Designing for Interactive Dimension Reduction Visual Analytics Tools to Explore High-Dimensional Data

Jessica Zeitz Self, Xinran Hu, Leanna House, Scotland Leman, Chris North

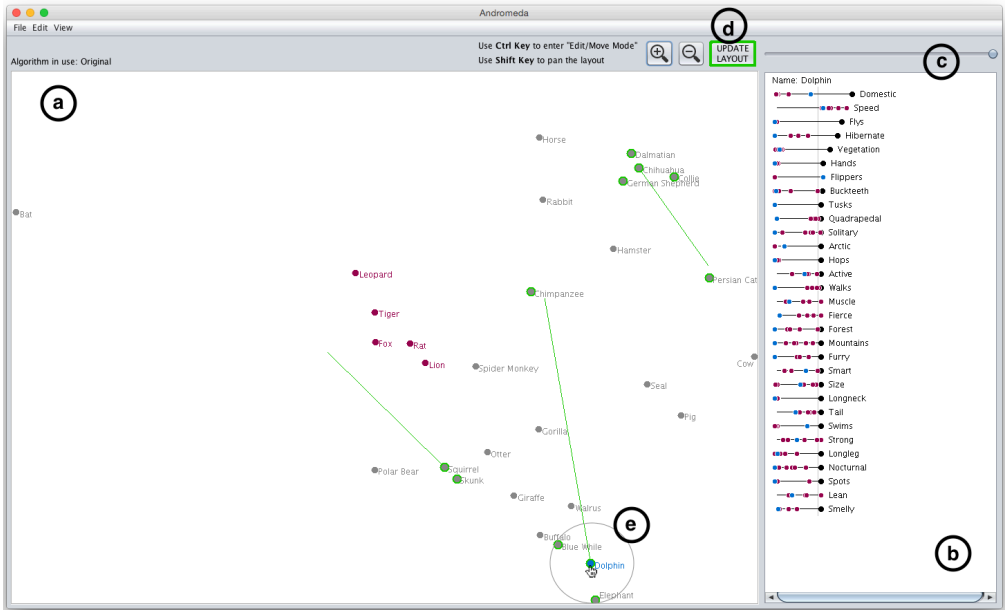

Fig. 1. Andromeda interface exploring an animal dataset: (a) the object view visualizing low-dimensional data points, (b) the parametric view displaying all dimension weights, (c) the slider tool to animate between transitions, (d) the button to update the layout, and (e) the radius highlighting near and possibly important data points.

Abstract— Exploring high-dimensional data is challenging. As the number of dimensions in datasets increases, the harder it becomes to discover patterns and develop insights. Dimension reduction algorithms, such as multidimensional scaling, support data explorations by reducing datasets to two dimensions for visualization. Because these algorithms rely on underlying parameterizations, they may be tweaked to assess the data from multiple perspectives. Alas, tweaking can be difficult for users without a strong knowledge base of the underlying algorithms. In this paper, we present principles for developing interactive visual analytic systems that enable users to tweak model parameters directly or indirectly so that they may explore highdimensional data. To exemplify our principles, we introduce an application that implements interactive weighted multidimensional scaling (WMDS). Our application, Andromeda, allows for both parametric and object-level interaction to provide in-depth data exploration. In this paper, we describe the types of tasks and insights that users may gain with Andromeda. Also, the final version of Andromeda is the result of sequential improvements made to multiple designs that were critiqued by users. With each critique we uncovered design principles of effective, interactive, visual analytic tools. These design principles focus on three main areas: (1) layout, (2) semantically visualizing parameters, and (3) designing the communication between the interface and the algorithm.

Index Terms—Dimensionality reduction, object-level interaction, visual analytics, interface design

1 Introduction

With the amount of analyzable data growing rapidly, we must continue to develop tools to strengthen our ability to learn all that we can from data. Statistical mathematical models enable us to simplify and formalize our understanding of data. The goal of visual analytics is to design usable methods for interacting with these models to improve user data exploration techniques. For example, dimension reduction algorithms, such as Weighted Multidimensional Scaling (WMDS), project high-dimensional data onto low-dimensional (e.g. two-dimensional), space. The purpose of the algorithms is to summarize high-dimensional information in a form that is accessible to users, such as a two-dimensional graph. The visual analytics community may further improve the utility of these models by enhancing them with information visualization and developing tools that allow for visual interaction with the mathematical models. We in the visual analytics community may further improve the utility of these models by developing tools that allow for visual interaction with the mathematical models. In previous work, parametric and observation level interaction (OLI) with data visualizations has been defined and shown helpful for data exploration [1]–[3]. Both forms of interaction enable users to adjust display-generating models directly and/or indirectly. However, there are constraints to these interactions and points to consider while developing software with parametric and OLI capabilities that have not been formalized. In this paper, we present principles to develop technically sound and useful visual analytic software to exploit parametric interaction and OLI. We exemplify these principles in a tool we developed called Andromeda.

In Section 3, we describe Andromeda in detail. It is a visual analytics tool that spatializes high dimensional data in two dimensions using an algorithm called Weighted Multidimensional

Scaling (WMDS) [4], [5]. In the spatialization, distance reflects relative similarity; e.g., two points close to each other in the spatialization are more similar to each other in the high-dimensional space than two points far from each other. To set the spatial coordinates of the observations, WMDS relies on one parameter for each variable in the dataset. We refer to the parameters as variable weights because variables with large weights are considered heavily in the spatialization and those with low weights are not. Thus, one can deepen their interpretation of a visualization in Andromeda by considering both distance and weights; e.g., two points close to each in a spatialization are more similar to each other *in the variables with large weights* than two points far from each other. Andromeda enables users to adjust spatializations using either parametric interaction or OLI.

Parametric interaction is available in several tools [6]–[8] where users may specify underlying model parameters by adjusting dials and/or sliders. Although it has been shown useful, parametric interaction can challenge users without a strong knowledge of the underlying model. Thus, in previous work, we invented a new way to interact with mathematical models called OLI [1], [2]. With OLI, an automated procedure transforms user interactions with data visualizations (visual feedback) to parametric feedback which in turn adjusts an entire visual space. For example, in Andromeda, users may change the distance between observations by relocating them so that an automated procedure may adjust the parameters (i.e., variable weights) in response.

