by the Intergovernmental Panel on Climate Change (IPCC). The data set records monthly 30-year averages for 11 climate attributes. Data is sampled in $\frac{1}{2}^\circ$ latitude and longitude steps at every positive elevation throughout the world. In total, D contains approximately 750,000 data elements and 8.25 million attribute values. After discretization, D is reduced to 833 distinct ranges.

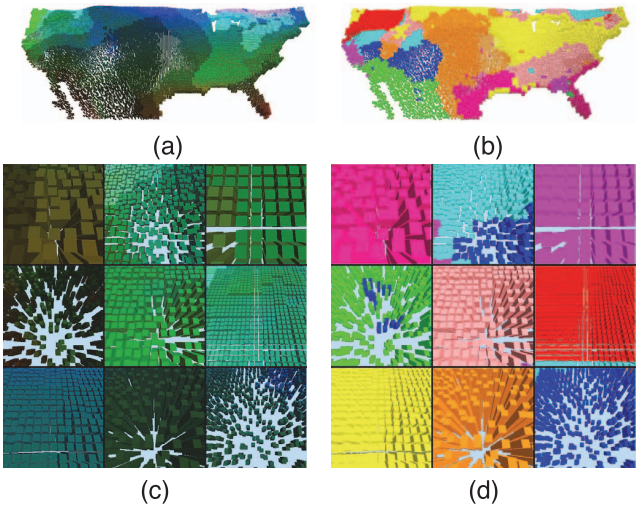

Fig. 3. Initial views. (a) Climatology data clustered into neighborhoods of similar elements. (b) Elements colored to identify cluster membership. (c) Initial views of similar elements. (d) elements colored to identify cluster membership.

Data elements e_i are represented with 3D "tower" glyphs that vary their color and texture properties to visualize e_i's attribute values. We mapped

- $A_1 = cloud\ coverage \rightarrow$ density,
- $A_2 = temperature \rightarrow$ color,
- $A_3 = wind\ speed \rightarrow$ height,
- $A_4 = wet\ day\ frequency \rightarrow$ luminance, and
- $A_5 = temperature\ range \rightarrow$ regularity of placement.

Research in our laboratory has shown that these features are perceptually salient, both in isolation, and in combination with one another. The mapping order is based on attribute importance: important attributes are assigned to perceptually strong visual features. Interested readers are directed to [3] for more details.

Fig. 4 displays all data elements for February. This demonstrates how viewing D' in its entirety degrades the effectiveness of the visualization. Although regions with similar luminances, colors, and densities are visible, differences in height and regularity are difficult to see. Moreover, certain values of one feature—for example, low glyph densities—can mask other features like luminance and color. A user must zoom in to obtain a reasonable level of detail, causing large parts of D' to move offscreen.

Our example scenarios were based on explorations climatology colleagues described as part of their work. For example, attribute correlations are used to

1. Locate areas with attributes values that may identify features of interest.
2. Visually examine the regions to see whether the features exist.
3. Determine if other attribute values correlate with the presence of the feature.
4. Modify the attribute value set to improve feature detection.

Scientists also compare historical conditions to computational models by checking to see whether the types and distributions of attribute values from real-world data match simulated results. We focus here on describing the theoretical

