AxiSketcher: Interactive Nonlinear Axis Mapping of Visualizations through User Drawings

Bum Chul Kwon, Hannah Kim, Emily Wall, Jaegul Choo, Haesun Park, and Alex Endert

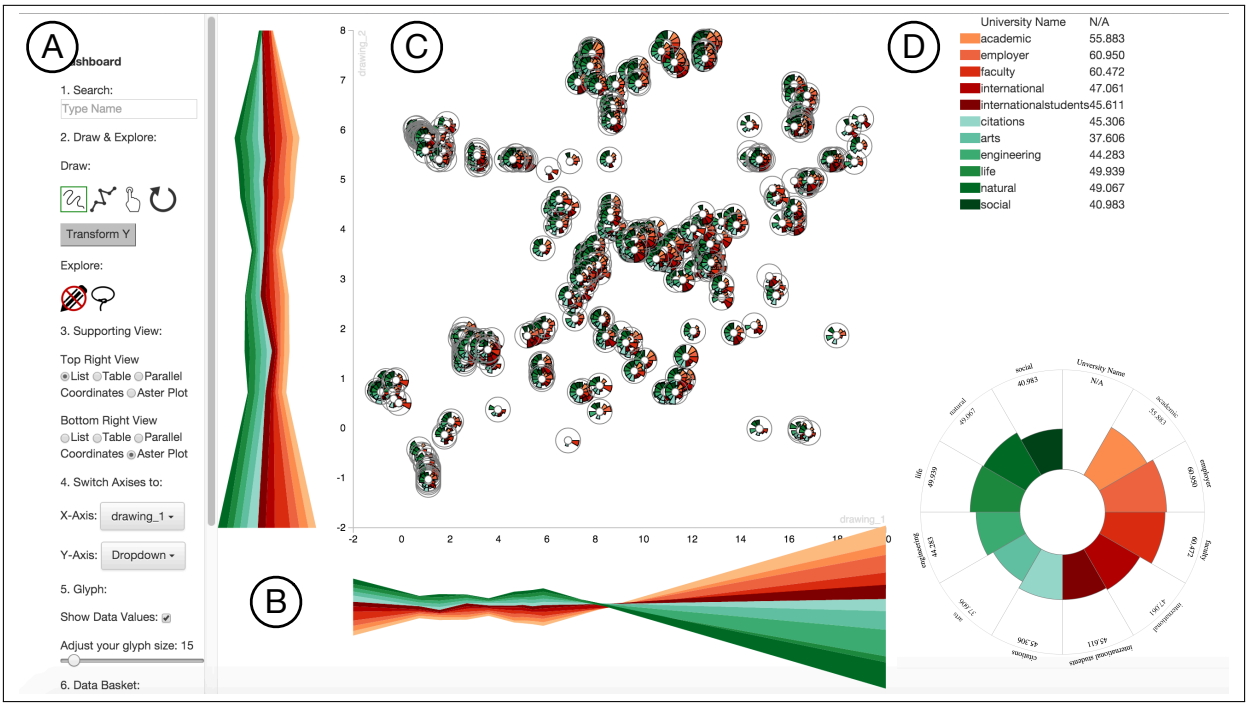


Fig. 1: An overview of our visual analytics system, called AxiSketcher. (a) Dashboard allows users to choose a sketching method, to change axes to previously constructed axes, and to change other peripheral settings; (b) Axis Rainbow visualizes nonlinear progression of data attributes in ThemeRiver-style visualizations; (c) Scatterplot shows data points mapped to user-defined axes; (d) Detail View shows the original attribute values of selected data items as a table (top) and an aster plot (bottom).

Abstract— Visual analytics techniques help users explore high-dimensional data. However, it is often challenging for users to express their domain knowledge in order to steer the underlying data model, especially when they have little attribute-level knowledge. Furthermore, users' complex, high-level domain knowledge, compared to low-level attributes, posits even greater challenges. To overcome these challenges, we introduce a technique to interpret a user's drawings with an interactive, nonlinear axis mapping approach called *AxiSketcher*. This technique enables users to impose their domain knowledge on a visualization by allowing interaction with data entries rather than with data attributes. The proposed interaction is performed through directly sketching lines over the visualization. Using this technique, users can draw lines over selected data points and the system forms the axes that represent a nonlinear, weighted combination of multidimensional attributes. In this paper, we describe our techniques in three areas: 1) the design space of sketching methods for eliciting users' nonlinear domain knowledge; 2) the underlying model that translates users' input, extracts patterns behind the selected data points, and results in nonlinear axes reflecting users complex intent; and 3) the interactive visualization for viewing, assessing, and reconstructing the newly formed, nonlinear axes.

Index Terms—axis mapping, interactive model steering, sketch, axis visualization, human-centered visual analytics

1 Introduction

- *Bum Chul Kwon is with IBM T.J. Watson Research Center in Yorktown Heights, NY, USA. E-mail: bumchul.kwon@us.ibm.com*
- *Hannah Kim, Emily Wall, Haesun Park, and Alex Endert are with Georgia Institute of Technology, Atlanta, GA, USA. E-mail: {hannahkim, emilywall, hpark, endert}@gatech.edu*
- *Jaegul Choo, the corresponding author, is with Korea University in Seoul, South Korea. E-mail: jchoo@korea.ac.kr*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.

Visual analytic techniques help to facilitate analytical reasoning and decision-making in the presence of data [25, 26, 46]. By providing people with interactive data visualizations created using powerful analytic and computational models, people can leverage their perceptual and cognitive abilities to explore the data, reason about it, and ultimately arrive at insights and new knowledge [38]. People use visual analytics for a variety of tasks, including data exploration, intelligence analysis, sensemaking, and decision-making.

For visual analytic techniques, the careful combination of analytic model, visualization technique, and interaction paradigm is central to

Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

ensuring the systems usability and effectiveness [16]. Through such systems, people are able to not only observe the visual output of models, but are also able to interactively explore the underlying data. Such "human-in-the-loop" approaches are central to fostering interactive data exploration in visual analytics, promoting greater insight.