To design useful, efficient, and analytically accurate tools that enable parametric interaction and OLI requires intimate knowledge of both mathematical models and how users process information. Thus, in this paper, we develop 16 principles that visual analysts should consider when developing these tools. We categorize these principles as layout, parametric, and algorithmic. Layout principles refer to the design of how to represent the data. Parametric principles deal with how to visualize and interact with parameters that do not necessarily have intrinsic meaning. For example, most parameters have domain constraints within models and algorithms. Users should not be able to specify values for parameters outside their domains. Algorithmic principles define the considerations for connecting the visualization to the mathematical model. The effort spent iterating through multiple designs of Andromeda resulted in the aforementioned design principles for interactive visual analytics tools. Our contributions are as follows:

- design principles for interactive visual analytics tools to explore high-dimensional data;
- an interface design, Andromeda, that encompasses all design principles.

2 Related Work

Visual analytics tools aid users in exploring data. However, the tools are only useful if the design makes sense to the user, correctly portrays the underlying model correctly, and allows the user to conduct analyses efficiently. Much research exists to guide designers in the creation of interfaces for information visualization [9]–[12]. The visual analytics field specifically focuses on the design of interactive visualizations that provide an intuitive space that fosters insight creation and data understanding [13]. High-dimensional data is particularly difficult for users to comprehend because humans have trouble thinking about a large number of dimensions simultaneously. As discussed earlier, dimension reduction models reduce the data so that it is more manageable. Other non-traditional methods have been developed to support high-dimensional data exploration [14]–[16]. These techniques have yet to be incorporated into an interactive visual analytics tool.

IN-SPIRE's Galaxy View displays text documents as data points in topical clusters in a two-dimensional space where proximity implies relatedness [8]. Star Coordinates plots objects in high-dimensional space and then projects this space onto two-dimensions [17]. However, within both of these tools, only surface-level interactions are possible. In IN-SPIRE, the user can explore the data by selecting groups of points. The selection is cross-referenced with other types of visualizations and graphs for the user to gain more insight. Star Coordinates allows users to rotate and scale the projection. These surface-level interactions are useful, however, the user has no control over the parameters that are used to process the data.

Models have underlying parameters that can be adjusted to control how the data are reduced. Many tools exist to allow users to not only visualize high-dimensional data, but also adjust the parameters of the model to visualize the data from multiple perspectives. Systems such as STREAMIT [18] and Dust & Magnet [19] allow parametric interaction where the user inputs feedback to update the model and in turn the model updates the visualization. STREAMIT uses a force-directed layout to visualize streaming text documents based on keyword similarity. Users can modify the numeric parameters (i.e. importance of keywords) to update the visualization. Dust & Magnet displays the parameters (i.e. dimension weights) as "magnets" within the object visualization. Users can directly interact with the magnets to modify their importance. This direct manipulation can be more intuitive for a user than increasing or decreasing a numerical value. However, both approaches still solely provide parametric interaction which limit the depth and effectiveness of data exploration.

Dis-Function incorporates OLI for the user to adjust the data points projected by WMDS [6]. As in Andromeda, adjusting points in the visualization recalculates the parameters (i.e. weight dimensions) as feedback to the user. However, these tools define the inverse model differently which we will discuss later. Despite providing OLI, Dis-Function does not provide parametric interaction which could enhance the user experience. The goal of Dis-Function differs from that of Andromeda. Dis-Function focuses on learning a distance function that represents the users understanding of a dataset, whereas Andromeda provides an exploratory space to gain insights about a dataset.

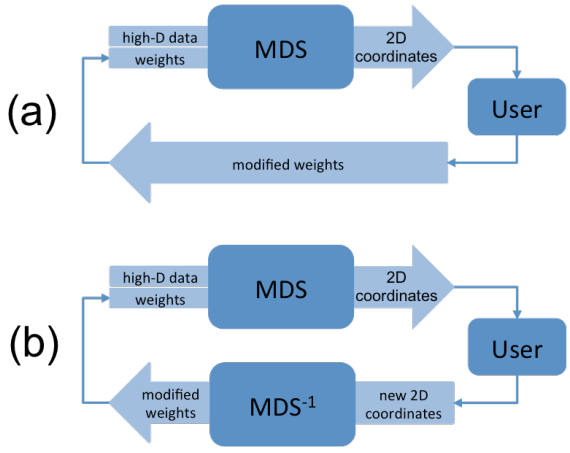

Fig. 2. Algorithmic pipeline (a) for parametric interaction and (b) for OLI or visual to parametric interaction.

3 User Interface Design

Andromeda is an interactive visual analytics tool designed to aid users in the analysis of high-dimensional data. It provides a way for users to interact with the input and output of weighted multidimensional scaling (WMDS). Andromeda supports both OLI and parametric interaction. The algorithmic pipelines of OLI and parametric interaction are shown in Fig. 2a and Fig. 2b respectively. The interface is implemented using Java Swing and WMDS uses the MDSJ Java Library [21]. Andromeda is composed of two main sections: the object view (Fig. 1a) and the parameter view (Fig. 1b). Examples throughout this paper use a modified version of an animal

dataset provided by Lampert et al. which contains 30 animal objects and 31 dimensions [22]; i.e. that data include 30 31-dimensional data points.

The object view. The resulting object layout calculated by WMDS is displayed in the object view. Each point represents one row of the high-dimensional data. This view has two modes toggled between by the control modifier key. Without the modifier key, users can explore and view the data points. A user can hover over (blue point) and select points (maroon points) to view the corresponding raw data. We use color to link selected points to the parameter view where the raw data is displayed. With the modifier key, the user enters move mode that encompasses object-level interaction. The user can manipulate points on the screen to provide input to the algorithm. When a point is moved, it is encoded with a green ring and a line from its original location to its new location (see Dolphin, Squirrel, and Chihuahua in Fig. 1a). Points that are selected, but not moved are considered highlighted. These points are encircled with a green ring, but do not have a line since they were not moved (see Elephant, Blue Whale, Skunk and others in Fig. 1a). The green outline matches the outline of the "Update Layout" button as a visual cue that all outlined points are important to the algorithm (Fig. 1d). Algorithmically, moved points represent explicit user input and highlighted points represent implicit user input as discussed in Section 4.3.

After points have been moved, the user can click the "Update Layout" button to recalculate the layout based on the new coordinate locations of the moved points. An optimization algorithm is run to find a weight vector that best represents the new coordinates of only the moved points. WMDS is then run to update the coordinates of all points given the new weight vector. Within the object view, the points animate to the new locations to give the user a visual representation of the movement of the points. The user can repeat this visualization by engaging with the slider (Fig. 1c). The slider allows the user to manually trace all points between the previous and current locations.