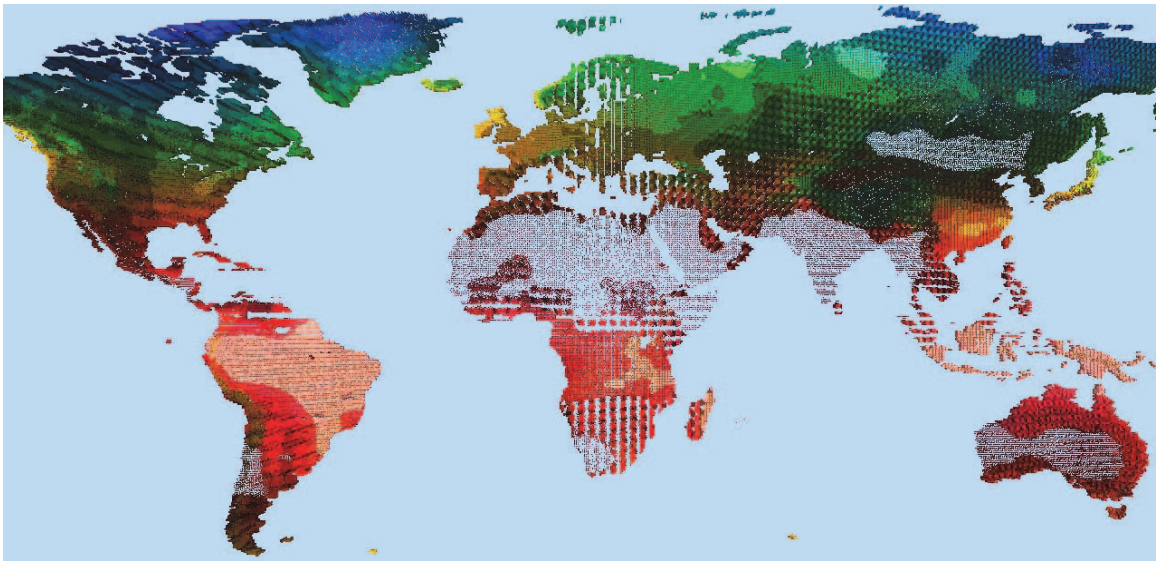

Fig. 4. A perceptual color and texture-based visualization of February's historical climate patterns throughout the world.

details of our navigation system and demonstrating how it captures user input to estimate user interests.

6.1 Broad Critiquing and Fine Focus

We provide two example scenarios to demonstrate identifying a viewer's interests from a small number of interactions. The Bayesian classifier examines each e_i to determine how well it fits interest rules the classifier has discovered. Spatial groups of interesting elements are highlighted to allow users to quickly locate them.

The first example demonstrates how a preference model is refined with explicit broad critiquing—adding or removing t_i from T—and implicit fine focus events —observing a user's actions after viewing e_i's information balloon. The goal is to examine areas with dense *cloud coverage* and moderate *temperature*, to see if the other attributes form patterns within these regions.

Step 1. Following an initial view selection, the user moves to western Canada (Fig. 5a) and explicitly adds $(4, 3, 5, 5, 1)$ and $(4, 3, 5, 4, 2)$, and rejects $(4, 4, 5, 5, 1)$ and $(3, 3, 5, 3, 3)$. Elements of interest are outlined in blue, and navigation graph edges are displayed as red lines (Fig. 5b). Each addition or removal requires an information balloon to query an element's values—a fine focus event—followed by

