Visual Perception and Mixed-Initiative Interaction for Assisted Visualization Design

Christopher G. Healey, Sarat Kocherlakota, Vivek Rao, Reshma Mehta, and Robert St. Amant

***Abstract*— This paper describes the integration of perceptual guidelines from human vision with an AI-based mixed-initiative search strategy. The result is a *visualization assistant* called ViA, a system that collaborates with its users to identify perceptually salient visualizations for large, multidimensional datasets. ViA applies knowledge of low-level human vision to: (1) evaluate the effectiveness of a particular visualization for a given dataset and analysis tasks; and (2) rapidly direct its search towards new visualizations that are most likely to offer improvements over those seen to date. Context, domain expertise, and a high-level understanding of a dataset are critical to identifying effective visualizations. We apply a mixed-initiative strategy that allows ViA and its users to share their different strengths and continually improve ViA's understanding of a user's preferences.**

We visualize historical weather conditions to compare ViA's search strategy to exhaustive analysis, simulated annealing, and reactive tabu search, and to measure the improvement provided by mixed-initiative interaction. We also visualize intelligent agents competing in a simulated online auction to evaluate ViA's perceptual guidelines. Results from each study are positive, suggesting that ViA can construct high-quality visualizations for a range of real-world datasets.

***Index Terms*— computer graphics, perception, search, user interfaces, visualization**

I. Introduction

An important problem in computer graphics is visualization, the conversion of collections of strings and numbers (or datasets) into images that viewers can use to explore, discover, and analyze within their data [37], [46]. The rapid growth in our ability to generate, capture, and archive vast amounts of information has dramatically increased the need for effective visualization techniques. Unfortunately, methods to display information in useful and meaningful ways have not always kept pace.

Numerous research efforts are now underway to identify new visualization algorithms [27], [28], [53]. One promising approach is the use of guidelines from visual perception. The human visual system can detect certain image properties very rapidly, often in only a few hundred milliseconds. Visualization techniques that harness human perception have the potential to significantly increase data throughput and improve viewer comprehension. The need to consider perception during visualization was highlighted as an important research issue in the original NSF report on scientific visualization [37], and it continues to offer the potential to improve a wide range of visualization algorithms [27], [28], [53].

Our specific interest in this paper is the visualization of *multidimensional* datasets that encode multiple attributes. Consider a dataset D representing n data attributes $A = (A_1, \ldots, A_n)$, $n > 1$ and containing m data elements e_j, $D = (e_1, \ldots, e_m)$. Each element encodes one value for every attribute, $e_j = \{a_{j,1}, \ldots, a_{j,n}\}$, $a_{j,i} \in A_i$. One way to visualize D is to select n visual features $V = (V_1, \ldots, V_n)$ to represent each A_i. Functions $\Phi = (\phi_1, \ldots, \phi_n)$ map the domain of A_i to the range of displayable values in V_i, $\phi_i : A_i \mapsto V_i$. Described in this way, visualization is the selection of a data-feature mapping $M(V, \Phi)$, together with an evaluation of a viewer's ability to comprehend the images generated by M. Results from human psychophysics can be used to construct perceptually effective M that allow viewers to rapidly and accurately understand their data.

Fig. 1 presents a multidimensional visualization of a simulated supernova collapse, a massive explosion that occurs at the end of a star's lifetime. We visualize slices through the flow volume using nonphotorealistic brush strokes built from guidelines on perception and aesthetics. Stroke color represents flow magnitude (dark blue for low to bright pink for high), stroke orientation represents flow direction, and stroke size represents flow pressure (larger for higher). Anecdotal feedback from our astrophysics collaborators at North Carolina State University confirms that the visualizations provide important advantages, particularly during analysis of data attribute interactions [17], [52].

Since most users are not visualization researchers, they cannot be expected to know how to construct effective visualizations. Even if users have experience in building visual representations for their data, they often repeat the same basic design strategy. This can lead to a number of inefficiencies. For example, it is difficult to determine if the resulting visualization is of a high quality, or if there are simple ways it could be improved. Offering only a single visualization mapping can also limit data analysis. Providing a collection of visualization designs that show the same data in different ways will often provide new insights into important properties that exist in the dataset.

Our goal is to empower our users by providing access to existing visualization knowledge. To do this, we propose a *visualization assistant*, a mixed-initiative artificial intelligence search system that helps users construct visualizations for their data. We defined a number of requirements for this system, specifically, that its operation and resulting visualizations are:

- *effective:* the data-feature mappings that produce the visualizations must display data in ways that allows viewers to rapidly and accurately complete their analysis tasks,
- *multidimensional:* the resulting visualizations should be capable of representing datasets with multiple values encoded at each data element,
- *transparent:* viewers must be able to understand and guide the assistant (e.g. by changing the initial inputs, or through built-in interaction mechanisms) to impose constraints, to define preferences, or to maintain context specific to the data being visualized,
- *application independent:* the assistant should not depend on properties specific to a particular application area or dataset type, but instead should be capable of producing visualizations for a range of different domains, and

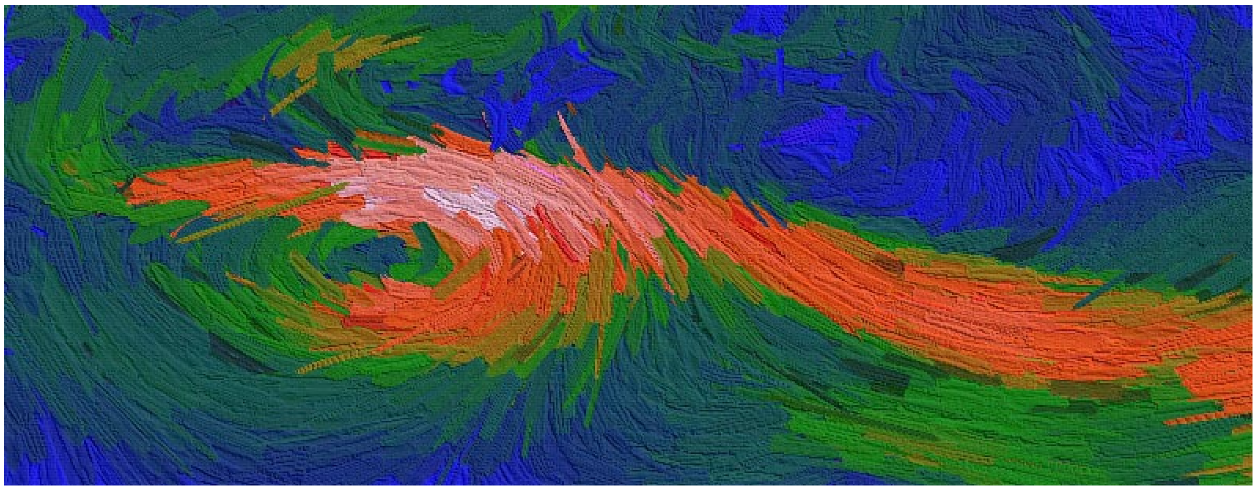


Fig. 1. A painterly visualization of 2D flow in a simulated supernova collapse, *flow direction* → stroke orientation, *magnitude* → color, *pressure* → size

- *extensible:* the assistant should easily extend to support new visualization results as they are discovered.

The key idea is to divide the visualization process between the computer and the end-user in a way that harnesses the unique strengths of each participant. Since most users are not visualization experts, details about how to construct and evaluate a multidimensional visualization are assigned to the visualization assistant. Users *are* experts on their data, however, so they are asked to provide information about the dataset and analysis tasks to be performed, to make decisions about how the data can be preprocessed, and to consider domain-specific constraints to enforce any visual context required in the final visualizations

II. Perceptual Foundations

Our visualizations are constructed from psychophysical studies of how the human visual system "sees" fundamental properties of color and texture in an image.

Color is a common visual feature used in many visualization designs. This belies its complex nature, however [60]. Simple color scales include the rainbow spectrum, the grey-red saturation scale, and red-blue or red-green ramps [55]. More sophisticated methods divide color along basic dimensions like luminance, hue, and saturation to better control the differences viewers perceive [32]. Perceptually balanced models like CIE LUV or Munsell roughly equate perceived color difference to Euclidean distance [8], [39]. Methods have been proposed to emphasize specific data ranges with non-linear paths through a color model [42], to choose color scales based on data attribute properties [45], to control color surround errors [55], or to pick small collections of distinguishable colors [16].

Our current color scales combine many of these findings [17]. We build paths along a monitor gamut's boundary in CIE LUV, either as a single spiral that monotonically increases in luminance, or as a single loop with a constant luminance. A path is subdivided into named regions that are parameterized to have equal arc length. The result is a color scale that: (1) is balanced along its length; (2) controls color surround errors; (3) can vary hue and luminance independently; and (4) can suggest small sets of equally distinguishable colors.

Like color, texture can be decomposed into a collection of fundamental perceptual dimensions. Mapping data attributes to texture properties produces texture patterns that change their visual appearance based on the data they represent [16], [26], [56], [58]. Texture dimensions have been identified both in computer vision and visual perception [7], [29], [35], [40], [41], [54], [59]. For example, automatic texture segmentation measure properties like size (or height), spatial packing density, orientation, and regularity. Results from computer vision cannot always be applied directly during visualization design, however, so care must be taken to confirm the visual system's ability to distinguish and identify individual texture properties [16], [25], [30], [56].

Visual properties that work well in isolation do not necessarily function with the same efficiency if they are shown together. For example, random changes in luminance can interfere with a viewer's ability to recognize hue patterns [6], [16], [50]. This interference effect is asymmetric: random variations in hue have no effect on a viewer's ability to perceive luminance. Both hue and luminance interfere with texture dimensions, again in an asymmetric manner: random hue or luminance masks texture properties, but random texture properties have no effect on hue or luminance [7], [16], [50]. Small but significant interference effects also occur between texture dimensions: density or regularity can mask small targets, and size or regularity can mask sparse targets. We do not want to map a low-relevance attribute to a high-salience visual feature, since this could obscure important data.

Our visualization designs focus on visual features that are easily recognized, both in isolation and in combination. We map individual data attributes to features in ways that draw a viewer's focus of attention to important areas in a visualization. The ability to harness low-level human vision is attractive, since:

- high-level exploration and analysis tasks are rapid and accurate, usually requiring 200 milliseconds or less to complete,
- the time to perform a task is independent of the number of elements in the display, and
- different features can interact with one another to mask information; psychophysical experiments allow us to identify and avoid these visual interference patterns [15]–[17].