Table 1. Andromeda Interactions

Feature	Encoding	Description
Increase/decrease dimension weight	Hand cursor over handle	Drag handle to the right (increase) or to the left (decrease)
View Mode		
Hover over point	Blue in color	Hover cursor over point to view raw data for hovered point on weight lines
Select point(s)	Maroon in color	Click single point or draw a box around multiple points to view raw data for selected point(s) on weight lines
View raw data	Tooltip over dimension handle	Hover cursor over dimension line to view raw data for selected points
Edit/Mode Mode (Must use Ctrl modifier key)		
Drag point	Green outline around point and green line from previous location	Ctrl key + drag a point with the mouse
Highlight point(s)	Green outline around point	Ctrl key + click a single point or draw a box around multiple points

The parameter view. This view displays the weighted dimensions (Fig. 1b). Due to the algorithm only handling continuous numerical data, the interface shows categorical or informational dimensions as static text for viewing only. Each numerical dimension is represented by an interactive line that serves as a visual representation of the relative weight compared to all other dimension lines. The user can drag the circular handle at the end of a line to adjust the weight of that dimension. Since all weights must sum to 1, the interface automatically modifies all other dimensions when one dimension is increased or decreased. Modifying dimensions causes dynamic updates to the layout. Each time a user increases or decreases a dimension, WMDS recalculates the object layout based on this new weight vector in real time.

The parameter view also displays the raw data values of the high-dimensional data. All raw data values are normalized to fit a constant scale across all dimensions. This scale is used to plot the raw data onto the weight lines. When a point is selected in the object view, the corresponding raw data values are drawn onto each dimension line as a colored dot. For example, the maximum raw data value for a specific dimension will be placed on the far right of the line. A lower raw data value will appear closer to the left of the line. In Fig. 1, the selected maroon data points in the object view are animals that do not fly (third dimension line from the top), therefore the raw data points appear toward the left of the line. As a dimension weight is increased, the plotted raw data dots are stretched to fill the line. The raw data is not changing, however the relative distances between the values are changing based on the emphasis placed on that particular dimension.

4 Benefits of the Design

Throughout development, we had users informally assess the design of our system. We applied Andromeda in an education setting with a graduate level visual analytics course and a graduate level information visualization course during separate semesters. Each course used a different iteration of the system so that we could learn from the students' analyses. That involved collecting data about the students to attain two datasets with information for them to explore. Example survey questions included car mileage, number of apps on your phone, number of restaurants visited in town, and on a scale from 0 to 100 how extroverted would you consider yourself. Each class explored its respective dataset using Andromeda and developed insights about each other. An example sequence of user interactions is show in Fig. 3. We had the users reflect on their processes and explain any challenges they encountered with the system. We analyzed the insights and processes from each class to see if users did what we expected. If not, we developed new interactions and design choices that encouraged more efficient usage and addressed the challenges. The discoveries from the visual analytics course led to interface modifications for the next interface design, which was given to the information visualization course to repeat the process.

We hypothesized that OLI increases usability in the following three ways:

(1) Users are more likely to understand the objects themselves and how to manipulate them rather than the parameters and how to adjust those. The objects in the dataset will presumably be familiar entities to the user, but the particular dimensions describing the objects may be different from the dimensions the users have in their head. User domain knowledge may include additional dimensions about the objects that do not appear in the dataset. The user may also have meta-dimensions that they may or may not be able to describe using the dataset dimensions. For example, in the animal data, a user may be interested in a dimension he refers to as cat-like. He may not be able to explain it, but he has a preconceived notion in his head about whether an animal fits this meta-dimension. The user can perform OLI and the system will calculate dimensions within the dataset that define cat-like. Similarly, users have preexisting notions of groupings, but may not have a meta-dimension to describe the grouping. Again, OLI offers the ability to instinctively group similar objects and have the mathematical model find supporting dimensions.

(2) OLI provides the ability to pose what-if questions as a separate type of hypothesis testing. Users know and understand preexisting relationships between objects. For example, a user can pose that she thinks certain objects are similar. OLI can discover whether there is data to support this claim. Another question might focus on forcing a outlier into a cluster. With parametric interaction, this would require many trial and error iterations until the system converged on an appropriate parameterization. With OLI, the user can drag the outlier into the cluster and let the mathematical model do the work.

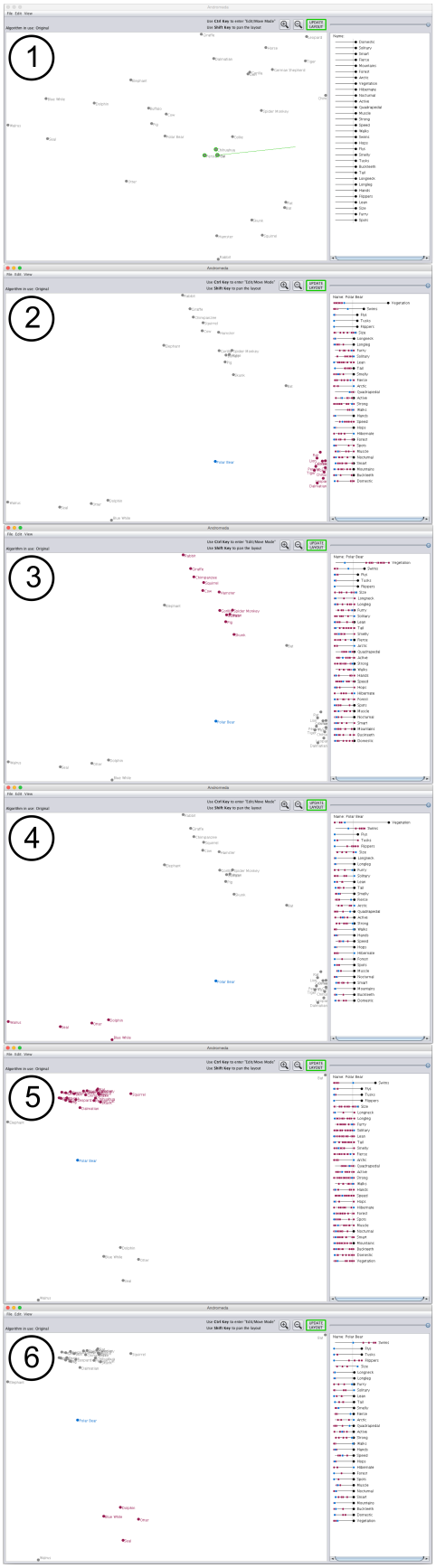

Fig. 3. This is a sequence of interactions in Andromeda. (1) Initial view with moved points. (2)-(4) Updated layout with different clusters selected. (5)-(6) Update layout after decreasing vegetation dimension.