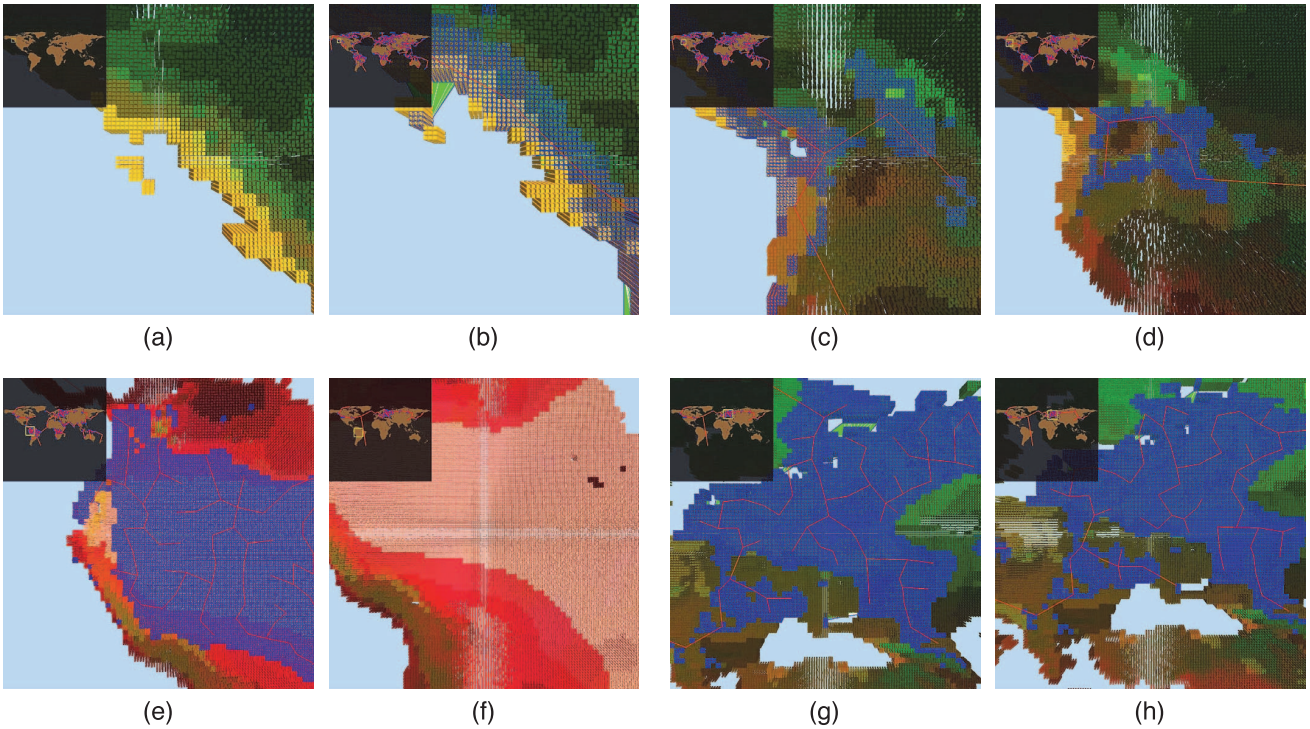

Fig. 5. Before and after visualizations, with the navigation graph overview inset in the top-left corner. (a) and (b) Step 1. (c) and (d) Step 2. (e) and (f) Step 3. (g) and (h) Step 4.

TABLE 3
The Top Four Rules After Each Step in the Broad Critiquing and Fine Focus Example

Step	Top Four Rules
1	$(*,*,*,*,2),\ w=1.40;\quad (*,*,*,\{4,5\},*),\ w=1.40;\quad (4,*,*,*,*),\ w=1.18;\quad (*,*,5,*,2),\ w=1.18$
2	$(4,*,*,\{3,4,5\},*),\ w=3.85;\quad (4,*,*,*,*),\ w=3.64;\quad (4,*,5,*,*),\ w=3.14;\quad (4,*,\{4,5\},*,*),\ w=2.76$
3	$(4,\{2,3\},5,*,*),\ w=7.47;\quad (4,\{2,3\},*,*,*),\ w=7.47;\quad (4,3,5,*,*),\ w=4.52;\quad (4,3,*,*,*),\ w=4.52$
4	$(4,3,5,*,*),\ w=10.02;\quad (4,3,*,*,*),\ w=10.02;\quad (4,3,5,3,*),\ w=5.30;\quad (4,3,*,*,3),\ w=5.30$
5	$(4,3,5,\{3,4,5\},*),\ w=29.46;\quad (4,3,*,\{3,4,5\},*),\ w=29.46;\quad (4,3,*,*,*),\ w=23.93;\quad (4,5,5,*,*),\ w=23.93$

an action—an explicit broad critiquing event. This produces 104 training examples. The four highest ranked interest rules are shown in Table 3.