Model steering is way to incorporate users domain expertise into visual analytics through user interaction. This method entails giving the user control over parameters of the analytic model, augmenting the computation in specific, user-defined ways. The complexity of the models included in these systems challenges the community to derive new paradigms for user interaction in visual analytics that ensure usability, trust, and interpretability [10,37]. Methods for steering axes of scatterplots generated from dimension reduction techniques have been proposed [5, 14, 15, 21, 33]. The method most similar to that of this work is *InterAxis* [27]. This visual analytic technique allows people to steer and leverage dimension reduction models to create custom scatterplot axes that are formed by linear combinations of data attributes. Users can form an axis that represents a weighted combination of multiple data attributes that comprise a higher-level concept. For example, if one analyzes a dataset about cars, an axis might include a weighted combination of attributes such as price, horsepower, and weight to represent "muscle cars." In general, visual analytic techniques like *InterAxis* help people form such concepts explicitly as a scatterplot axis.

However, what if the concepts people try to represent are too complex to be represented as a fixed weighted combination of attributes? How can we adapt the user interaction and model steering methods for more complex, nonlinear models? In addition to the complexity of the user-generated concept to be translated to an axis, the characteristics of the data may be better suited to a nonlinear model.

That is, a nonlinear approach can allow users to construct much more flexible representational models, with varying weight values along an axis. For instance, a user's preference regarding cars may vary depending on the price range. For expensive cars, the user may find that design is very important, while for less expensive cars, the user may consider fuel economy to be far more significant. This preference cannot be adequately modeled using linear techniques. Furthermore, some high-level concepts cannot be represented via a linear model by using only the original low-level attributes. For example, creating an axis representing a variety of facial expressions with low-level pixel values would inherently require a nonlinear model.

Even with these superior capabilities, interactively mapping nonlinear models to scatterplot axes raises new challenges over existing techniques for creating and steering linear models for axes (e.g., *InterAxis*). First, intuitive, flexible methods are needed that allow users to elicit their preferences for generating the nonlinear models for a new axis. Second, the nonlinear models along axes must be visualized so that users can correctly understand the newly formed axes. Resolving these challenges will help users construct, evaluate, and edit nonlinear models interactively.

In this paper, we present *AxiSketcher*, a sketch-based interaction technique for creating flexible, nonlinear axes in a user-driven manner. *AxiSketcher* allows users to create, understand, and refine nonlinear axes and use them for in-depth data analysis and decision-making tasks. We present novel user interactions for creating nonlinear axes, e.g., drawing lines passing through multiple data points and drawing freeform lines on the scatterplot. Next, we propose novel algorithms and models that generate nonlinear axes given such interactions. In addition, to facilitate understanding of the model output and further iterative steering of the model, we develop a ThemeRiver-style visualization [20] (called *Axis Rainbow*) along each axis, which allows users to understand and change the relative importance of each attribute across different regions along the axis.

The primary contributions of this work include:

1. A visual analytic system where a user can create, visualize, and adjust flexible nonlinear models as scatterplot axes.
2. A bi-directional ThemeRiver visualization technique, called *Axis Rainbow*, for displaying nonlinear models on axes.
3. Connect-the-dot and freeform line drawing interactivity in scatterplots, with which users can create and steer a nonlinear axis.

In this paper, we provide an overview of the nonlinear axis mapping process in Section 3. In Section 4, we present technical details of the methods that help users interactively create, evaluate, and edit the nonlinear model. Then, we discuss the implementation of *AxiSketcher* in a visual analytics system with additional visualizations and interactivity to support the interactive model steering process in Section 5. In Section 6, we demonstrate the utility of *AxiSketcher* through several use cases in three scenarios. Finally, we discuss takeaways and the implications of this work in Section 7.

2 Related Work

In this section, we discuss previous work on dimension reduction methods and interaction techniques that inspired us to build a new interactive nonlinear axis mapping technique.

2.1 Manifold Learning

Unlike traditional dimension reduction methods, such as principal component analysis [23] and multidimensional scaling [28], manifold learning is a class of nonlinear dimension reduction techniques [32] that tries to reveal the low-dimensional curvilinear structure of high-dimensional data. Such a curvilinear structure is often composed of the intrinsic dimensionality of the data, which is often semantically more meaningful than the original dimensions. Manifold learning finds such intrinsic dimensions and maps data into those dimensions. For instance, in facial image data, intrinsic dimensions can include viewing angles (from left to right), face sizes (small to big), and facial expressions (from smiling to frowning), whereas the original dimensions of the image include a set of pixel values of an image. Representative methods include isometric feature mapping [45], locally linear embedding [39], and Laplacian Eigenmaps [4].

However, the applicability of these methods to various real-world data has been limited due to numerous issues, including high levels of noise and scarcity in data. Considering these issues in practice, one cannot easily expect a nonlinear dimension reduction method to nicely map our high-level notions to low-dimensional representations in a fully automated manner.

Modifications to this method have been developed with great success. One class of techniques, principal curves and surfaces, restricts the intrinsic nonlinear dimensionality to one or two dimensions [12, 19]. Kegl et al. later proposed an improved algorithm that models a principal curve as a polygonal line with a length constraint [24]. From a practical perspective, these principal curve techniques can give a much more faithful representation of data, revealing a single meaningful intrinsic dimension at a time, rather than trying to find all intrinsic dimensions at once, as with the manifold learning methods.

The technique for one-dimensional principal curves is a natural fit for our proposed user interactions. User interactions can allow users to progressively construct new axes, as discussed in Williams and Munzner [51], to project high-dimensional data onto 2D space. These interactions include allowing users to draw lines on a visualization using multiple different techniques. Once a user draws a line along some data pattern, the principal curve technique can help to recover the major underlying pattern in the high-dimensional data based on a user-drawn line in a 2D scatterplot visualization. In this sense, the backend computational technique that we propose in this paper can be viewed as a semi-supervised principal curve approach using natural forms of user interactions in 2D scatterplots.