III. Architecture

Numerous systems exist with extensive capabilities for the presentation of data, for example, Vis5D, AVS, or the Visualization Toolkit (vtk) [21], [48]. What these systems lack is a method of suggesting to users *how* to represent their data in ways that are best-suited to their specific analysis and exploration needs. This is exactly the problem that we are trying to address.

Some previous work in visualization has investigated automating the selection of a visual mapping M. Wehrend and Lewis built a classification system to describe visualization techniques in a domain-independent manner [57]; these classifications are used to try to suggest an appropriate M. A similar technique is described by Lohse et al. [33]. Robertson uses a natural scene paradigm to guide the choice of visual representations for data [43], [44]; this methodology identifies the types of information conveyed by a particular representation, then tries to match these to the underlying characteristics of a dataset. Mackinlay proposes automated methods that measure expressiveness and effectiveness to develop 2D graphical presentations [34]. Beshers and Feiner apply similar rules to build graph-based "worlds within worlds" to visualize multidimensional data [4]. Senay and Ignatius extend Mackinlay's work to 3D using a visualization system built on heuristic rules [49]. Gallop proposes data models and structures to classify visualizations [12]. Rogowitz and Treinish describe a rule-based visualization architecture for representing continuous surfaces [45], built on perceptual rules to guarantee that an n-fold increase in an attribute's value results in a perceptual n-fold increase in the visual presentation for that value. Bergman et al. [3] describe a colormap tool that uses system-generated and user-provided information about a dataset to limit a viewer's choice of color scales during visualization.

Unfortunately, the construction of the perceptual rules used by these systems is often left as work-in-progress. As well, the techniques include a number of potential limitations, for example: (1) only one M is recommended for each type of dataset; (2) the parameters used to categorize a dataset are relatively coarse, so many different D will map to the same M; and (3) there is no simple way to intelligently modify M to support context or user preferences. Design galleries [36] address the first limitation by converting input parameters to images via a mapping function; a set of images maximally dispersed from one another can be automatically identified, arranged, and displayed to provide an overview of how different inputs affect the resulting image. Although expressive, perceptual knowledge and expertise are still needed to select the "best" M for the user's visualization and exploration needs.

A. *ViA*

We propose an AI-based visualization assistant called ViA [18], [19], built with perceptual guidelines from human vision, heuristic AI search strategies, and mixed-initiative interactions. ViA collaborates with its users to design high quality, perceptually salient visualizations that are well-suited to the underlying data and analysis needs.

ViA begins by asking a short set of questions about the dataset and the user's analysis tasks. These initial constraints allow ViA to evaluate the applicability of different visualization mappings M. Users can modify the constraints, either through mixed-initiative interaction during the search process, or after a set of visualizations are proposed by ViA. Once the dataset properties and analysis tasks are defined, ViA begins constructing and testing visualizations. A potential mapping M is decomposed into its n data attribute-to-visual feature pairs (A_i, V_i). For each pair, an *evaluation engine* assesses the use of visual feature V_i. The engine returns a normalized evaluation weight to rate the effectiveness of the pairing. For pairings with low weights, an engine may also return one or more *hints* on how the pairing could be improved, together with an estimated evaluation weight increase if the hint were applied.

The simple strategy of exhaustively searching the state space of all possible data-feature mappings for M with the largest evaluation weight quickly becomes infeasible, even for small numbers of data attributes and visual features. Allowing users to add, remove, or modify their initial inputs while ViA runs changes the evaluation constraints, further increasing the number of different visualizations that must be considered.

Rather than applying a brute-force approach, ViA tries to restrict its searches to locations that are most likely to contain high-quality mappings. The search algorithm collects weights and hints for all the attribute-feature pairs in M. Chains of non-conflicting hints are bundled with M and placed on a priority queue in order of estimated evaluation weight improvement. The chain with the largest expected improvement is then removed from the queue and applied to M to form a new mapping M'. M' is evaluated in an identical manner, producing new hints that may lead to even better visualizations. This allows the search engine to focus on mappings that have a high probability of representing better visualizations. Searching continues until the queue is empty, or until a user-specified stopping conditions is reached.

B. *User Input*

Users are asked to enter a small amount of application-independent information prior to initiating a search. These inputs are used as an initial set of constraints when candidate visualizations are evaluated. For each data attribute, the user defines:

- *attribute importance:* a normalized importance weight, to order the attributes and to identify which attributes are most important to the user,
- *spatial frequency:* the spatial frequency of the attribute's values (high or low); an initial guess is made by ViA, and can be accepted or modified by the user,
- *continuous or discrete:* whether the data represents discrete values, or samples from an underlying, continuous data source [1], and
- *task:* the analysis tasks, if any, the user wants to perform on the attribute; ViA supports searching for a specific value (*search*), identifying spatial boundaries between regions with common values (*boundary detect*), estimating the number or ratio of data with a particular value (*estimate*), and tracking regions with common values as they move over time (*track*).

These properties were derived in part from our own experiences in decomposing domain-specific analysis requests into fundamental analysis tasks, and from existing automated visualization systems and task analysis research, which use many of these same properties to drive their visualization selection strategies.

C. *Evaluation Engines*

Evaluation engines are the basic building blocks that determine the quality of a given visualization M. The evaluation is based in part on M's perceptual strengths and limitations, and in part on the data being visualized, the user's stated interests in the data, and the analysis tasks the user wants to perform.

[1]ViA currently supports numeric data only, so nominal data is not considered

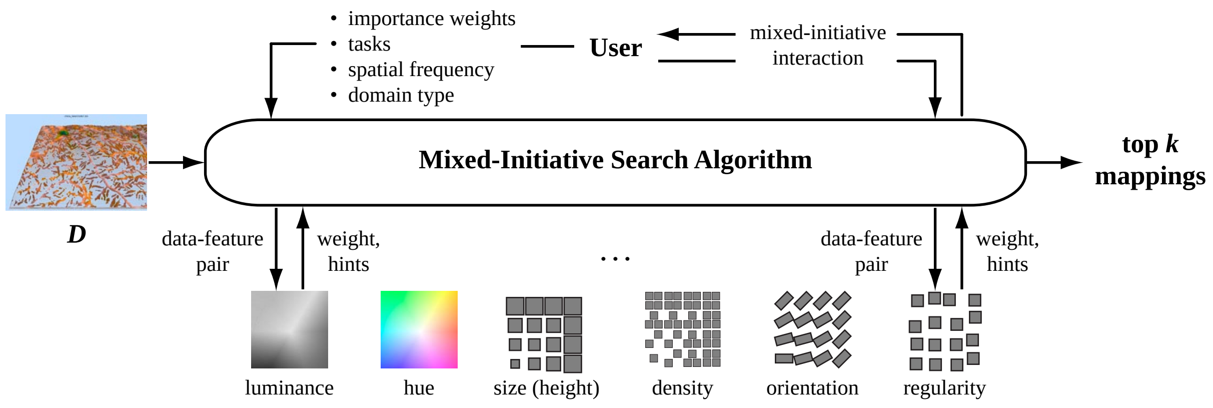


Fig. 2. ViA's architecture, made up of a dataset D, initial inputs from a user, a mixed-initiative search algorithm, and visual feature evaluation engines

Evaluation engines are categorized by visual feature. ViA currently includes engines for luminance, hue, size, density, orientation, and regularity. This design makes it easy to extend ViA to include new visual features. Once sufficient perceptual knowledge is available, a new engine can be implemented and integrated directly into ViA's search algorithms.

An evaluation engine performs four different classes of tests for each attribute-feature pair (A_i, V_i):

1) *Spatial frequency:* Certain visual features are best applied to either high or low spatial frequency data (e.g. luminance is appropriate for high spatial frequency data, while isoluminant hues are better suited for low spatial frequency patterns). ViA checks A_i's spatial frequency against what V_i can best support.
2) *Interference:* Visual features can interact with one another, causing visual interference patterns (e.g. luminance can mask hue patterns, and luminance and hue can mask texture patterns). Features that lie near the top of the visual salience hierarchy should not be used to represent data attributes with low importance. ViA searches M for cases where less salient visual features V_j are mapped to more important data attributes A_j, that is, $V_j < V_i$ but $imp_j > imp_i$.
3) *Task type:* Different visual features are best-suited for different types of analysis tasks. ViA checks to see which tasks a user has asked to perform on attribute A_i, to determine whether V_i can support the given task fully, partially, or not at all.
4) *Attribute type:* Different visual features work better with either continuous or discrete data. ViA compares V_i's capabilities to A_i's domain, and to the total number of unique values of A_i contained in D.

Each test is weighted evenly, comprising 25% of the total evaluation score. Their sum is returned as the final evaluation weight for the current (A_i, V_i) pairing in M. Alg. 1 shows a pseudocode overview describing how the orientation evaluation engine operates. Although the specifics for number of recommended values, maximum number of allowed values, invalid task and domain pairs, weight penalties, and so on vary between engines, their basic structure is identical.

We considered different ways to penalize flaws in a visualization mapping, including some that were fairly complicated. In the end, we decided to begin with a simple set of weight reductions, for example, 1 or 0 for supporting a particular constraint, or a liner increase in a penalty as the mapping moves away from a known optimal value. This strategy was selected as a starting point for two reasons. First, past research in artificial intelligence has show that very simple weighting schemes for automated search can often produce results that are as good as more complicated techniques [9]. Second, choosing basic, recognizable values makes it easier to critique and correct the performance of each evaluation engine.

Results to date for real-world datasets have been positive, to the extent that although we are considering minor improvements to the evaluation engines' weighting schemes, we are not planning wholesale changes to the underlying strategies they employ. As long as the existing approach continues to return good results, we intend to leave the evaluation framework intact.

D. Hints

Weights allow the search algorithm to compare different visualizations, but they do not offer any clues on how to improve them. Since the evaluation engines have the specific knowledge needed to provide this information, they are also responsible for proposing possible improvements for each (A_i, V_i) they test. These suggestions are returned in the form of hints, a specific modification to M together with an estimate of how much M will improve if the modification is applied. The lower the evaluation weight for a particular attribute-feature pair, the more hints the evaluation engine is likely to return.

Four types of modifications can be suggested within a hint:

1) *Feature swap:* Swap features V_i and V_j (e.g. in a situation where visual interference is occurring between V_i and V_j).
2) *Importance weight modify:* Increase or decrease the importance weight of attribute A_i by a set amount.
3) *Discretize:* Bin A_i's values into a fixed number of equal-width intervals.
4) *Task remove:* Remove a task a user has stated he may want to perform on A_i.