Step 2. Following the navigation graph, the user shifts to the Pacific Northwest to view a large cluster of elements of interest (Figs. 5c and 5d). The user adds $(4,3,5,4,3)$ and $(4,3,5,3,3)$, and rejects $(4,4,5,4,3)$, $(3,3,5,4,2)$, and $(2,3,5,2,4)$. At this point the assistant begins to identify the correct interest rules (Table 3). The number of training examples is reduced to 60, and each of the four top interest rules reference dense *cloud coverage*, $A_1 = 4$.

Step 3. The user again follows the navigation graph to South America to view a large AOI (Fig. 5e). Here, the user adds $(4,3,5,5,1)$ and removes $(4,6,5,5,2)$ and $(4,5,5,4,3)$. In response, the preference model removes the entire AOI (Fig. 5f), and reduces the number of training examples to 25. The top four interest rules now contain the appropriate attribute values for *cloud coverage* and *temperature*, $A_1 = 4$ and $A_2 = 3$ (Table 3).

Step 4. The user moves to Europe to evaluate another large AOI (Fig. 5g), adding $(4,3,5,3,1)$ and $(5,3,5,4,1)$, and removing $(3,3,5,3,1)$ and $(4,2,4,3,2)$. The AOI remains stable since most of its elements are of interest to the user (Fig. 5h). Only 17 training examples remain.

Step 5. The user concludes by exploring Japan (Fig. 6), where elements $(4,1,5,3,3)$, $(4,5,5,3,1)$, and $(1,3,5,1,3)$ are rejected. This final tweaking allows the preference model to identify outliers in South America that were missed during the user's initial visit. At this point, more complex rules are beginning to emerge, for example, a possible correlation with moderate to high *wind speed* and *wet day frequency*, $A_3 = \{3,4,5\}$ and $A_4 = \{3,4,5\}$ (Table 3).

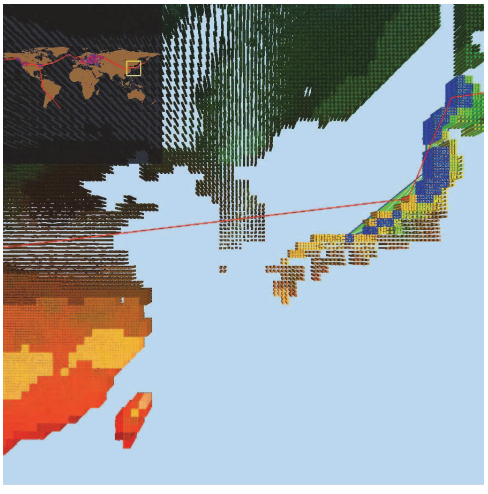

Fig. 6. Visualization after tweaking at Step 5.

6.2 Implicit Focus and Explicit Critiquing

The second example demonstrates the use of implicit focus events and explicit critiquing to define incomplete training examples that capture complex preferences. The goals are to confirm that areas in the Americas exist with both high *wet day frequency* and low *temperature*: $A_4 = \{4,5\}$ and $A_5 = \{1,2\}$, and to see if other attributes form patterns within these regions.

Step 1. Following an initial view selection, the user begins exploring in Central America, implicitly rejecting $(3,5,3,3,3)$, $(3,6,2,2,4)$, and $(2,6,4,2,3)$. The user locates and adds interesting element $(2,6,5,5,1)$. Failing to find additional interesting elements, the user critiques the element he just selected, indicating interests in the values of *wet day frequency* and *temperature range* in a fine critiquing window (Figs. 7a and 7b). The top four interest rules are shown in Table 4. Although the model already has a good estimate of the user's preferences. more input is needed to expand its interest rules.

Step 2. The user moves to the Great Lakes region (Figs. 7c and 7d), adding $(4,3,5,5,2)$, $(4,3,5,4,2)$, $(4,3,5,4,3)$, $(4,2,5,4,3)$, and $(4,3,5,4,2)$ by exploring the local neighborhood. More training examples are implicitly added to T as the user examines the AOI. The interest rules show how the new elements change the model (Table 4). The top rules remain focused on A_4 and A_5, but do not (yet) identify a dependency between them.