2.2 Interaction Techniques in Scatterplots

A scatterplot is a commonly used visualization technique for multidimensional data analysis [11, 17, 48]. The two axes of a 2D scatterplot each show either one of the original data attributes or a newly computed dimension generated using a dimension reduction method. In the case of nonlinear dimension reduction methods, the axes are sometimes not defined at all, and only the relative distances between data items convey information about their original relationships.

Scatterplots have been used as one of the main visualizations in numerous data and visual analytics systems, e.g., *Tableau* [44], *GGobi* [43], and *Spotfire* [1]. Standard interactions supported in these

systems include the ability to choose particular attributes to assign to the x- and y-axes, brushing and linking with other views, and showing details on demand for selected points. Other systems, e.g., *iPCA* [22] and testbed [9] focused on changing the specification of the dimension reduction methods that generate the resulting scatterplot.

Various types of novel interactions with scatterplots have also been proposed. For example, Yi et al. [53] proposed a technique called Dust & Magnet, which interactively places data attributes on a scatterplot. The attributes act like magnets, attracting dust (data items) with large values in the corresponding attributes. Another type of interaction, semantic interaction [13], allows for the direct manipulation of data points in a scatterplot, where an underlying dimension reduction method updates its model parameters accordingly [5, 6, 15].

Myriad other interaction techniques for scatterplots have been studied, many of which are tightly coupled with specific application domains, e.g., image classification [8], document topic modeling [2, 7] and bibliographic analysis [54].

Our proposed interaction technique differs from existing techniques in that ours reveals sophisticated nonlinear patterns hidden in high-dimensional data in a user-driven manner. Our technique then explicitly maps such patterns to an axis on the scatterplot.

2.3 Sketch-based Interaction Techniques

Sketch-based interaction techniques are part of a larger movement toward natural user interfaces. Sketching is an intuitive form of interaction with an interface and a growing trend in both information visualization and scientific visualization [42]. Sketching also provides an effective medium with which to elicit users' insights at various levels of abstraction [51]. In scientific visualization, sketching has been used to modify volumetric visualizations by allowing users to erase attributes and modify properties such as color, transparency, image clarity, contrast, and brightness, all with a few strokes of a pen [18]. Sketching has also been used to generate vector fields, allowing users to draw streamlines and quickly simulate fluid flows [40, 50]. In another example, Natali et al. used sketching to create 3D geological models that track the evolution of an area based on the shape of its land and waterways [36].

Sketch-based interaction techniques are also becoming increasingly popular in information visualization. Wang et al. utilized a sketching technique to generate and modify high-dimensional data in an intuitive manner [49]. In addition to providing a natural interaction technique, sketching also provides a different user experience. Schumann et al. conducted a study showing that sketch-based renderings provide a higher level of engagement in the early stages of computer-aided design [41]. Based on this idea, Wood et al. created a processing library to render real data in the style of hand-drawn sketches [52]. In another example, Lee et al. used sketching to create visualizations for storytelling [31]. SketchSliders show sketch-based data exploration methods on mobile devices [47].

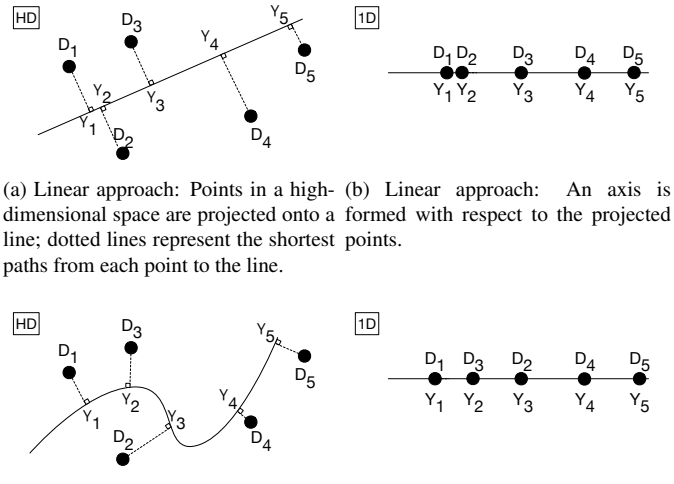
Inspired by the prior research studies, *AxiSketcher* utilizes sketch-based interactions as an intuitive means for users to create a new axis by drawing a line passing near data points of interest. Specifically, our system allows users to form new complex, nonlinear axes that reflect their intent through a natural form of user interactiondrawing via a pen or cursor on a scatterplot.

3 Overall Workflow of User-Driven Axis Mapping

In this section, we describe the design goals and the overview of *AxiSketcher* and describe its novelty by comparing it to the previous linear technique, InterAxis.

3.1 Overall Workflow

The main contribution of this paper is the proposal of interaction techniques that allow users to create scatterplot axes that reflects the user's mental model. As Fig. 2 shows our primary task, creating a user-driven axis, which can be described as follows. First, through particular forms of user interactions (described later), we obtain a line in the high-dimensional space of the original data. This line can be a simple straight line (Fig. 2(a)) or a complex curve (Fig. 2(c)). In an either



(a) Linear approach: Points in a high-dimensional space are projected onto a line; dotted lines represent the shortest paths from each point to the line.

(b) Linear approach: An axis is formed with respect to the projected points.

(c) **AxiSketcher**: Points in a high-dimensional space projected onto a curve modeled by a piecewise line; dotted lines represent the shortest paths from each point to the line.

(d) **AxiSketcher**: An axis is formed with respect to the projected points.