(3) With OLI, users have the opportunity to manipulate the data on the object level. We claim an object level view is a more meaningful space to interact than a view consisting of a list of parameters. As the data grows, it is easier to manipulate lots of objects instead of adjusting lots of parameters. We can imagine quickly and fluidly manipulating many objects at one time. In order to provide this for parameters, we would have to considerably modify our parameter view design choices.

The above hypotheses were validated through our assignments given to the courses. The rest of this section discusses the tasks performed by the students through the use of Andromeda.

Our final design encouraged users to perform new tasks creating analytical gains of the system.

Injecting domain knowledge. Users inject domain knowledge into their explorations. For example, users clustered their peers into an international group and a non-international groups and international was not a dimension.

Single dimension trends across all objects. Users described single dimension trends across all data points by selecting all data points in the object view to view the raw data values in the parameter view (Fig. 1b).

Comparison of clusters. Users also compared clusters and explained what dimensions would need to change to make one cluster like the other. For example, one user stated that introverts could be like extroverts if they ate at restaurants more often, spent more money and had more Facebook friends.

Filling in data gaps. Another new task was hypothesizing about missing data. The dataset included people who did not want to provide their gender. One user adjusted the data points in the object view to guess the gender of peers who did not provide it. OLI completely afforded this task. The user was then able to also state what dimensions characterize males and females.

Solving a subjective question. A unique task led a user to find his perfect roommate out of his peers. He used parametric interaction to increase the weight of two dimensions (typical bedtime and preferred outdoor temperature). He continued to increase certain dimensions and noted the clusters forming close to his data point. He was progressively decreasing the number of potential roommates.

Finding relationships between dimensions. Users can find out if different dimensions are related to each other. For example, people who liked to cook, may not eat in restaurants as much.

Forcing outliers into clusters. Users forced outliers into clusters in order to discover what dimensions must be emphasized to include those outliers in clusters.

Extremes in the dataset. Users state which two or three data points are most similar or different, and group dimensions based on categories (e.g. number of US states visited and number of countries visited grouped to describe how well travelled people are).

The above new tasks would not have been possible without the combination of OLI and parametric interaction into one tool. Throughout our interface iterations, we have seen that Andromeda continues to allow the discovery of trivial insights through simple tasks, but also provides the opportunity for users to develop more complex insights through creative tasks. The use of OLI together with parametric interaction within an interface design delivers a well-equipped tool for visual analyses. Throughout the design iterations, we noted the types of tasks users were able to accomplish. Many tasks were consistently performed with all iterative interface designs. These tasks were simple and typically resulted in simple insights.

5 Design Principles

We established design principles that are necessary to consider when developing an interactive high-dimensional data visual analytics tool. These principles resulted from thorough evaluations of multiple iteratively-developed interfaces. User challenges and misunderstandings gave rise to new design choices that better articulated the appropriate usage of the tool and the comprehension of the underlying model. We grouped the design principles into three categories: layout visualization, parameter visualization and algorithm communication.

5.1 Projection Visualization

The layout visualization provides a space to visualize and manipulate the data points. The design principles in this section focus on the interactions and encodings necessary to utilize this space efficiently.

(1) The projection visualization should utilize object-level interaction. Since OLI is familiar for users [1], manipulating the model output to exploit OLI provides an intuitive space for exploration. OLI [1] provides an intuitive interactive workspace for exploring complex high-dimensional data with the aid of powerful parameterized statistical models. WMDS and those alike provide a perfect opportunity to utilize OLI since it reduces data points in high-dimensional space to low-dimensional space. In our case, Andromeda displays the data in two-dimensional Euclidean space, which is easier for users to understand. It follows naturally that users may adjust these data points and receive feedback based on their new relative pairwise distances.

All of our interface designs provide object-level interaction. It allows the model to be hidden and instead relies on a familiar metaphor, near is similar. Users can drag the points around the screen to form clusters, force outliers or create other patterns. Points that get dragged closer are considered more similar and points that get dragged apart are deemed different. Operating on this metaphor is more familiar to users than modifying parameters that are specific

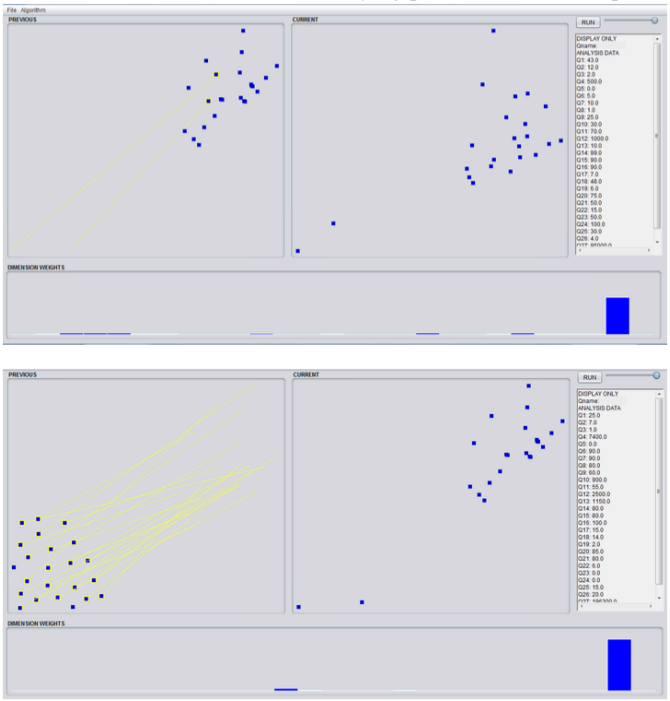

Fig. 5. These screenshots are of the first Andromeda iteration. The top displays where a user tried to force two outliers into the cluster. The bottoms shows how the user tried to force the two outliers by dragging all members of the cluster to the outliers. The current visualizations on the right side of each screenshot show that the result is not what the user expected since the two outliers are still outliers.