A hint's expected improvement weight is calculated directly from the penalty for the flaw it is meant to correct. For example, a hint by the orientation evaluation engine to reduce the number of unique values from above the maximum allowed to below the maximum recommended would have an estimated weight improvement of 0.25 (i.e. it will increase `domain_wt` from 0 to 1, improving the overall evaluation weight by $\frac{1}{4}$, see also Alg. 1).

E. Search Algorithm

Once hints are collected, they are bundled together to form *hint chains*. Not all combinations of hints are valid. Each evaluation

```
Input: Attribute $A_i$ mapped to orientation
Output: Evaluation weight, hints for ($A_i$,orientation)
1  if domain is continuous then domain_wt = 1.0;
2  else if unique vals $\leq$ max recommend then domain_wt = 1.0;
3  else
4      if unique vals $>$ max allow then  domain_wt = 0.0;
5      else  domain_wt = 1.0 - ( unique vals / max allow );
6      hint to discretize to max recommend values;
7      forall lower salience features F do
8          if unique vals $\leq$ max recommend for F then
9              hint to swap features to F;
10 freq_wt = 1.0;
11 interfere_wt = 1.0;
12 forall V $\in$ (Color, Luminance) do
13     if another A uses V && A's importance $<$ importance
       then
14         interfere_wt = 0.0;
15         hint to swap features with attribute A;
16         diff = | importance - A's importance |;
17         if diff $<$ min importance diff then
18             hint to make importance -= diff;
19             hint to make A's importance += diff;
20 task_wt = 1.0;
21 forall T $\in$ (estimate/continuous, search/continuous ) do
22     if task and domain match T then
23         if T is (estimate/continuous) then task_wt = 0.25;
24         else if T is (search/continuous) then task_wt = 0.5;
25         if importance $<$ min importance wt then
26             hint to remove task T;
27 wt = ( domain_wt + freq_wt + interfere_wt + task_wt ) / 4.0;
28 if wt $<$ min release wt then
29     forall attributes A not mapped to any feature do
30         hint to release feature to A;
31 return wt, hints
```

Alg 1. Pseudo-code for the orientation evaluation engine

engine works independently to improve its own attribute-feature pair, so different hints can conflict with one another. For example, one hint might suggest discretizing attribute A_i (discretize hint), while another might recommend swapping A_i to use a different visual feature (feature swap hint). The basic rule for identifying conflicting hints is straight-forward: a data attribute A_i or visual feature V_i can be modified by at most one hint in a chain. Hint chains that include conflicting hints are discarded.

Once valid hint chains for M are identified, they are inserted onto a priority queue ordered by the expected improvement in M's weight. The expected improvement is defined as the sum of M's current weight plus the largest expected improvement in the chain (i.e. the expected improvement for the hint with the largest improvement weight), together with a bonus for the length of the chain. Longer chains are favored over shorter ones, since these have a higher potential to improve M's overall evaluation weight. Moreover, a longer chain moves farther from M's current position in the search space, helping to escape from local maxima.

The search algorithm proceeds by removing the hint chain at the top of the queue and applying it to the mapping M that generated the chain's hints. This produces a new visualization M' that is likely to have a better evaluation weight than M. If ViA has not seen M' previously, it is evaluated, generating new hints and hint chains that are added to the queue. This allows the search algorithm to restrict its efforts to small areas of the much larger search space of all possible visualizations. ViA focuses on visualizations that are expected to have the highest evaluation weights. The search continues until: (1) the priority queue is empty; (2) a user-specified number of visualizations are evaluated; or (3) a user-specified period of time passes.

Once the search is complete, ViA presents the top k visualizations. The mappings can be applied directly to D, allowing users to view and explore their data with each candidate visualization. The ability to show the same dataset in different ways is one of ViA's important strengths. Even when each proposed visualization has a high evaluation weight, displaying the data from different perspectives will often highlight different areas of interest to the user. ViA removes the need to manually construct multiple visualizations, a task that can be difficult when both perceptual and dataset constraints must be considered. It also avoids the tendency of a visualization expert to design visualizations using common procedures or templates, producing visualizations with similar structures and therefore potentially less power to display the data in significantly different ways.

IV. Mixed-Initiative Interaction

Researchers in human-computer interaction have pursued numerous important goals. One area investigates intelligent agents that try to automatically identify and complete a user's tasks. Another studies interfaces for direct manipulation. A third approach integrates these techniques, combining automated services and user control to form mixed-initiative interaction [1].

Mixed-initiative interaction allows participants to contribute their unique strengths towards solving a common goal [5], [10], [38]. The basic idea is that initiative (i.e. control) should shift depending on who is most qualified to solve the current step of a problem. Humans have knowledge of high-level goals, and are guided by principles that are often difficult or impossible to fully automate. A computer excels at managing low-level details and performing repetitive tasks. Humans can formulate and plan, collect and evaluate relevant information, supply estimates for uncertain factors, and perform visual and spatial reasoning. Computers can rapidly conduct systematic searches, and manage and communicate large volumes of data. Mixed-initiative algorithms try to minimize the number of explicit actions required of the user. They also focus on asking the user to perform high-value actions, where answers provide additional knowledge for automatically handling future decisions.

A number of mixed-initiative systems have been presented in the literature. Lookout extends Microsoft Outlook by scanning new email messages and collaborating to schedule appointment requests [23]. TRAINS95 constructs optimal transportation routes in the presence of uncertainty (e.g. changing weather and traffic conditions, unexpected events, and so on) [11]. AIDE (Assistant for Intelligent Data Exploration) applies AI planning techniques to help users analyze univariate and bivariate statistical relationships [51]. Design-A-Plant uses an animated pedagogical agent to teach botanical anatomy and physiology to middle school students [31].

One important consideration is how to manage uncertainty about a user's goals during problem solving. A common solution is to use Bayesian agents to model goals and construct utility measures based on probabilistic relationships [22]. Combining statistical models of user goals with the expected utility of action or inaction forms a critical component of a mixed-initiative system [23].

A. Mixed-Initiative Search

If a user's initial inputs were fixed, or if ViA's evaluation weights were insensitive to changes in these inputs, the ability to support modifications as the search unfolds would not be necessary. Unfortunately, these assumptions do not hold. Many of ViA's hints (e.g. discretize, importance weight modify, or task remove) require explicit changes to the user's initial constraints or to the data itself. These hints represent situations where ViA suggests: *If* a change can be made to the format of the data or the users' choices about what they want to do with the data, *then* a significantly better visualization may be available. The question becomes, should ViA accept the change a hint recommends?

Certain considerations can be used to help with this decision. Larger changes to the data or the user's inputs are less likely to be allowed. The probability of a hint being accepted may be higher if its expected improvement weight is larger. Finally, the way a hint was managed in the past can provide clues about future decisions. For example, if ViA was previously told not to discretize an attribute, new discretize requests are less likely to be accepted.

More formally, consider a situation where the user has a desired goal state G, and a hint suggests performing a given operation O (e.g. discretizing an attribute, or changing its importance weight). One simple way to resolve the hint is to ask the user each time a change is desired. This often produces a long sequence of requests that quickly lead to situations where users answer without significant consideration (e.g. always answering "Yes" or "No"). Allowing ViA to decide whether or not to apply O has its own drawbacks. Even with sophisticated heuristics, ViA cannot correctly anticipate how its users want to proceed for every operation. A fully automatic approach does not allow ViA to benefit from a user's expertise about the data.

Instead of relying on either extreme, we extend a probabilistic utility model proposed by Horowitz to guide ViA's actions [23]. For any operation O and goal state G, there are four possible actions: applying O when the goal is G, applying O when the goal is not G, not applying O when the goal is G, and not applying O when the goal is not G. We denote the utility of these actions $u(O,G)$, $u(O,!G)$, $u(!O,G)$, and $u(!O,!G)$. If we assume O is designed to facilitate achieving G, then $u(O,G) > u(!O,G)$ and $u(!O,!G) > u(O,!G)$. Given evidence to date E observed by ViA, the probability that G is a valid goal state is denoted $p(G|E)$. We can now compute the expected utility for accepting O:

$$\begin{aligned} eu(O|E) &= p(G|E)u(O,G) + p(!G|E)u(O,!G) \\ &= p(G|E)u(O,G) + (1-p(G|E))u(O,!G) \end{aligned} \tag{1}$$

Similarly, the expected utility for rejecting O is:

$$eu(!O|E) = p(G|E)u(!O,G) + (1-p(G|E))u(!O,!G) \tag{2}$$

By comparing $eu(O|E)$ and $eu(!O|E)$, we can determine whether accepting or rejecting O produces a higher expected utility. Fig. 3 graphs the two utility curves $u(O)$ and $u(!O)$, and identifies a break-even probability p^* where either accepting or rejecting O leads to the same expected utility. For all $p(G|E) < p^*$, we automatically reject O, and for all $p(G|E) \geq p^*$ we can automatically accept O.

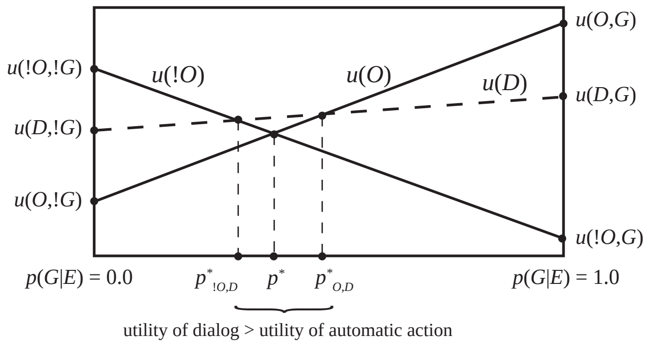


Fig. 3. Utility curves for automatically accepting operation O, for automatically rejecting $!O$, and for querying the user through a dialog D

A final factor to consider is the value of interacting with the user. If G is valid, the utility of asking the user and then applying O is higher than automatically rejecting O (since this incorrectly defines G as not valid), but lower than automatically applying O (since additional work is required by the user to validate G). That is, given a dialog action D, $u(!O,G) < u(D,G) < u(O,G)$. Similarly, when G is not a valid goal state, $u(O,!G) < u(D,!G) < u(!O,!G)$. The utility of presenting a dialog $u(D)$ can be added to the probability graph (Fig. 3). This defines a probability region where $eu(D|E)$ is higher than both $eu(O|E)$ and $eu(!O|E)$. In this region the best choice is to ask the user how to proceed. Given the intersection points between $u(!O)$ and $u(D)$ at $p^*_{!O,D}$ and between $u(O)$ and $u(D)$ at $p^*_{O,D}$, we can now define ViA's actions as:

1) $p(G|E) \leq p^*_{!O,D}$, automatically reject O
2) $p(G|E) \geq p^*_{O,D}$, automatically accept O
3) $p^*_{!O,D} < p(G|E) < p^*_{O,D}$, ask the user about O

When a dialog is presented, the user's answer is important for more than simply deciding about O. The choice changes E, the evidence observed about the probability of G, allowing us to infer more information about $p(G|E)$.