Fig. 2: Comparison of the axis mapping processes between a linear approach and **AxiSketcher**

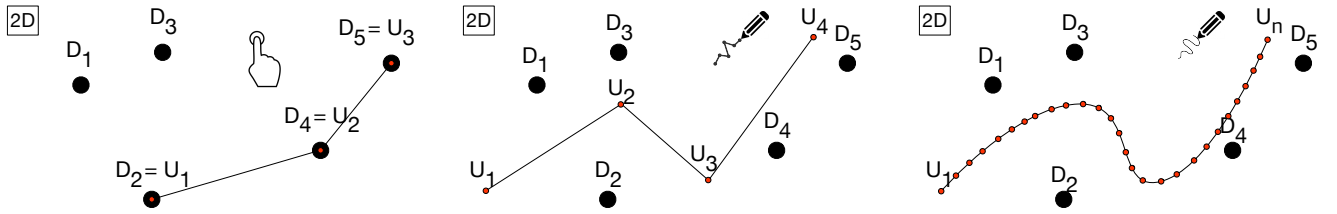
case, the line represents an increasing pattern with respect to the user's intended model, e.g., a user's preference regarding cars, along a particular direction of the line. Second, each data point is projected onto this line in high-dimensional space, and the projected point in this line determines the coordinate of the data point along this line (Figs. 2(a) and (c)). Finally, this line, along with the data coordinate information, becomes our new scatterplot axis (Figs. 2(b) and (d)).

Backend and frontend perspectives. In fact, this process involves several important design choices and considerations from the two perspectives: a backend model and a frontend user interaction. That is, (1) the backend part poses the question of how we model such a line and what information we should require from a user to construct the line. On the other hand, (2) the frontend part presents the issue of what kind of user interactions will provide the required information so that the model can properly reflect a user's intent.

3.2 Previous Linear Approaches

There exist previous approaches to this form of user interaction for linear models. Most similarly, Interaxis [27] takes a simple but effective approach for allowing users to create an axis of a scatterplot in a user-driven manner. In particular, Interaxis uses a linear model in the backend, where a line is formed between two points in a high-dimensional space. In the frontend, the required user interaction to specify this line is performed by selecting two data items that determine a line passing through them. That is, when a user selects two data items, whose high-dimensional vector representations are $a_1 \in \mathbb{R}^m$ (positive end) and $a_2 \in \mathbb{R}^m$ (negative end), respectively, the direction of the resulting line l is formed as $l = a_1 - a_2$. Afterwards, data items are projected onto this line, and their coordinate values are determined accordingly. Mathematically, the new coordinate value $y_i \in \mathbb{R}$ of a data item $a_i \in \mathbb{R}^m$ along the axis formed by l is defined as $y_i = l^T a_i = (a_1 - a_2)^T a_i$. This inner product between $a_1 - a_2$ and a_i can be viewed as a weighted combination of the differences between the two points a_1 and a_2 in original dimensions, where the individual value of a_i in each dimension works as a weighting coefficient in the corresponding dimension. In this manner, the resulting axis and the corresponding coordinate values of data items are formed such that the differences between the two selected data items are highlighted.

As seen in this model, the main limitation of the linear approaches is that it can use only a fixed set of weight values to describe a user's mental model throughout the entire region of an axis.



(a) **Object Selection**: a series of segmented lines connecting data points. Vertices of this line are determined by directly selecting data points.

(b) **Polyline**: a series of segmented lines. The low-dimensional vertices are determined based on where the user clicks in the scatterplot.

(c) **Freeline**: a curved line that represents a series of segmented lines. Low-dimensional vertices are determined by periodically sampling points along the curve.

Fig. 3: Three different methods for drawing lines over data points and computing vertices of the user-drawn lines.

4 AxiSketcher: Interactive Nonlinear Axis Mapping

Our proposed nonlinear axis mapping technique is more generalized than previous linear techniques and significantly advances the task of creating a user-driven axis, from both backend and frontend perspectives. In this section, we present our design goals and the interaction techniques used to fulfill them, as well as the underlying algorithms for constructing a nonlinear model from user interactions.

4.1 Design Goals

We summarize the four major goals of AxiSketcher as follows:

G I. Allow the creation of a nonlinear axis by clicking on data points. Users should be able to click on individual data points in sequence on the scatterplot. The system should then interpret the sequence of points in high-dimensional space to create an axis that reflects the given sequence. Other data points should be mapped onto this new axis such that points that are nearby are similar to each other.

G II. Allow the creation of a nonlinear axis by drawing on the visualization. Users should be able to create an axis by drawing directly onto the scatterplot. This should be possible by clicking on the scatterplot to create a piecewise line or by drawing freely on the scatterplot. The system should then interpret these interactions in high-dimensional space to create a corresponding axis.

G III. Visualize attribute contributions along a nonlinear axis. Once an axis has been created, the system should visualize the attribute contributions along the axis. Because the axes do not use a linear model, attribute contributions are not uniform along an axis. Consequently, traditional visualizations of attribute contributions for axes (e.g., bar charts) are not suitable.

G IV. Allow modification of attribute contributions along an axis. Users should be able to modify the attribute contributions along an existing axis. In addition, the user should be able to specify a position along the axis at which to apply modification.

4.2 Sketch-based Methods to Capture User Input

From a frontend perspective, we propose sketch-style user interaction techniques that specify nonlinear curves in a user-driven manner. In general, it is a challenging task for a user to specify a nonlinear curve in a high-dimensional space so that it properly reflects her intent. Furthermore, a critical challenge is that user interactions should be performed in low-dimensional visualization space, while we have to infer from such interactions a nonlinear curve in high-dimensional space.

To overcome this issue, we propose three different techniques that allow user-friendly, sketch-style interactions, as shown in Fig. 3.

Selecting points. The first method is the most straightforward of the three, in which a user selects a sequence of data points in a scatterplot (Fig. 3(a)). Once the sequence of data points is obtained, we form a polyline that connects through these points in the same order in a high-dimensional space. Such a high-dimensional polyline works as an approximation of a curve (Fig. 2(c)). Through this interaction, we interpret the given sequence of data points as a monotonically increasing (or decreasing) pattern following a user's intended mental model, e.g., an increasing (or decreasing) user preference regarding a car.