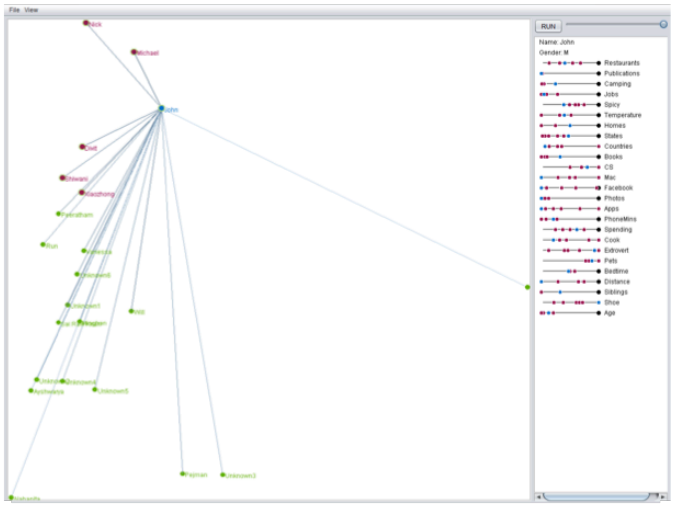

Fig. 4. This screenshot displays the stress lines in the second Andromeda iteration. Darker lines depict higher stress between the distances in low-dimensional and high-dimensional space.

to the model [23]. Users can easily manipulate data points that represent all dimensions. It also gives users another angle at which to approach a hypothesis when analyzing a dataset. The focus is on the data points instead of the dimensions. Users can form clusters and hypothesize about the similarities and differences among the included data points. In our tool, object-level interaction is possible because the WMDS model can be inverted to calculate a new weight vector that best represents the new point locations to which points were dragged. This inversion is discussed further in Section 4.3.

(2) The interface should distinguish between view mode and edit/move mode. Visualization tools that implement mathematical models should support two modes: view mode and edit/move mode. View mode should allow the user to explore the data. Clicking should invoke brushing and linking between all views [10]. In Andromeda, clicking a point in view mode displays the raw data associated with that point in the parameter view. The user is able to explore the data points and dimensions to make insights about the patterns within the current layout. Edit/move mode is entered with the use of a modifier key. The user can manipulate points within the object layout in this mode. By moving points, the user is conveying input to the algorithm. This is an important concept for the user to understand since the optimization algorithm is designed to recalculate based on only the user moved points. Viewing the data and updating the algorithm are two distinct tasks. Having to physically press a modifier key enforces the separate modes. It is important to distinguish between the two to help the user understand the input she is providing to the algorithm.

(3) User input to the model should be visually denoted within the visualization. As mentioned in the previous design principle, it is essential for users to understand the input they are providing to the algorithm. Our first iteration did not distinguish between view and edit/move mode. Only user moved points visually differed from other points with a yellow outline and straight line from the previous location. Users would manipulate points on the screen to represent their internal understanding of the similarities and differences. For example, one user wanted to figure out in what way two outliers were similar to a cluster. She dragged the two outliers closer to the cluster and then clicked the button for the algorithm to recalculate. In the resulting layout, the outliers returned to their original location still removed from the cluster (Fig. 5). Next she tried to drag all data points in the cluster closer to the outliers, which again resulted in the same layout with the cluster returning to its original location. Both results occurred because the algorithm considers only user moved points when recalculating the weight vector. Therefore, since she did not highlight any points in the cluster as well as move the outliers

closer to the cluster, the algorithm only considered the two outliers as important and not the fact that she was moving the outliers in relation to the cluster.

To help elicit more input from the user, our final design automatically highlights surrounding points around the current location of the user moved point. As the user is moving the point, radial points are highlighted to demonstrate to the user the significance of highlighting reference points. For example, in Fig. 1e the system automatically highlighted Blue Whale and Elephant in response to the user dragging Dolphin closer. The system also highlighted Chimpanzee since the user dragged Dolphin away. We designed the interaction to not only help the results of the algorithm, but also teach the users how to specify the best input to receive insightful output.

In Andromeda, the user decides when to update the layout and initiate algorithm input. After the user is satisfied with the adjusted layout by OLI, she clicks the "Update Layout" button. We had much discussion about the existence of the update button versus dynamic interaction where the layout would recalculate after each data point user movement. Dynamic feedback would provide a fluid transition between layout updates and free the user from deciding when the algorithm should recalculate. However, the optimization algorithm for WMDS inverse does not provide appropriate feedback when less than three low-dimensional data points are adjusted. Andromeda could account for this and wait until the user has manipulated three points, but what if the user was planning to create a cluster containing ten data points? The layout would update after three interactions and the user's thought process might be interrupted. Since WMDS works most efficiently when many points are moved and the interface cannot predict the user's future interactions, we concluded it is best to let the user be responsible for when the layout should be updated.

(4) Any reasonable output from the model should be appropriately visualized in the interface as additional feedback to the user. The visual algorithmic feedback to the user is just as important as the input from the user. In order for the user to give feedback for the next iteration and continue the exploratory cycle efficiently, it helps to understand the algorithmic feedback. In Andromeda, algorithmic feedback includes: (1) a new layout, (2) the weighted dimensions applied to the real high-dimensional distance, and (3) a stress factor(s). Stress in WMDS represents a quantitative measure of the discrepancy between actual pairwise distances in the high-dimensional space and those plotted in the low-dimensional space. It may reported for each pair of observations, a subset of pairs, or as a global metric (summing the stress for all pairs).

High stress denotes that the distance in low-dimensional space does not match the distance in high-dimensional space. After our first iteration of Andromeda we thought that users could motivate some interactions based on pairwise stress factors. Thus, in the second iteration, we designed stress lines to display upon request. That is, when a user clicked on a projected data point in the object view, the system would draw gradient lines from the clicked data point to all other data points (see Fig. 4) to represent stress. Darker lines represented higher stress meaning the two points in low-dimensional space wanted to be either closer together or farther apart. However, users did not find the stress lines helpful since they did not depict whether the points were too close or too far compared to the distances in high-dimensional space. A more useful distance comparison to display in the object view might be to visualize the real high-dimensional pairwise distances in reference to a single point instead of the stress.

The optimization we use iterates through many layouts to maximize the stress. This calculation can take a long time depending on the number of objects and dimensions in the data. Future visual feedback might include a preview of the layout as the calculations are occurring. Another possibility might be to recalculate the layout using less strict constraints on the model just to give the user an idea of how the layout will change given the adjustments. The user could then decide whether to continue with the stricter model. All design choices should provide the user with feedback that represents all aspects of the mathematical model. This provides the opportunity for the user to better understand the data from many different angles.