B. ViA's Expected Utility Graph

In order to evaluate a hint to perform operation O, we need the six utility values $u(O,G)$, $u(O,!G)$, $u(!O,G)$, $u(!O,!G)$, $u(D,G)$, and $u(D,!G)$. For a visualization M with an overall weight w, and a hint to perform operation O with an expected improvement w_O, the four utilities for accepting or rejecting O are defined as:

- $u(O,G) = w + w_O$, applying a hint the user wants increases utility by O's expected improvement weight.
- $u(O,!G) = 0$, applying a hint the user does not want reduces utility to zero.
- $u(!O,G) = w - w_O$, not applying a hint the user wants reduces utility by the improvement that O could have provided.
- $u(!O,!G) = w$, not applying a hint the user does not want provides no improvement or penalty.

Two additional values are used to determine the expected utility of presenting a dialog to the user. A *conservation factor* $0 \leq f \leq 1$ defines a user's level of concern about allowing ViA to make

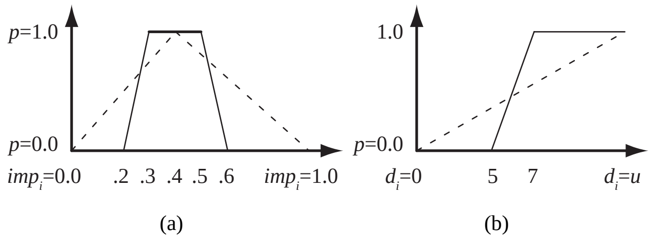


Fig. 4. Probability graphs for ViA's hints: (a) *importance weight modify*, initial graph (dashed) with $imp_i = 0.4$, updated graph (solid) after accepting $imp'_i = 0.3$ and $imp'_i = 0.5$, but rejecting $imp'_i = 0.2$ and $imp'_i = 0.6$; (b) *discretize*, initial graph (dashed) with u unique values, updated graph (solid) after accepting $d = 7$ ranges, but rejecting $d = 5$ ranges

decisions without consultation. A value $f = 1$ indicates the user is very conservative, and wants to review all decisions ViA makes. The importance weight imp_i of A_i is also considered. Operations on more important attributes are more likely to generate requests to the user. Given these additional weights, the two utilities for presenting a dialog D are defined as:

- $u(D, G) = w + (f \times imp_i \times w_O)$, presenting a dialog for a hint the user wants increases utility to between w and $w + (imp_i \times w_O)$, depending on f and imp_i.
- $u(D, !G) = f \times imp_i \times w_O$, presenting a dialog for a hint the user does not want decreases utility to between 0 to $imp_i \times w_O$.

The utility values allow us to build utility graphs that define $p^*_{!O,D}$ and $p^*_{O,D}$. ViA automatically rejects O when $p(G|E) \leq p^*_{!O,D}$, automatically accepts O when $p(G|E) \geq p^*_{O,D}$, or queries the user when $p^*_{!O,D} < p(G|E) < p^*_{O,D}$.

C. Conditional Probability Graphs

The final value needed to process a hint is $p(G|E)$, the conditional probability that users want goal G based on their previous interactions with ViA. Initial probability graphs are built for each attribute A_i for the hints importance weight modify, discretize, and task remove. Feature swap hints can be managed automatically, since they depend on perceptual rules alone.

Importance Weight Modify. An importance weight modify hint requests to change A_i's importance weight from imp_i to imp'_i, for example, to remove visual interference between attributes A_i and A_j by reversing their importance ordering.

Given A_i with importance weight imp_i, the initial importance weight probability graph assigns $p(G|E) = 1$ for weight imp_i, and $p(G|E) = 0$ for weights 0 and 1. Linear interpolation is used to compute probabilities between these three points. A user's responses to importance weight modify hints (accept or reject) update the probability graph. For example, consider A_i with $imp_i = 0.4$ (Fig. 4a). Suppose ViA suggests changing imp_i first to 0.3 and then to 0.5, both of which the user accepts, then ViA suggests weights of 0.2 and 0.6, both of which the user rejects. The updated probability graph is shown in Fig. 4a. All $0.3 \leq imp_i \leq 0.5$ are now assumed to be accepted, and all $imp_i \leq 0.2$ and $imp_i \geq 0.6$ are assumed to be rejected. Linear interpolation is used to estimate probabilities over the unknown regions $0.2 < imp_i < 0.3$ and $0.5 < imp_i < 0.6$.

Discretize. A discretize hint requests that A_i be divided into d equal-width ranges, for example, to reduce A_i to a distinguishable number of unique values.

A_i's initial discretization probability graph assigns $p(G|E) = 1$ for no discretization (i.e. for any hint with $d \geq u$), and $p(G|E) = 0$ for a discretization of $d = 0$. Linear interpolation is used to compute probabilities between these endpoints. A user's responses to discretize hints (accept or reject) update the probability graph. Consider A_i with $u > 7$ unique values (Fig. 4b). Suppose ViA suggests discretizing A_i to $d = 7$ values, which the user accepts, then ViA suggests discretizing to $d = 5$ values, which the user rejects. The updated probability graph is shown in Fig. 4b. All $d \geq 7$ are now assumed to be accepted, and all $d \leq 5$ are assumed to be rejected. Linear interpolation is used to estimate probabilities over the unknown region $5 \leq d \leq 7$ (i.e. $p(G|E) = 0.5$ for $d = 6$).

Task Remove. A task removal hint requests that a task assigned to A_i be removed from consideration. This hint is generated when visual feature V_i representing A_i cannot support the given task.

Task removal operations are binary in nature, since there are no "parameter values" attached to the request. An initial probability $p(G|E) = 1 - imp_i$ is used to estimate how likely a user is to accept the hint. A single query fixes the task removal probability. If a user allows the task removal, the probability is set to $p(G|E) = 1$. If a user rejects the task removal, the probability is set to $p(G|E) = 0$.

V. Evaluation

In order to evaluate ViA's performance, we considered three separate components. First, we measured ViA's ability to locate the best visualization mappings, compared to an exhaustive search of all possible visualizations, and to the common simulated annealing and tabu search algorithms. This provides evidence that ViA's hint-based search algorithm can locate high-quality visualizations, hopefully in significantly fewer steps than are required for exhaustive search.

Second, we studied how adding mixed-initiative interaction affects the quality of the visualizations ViA recommends, and how well it buffers a user from having to answer repeated queries about how the search should proceed.

Both the search and mixed-initiative investigations were conducted using a real dataset containing historical weather conditions. We concluded by applying ViA to a very different kind of dataset representing artificial intelligence agents competing in a simulated e-commerce auction environment. Researchers from this project used ViA's visualizations to identify important new strategies the agents employed during the competition.

A. Search Performance

We began by comparing ViA's search capabilities to an exhaustive search of all possible visualization mappings, and to two common artificial intelligence search algorithms: simulated annealing (SA) and reactive tabu search (RTS). Three metrics were calculated to measure performance:

1) *Optimality:* the evaluation weight of the best mapping found by an algorithm.
2) *Efficiency:* the number of visualization mappings evaluated by an algorithm to find the first optimal mapping.
3) *Completeness:* the number of optimal mappings an algorithm finds, relative to the total number of mappings with the maximum evaluation weight.

The number of visualizations in a search space depends on the number of data attributes n and the number of available visual

TABLE I
WEATHER DATASET ATTRIBUTES AND ASSOCIATED PROPERTIES

Attr	Domain	Freq	Task	Impt
temperature	discrete ($u = 7$)	high	search	1.0
wind speed	discrete ($u = 23$)	low	boundary	0.75
pressure	continuous	low	boundary	0.15
precipitation	discrete ($u = 82$)	high	search	0.75

features v. Given $n \geq v$, the total number of visualization mappings is $\binom{n}{v}v!$. If users modify their initial constraints, the number of states will increase rapidly, since a change to A_i means that any visualization mapping containing A_i may evaluate to a new weight and set of hints.

We used a meteorological dataset containing weather conditions collected and averaged by the Intergovernmental Panel on Climate Change (IPCC) for the years 1961 to 1990[2]. Monthly averages for eleven separate weather conditions are provided at $\frac{1}{2}^\circ$ latitude by $\frac{1}{2}^\circ$ longitude steps for positive elevations throughout the world.

In order to limit the size of the search space during exhaustive searching, we focused on $n = 4$ data attributes: mean *temperature*, *wind speed*, *pressure*, and *precipitation* over the continental United States. Table I summarizes the attributes and the initial inputs that were provided to ViA. We allowed ViA to consider using $v = 4$ visual features to represent the weather data: color, density, orientation, and height. We also pre-selected how users respond to each type of hint for each attribute. This allowed us to fully specify a search space with 1,680 nodes containing all possible combinations of visualization mappings and user inputs.

Simulated Annealing. Simulated annealing (SA) is a heuristic technique that applies a probabilistic model to iterate towards a global optimum in a large search space. SA has been compared to cooling metals to a minimum crystalline structure through an annealing process, where the metal is set to a high temperature and slowly cooled to maintain an approximate thermodynamic equilibrium. If the energy function of the physical system is replaced by an objective function, the progression towards a ground state represents iterating towards a global optimum. SA performs gradient descent by picking a random move, then allowing the move if it improves the current state, or allowing it with probability less than 1 if it does not. This probability is lower when a move produces a larger decrease in the current state, and when the current temperature of the system is lower. As with the annealing process, avoiding local maxima depends on the choice of the initial and final temperatures, and the change in temperature as each cooling step is applied [14], [47].

SA begins with an initial data-feature mapping M that evaluates to weight w. All permissible operations on M (e.g. discretize, feature swap, and so on) are generated, with one being randomly selected and applied to produce M'. If M' has a higher evaluation weight $w' > w$, it is automatically accepted, otherwise it is accepted or rejected with a probability based on $w - w'$ and T, a monotonically decreasing current temperature of the system. This process continues until a final system temperature is reached. Mappings located during the search with the highest evaluation weights are then returned by the system.