Step 3. The user shifts to northern Canada to investigate another AOI (Fig. 7e). Here, the user explicitly removes $(2,1,5,2,1)$. The new global graph shows how this single rejection produces a significant change in the preference model (Fig. 7f). The interest in numerous elements decreased to a point where the model no longer highlights them (Table 4). The model is again identifying a dependency between A_4 and A_5.

Step 4. The user finishes by visiting the Pacific Northwest (Figs. 7g and 7h). He adds $(4,4,5,5,1)$, $(4,4,4,4,1)$, $(4,4,2,4,2)$, $(4,4,3,4,2)$, and $(3,4,2,4,2)$, and removes $(4,3,4,3,2)$. After this input, the assistant continues to report a strong link between high *cloud coverage* and low *temperature* (Table 4). Unlike the previous example, however, no other attribute patterns are reported, with $\{A_1, A_2, A_3\} = \{*,*,*\}$ in all four interest rules.

6.3 Feedback

Although we did not run formal validation studies, anecdotal feedback from our colleagues was positive, highlighting the following strengths of interest-driven visualization:

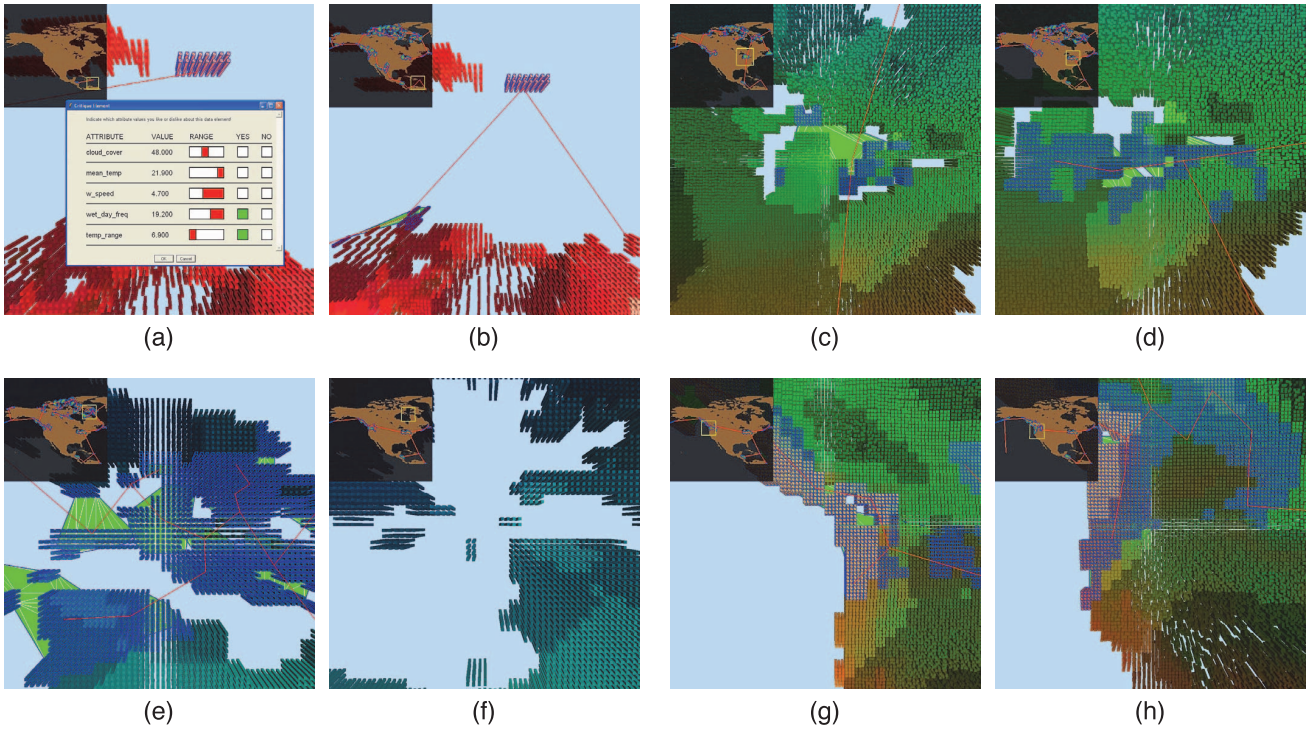

Fig. 7. Before and after visualizations, with the navigation graph overview inset in the top-left corner. (a) Fine focus critiquing. (b) After critiquing. (c) and (d) Step 2. (e) and (f) Step 3. (g) and (h) Step 4.