(5) The interface should provide the user with visual feedback in between model updates. In our first iteration, the interface consisted of two object layouts: previous and current. The previous object view displayed the data point layout from the previous algorithm calculation. The current object view displayed the most recent data point layout. This allowed users to compare the state of the layout between algorithm updates. The comparison was useful, but visually comparing the two layouts side by side, even with brushing and linking, was challenging. It also decreased screen real estate needed for exploring and manipulating data points. For the second iteration, we designed a slider that animated all points between their previous and current locations in a single view. This interaction allowed users to manually trace the points' paths while maximizing data exploration space. However, despite the new design, the comparison is misleading since all distances are relative. Because scale does not persist across iterations of WMDS, the distance between two points in one layout cannot be compared to the distance between the same two points in a future layout. This requires that we transpose the WMDS coordinates to fit into the size of the visualization window after each update. Despite the confusion, we decided this comparison was more helpful than harmful for users. Since our final design does not include a permanent display of the previous layout, but solely provides an animation, we concluded it was beneficial for the user to visualize the transition between states.

(6) An interactive visual analytics tool that visualizes iterative updated layouts should transition between these layouts smoothly. In our first iteration, when a user moved points and then updated the layout, the recalculation would occur and the low-dimensional data points would abruptly relocate. As discussed in the previous principle, the user was able to use the slider to visualize the transition between the layouts. It became obvious that this transition needed to occur automatically so that the user saw the animation directly after the update calculation. In the final design of Andromeda, the low-dimensional data points animate from their current location to their updated location directly after a layout recalculation as a smooth transition to the new visualization. As discussed in Section 3, user adjusted points are encoded with a green outline. The green outline is displayed throughout the animation so that users can track the projected data points of interest to the updated locations. As this animation is happening, the slider handle animates along its track. This coordination teaches the user how to use the slider after the transition animation is complete. The animation provides a smooth transition as well as a functionality depiction.

(7) Interactive visual analytics tools should retain object persistence between model visualization updates. Each time the model recalculates, the objects relocate to a new coordinate position in the low-dimensional space. Because the WMDS model we employ is a projection of the high-dimensional space, the same weight vector

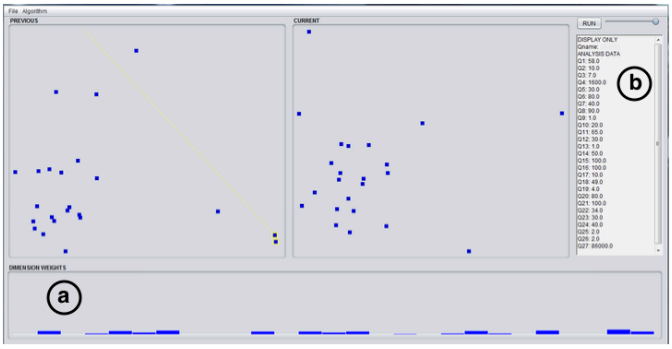

Fig. 6. Andromeda's first design displayed the parametric weights as a bar chart, shown in (a), with each dimension as a bar. Raw data values were statically displayed as seen in (b).

could produce multiple data point layouts. Even a small weight modification could produce a completely new projection or one that is rotated when compared to the original layout. According to the model, all projected layouts are correct despite rotation. However, to a user, a rotation portrays a very different layout even though the relative distances between points might match the relative distance of the original layout. To combat this confusion, it is necessary to create object persistence between layout updates. Andromeda's implementation for persistence is discussed in Section 4.3.

5.2 Parametric Visualization

In Andromeda, the parameters are the weights placed on all dimensions. We visualize each weight using a horizontal line with a handle at the end for adjusting the value of that weight. Our first iteration displayed the weights using a bar chart running along the bottom of the interface (Fig. 6a). Each weight value was represented as an interactive vertical bar within the bar chart and raw data values were displayed as static text as shown in Fig. 6b. By substituting the bars for lines and merging the lines and the raw data view, we gained more space for the layout visualization. It also more strongly connected the dimensions to the raw data for more efficient data exploration. The following principles govern the parametric visualization of the model.

(1) It is important to design an abstract way for the users to instinctively adjust the parameters without having to be experts about the model. Parametric interaction allows users to adjust the underlying parameters that define the model. Andromeda allows users to increase and decrease the dimension weights by dragging handles. In WMDS, the actual numerical value of each dimension is not necessarily useful because how the weights relate to one another is more telling. For example, if a user increases one dimension to be more highly weighted than a second dimension, she is saying that the first dimension is more important than the second. Exactly by how much does not matter as much as the relative differences between the dimension weights. Therefore, the semantic interaction of weight lines is more intuitive than typing in numerical values. Providing a visual representation of the numerical parameters may not make sense for all mathematical models, but designing an appropriate parametric interaction that decouples the interaction from the complexity of the parameters allows the user to focus on the data and not the model.

(2) Interactions must adhere to the model constraints. It is important to design tool interactions that are in keeping with the model constraints. Parameters must be contained within a feasible range of the parameter space. Dimension weights define the parameters of WMDS. WMDS requires that all weights sum to 1. Because of this constraint, the parametric interaction of increasing a weight requires the decrease of all other weights. As a visual cue, Andromeda dynamically decreases all other weights as a user increases a single weight. The model constraint is visually expressed to the user.

WMDS is also constrained by the real high-dimensional distances between the data points. However, these distances are altered when a weight is emphasized over other weights. Similarities and differences of the data points are enhanced when a weight is increased. We overlay the raw data values on the weight lines to show the relative distances between data points as the weights are adjusted. For example, in Fig. 7a the squirrel and skunk seem to have about the same level of fierceness. However, when fierceness is emphasized in Fig. 7b (i.e. the fierce dimension weight is increased) either by parametric interaction or by OLI, the skunk appears relatively more fierce than the squirrel. Since the user increased the importance of the fierce dimension, the degree to which the animals differ will become more pronounced. Overlaying the raw data onto the weight lines requires the lines to be at least a certain length so that we do not lose the ability to inspect the raw data values. Therefore, Andromeda places an arbitrary vertical gray line to denote zero weight for each dimension. For example, if we were to decrease the fierce dimension weight in Fig. 7 all the way to the left past the gray line, then we would not longer have a line on which to see the raw data values for the squirrel and skunk. Users must remember that the weight line does not disappear when the value is close to zero; the handle is merely close to the gray line.

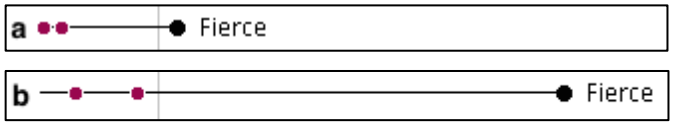

Fig. 7. (a) and (b) show the difference in raw data point placement on the weight line after the fierce dimension is increased. The two objects' relative distance in reference to fierceness has increased since that dimension was emphasized.