Reactive Tabu Search. One drawback of SA is that, due to the random selection of operators, it can become trapped in a cycle. Tabu search (TS) allocates additional memory to remember previous states, allowing it to avoid this problem [13], [47]. At each iteration in the search process, the most recently visited nodes are marked as tabu, and are not considered when making the next move. The size of the recently visited node list affects the algorithm's performance. Strict-TS treats any node ever visited as tabu. This leads to slow convergence, however, since previously visited nodes act as barriers to improved areas of the search space. Fixed-TS treats the last p nodes visited as tabu. p must be chosen to be large enough to avoid cycles, but small enough not to overconstrain the search. Reactive-TS (RTS) varies p dynamically [2]. RTS maintains a separate, long-term memory of all nodes visited. Whenever a node is repeated, the tabu list size is increased. A separate, slower mechanism reduces the size of the list when new nodes are located. If the number of node repetitions becomes significant, a diversification stage is applied to escape the local maxima by making a random walk to a different region of the search space.

As with SA, RTS begins with an initial data-feature mapping M that evaluates to weight w. The node containing M is placed on the tabu list, and all permissible operations on M are generated. The move that produces M' with the largest evaluation weight w' is applied. RTS continues searching local non-tabu mappings, diversifying as necessary, until a stopping condition is reached. Mappings located during the search with the highest evaluation weights are returned by the system.

Performance Results. Given the meteorological dataset, initial user inputs, and allowed modifications to the inputs, an exhaustive search of all 1,680 nodes identified 21 optimal mappings, each with an evaluation weight of $w = 0.844$. We ran ViA's hint-based search, simulated annealing, and reactive tabu search to compare their performance to this complete evaluation.

How each search proceeds depends in part on its starting position within the state space. By default, ViA builds an initial data-feature mapping at starting position S_0 by sorting the data attributes by importance weight, then assigning visual features in order of perceptual salience. This maps the most salient visual features to the most important data attributes, and produces good results in practice. In addition to S_0, we selected five additional starting positions $S_1, \ldots, S_5$ that were distributed throughout the state space, and that had both high and low evaluation weights (i.e. represented both strong and weak initial visualizations). For each S_i, an algorithm was allowed to perform 200 evaluations, and was then asked to return the best mappings that it found.

Hint-based search runs in a deterministic fashion. From a given starting point, it will always evaluate the same states in the same order. SA and RTS have random components, however, so they can visit different nodes for a given starting position and fixed number of evaluation steps. To address this, we ran both algorithms from each S_i until the variance between runs fell below a pre-selected confidence threshold, then averaged the results to produce an overall estimate of the algorithm's performance.

We started by calculating the number of evaluations required to find the first optimal mapping. Averaged over all six starting positions, hint-based search needed to perform 63.2 evaluations, ranging from a low of 4 evaluations for starting position S_5 to a high of 120 evaluations for S_1 (Table II). SA needed 85.8 evaluations, ranging from an average low of 74 evaluations for S_3 to an average high of 98 evaluations for S_0. RTS needed 84.2 evaluations, ranging from an average low of 14 evaluations for S_3 to an average high of 190 evaluations for S_4.

[2]`http://ingrid.ldeo.columbia.edu/SOURCES/.UEA/.CRU/.New/.CRU05/.climatology/.c6190/`

TABLE II
RESULTS FOR EXHAUSTIVE SEARCH VERSUS VIA'S HINT-BASED HEURISTIC, SIMULATED ANNEALING, AND REACTIVE TABU SEARCH

	First Optimal							**Total Optimal**						
	S_0	S_1	S_2	S_3	S_4	S_5	$\overline{S}$	S_0	S_1	S_2	S_3	S_4	S_5	$\overline{S}$
Full	—	—	—	—	—	—	—	21	21	21	21	21	21	21
Hint	76	120	102	$>$200	14	4	63.2	3	5	3	0	4	2	2.83
SA	94	98	80	74	90	79	85.8	3	4	4	4	3	4	3.83
RTS	68	36	44	14	190	153	84.2	16	17	17	17	6	6	13.17

Interestingly, from starting position S_3 hint-based search was unable to locate an optimal result during its 200 allowed evaluations. The best mapping it found had an evaluation weight 98% of the global maximum. The hint-based search arrived at the neighbor of an optimal node, but never moved to the optimal node itself. Moves are dictated by the hints the evaluation engines return, and none of the hints suggested pushed the search onto the optimal node.

Simulated annealing also failed to find optimal nodes during some (but not all) of its runs for all six starting positions. In every case, however, SA was able to locate mappings that had an evaluation weight at least 97% of the global maximum. The random nature of how SA selects its moves caused it to miss nearby optimal mappings during approximately 23% of its searches.

We also measured how many of the 21 optimal mappings each algorithm identified. Hint-based search located, on average, 2.83 optimal mappings (ranging from a low of zero mappings for starting position S_3 to a high of five mappings for S_1), SA located, on average, 3.83 optimal mappings (ranging from a low of three mappings for S_4 to a high of four mappings for S_0, S_1, S_2, S_3, and S_5), and RTS located, on average, 13.17 optimal mappings (ranging from a low of six mappings for S_4 and S_5 to a high of 17 mappings for S_1, S_2, and S_3).

RTS identified more optimal mappings, compared to the hint-based and SA searches. This algorithm did particularly well due to its exhaustive evaluation of local regions of the search space (the intensification stage) before moving on to new locations (the diversification stage). The space of visualizations we searched during our testing contained 12 optimal mappings clustered near one another. RTS was able to quickly identify all 12 mappings from four of the six starting positions. In the other two cases RTS did not diversify fully into this region. It was still able to identify six optimal mappings, however, more than the best results for both hint-based and SA search.

Although RTS may find more optimal mappings when the mappings cluster together, these mappings will, by definition, be similar to one another. This makes them potentially less likely to highlight different aspects of the dataset, since they visualize data in very similar ways. Hint-based search does a better job of spreading its evaluations throughout the search space by following promising hint chains to diversify quickly into different local regions. This has the potential to return optimal mappings that are very different from one another, although at the expense of a reduction in the total number of optimal mappings identified.

None of the search algorithms produced poor results. ViA's hint-based search completed its 200 evaluations in less than one second, although more time would be needed if more attributes were included (e.g. approximately 10 seconds for six or more attributes). There is room for improvement, however, mostly in guaranteeing that at least some optimal mappings are located in a timely fashion. Based on our results, we believe a combination of hint-based diversification between regions and RTS intensification within a region could improve our results. This algorithm is being implemented as part of our future work.

B. Mixed-Initiative Interaction Performance

We continued our investigations by studying ViA's recommendations, both with and without mixed-initiative interaction, for visualizing the same mean *temperature*, *wind speed*, *pressure*, and *precipitation* attributes from the IPCC weather dataset (Table I).

ViA was executed in three separate modes. In the first, no user interactions were conducted except for the initial input of data properties and analysis task requirements. We denote this version of ViA with no interaction ViA-N. Fixed cutoffs were applied to automatically determine whether to accept or reject hints from the evaluation engines:

- *feature swap:* automatically allowed, since these hints are perceptually based and do not change the user's initial inputs,
- *importance weight modify:* allowed if the difference between the new importance weight imp'_i and the original weight imp_i is no more than 0.15, $|\, imp_i - imp'_i \,| \leq 0.15$, otherwise rejected,
- *discretize:* allowed if the number of discrete ranges d is no less than half the number of unique values u originally contained in A_i, $d \geq \frac{1}{2}u$, otherwise rejected, and
- *task remove:* allowed if the importance imp_i of A_i is 0.25 or less, $imp_i \leq 0.25$, otherwise rejected.

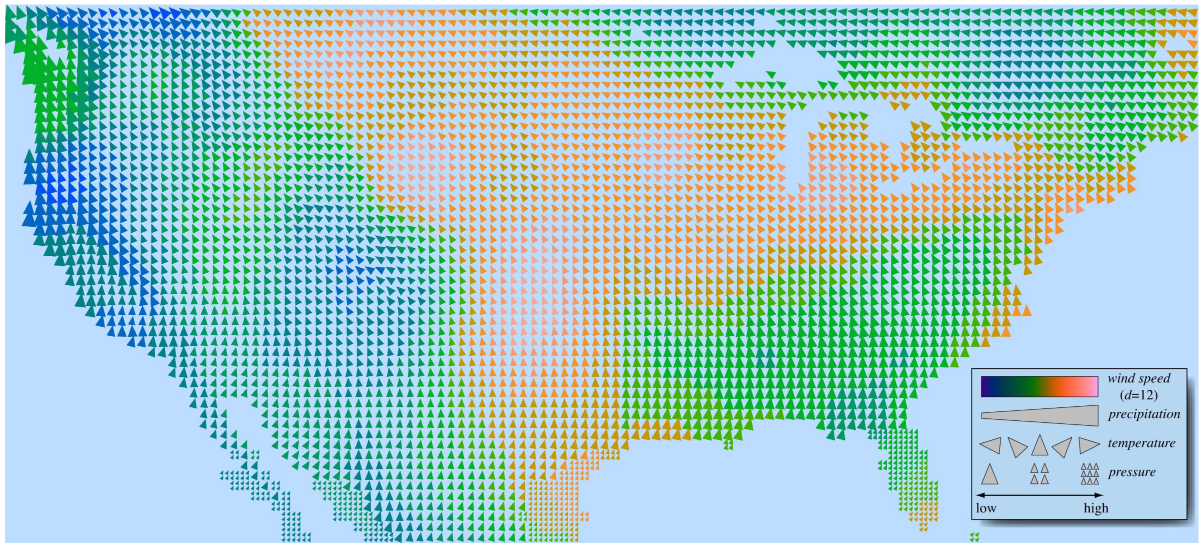
In the second mode, ViA runs with the mixed-initiative engine enabled. Decisions about whether to automatically accept or reject a hint, or to ask the user for advice, are controlled by expected utilities. We denote this version of ViA with mixed-initiative interaction ViA-MI.

Finally, we ran ViA with full, explicit interaction. The user is asked to accept or reject each importance weight modify, discretize, or task remove hint. Apart from avoiding duplicate queries, answers to past queries are not analyzed to try to infer future answers. We denote this version of ViA with complete interaction ViA-UI.