Notation	Description
n	The number of data points
d_i	The ith **d**ata point
$U = (u_1, \cdots, u_m)$	The **u**ser-constructed polyline of a sequence of m vertices $u_1, \cdots, u_m$
u_i	The ith vertex of **u**ser-constructed polyline U
y_i	The projected coordinate value of the ith data point
p^h	The attribute values of point p in high-dimensional space
p^l	The 2D coordinate values of point p in scatterplot

Table 1: Notation description for equations used in the paper.

Drawing a polyline. The second interaction technique we propose for specifying a high-dimensional line is to draw an arbitrary polyline in a scatterplot (Fig. 3(b)). This interaction type is similar to the previous one in that the user's interaction forms a polyline in a scatterplot, which is, in turn, used to reconstruct the polyline in a high-dimensional space. However, this interaction type is not restricted to a polyline connecting data points, but instead allows users to freely draw a polyline anywhere in the scatterplot. This enhanced flexibility can be useful when a user wants to specify her intended sequence of patterns based on aggregated patterns of multiple data items, which are not exactly represented via a single data item. For example, once a user finds a general pattern in a particular region of the scatterplot, e.g., a group of cars with relatively high fuel economy and a small size, she can draw a polyline starting from this region rather than having to select a discrete data point.

Drawing a freeform line. The third interaction technique we propose is to draw a freeform curve in a scatterplot (Fig. 3(c)). This interaction can be viewed as a straightforward extension of the second interaction, in which the polyline is formed by sampling a large number of points along the curve drawn by the user. This interaction type provides a more natural, sketch-style user experience than the previous two, since it can be performed by simply drawing a freeform line.

All three interactions we describe here provide the information required to specify a line in the original high-dimensional space; the details of this process are described in the next section.

4.3 Algorithms for Inferring High-Dimensional Lines

In this section, we describe the underlying algorithms for inferring a high-dimensional line from the three different interactions that we just described: (1) selecting points, (2) drawing a polyline, and (3) drawing a freeform line. Based on these interactions, the main goal of this step is to obtain a user-driven line as a sequence of high-dimensional points $U^h = \left(u_1^h, u_2^h, \cdots, u_m^h\right)$. Please refer to Table 1 for descriptions of our notation.

Selecting points. Suppose a user selects a sequence of m data points represented as two-dimensional vectors in a scatterplot, for example, $U^l = \left(d_{u1}^l, d_{u2}^l, \cdots, d_{um}^l\right)$. Each of these selected data points,

e.g., d^l_{ui}, has a high-dimensional representation, d^h_{ui}. Thus, we construct the high-dimensional line as $U^h = \left(d^h_{u1}, d^h_{u2}, \cdots, d^h_{um}\right)$.

Drawing a polyline. In this interaction, a user selects an arbitrary polyline, e.g., a sequence of two-dimensional points in a scatterplot that need not overlap with data points (Fig. 3(b)). Here, let us denote the set of these two-dimensional vector representations as $U^l = \left(u^l_1, u^l_2, \cdots, u^l_m\right)$, where m is the number of selected points forming the polyline. In order to construct the corresponding high-dimensional polyline, we first determine the high-dimensional vector u^h_i corresponding to u^l_i for $i = 1, \cdots, m$. To this end, we utilize the k-nearest neighbors approach as follows. For each point u^l_i, we identify the k nearest data points in a 2D scatterplot, for example, $\left\{d^l_{i1}, d^l_{i2}, \cdots, d^l_{ik}\right\}$. We denote the corresponding high-dimensional vector of data item d^l_{ij} as d^h_{ij}. We define the high-dimensional representation u^h_i of each point in U^h as a weighted linear combination of the high-dimensional representations of the nearest data items to u^l_i. In this step, we assign more weight to those data items closer to u^l_i in a scatterplot, since it is likely that a user considered close data items more significantly when drawing a polyline. Specifically, we compute u^h_i as

$$u^h_i = \sum_{j=1}^{k} w_{ij} d^h_{ij}, \text{ where } w_{ij} = \frac{1/(u^l_i - d^l_{ij})^2}{\sum_{j=1}^{k} 1/(u^l_i - d^l_{ij})^2}.$$

We then assign $U^h = \left(u^h_1, u^h_2, \cdots, u^h_m\right)$.

Finally, we adjust the resulting high-dimensional line via an automatic curve-fitting algorithm in a high-dimensional space, called principal curve [24]. That is, we perform this automatic curve-fitting using as input the set of data points used as the k-nearest neighbors, d^h_{ij}'s, of any points in U^l. The algorithm returns its own high-dimensional polyline as an output, which reflects the actual manifold in which the high-dimensional data lie. Afterwards, we compute the weighted average between the two polylines: the polyline generated from curve-fitting and the one generated by user interaction. In this manner, we make our high-dimensional polyline inference process robust and reliable by considering the principal manifold in which the data actually lie.

Drawing a freeform line. When a user draws a freeform line in a scatterplot, we convert it to an approximate polyline using m equidistant points along the freeform line. That is, suppose that the two-dimensional vectors of these m points in the converted polyline are $U^l = \left(u^l_1, u^l_2, \cdots, u^l_m\right)$. Afterwards, the rest of the process is the same as that in the previous case of "drawing a polyline."

4.4 Projecting Data onto a High-Dimensional Line

Projection algorithm. Given that a high-dimensional line is identified in the previous step, we perform the projection of other data items onto the closest point on the line, as shown in Fig. 2(c). To this end, we form a high-dimensional polyline $f\left(U^h\right)$ connecting the points in U^h sequentially. Additionally, we extend the leftmost and rightmost parts of the polyline as rays, which go to either infinity, i.e., $\overleftarrow{u^h_1 u^h_2}$ and $\overrightarrow{u^h_{m-1} u^h_m}$, respectively. In other words, $f(U^h) = \overleftarrow{u^h_1 u^h_2} \cdot \overline{u^h_2 u^h_3} \cdots \overrightarrow{u^h_{m-1} u^h_m}$. The projection of a point onto $f\left(U^h\right)$ is performed via an orthogonal projection onto each of the piecewise line segments in $f\left(U^h\right)$, and among multiple projected points, we choose the one that has the smallest distance from the original point to the projected one. Once the projection for each data point is done, we straighten the line that forms the new axis for use in our scatterplot (Fig. 2(d)). Here, we consider the points corresponding to u^h_1 and u^h_n as the low and high ends of the axis, respectively. Accordingly, the final coordinate value y_i of each data point d_i is computed as the distance from u^h_1 to its projected points along $f\left(U^h\right)$. Intuitively, if two points are close to each other in terms of their projection onto $f(U^h)$, they are similar with respect to the new axis.