- **Pattern reporting.** Select elements of interest, then have the system report the attribute patterns that distinguish the selected elements.
- **Interesting regions.** Automatically identify clusters of interesting elements from a few exemplars, then use the system to locate and analyze the clusters.
- **Hypothesis testing.** Define a high-level hypothesis with a few explicit rules, identify elements of interest based on the rules, use critiquing to remove elements from this set that are *not* interesting, then look at the rules that remain to see how they differ from the original hypothesis.

7 Conclusions

The ability to locate and explore interesting offscreen data is a critical problem for visualizing large data sets. These data sets can overwhelm a user's ability to see the data at full detail in a single view, causing the user to become disoriented and unsure about where to look next in the visualization.

Previously we described a navigation assistant that addresses these issues by building graphs of elements of interest and exploring them with intelligent camera planning [2]. This paper proposes replacing manual identification of elements of interest with a preference model that automatically defines rules to locate the elements. A boosted Bayesian network classifier is built to achieve this goal. The BBNC is carefully designed to consider uncertainty during classification. A simple set of explicit operations allow a user to critique the preference model. Common actions performed during visualization implicitly capture additional details about a user's preferences. Two example visualization sessions are described to demonstrate the capabilities of both the BBNC and the navigation assistant.

7.1 Relevance to Visualization

The ability to capture a user's interests is applicable to a wide range of visualization techniques. We believe our BBNC can be used in numerous situations where understanding a user's interests is important. For example, a visualization algorithm could filter the data it renders based on importance. Viewer interest could serve as a semantic cue to determine whether to display or hide elements during *focus+context* zooming. Elements with strong viewer interest could act as representative examples, allowing a data set to be clustered into subsets of interesting elements with distinguishable patterns in their attribute values.

TABLE 4
The Top Four Rules After Each Step in the Implicit Focus and Explicit Critiquing Example

Step	Top Four Rules
1	$(*,*,*,\{4,5\},1),\ w=4.31;\quad (*,*,*,*,1),\ w=4.26;\quad (4,*,*,*,1),\ w=3.14;\quad (*,*,5,*,1),\ w=3.11$
2	$(*,*,*,*,1),\ w=4.39;\quad (*,*,5,*,\{1,2\}),\ w=4.04;\quad (*,*,*,5,*),\ w=3.84;\quad (*,*,5,*,1),\ w=3.82$
3	$(*,*,*,5,\{1,2\}),\ w=19.58;\quad (*,*,*,5,*),\ w=14.88;\quad (*,*,*,5,1),\ w=12.37;\quad (\{3,4\},*,*,5,*),\ w=11.99$
4	$(*,*,*,\{4,5\},\{1,2\}),\ w=7.46;\quad (*,*,*,\{4,5\},*),\ w=4.15;\quad (*,*,*,\{4,5\},1),\ w=3.98;\quad (*,*,*,4,\{1,2\}),\ w=3.65$

Our approach is designed to be flexible in the types of data sets it can analyze. If a viewer's interests can be defined by combinations of the attribute values, our preference algorithm will be able to extract rules that identify interesting elements.

REFERENCES

[1] Y. Jing, V. Pavlovic, and J.M. Regh, "Efficient Discriminative Learning of Bayesian Network Classifiers via Boosted Augmented Naive Bayes," *Proc. 22nd Int'l Conf. Machine Learning (ICML '05),* pp. 369-376, 2005.