The type of model parameters should guide how they are displayed within the interface. In Andromeda, we display each weight parameter as a horizontal line. The number of lines displayed in the parameter view depends on the number of dimensions in the data and the physical screen space limits the number of weight lines that can be visualized. The parameter view is scrollable to allow an infinite number of visualized dimensions. However, to support fluid interactions and visualization updates, Andromeda sorts the dimension weights based on value from highest to lowest. This limits the amount of time the user has to spend scrolling through dimensions. It also places the feasibly most important weights in the user's immediate view. Also, more dimensions means lower overall weights since all weight values must sum to 1.

(3) Smooth transitions of the parameters should occur in parallel with transitions in the object view. Earlier we discussed the importance of smooth transitions between updates to the low-dimensional data point projection. Smooth transitions are also necessary for updating the parametric feedback. For example, as a single weight is adjusted, Andromeda dynamically increases or decreases all other weights to fulfil the model constraint that all weights must sum to 1. The real time animation indicates the change of all the weights to the user and eliminates a jumpy update on mouse release. If the parametric feedback is connected to the object-level feedback, then the transitions should happen in parallel. Andromeda's visualization of the weights displays a line for each dimension. After a recalculation, the weight lines grow and shrink in parallel with the animation of the data points in the object view. The user can visually track which dimension weights are increasing and which are decreasing. The animation is repeated when the user engages the slider.

(4) The interface should support dynamic parametric interaction. Interactions with parameters should happen in real time. While a user is performing an analysis, it is inefficient and interrupts cognitive processes for him to wait for the model updates. Dynamic interaction allows for a more fluid exploratory process because there is not lull. By implementing dynamic interaction, we also eliminate the need for a button that updates the layout with the new parameters. Andromeda's first iteration required the user to adjust the weights of interest and then click a button. This button was also used to update the object view which brought up challenges. Since Andromeda encompasses two separate algorithms for OLI and parametric interaction, if a user adjusted a parameter, the system disabled the object view until the parameters were updated. Disabling part of the visualization only causes confusion because the user has to pause the analysis to figure out why he cannot perform the interaction he had planned.

Modifications to the implementation of the WMDS parameter model hastened the calculation to real time speed. The final design eliminated the button for parameter updates because the real time speed supported dynamic interaction.

5.3 Algorithmic

Analytic tools with OLI capabilities have automated procedures in place to update display-generating parameters in response to

specific user interactions with visualizations. These procedures rely intimately on the models or algorithms chosen to generate the data visualizations. Some algorithms are more conducive for OLI than others. Principles of good algorithms are highlighted below.

(1) The algorithm should be invertible. Typical visual analytic algorithms rely on parameters to reduce data dimensionality for visualization purposes. Ideally, visualizations are functions of these parameters so that when visual adjustments are made, an inverted form of the function may solve for new parameter specifications.

For example, Andromeda relies on the algorithm called WMDS. In this algorithm, there are parameters $\boldsymbol{\omega}=[\omega_1,\omega_2,...,\omega_p]'$ that reflect the importance in a visualization of each variable in the p-dimensional data space. We refer these parameters $\boldsymbol{\omega}$ as weights. Low-dimensional (e.g., two-dimensional) coordinates $\boldsymbol{r}$ of high-dimensional data points $\boldsymbol{d}$ are determined based on minimizing a stress function with respect to $\boldsymbol{r}$ given $\boldsymbol{\omega}$,

$$r = \min_{r_1,\dots r_n} \sum_{i=1}^{n}\sum_{j>i}^{n} |dist_L(r_i, r_j) - dist_H(\omega, d_i, d_j)| \quad , \qquad (1)$$

where, n represents the number of data points; $dist_L(r_i, r_j)$ represents a distance measure between low-dimensional points r_i and r_j; and $\text{dist}_H(\boldsymbol{\omega}, d_i, d_j)$ represents a distance measure between high-dimensional points d_i and d_j. To be clear, r_i, r_j are low-dimensional representations of d_i, d_j. In Andromeda, $\text{dist}_L()$ is Euclidean distance and $\text{dist}_H()$ is weighted Euclidean distance,

$$dist_H(\omega, d_i, d_j) = \sqrt{\sum_{k=1}^{p} \omega_k (d_{ik} - d_{jk})^2} \quad .$$

When users either change the coordinates of some data points or highlight points to consider for an updated visualization, Andromeda inverts the optimization in Equation (1). That is, Andromeda solves for $\boldsymbol{\omega}$ given moved or selected coordinates $\boldsymbol{r}^*$,

$$\omega = \min_{\omega_1,\dots\omega_p} \sum_{i=1}^{n}\sum_{j>i}^{n} |dist_L(r_i^*, r_j^*) - dist_H(\omega, d_i, d_j)| \quad .$$

As a result, there is a clear, quantitative, and parametric interpretation of moving low dimensional coordinates in visualizations.

(2) The inverted algorithm should emphasize explicit user input. When interacting with a screen full of objects, users tend to concentrate on a small number of objects. These explicit interactions contain more information about user semantics than the other objects on the screen [24]. Some early OLI systems opt for considering all objects in the projection with equal weights [6]. This approach may distort the interpretation by introducing too much noise. OLI systems, on the other hand, always allocate more attention toward objects with which users have specifically interacted. This approach: (1) increases the likelihood of correctly identifying the semantics and (2) reduces the computational burden because it lessens the number of objects the model considers.

(3) The inverted algorithm should consider implicit user input. Objects that are not directly manipulated by the user may still express user semantics. For example, a user may decide to move some objects toward a reference point (say Object A) in order to express similarity. Object A will be unmoved during the interaction, but object A is still of high value in understating the user semantics. Identifying these implicit objects is a tricky task. Two approaches should be considered: (1) provide appropriate tools to assist users in being more explicit about their semantics (i.e. highlighting in V2PI-MDS) and (2) nominate objects in a close vicinity of an explicitly interacted object and allow the users to confirm or overrule these suggestions (i.e. Andromeda).