Projection uncertainty. In this process, we consider the approximation error due to the projection of a data point onto $f(U^h)$, which can be defined as the distance between the original data point and its projected point onto $f(U^h)$. From the user's perspective, a high uncertainty value, or a large approximation error, indicates that the corresponding point is far from the axis in high-dimensional space; thus, we are less confident about the accuracy of its mapping onto the axis as an indication of its similarity to nearby points on the axis. We visually encode such uncertainty values via an alpha channel, so that uncertain data points are visualized with high transparency.

4.5 Visualizing the Axis

We next discuss how to visualize the newly formed axis from Section 4.4. As previously discussed, the new axis contains piecewise line segments between vertices with high-dimensional attribute values (Fig. 4b). We visualize the piecewise linear progression of attribute values by drawing bi-directional ThemeRiver-style visualization, which we call *Axis Rainbow*. As Fig. 1 shows, the Axis Rainbow is drawn next to its corresponding axis: on the left side of the y-axis and below the x-axis. The thickness of each layer shows the values of each attribute at a particular point along the axis. By inspecting how the thickness of a layer changes along the axis, the user can understand how the given attribute value changes from points A to B on the axis. Each layer is highlighted as the user hovers over it; the other layers become semi-transparent with an opacity of 0.1. The user can see axis attribute values using the Axis Rainbow by hovering over the Axis Rainbow, which then shows the value of a data attribute at a particular point.

4.6 Refining the Axis

A user can refine an axis by directly interacting with the Axis Rainbow. For example, the user can initiate the context menu by right-clicking a particular point in the Axis Rainbow. The pop-up menu contains an aster plot showing the attribute values at that point (see Fig. 4c). The user can click-and-drag the wedges of the aster plot to increase or decrease the attribute values at that point. As the user completes the modification and clicks the "update" button, the system sends the new point to the axis mapping algorithm in the backend, which then updates the Axis Rainbow accordingly (see Fig. 4d).

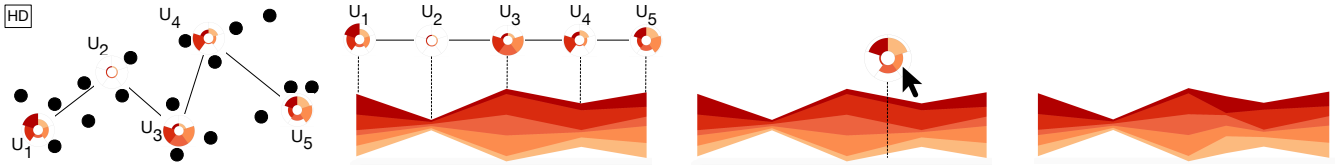
In detail, a new vertex is created at the position at which the user clicks, e.g., u_* between the adjacent vertices u_i and u_{i+1} in a high-dimensional space. The attributes of the new vertex u_* are determined by user specification of the attributes in the aster plot. The attributes that are not specified by the user are interpolated using the values of the corresponding attributes of u_i and u_{i+1}. Accordingly, the piecewise definition of the axis $f(U^h)$ is updated to accommodate the new vertex u_*. At the same time, the segment between u_i and u_{i+1} is removed, and the two new segments are created between u_i and u_* and between u_* and u_{i+1}. In the case of $i = 1$ or $i = m$, no segment is removed from $f(U^h)$, and only one segment is added: between u_* and u_1 or between u_m and u_*, respectively.

5 Designing a Visual Analytics System for Interactive Axis Mapping

Here, we build a visual analytics system that incorporates *AxiSketcher* with additional views and functions for multidimensional data analysis (henceforth called the AxiSketcher system).

5.1 Tool Overview

Fig. 1 shows the overview of the *AxisSketcher* system. The AxiSketcher interface consists of three components: 1) Interaction Dashboard, 2) Main View, and 3) Detail View. The Interaction Dashboard contains settings to modify the Main View and the Detail View. For example, the user can change the glyph representations in the Main View or set the Detail View such that the show parallel coordinates are shown. The Main View allows the user to draw sketches over the scatter plot, explore data points, and analyze relationships between the data points; furthermore, the user can interact with the *Axis Rainbow*,



(a) **Compute Attribute Values**: We compute a distance-weighted average of k-nearest neighbor points for vertices.

(b) **Construct Axis Rainbow**: We find the piecewise linear progression of attribute values between vertices and draw area charts per each attribute.

(c) **Refinement**: a user can directly modify attribute values of a certain point along Axis Rainbow.

(d) **Resulting Axis Rainbow**: Axis Rainbow is reconstructed based on the user's input.

Fig. 4: Nonlinear axis construction and refinement process

which visualizes the newly generated axis based on the user's sketch. The Detail View shows additional details of the data points that are hovered over or selected in the Main View. In the following sections, we discuss each view in more detail.

5.2 Interaction Dashboard

In the Interaction Dashboard, we provide six menu items to control settings in different views (Main View and Detail View). The Interaction Dashboard shows the supported settings.