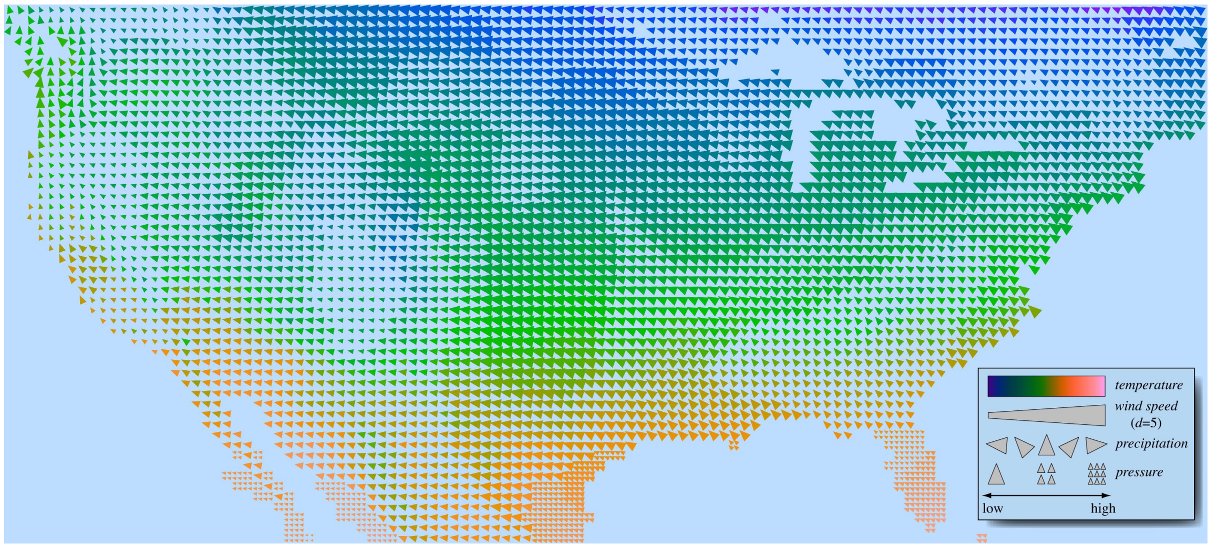
ViA-N versus ViA-MI. Our first experiments tested the abilities of ViA with no interaction versus ViA with a mixed-initiative interface. Four conditions were measured: evaluation weights of the best visualizations ViA found, the number of visualization mappings evaluated, the total number of possible mappings over the entire search space, and the number of queries made to the user. ViA-N evaluated 89 of 3,840 possible mappings (2.3%), returning the following mappings as the best it identified (Table III):

- $w = 0.84$, *temperature* $\rightarrow$ orientation; *wind speed* (discretized to $d = 12$ ranges) $\rightarrow$ hue; *precipitation* $\rightarrow$ size; *pressure* $\rightarrow$ coverage
- $w = 0.84$, *temperature* (discretized to $d = 5$ ranges) $\rightarrow$ size; *wind speed* (discretized to $d = 12$ ranges) $\rightarrow$ hue; *precipitation* $\rightarrow$ orientation; *pressure* $\rightarrow$ coverage

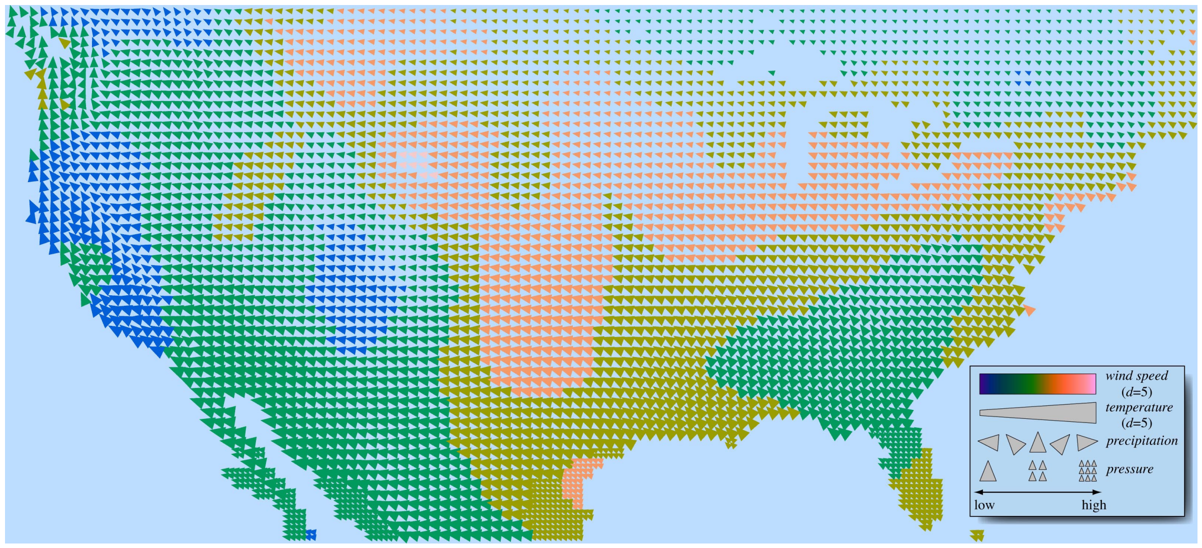
Fig. 5a shows the first ViA-N mapping used to visualize average weather conditions for the month of January. An area of high *wind speed* appears in the center of the country (orange and pink glyphs), while an area of high *precipitation* occurs in the Pacific Northwest (larger glyphs). Higher *temperature* and *pressure* are



(a)



(b)



(c)

Fig. 5. Average historical weather conditions for January over the continental United States: (a) ViA-N; (b,c) two different ViA-MI

TABLE III
SEARCHES WITH FIXED CUTOFFS (VIA-N), MIXED-INITIATIVE INTERACTION (VIA-MI), AND EXPLICIT USER INTERACTION (VIA-UI)

	Maximum w	**Queries Issued**	**Hints Accepted**
ViA-N	0.84	0	5
ViA-MI, first run	0.875	6	4
ViA-MI, second run	0.94	9	8
ViA-UI, always no	0.83	20	0
ViA-UI, fixed cutoffs	0.84	43	5
ViA-UI, always yes	0.94	2,324	2,324

visible in southern Texas and Florida (a denser packing of glyphs with a rightward orientation).

For ViA-MI, a total of six queries were made. The user answered "Yes" to the following request:

- Discretize *wind speed* to 5 values?

The user answered "No" to the following five requests:

- Discretize *temperature* to 5 values?
- Discretize *precipitation* to 7 values?
- Change the importance weight of *temperature* to 0.875?
- Change the importance weight of *wind speed* to 0.875?
- Change the importance weight of *precipitation* to 0.875?

Based on these responses, ViA-MI evaluated 49 of 1,920 possible mappings (2.6%). Even with this single allowed change, ViA-MI was able to identify better visualizations that ViA-N. The top mappings returned by ViA-MI were (Table III):

- $w = 0.875$, *temperature* $\rightarrow$ hue; *wind speed* (discretized to $d = 5$ ranges) $\rightarrow$ size; *precipitation* $\rightarrow$ orientation; *pressure* $\rightarrow$ coverage
- $w = 0.86$, *temperature* $\rightarrow$ size; *wind speed* (discretized to $d = 5$ ranges) $\rightarrow$ hue; *precipitation* $\rightarrow$ orientation; *pressure* $\rightarrow$ coverage

Fig. 5b shows the first ViA-MI mapping used to visualize January. The *temperature* patterns appear easier to identify with hue than with the orientations in Fig. 5a. The strong *wind speed* region in the center of the country is visible as larger glyphs. The region of high *precipitation* now appears as a set of glyphs with a strong rightward orientation.

We observed a number of interesting effects produced by enabling mixed initiative interactions. Discretizing *wind speed* to $d = 5$ ranges allowed both hue and size to visualize the attribute (hue supports up to seven distinguishable values, and size supports up to five values). Since ViA-N's fixed constraints do not allow discretizing an attribute to less than half its original number of unique values, size could not be used by ViA-N without some penalty. Discretizing *wind speed* also allowed ViA-MI to evaluate 30 new mappings that were not considered by ViA-N.

Because the user did not allow *temperature* to be discretized, ViA-MI did not consider 42 mappings that were evaluated by ViA-N. By not allowing any importance weight changes, ViA-MI eliminated an additional 40 mappings that ViA-N had to evaluate

Although ViA-MI was not allowed to discretize *temperature* below its initial $u = 7$ values, ViA-N did reduce it to $d = 5$ ranges. On the one hand, this is an example of ViA-MI using a query to identify and enforce a user's preferences. On the other hand, new visualizations of potential interest to the user were generated by reducing *temperature* to five ranges. To gain this benefit, users can run ViA multiple times and answer certain queries differently to explore how their preferences and constraints affect the types of visualizations ViA suggests.

In order to investigate this question of how user responses affect ViA-MI's performance, we re-ran ViA-MI, but answered the dialog queries in a slightly different manner. A total of nine queries were made. The user changed his answer from "No" to "Yes" for the following three requests:

- Discretize *temperature* to 5 values?
- Change the importance weight of *temperature* to 0.875?
- Change the importance weight of *wind speed* to 0.875?

Based on these responses, ViA-MI evaluated 343 of 30,720 possible mappings (1.1%), returning a visualization with an increased weight of $w = 0.94$ (Table III). The mapping assigned *temperature* (discretized to $d = 5$ ranges, and with importance weight reduced to 0.875) $\rightarrow$ size, *wind speed* (discretized to $d = 5$ ranges, and with importance weight increased to 0.875) $\rightarrow$ hue, *precipitation* $\rightarrow$ orientation, and *pressure* $\rightarrow$ coverage. The following hints were needed to improve upon the best mapping found during the first test of ViA-MI:

- size is better-suited to representing the high spatial frequency details in *temperature* than (isoluminant) hue, but in order to use size without penalty, it needed to be discretized to $d = 5$ ranges,
- hue can be used to represent (low spatial frequency) *wind speed* once it is discretized to $d \leq 7$ ranges, and
- the initial importance weights of 1.0 for *temperature* and 0.75 for *wind speed* would produce visual interference (lower importance *wind speed* represented with a more salient visual feature hue); to remove this, the importance weight for *temperature* is reduced to 0.875, and the importance weight for *wind speed* is increased to 0.875.

This demonstrates how ViA tries to use sequences of hints to iteratively improve its visualization recommendations. All three hints described above were needed to arrive at the final, high quality visualization ViA suggested.

Fig. 5c shows the new ViA-MI mapping used to visualize January. All of the weather patterns seen in the previous visualizations appear in this final image. One prominent feature of Fig. 5c is the small area of very high *wind speed* visible as bright, pink glyphs in Wyoming. This is due to the use of hue and the discretization to $d = 5$ ranges. Another interesting result is that *temperature* appears less prominent in Fig. 5c, compared to the original ViA-MI mapping in Fig. 5b. This is due to the user accepting changes to the importance weights for *temperature* and *wind speed*. Although modifying the initial user inputs can improve the visualization mapping's evaluation weight, the user must decide whether emphasizing *wind speed* at the expense of *temperature* is appropriate. ViA allows users to experiment with these types of questions and compare the resulting visualizations to see which mappings produce the best results for their analyses.