[2] B.M. Dennis and C.G. Healey, "Assisted Navigation for Large Information Spaces," *Proc. IEEE 13th Visualization Conf. (Vis '02),* pp. 419-426, 2002.

[3] C.G. Healey and J.T. Enns, "Large Data Sets at a Glance: Combining Textures and Colors in Scientific Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 5, no. 2, pp. 145-167, Apr-June 1999.

[4] S.K. Card, J.D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann Publishers, Inc., 1999.

[5] B. Shneiderman, "Tree Visualization with Tree-Maps: A 2D Space Filling Approach," *Trans. Graphics,* vol. 11, no. 1, pp. 92-99, 1992.

[6] G.W. Furnas, "Generalized Fisheye Views," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '86),* pp. 16-34, 1986.

[7] J. Lamping and R. Rao, "The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies," *J. Visual Languages and Computing,* vol. 7, no. 1, pp. 33-55, 1996.

[8] G.G. Robertson, J.D. Mackinlay, and S.K. Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Information," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '91),* pp. 189-194, 1991.

[9] T.A. Slocum, *Thematic Cartography and Geographic Visualization.* Prentice-Hall, Inc., 2005.

[10] M.D. Lee, R.E. Reilly, and M.E. Butavicius, "An Empirical Evaluation of Chernoff Faces, Star Glyphs, and Spatial Visualizations for Binary Data," *Proc. Asia-Pacific Symp. Information Visualization,* vol. 24, pp. 1-10, 2003.

[11] D.H. Laidlaw, E.T. Ahrens, D. Kremers, M.J. Avalos, R.E. Jacobs, and C. Readhead, "Visualizing Diffusion Tensor Images of the Mouse Spinal Cord," *Proc. Visualization,* pp. 127-134, 1998.

[12] R.M. Kirby, H. Marmanis, and D.H. Laidlaw, "Visualizing Multivalued Data from 2D Incompressible Flows Using Concepts from Painting," *Proc. Visualization,* pp. 333-340, 1999.

[13] C.G. Healey, J.T. Enns, L.G. Tateosian, and M. Remple, "Perceptually-Based Brush Strokes for Nonphotorealistic Visualization," *ACM Trans. Graphics,* vol. 23, no. 1, pp. 64-96, 2004.

[14] A. Inselberg, *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications.* Springer, 2009.

[15] B. Shneiderman, "Dynamic Queries, Starfield Displays, and the Path to Spotfire," http://www.cs.umn.edu/spotfile, 1999.

[16] C. Silva, E. Anderson, E. Santos, and J. Freire, "Using VisTrails and Provenance for Teaching Scientific Visualization," *Computer Graphics Forum,* vol. 30, no. 1, pp. 75-84, 2011.

[17] D. Kelly and J. Teevan, "Implicit Feedback for Inferring User Preference: A Bibliography," *ACM SIGIR Forum,* vol. 37, no. 2, pp. 18-28, 2003.

[18] W.S. Lam, J.M. Mukhopadhyay, and M. Palakal, "Detection of Shifts in User Interests for Personalized Information Filtering," *Proc. 19th Int'l Conf. Research and Development in Information Retrieval (SIGIR '96),* pp. 317-325, 1996.

[19] H.-R. Kim and P.K. Chan, "Learning Implicit User Interest Hierarchy for Context in Personalization," *Applied Intelligence,* vol. 28, no. 2, pp. 153-166, 2008.

[20] J. Goecks and J. Shavlik, "Learning Users' Interests by Unobtrusively Observing Their Normal Behavior," *Proc. Fifth Int'l Conf. Intelligent User Interfaces (IUI '00),* pp. 129-132, 2000.

[21] M. Claypool, P. Le, M. Waseda, and D. Brown, "Implicit Interest Indicators," *Proc. Sixth Int'l Conf. Intelligent User Interfaces (IUI '01),* pp. 14-17, 2001.