- *Search*: The user can search for a particular data point by entering its name (text). As the user enters each letter, the Main View highlights the data points whose name fields include the entered text as a substring.
- *Draw & Explore*: The user can choose among three different options for drawing: 1) freeline, 2) polyline, and 3) object selection. After choosing an option, the user can interact with the Main View to draw or select points accordingly. Users can choose two different selection mechanisms: 1) click and 2) lasso selection. After choosing an option, the user can select the data points on the Main View to enable the display of the details on the Detail View.
- *Detail View*: The user can choose among four different types of visualizations to populate each of the two sections comprising the Detail View. The four supported visualizations are: 1) List, 2) Table, 3) Parallel Coordinates, and 4) Aster Plot.
- *Axis Switch*: The user can change each axis to represent one of the data attributes or a newly generated axis based on one of the user's previous drawings.
- *Glyph Representation*: The user can modify the size of the glyph as well as toggle between an aster glyph and a circle glyph to show or hide attribute details.
- *Data Basket*: In the Data Basket, the user can view data items selected in the Main View. The user can also remove the items from the Data Basket.

5.3 Main View

In the Main View, we visualize the data points on a scatterplot, where each glyph represents a data point. Each data point can be visualized in an aster plot glyph. Each aster glyph visualizes attributes as wedges of the circle, and attribute values are visualized as the radius of the colored portion of the wedge. The Axis Rainbow appears as a result of the axis learning algorithm next to the corresponding axis. It uses a Theme River representation to visualize the value of each attribute to the formation of the axis at a given point [20]. In the following paragraphs, we discuss the aforementioned aspects of the Main View.

There are two main activities that the user can perform in the Scatterplot. First, the user can explore data items and analyze trends. Second, the user can draw sketches or select points to generate a new axis.

In the scatterplot, the user can view data items in the two dimensional plane, wherein each point is plotted according to the two axes. Initially, the scatterplot shows similarities of data using t-Distributed Stochastic Neighbor Embedding (*t-SNE*). The user can set the axes by choosing among data attributes in the Interaction Dashboard. The user can then hover over data items or select multiple items, which prompts the Detail View to show attribute values of the selected items.

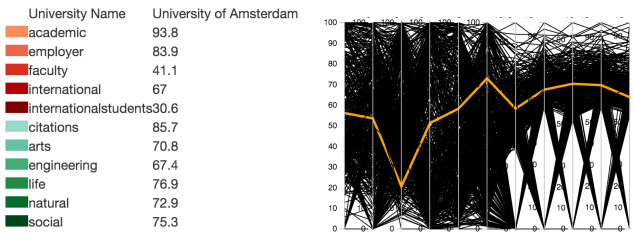
Individual data points on the scatterplot are shown using an Aster Plot glyph representation. As Fig. 1 (d) shows, the glyph shows the values of the data attributes in each wedge. The attributes are each normalized according to the size of the wedge. Each wedge is color-coded according to the attribute it represents. The color scheme is consistent across all views, e.g., the Axis Rainbow view and the list view. The user can toggle the Aster Plot glyph on the Interaction Dashboard.

In the scatterplot, the user can form a new axis by selecting points or drawing lines (Fig. 3). With freeline (Fig. 3(c)), the user can draw a free-form curve on the scatterplot by simply clicking and dragging. As the user clicks and drags across the scatterplot, the system collects coordinates of the user's clickpath. The system then draws lines over the collected points. The segmented lines are smoothed using a basis spline interpolation function. With polyline (Fig. 3(b)), the user can draw a piecewise line by clicking on a series of points on the scatterplot. With object selection (Fig. 3(a)), the user can draw a line by clicking on a series of glyph representations of data items in a sequential order. As users complete their drawing and hit the "change y-axis" button, the system sends the coordinate points to the axis learning algorithm, described in Section 4.3, and updates the view accordingly (x-axis remain unchanged). Finally, the system draws the Axis Rainbow next to the y-axis and animates the repositioning of data points according to the new axis. It also modifies the opacity level of data points based on its uncertainty of axis transformation. The uncertainty of a data point is the shortest distance between the given point and the user-drawn line in the high dimensional data space.

5.3.1 Detail View

The Detail View and the Main View are connected via brushing and linking. Users can see the details of selected data points in the Detail View. There are four different views to choose from: 1) List; 2) Table; 3) Parallel Coordinates; 4) Aster Plot.

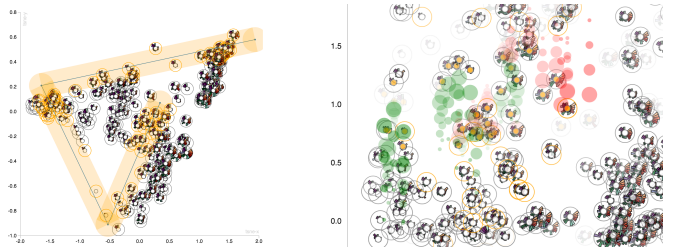
The user can show details of a group of data points by selecting them in the Main View. The system then highlights the data points in the Parallel Coordinates and Table views (bringing them on top of the Table); in the case of the Aster Plot and the List, the system visualizes arithmetic means over the data attributes for selected operates (Fig. 5). The user can select two views out of the four available at any given time by using the Interaction Dashboard.



(a) List

(b) Parallel Coordinates

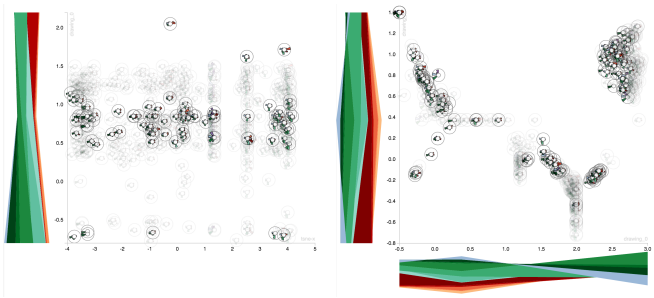
Fig. 5: List and Parallel Coordinates showing data attributes of selected data item(s).



(a) Draw a line brush to highlight data points that can be used for axis formation.

(b) Select a group of data points to trace their travel path from the original coordinates to the current (new) position.

Fig. 6: Supporting views to assess and refine axis transformation.



(a) Initial axis creation.

(b) Forming a new axis.

Fig. 7: Use case with USDA food data.