Because the visualizations in Fig. 5 are the "best" mappings suggested by ViA, they are all of a high quality. This explains why many of the differences between them are subtle. Mappings with lower evaluation weights would exhibit larger visual differences.

ViA-UI versus ViA-MI. Next, we compared ViA-MI to a system where the user is always asked to decide whether to accept or reject each hint. This scenario is similar to ViA-N, since it runs without using expected utilities to determine how to handle a hint. Instead of applying fixed cutoffs, however, the user is required to tell ViA how to act for each hint that is generated. Our main interests were to see: (1) what improvement in evaluation weight can we obtain by asking the user to control the hint selection process; and (2) how well ViA-MI insulates the user, that is, how many hints does ViA-MI manage automatically.

We did not want to force our users to answer the long sequences of queries generated during this experiment. Instead, we executed ViA-UI in three separate scenarios. In the first, we assumed the user rejected every hint suggested by ViA. In the second, ViA-UI made answers identical to those made by the fixed constraints of ViA-N. In the final scenario, ViA-UI allowed all hints.

When every hint was rejected, the search space was kept small, resulting in 20 unique queries and a maximum visualization weight of $w = 0.83$ (Table III). In the second scenario ViA-UI mimicked ViA-N, producing 43 unique queries. The best visualization had weight $w = 0.84$, identical to the ViA-N scenario. In the final scenario, allowing every hint increased the size of the search space dramatically, producing 2,324 unique queries. The weight for the best visualization was $w = 0.94$, identical to ViA-MI's second experiment.

These results suggest that mixed-initiative interaction offers important advantages over fixed constraints or full user control, by increasing the quality of the suggested visualizations while protecting the user from answering numerous queries. ViA makes requests to relax the initial constraints, but focuses on situations where this knowledge could significantly improve the visualizations being generated. ViA caches responses in probability graphs to improve the likelihood of taking automatic actions for future hints. ViA also explains why a query is being made, and quantifies the expected improvement of accepting the proposed action. Keeping the number of queries small and explaining the query's purpose and results motivates a user to fully consider a hint and answer knowledgeably about how to proceed.

A final point to recall is that, given the same visual features and perceptual guidelines, ViA cannot suggest visualizations that are better than the very best mapping hand-built by a visualization expert. ViA's strength is its ability to rapidly identify visualizations that are well-suited to a given dataset and analysis tasks, without requiring any visualization expertise from its users. Our search results demonstrate that ViA is faster and, in many cases, more effective than the typical manual process of designing high-quality visualizations.

C. E-Commerce Datasets

The first two studies highlight the advantages of ViA's search algorithm and mixed-initiative interaction model. This ignores the question of whether the evaluation engines accurately identify the strengths and weaknesses present in a visualization mapping. The rules used within each engine are based on results from controlled psychophysical experiments, specifically to address this issue. Real-world validation is still needed to confirm ViA's practical usefulness, however. To address this need, we conducted a final evaluation by visualizing intelligent agents competing in the Trading Agent Competition[3] (TAC) [18], [20]. The TAC implements different types of online auction rules to mimic a wide variety of market games. Intelligent auction agents compete within the TAC to study different buying and selling strategies. For example, during the first version of the TAC we visualized, agents were tasked to assemble travel packages consisting of:

- a round-trip flight to Boston,
- a hotel reservation during the trip, and
- tickets to entertainment events during the trip.

All three products are traded in separate markets with different auction rules. For example, hotel room auctions run as follows:

1) One economy and one luxury hotel offer sixteen rooms every evening.
2) Each hotel-evening pair runs as a separate auction.
3) An auction ends when the simulation ends, or a random period of inactivity passes with no new bids.
4) All rooms are sold at the sixteenth bid price (i.e. the sixteen highest bids win, but they all pay the sixteenth bid price).

Other auctions run with slightly different rules. For airline tickets, one flight operates every day as a separate auction, with enough seats to satisfy any number of customers and with prices ranging from \$150 to \$600, changing by ±\$10 every 20 to 30 seconds. For entertainment tickets, every agent receives an initial allotment of tickets, which they buy and sell with other agents in a stock market fashion. As with hotels, a separate auction is held for each evening-event combination.

TABLE IV
TAC ATTRIBUTES AND ASSOCIATED PROPERTIES

Attr	Domain	Freq	Task	Impt
agent ID	discrete ($u = 8$)	high	search	1.0
price	continuous	low	boundary	0.5
quantity	discrete ($u = 10$)	high	estimate	0.5

We began by asking the TAC designers to identify the attributes to visualize. They chose the *time*, *auction ID*, *agent ID*, *price*, and *quantity* for every bid made during the simulation. ViA does not suggest spatial layout of information, so we consulted with the TAC designers to choose *time* and *auction ID* to define a bid's x and y-position on a two-dimensional grid. 3D tower glyphs that can vary in their hue, luminance, height (size), density, and regularity of placement were used to represent the remaining attributes: *agent ID*, *price*, and *quantity* (Table IV).

After further discussion, we allowed *quantity* to be discretized into (as few as) three equal-width ranges. *agent ID* was not modified, since viewers need to identify specific agents during the simulation. Finally, ViA was not allowed to change any importance weights or discard any analysis tasks.

Based on these restrictions, a total of nineteen M were evaluated. The smaller number of attributes and visual features, together with the constraints on how mappings could be modified, kept this number low (without these constraints, ViA would have evaluated 189 separate M). A number of promising M remained, for example:

- $w = 0.862$, *agent ID* → color, *price* → height, *quantity* → density (discretized to $d = 4$ ranges)
- $w = 0.787$, *agent ID* → color, *price* → density, *quantity* → height

Given these mappings, we chose a modified version M_f of the first mapping for the final visualizations. Instead of using density to represent *quantity*, M_f varies each tower's width. This supports a wider range of *quantity* values, and it uncouples *quantity* from vertical density, allowing us to use this spatial property to show multiple bids within a common *time* and *auction ID*.

Each year, an online round-robin competition is used to select TAC finalists, who compete against one another at a conference venue. Fig. 6 shows a dataset from the Fourth International Conference on Multiagent Systems (ICMAS-00), visualized with

[3] `http://tac.eecs.umich.edu`

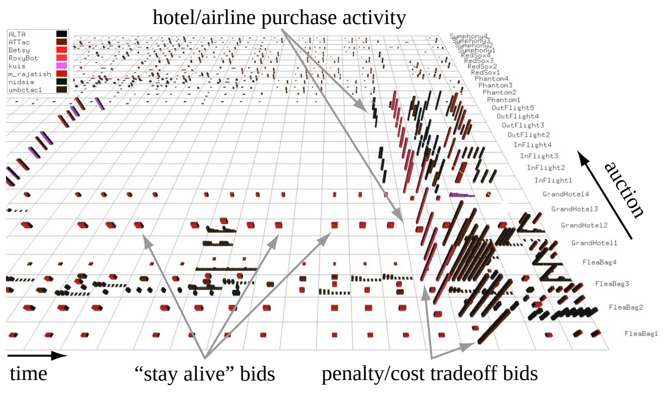


Fig. 6. ICMAS-00 TAC data visualized with M_f

M_f. Finalists at ICMAS used sophisticated agent strategies, many of which are visible in our visualizations. For example:

1) Most agents deferred purchasing hotel rooms and airline tickets until just before the simulation ended, since they felt there was no advantage to early purchase.
2) Some agents periodically made low buy bids b for hotel rooms to keep the hotel auctions open.
3) Some agents made very high buy bids b for hotel rooms at the end of the simulation. Without a hotel room, a penalty would apply for not completing the customer's trip, but the sixteenth winning bid for hotel rooms is likely to be $\ll b$, allowing the agent to secure a room for a reasonable price.

Each of the above findings were of interest to the TAC designers. They told us these particular techniques had not been identified in the event logs, graphs, and statistical analyses that were previously employed to study agents' actions. More importantly, these approaches represent major strategies. For example, agents that bid on hotel rooms using a minimum penalty-cost tradeoff won the ICMAS-00 TAC.

VI. Conclusions and Future Work

The paper describes ViA, a semi-automated visualization assistant designed to construct high quality multidimensional visualizations by combining knowledge of human visual perception with a mixed-initiative AI search algorithm.

Basic dimensions of color and texture are used to visualize individual data attributes. Guidelines on how we perceive these visual features are combined with a dataset's properties and a user's analysis needs to form a space of all possible visualization mappings. This space is explored with a hint-based search strategy that tries to quickly locate mappings that are best-suited to the user's data and tasks. A mixed-initiative interaction engine consults probability graphs and queries the user to decide when to modify the dataset or the user's initial inputs during the search for better visualizations. The top k mappings are returned, allowing the user to quickly visualize the same data in different ways to highlight different findings of interest.

ViA's hint-based search was compared to an exhaustive search, simulated annealing, and reactive tabu search. The mixed-initiative interactions were also studied to characterize the improvements they offer. Results for a meteorological dataset were positive, suggesting both components provide important advantages during visualization construction. We concluded by using ViA to visualize intelligent agent activity within a simulated e-commerce auction. Important agent strategies were found using ViA's visualizations, providing further evidence of the strength of ViA's perceptual evaluation approach.

ViA also contains a number of important limitations. Our visualizations use visual features attached to geometric glyphs to represent different data attributes. Certain datasets or analysis tasks may not fit well with this approach (e.g. when a very large number of attributes need to be displayed). ViA is designed to be application-independent. This was done to generalize to different problem domains, but it can also restrict how well ViA supports certain application-specific requirements. ViA searches for visualizations that fit either the original or any modified user constraints. Telling users which constraints a visualization is matched against can help them to understand how modifying constraints affects ViA's suggestions. Small evaluation weight differences imply subtle variations. In these situations, users may not agree with ViA about which visualization is the "best" choice. Finally, our evaluation of ViA's real-world capabilities is based on anecdotal feedback from domain experts. We have not conducted controlled experiments to formally quantify ViA advantages over existing analysis and visualization techniques.

These issues are being investigated as part of our plan for future work. Other improvements are also being pursued. Based on recently completed psychophysical studies, we are building three new evaluation engines to include flicker, direction of motion, and velocity of motion in ViA's visualizations [24]. We are also extending ViA's search strategy to include diversification through the existing hint-based scheme, and intensification within a local region using a reactive tabu search. Results from our evaluations suggest this may lead to better performance.

References

[1] Allen, J., Guinn, C. I., and Horowitz, E. Mixed-initiative interaction. *IEEE Intelligent Systems 14*, 5 (1999), 14–23.