(4) The algorithm should be fast. To enable users to explore data in parallel with how they are thinking about the data, the selected display-generating algorithm must be computationally expedient. That is, both the standard algorithm and its inverted form should operate in real time, or as close to real time as possible. When we first started developing Andromeda we explored several optimization schemes and opted to invoke a general purpose gradient descent algorithm [25]. In our case, the inverted form of WMDS is much slower than its standard form; i.e., solving for $\boldsymbol{\omega}$ given $\boldsymbol{r}^*$ is slower than solving for $\boldsymbol{r}$ given $\boldsymbol{\omega}$. Developing a speedy optimizer in both directions is an active research area, however, recent advances have led us to a Quadratic Programming solution to an approximated objective function, which greatly reduces computation times.

(5) The algorithm should be deterministic. Crucial to OLI is that users may create multiple visualizations in a sequence that parallels their sense making process. Thus, random perturbations in visualizations may confuse users; changes to visualizations should reflect added information provided by user interactions. If no information is added by a user, an "updated" visualization should not change. Thus, stochastic algorithms or optimization schemes that may get stuck in varying locations due to function multi-modality may not be appropriate for OLI software, unless added precautions or steps are taken. For example, WMDS is invariant to scale, rotation, and reflection. Thus, when solving for $\boldsymbol{r}$ given $\boldsymbol{\omega}$ (Equation (1)), it is possible to have reflected, rotated, or scaled versions of one data visualization; e.g., if the Update Layout button is hit twice, Andromeda may produce reflected versions of low-dimensional coordinates. To overcome this problem, Andromeda takes an extra processing step to align and scale coordinates after solving for $\boldsymbol{r}$. As a result new information in sequential visualizations is not masked by arbitrary mathematical properties of WMDS.

6 Conclusion

We formulated design principles for visual analytics tools encompassing multiple views and ways of interacting with mathematical models for exploring high-dimensional data. Specifically, the interactive layout visualization should display the objects and encode both user input to the model and model outputs to enhance the user's understanding of the model and how to interact with it. The interactive parameter view should be designed in a way that keeps the integrity of the model intact. We described the necessary principles to consider when designing the model itself as well as how it relates to the interface. Designers should consider all three categories of principles to fully understand the impact and interconnectivity of design choices within an interface.

We discussed the benefits of OLI and the important role OLI plays in a well-designed visual analytics interface for exploring high-dimensional data. We stressed the importance of including both parametric interaction and OLI. With both types of interaction, a user is able to gain more complex insights and accomplish new types of tasks.

In the future, we hope to see how these principles apply to other dimension reduction models so that they too are accessible to users without strong model knowledge.

Acknowledgments

Omitted for blind review.

References

[1] A. Endert, C. Han, D. Maiti, L. House, S. Leman, and C. North, "Observation-level interaction with statistical models for visual analytics," in *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2011, pp. 121–130.

[2] S. C. Leman, L. House, D. Maiti, A. Endert, and C. North, "Visual to Parametric Interaction (V2PI)," *PLoS One*, vol. 8, no. 3, p. e50474, Jan. 2013.

[3] L. House, S. Leman, and C. Han, "Bayesian Visual Analytics: BaVA," *Stat. Anal. Data Min. ASA Data Sci. J.*, vol. 8, no. 1, pp. 1–13, 2015.

[4] J. B. Kruskal and M. Wish, "Multidimensional Scaling," *Sage Univ. Pap. Ser. Quant. Appl. Soc. Sci.*, 1978.

[5] J. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[6] E. Brown, J. Liu, C. Brodley, and R. Chang, "Dis-Function: Learning Distance Functions Interactively," *IEEE Conf. Vis. Anal. Sci. Technol.*, pp. 83–92, 2012.

[7] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang, "iPCA: An Interactive System for PCA-based Visual Analytics," *Comput. Graph. Forum*, vol. 28, pp. 767–774, 2009.

[8] PNNL, "IN-SPIRE Visual Document Analysis." 2010.

[9] J. S. Yi, Y. A. Kang, J. T. Stasko, and J. A. . Jacko, "Toward a Deeper Understanding of the Role of Interaction in Information Visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 6, pp. 1224–1231, 2007.

[10] T. Munzner, *Visualization Analysis and Design*. CRC Press, 2014.

[11] B. Schneiderman and C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 4th ed. Reading, MA: Pearson Addison Wesley, 2005.

[12] S. Card, J. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*. Academic Press, 1999.

[13] J. J. Thomas and K. a Cook, "Illuminating the path: The research and development agenda for visual analytics," *IEEE Comput. Soc.*, vol. 54, p. 184, 2005.

[14] M. Gleicher, "Explainers: expert explorations with crafted projections.," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 2042–51, Dec. 2013.

[15] P. Joia, F. V. Paulovich, D. Coimbra, J. A. Cuminato, and L. G. Nonato, "Local Affine Multidimensional Projection," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2563–2571, 2011.

[16] E. Portes, E. V. Brazil, L. G. Nonato, and M. C. Sousa, "iLAMP : Exploring High-Dimensional Spacing through Backward Multidimensional Projection," *IEEE Conf. Vis. Anal. Sci. Technol.*, pp. 53–62, 2012.

[17] E. Kandogan, "Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions," *Proc. IEEE Inf. Vis. Symp. Late Break. Hot Top.*, vol. 650, pp. 9–12, 2000.

[18] J. Alsakran, Y. Chen, Y. Zhao, J. Yang, and D. Luo, "STREAMIT: Dynamic visualization and interactive exploration of text streams," *IEEE Pacific Vis. Symp. 2011, PacificVis 2011 - Proc.*, pp. 131–138, 2011.

[19] J. S. Yi, R. Melton, J. Stasko, and J. a Jacko, "Dust & Magnet: multivariate information visualization using a magnet metaphor," *Inf. Vis.*, vol. 00, no. April, pp. 239–256, 2005.

[20] I. Jolliffe, *Principal Component Analysis*, 2nd ed. John Wiley and Sons, Ltd, 2002.

[21] A. Group, "MDSJ: Java Library for Multidimensional Scaling (Version 0.2)." University of Konstanz, 2009.

[22] C. H. Lampert, H. Nickisch, S. Harmeling, and J. Weidmann, "Animals with Attributes: A Dataset for Attribute Based Classification," 2009. [Online]. Available: http://attributes.kyb.tuebingen.mpg.de/.

[23] A. Endert, P. Fiaux, and C. North, "Semantic Interaction for Sensemaking: Inferring Analytical Rea-soning for Model Steering," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 12, pp. 2288–2879, 2012.

[24] X. Hu, L. Bradel, D. Maiti, L. House, C. North, and S. Leman, "Semantics of Directly Manipulating Spatializations," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 2052–2059, 2013.

[25] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. New York: Cambridge University Press, 1992.