[22] R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives.* Cambridge Univ. Press, 1976.

[23] R. Burke, K. Hammond, and R. Young, "The FindMe Approach to Assisted Browsing," *IEEE Expert,* vol. 12, no. 4, pp. 23-40, July/Aug. 1997.

[24] G. Linden, S. Hanks, and N. Lesh, "Interactive Assessment of User Preference Models: The Automated Travel Assistant," *Proc. Sixth Int'l Conf. User Modeling (UM '97),* pp. 67-78, 1997.

[25] S. Shearin and H. Lieberman, "Intelligent Profiling by Example," *Proc. Sixth Int'l Conf. Intelligent User Interfaces (IUI '01),* pp. 145-151, 1999.

[26] D.W. Oard and J. Kim, "Modeling Information Content Using Observable Behavior," *Proc. 64th Ann. Meeting of the Am. Soc. for Information Science and Technology,* pp. 38-45, 2001.

[27] D. Kelly and N.J. Belkin, "Display Time as an Implicit Feedback: Understanding Task Effects," *Proc. 27th Int'l Conf. Research and Development in Information Retrieval (SIGIR '04),* pp. 377-384, 2004.

[28] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning,* vol. 29, nos. 2/3, pp. 131-163, 1997.

[29] P. Langley, W. Iba, and K. Thompson, "An Analysis of Bayesian Classifiers," *Proc. 10th Nat'l Conf. Artifical Intelligence (AAAI '92),* pp. 223-228, 1992.

[30] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach,* third ed. Prentice-Hall, Inc., 2010.

[31] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application of Boosting," *J. Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[32] Y. Jing, V. Pavlovic, and J.M. Regh, "Boosted Bayesian Network Classifiers," *Machine Learning,* vol. 73, no. 2, pp. 155-184, 2008.

[33] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning,* vol. 36, nos. 1/2, pp. 105-139, 1999.

[34] G. Ridgeway, D. Madigan, and T. Richardson, "Interpretable Boosted Naive Bayes Classification," *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD '98),* pp. 101-104, 1998.

[35] K.M. Ting and Z. Zheng, "A Study of AdaBoost with Naive Bayesian Classifiers," *Computational Intelligence,* vol. 19, no. 2, pp. 186-200, 2003.

[36] C.G. Healey, K.S. Booth, and J.T. Enns, "High-Speed Visual Estimation Using Preattentive Processing," *ACM Trans. Computer-Human Interaction,* vol. 3, no. 2, pp. 107-135, 1996.

[37] C.G. Healey and J.T. Enns, "Building Perceptual Textures to Visualize Multidimensional Data Sets," *Proc. IEEE Ninth Visualization Conf. (Vis '98),* pp. 111-118, 1998.

[38] D.E. Huber and C.G. Healey, "Visualizing Data with Motion," *Proc. IEEE 16th Visualization Conf. (Vis '05),* pp. 527-534, 2005.

[39] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys,* vol. 31, no. 3, pp. 264-323, 1999.

[40] R. Agrawal, T. Imielinski, and R. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 207-216, 1993.

[41] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Int'l Conf. Very Large Data Bases (VLDB '94),* pp. 487-499, 1994.

Christopher G. Healey received the BMath degree from the University of Waterloo, Canada, and the MSc and PhD degrees from the University of British Columbia in Vancouver, Canada. Following a postdoctoral fellowship at UC Berkeley, he joined the Department of Computer Science at North Carolina State University, where he is currently an associate professor. His research interests include visualization, graphics, visual perception, and data analytics. He is a senior member of the IEEE.

Brent M. Dennis received the BS degree in mathematics from Davidson College, North Carolina, and the MS and PhD degrees from North Carolina State University in Raleigh, North Carolina. Currently, he is a member of the technical staff at the Massachusetts Institute of Technology Lincoln Laboratory, where he works on designing tools to support the exploration of large information spaces. His research interests include visualization, perception, artificial intelligence, machine learning, and data mining.