[2] Battiti, R. Reactive search: Toward self-tuning heuristics. In *Modern Heuristic Search Models*, V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, Eds. John Wiley & Sons, Inc., New York, New York, 1996, pp. 61–83.

[3] 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.

[4] Beshers, C., and Feiner, S. AutoVisual: Rule-based design of interactive multivariate visualizations. *IEEE Computer Graphics & Applications 13*, 4 (July 1993), 41–49.

[5] Brown, M., and Chu-Carroll, J. An evidential model for tracking initiative in collaborative dialog interactions. *User Modeling and User-Adapted Interaction 8*, 3 (1998), 215–253.

[6] Callaghan, T. C. Dimensional interaction of hue and brightness in preattentive field segregation. *Perception & Psychophysics 36*, 1 (1984), 25–34.

[7] Callaghan, T. C. Interference and dominance in texture segregation. In *Visual Search*, D. Brogan, Ed. Taylor & Francis, New York, New York, 1990, pp. 81–87.

[8] CIE. *CIE Publication No. 15, Supplement Number 2 (E-1.3.1, 1971): Official Recommendations on Uniform Color Spaces, Color-Difference Equations, and Metric Color Terms*. Commission Internationale de L'Èclairge, 1978.

[9] Cohen, P. R. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, 1995.

[10] Cohen, P. R., Allaby, C., Cumbaa, C., Fitzgerald, M., Ho, K., Hui, B., Latulipe, C., Lu, F., Moussa, N., Pooley, D., Qian, A., and Dissiqi, S. What is initiative. *User Modeling and User-Adapted Interaction 8*, 3 (1998), 171–214.

[11] Ferguson, G., Allen, J. F., and Miller, B. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems* (Edinburgh, Scotland, 1996), pp. 70–77.

[12] GALLOP, J. Underlying data models and structures for visualization. In *Scientific Visualization: Advances and Challenges*, L. Rosenblum, Ed. Academic Press, San Diego, California, 1994, pp. 87–102.

[13] GLOVER, F., AND LAGUNA, M. Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems*, C. R. Reeves, Ed. Blackwell Scientific Publishing, Oxford, UK, 1993, pp. 70–150.

[14] GRAY, P., HART, W., PAINTON, L., PHILLIPS, C., TRAHAN, M., AND WAGNER, J. A survey of global optimization methods. `http://www.cs.sandia.gov/opt/survey/main.html.1997`, 1997.

[15] HEALEY, C. G., BOOTH, K. S., AND ENNS, J. T. Harnessing preattentive processes for multivariate data visualization. In *Proceedings Graphics Interface '93* (Toronto, Canada, 1993), pp. 107–117.

[16] 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.

[17] HEALEY, C. G., ENNS, J. T., TATEOSIAN, L. G., AND REMPLE, M. Perceptually-based brush strokes for nonphotorealistic visualization. *ACM Transactions on Graphics 23*, 1 (2004), 64–96.

[18] 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.

[19] HEALEY, C. G., ST. AMANT, R., AND ELHADDAD, M. ViA: A perceptual visualization assistant. In *28th Workshop on Advanced Imagery Pattern Recognition (AIPR-99)* (Washington, DC, 1999), pp. 1–11.

[20] HEALEY, C. G., AND WURMAN, P. R. Visualizing market data. *IEEE Internet Computing 5*, 2 (2001), 88.

[21] HIBBARD, B., AND SANTEK, D. The VIS-5D system for easy interactive visualization. In *Proceedings Visualization '90* (San Francisco, California, 1990), pp. 28–35.

[22] HOROWITZ, E. Uncertainty, action and interaction: In pursuit of mixed-initiative computing. *IEEE Intelligent Systems 14*, 5 (1990), 17–20.

[23] HOROWITZ, E. Principles of mixed initiative user interfaces. In *Proceedings SIGCHI '99* (Pittsburgh, Pennsylvania, 1999), pp. 159–166.

[24] HUBER, D. E., AND HEALEY, C. G. Visualizing data with motion. In *Proceedings Visualization 2005* (Minneapolis, Minnesota, 2005), pp. 527–534.

[25] INTERRANTE, V. Illustrating surface shape in volume data via principle direction-driven 3D line integral convolution. In *SIGGRAPH 97 Conference Proceedings* (Los Angeles, California, 1997), T. Whitted, Ed., pp. 109–116.

[26] INTERRANTE, V. Harnessing natural textures for multivariate visualization. *IEEE Computer Graphics & Applications 20*, 6 (2000), 6–11.

[27] JOHNSON, C. R. Top scientific visualization research problems. *IEEE Computer Graphics & Applications 24*, 4 (2004), 13–17.

[28] JOHNSON, C. R., MOORHEAD, R., MUNZNER, T., PFSITER, H., RHEINGANS, P., AND YOO, T. S. E. *NIH/NSF Visualization Research Challenges Report*. IEEE Press, Piscataway, New Jersey, 2006.

[29] JULÉSZ, B., GILBERT, E. N., AND SHEPP, L. A. Inability of humans to discriminate between visual textures that agree in second-order statistics—revisited. *Perception 2* (1973), 391–405.

[30] LAIDLAW, D. H., KIRBY, R. M., JACKSON, C. D., DAVIDSON, J. S., MILLER, T. S., DA SILVA, M., WARREN, W. H., AND TARR, M. J. Comparing 2D vector field visualization methods: A user study. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (2005), 59–70.

[31] LESTER, J. C., STONE, B. A., AND STELLING, G. D. Lifelike pedagogical agents for mixed-initiative problem solving in constructivist learning environments. *User Modeling and User-Adapted Interaction 9*, 1 (1999), 1–44.

[32] LEVKOWITZ, H., AND HERMAN, G. T. Color scales for image data. *IEEE Computer Graphics & Applications 12*, 1 (1992), 72–80.

[33] LOHSE, J., RUETER, H., BIOLSI, K., AND WALKER, N. Classifying visual knowledge representations: A foundation for visualization research. In *Proceedings Visualization '90* (San Francisco, California, 1990), pp. 131–138.

[34] MACKINLAY, J. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics 5*, 2 (1986), 110–141.

[35] MALIK, J., AND PERONA, P. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A 7*, 5 (1990), 923–932.

[36] MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., AND KANG, T. Design galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH 97 Conference Proceedings* (Los Angeles, California, 1997), T. Whitted, Ed., pp. 389–400.

[37] MCCORMICK, B. H., DEFANTI, T. A., AND BROWN, M. D. Visualization in scientific computing. *Computer Graphics 21*, 6 (1987), 1–14.

[38] MILLER, B. Is explicit representation of initiative desirable? In *Working Notes of AAAI 97 Spring Symposium on Mixed-Initiative Interaction* (Stanford, California, 1997), pp. 105–110.

[39] MUNSELL, A. H. *A Color Notation*. Geo. H. Ellis Co., Boston, Massachusetts, 1905.

[40] RAO, A. R., AND LOHSE, G. L. Towards a texture naming system: Identifying relevant dimensions of texture. In *Proceedings Visualization '93* (San Jose, California, 1993), pp. 220–227.

[41] REED, T. R., AND HANS DU BUF, J. M. A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Understanding 57*, 3 (1993), 359–372.

[42] RHEINGANS, P., AND TEBBS, B. A tool for dynamic explorations of color mappings. *Computer Graphics 24*, 2 (1990), 145–146.

[43] ROBERTSON, P. K. A methodology for scientific data visualisation: Choosing representations based on a natural scene paradigm. In *Proceedings Visualization '90* (San Francisco, California, 1990), pp. 114–123.

[44] ROBERTSON, P. K., AND DE FERRARI, L. Systematic approaches to visualization: Is a reference model needed? In *Scientific Visualization: Advances and Challenges*, L. Rosenblum, Ed. Academic Press, San Diego, California, 1994, pp. 239–250.

[45] ROGOWITZ, B. E., AND TREINISH, L. A. An architecture for perceptual rule-based visualization. In *Proceedings Visualization '93* (San Jose, California, 1993), pp. 236–243.

[46] ROSENBLUM, L. J. Research issues in scientific visualization. *IEEE Computer Graphics & Applications 14*, 2 (1994), 61–85.

[47] RUSSELL, S. J., AND NORVIG, P. *Artificial Intelligence: A Modern Approach, 2nd Edition*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2003.

[48] SCHROEDER, W., MARTIN, K., AND LORENSEN, B. *The Visualization Toolkit*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1998.

[49] SENAY, H., AND IGNATIUS, E. A knowledge-based system for visualization design. *IEEE Computer Graphics & Applications 14*, 6 (1994), 36–47.

[50] SNOWDEN, R. J. Texture segregation and visual search: A comparison of the effects of random variations along irrelevant dimensions. *Journal of Experimental Psychology: Human Perception and Performance 24*, 5 (1998), 1354–1367.

[51] ST. AMANT, R., AND COHEN, P. R. Interaction with a mixed-initiative system for exploratory data analysis. In *Proceedings 2nd International Conference on Intelligent User Interfaces (IUI '97)* (Orlando, Florida, 1997), pp. 15–22.

[52] TATEOSIAN, L. G., AND HEALEY, C. G. Engaging viewers with aesthetic visualizations. In *Proceedings 5th International Symposium Non-Photorealistic Animation and Rendering (NPAR 2007)* (San Diego, California, 2007), p. to appear.

[53] THOMAS, J. J., AND COOK, K. A. *Illuminating the Path: Research and Development Agenda for Visual Analytics*. IEEE Press, Piscataway, New Jersey, 2005.

[54] TRIESMAN, A. Search, similarity, and integration of features between and within dimensions. *Journal of Experimental Psychology: Human Perception & Performance 17*, 3 (1991), 652–676.

[55] WARE, C. Color sequences for univariate maps: Theory, experiments, and principles. *IEEE Computer Graphics & Applications 8*, 5 (1988), 41–49.

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

[57] WEHREND, S., AND LEWIS, C. A problem-oriented classification of visualization techniques. In *Proceedings Visualization '90* (San Francisco, California, 1990), pp. 139–143.

[58] WEIGLE, C., EMIGH, W. G., LIU, G., TAYLOR, R. M., ENNS, J. T., AND HEALEY, C. G. Oriented texture slivers: A technique for local value estimation of multiple scalar fields. In *Proceedings Graphics Interface 2000* (Montréal, Canada, 2000), pp. 163–170.

[59] WOLFE, J. M. Guided Search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review 1*, 2 (1994), 202–238.

[60] WYSZECKI, G., AND STILES, W. S. *Color Science: Concepts and Methods, Quantitative Data and Formulae, 2nd Edition*. John Wiley & Sons, Inc., New York, New York, 1982.