5.3.2 Filtering Data Points by Brushing

We provide an interaction technique to select data points for axis formation (Fig. 6(a)). When users draw segmented lines, the corresponding orange rectangles are drawn over data points within a chosen distance from the drawn lines. Then, only the points within the orange rectangle are chosen for axis transformation. This interaction lets users fine-tune the axis transformation outcome by filtering data points that can be used for the vertices of the newly formed axes.

5.3.3 Comet View for Tracing Projection of Data Points to New Axes

We provide a comet view that helps users trace the travel path of the data points from the original position (i.e., coordinates of data points before axis transformation) to the newly projected space. Once an axis is transformed, users can select a group of interested data points, which prompt a comet view (Fig. 6(b)). A comet view is a series of circles that connects the original coordinates to the new coordinates. As the space between circles is larger, the data points have traveled more. When axis transformation occurs with respect to *y*-axis, green circles indicate the points traveled north from the original coordinates, and red circles indicate the points traveled south from the original coordinates. This view is particularly useful when users are in the process of modifying axes because users can check how specific items are mapped on the old axis and on the new one. Depending on which direction the items are moved, users can determine whether they are modifying the axis appropriately. If some items are positioned at an unintended position, users can then change the axis accordingly.

6 Usage Scenarios

In this section, we provide use cases for *AxiSketcher* by exploring two different datasets.

6.1 Exploring a Food Dataset

Bob is using *AxiSketcher* to analyze the USDA national nutrient database (`https://ndb.nal.usda.gov/`) of food items. Each food item has 14 numeric attributes (nutrients), such as protein, calcium, and fiber. The initial layout in *AxiSketcher*, generated using *t-SNE*, shows multiple groups of food items that possess similar nutrient values.

Bob found an example of a less healthy food item, *candy*, which has more sugar and less fiber; then subsequently found an example of a preferred item, *cheshire cheese*, which has higher values of protein and calcium. He also found a group of drinks with high water content and relatively small quantities of other nutrients. He draws a segmented line by clicking three representative items (**selecting points**), water, candy, and cheshire cheese, respectively. This action resulted in a new axis that distributed food items in a new order as shown in Fig. 7(a). It successfully found a new item, Swiss cheese, which Bob likes, in the top area and drinks in the bottom area. The new axis found a variety of food items in the middle area.

Bob decided to draw another segmented line based on representative items in the middle area. This time, he did not choose individual objects, but clicked on the areas (**drawing a freeform line**) in the order of his preferences: from those with a high concentration of fat or carbohydrates to those with balanced nutrients. He assigned the new axis to the *x*-axis. As Fig. 7(b) shows, the new axes grouped items into three food groups. The food item group in the top right corner contained items like nuts, cheese, and crackers. Those on the left side were restaurant foods.

6.2 Exploring a Quality of Life Dataset

This usage scenario uses the 2015 Quality of Life Index dataset from Numbeo (`http://www.numbeo.com/quality-of-life/`), which is a crowd-sourced database about worldwide living conditions. The dataset includes 86 countries and seven numeric attributes (purchasing power index, safety index, healthcare index, consumer price index, property price to income ratio, traffic commute time index, and pollution index).

The initial visualization shows countries with high purchasing power, safety, and healthcare indexes in one corner and countries with a high pollution index in the opposite corner. Mary wants to know if this is a global pattern, so she selects all data points by **drawing a polyline** that goes through all points (Fig. 8) to generate an axis based on the underlying patterns of the high-dimensional data. As a result, the system forms a principal curve and maps it onto the *y*-axis.

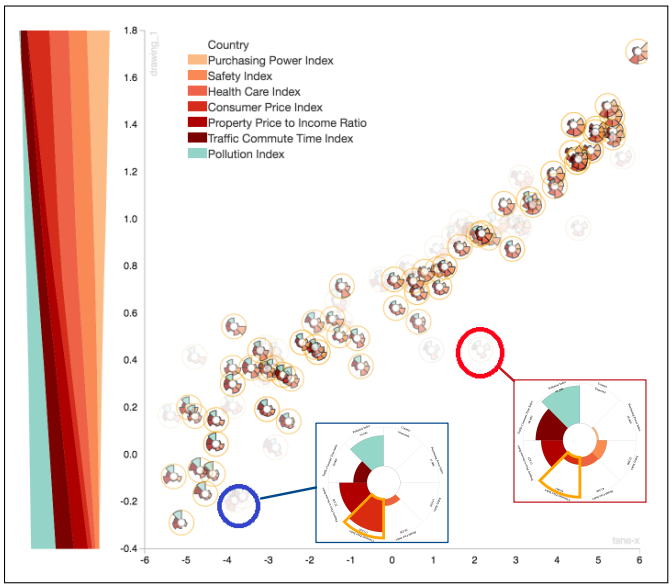


Fig. 8: Use case with quality of life data from 86 countries. The red circle represents South Africa; the blue circle represents Venezuela.

In the *Axis Rainbow* view in Fig. 8, she observes that as the *y* coordinate increases, the purchasing power, safety, healthcare, consumer price indexes (yellow, orange, dark orange, and red, respectively) also increase, while the pollution, traffic commute, and property price to income indexes (green, wine, and dark red, respectively) decrease. Upon exploring individual data points, she concludes that the new *y*-axis represents quality of life. For example, Switzerland, Norway, and Australia are above Mongolia, Venezuela, and Vietnam.

Now Mary is interested in outliers, which are detected by their higher distances from major patterns and their lower opacity. Among them, she selects South Africa (the red circle in Fig. 8), and compares it to other countries that have similar *y*-coordinate values. She discovers that South Africa's safety index is notably lower than the countries with which it is being compared and understands that this is why South Africa is visualized as an outlier. She then selects Venezuela (blue circle in Fig. 8). Since there is no country with similar *y* values, she compares it (top in Fig. 8) with the expected values, which are shown when hovering over the *Axis Rainbow* (bottom in Fig. 8). Venezuela has a significantly lower consumer price index (dark orange) than the expected value. Upon searching news articles, she finds out about Venezuela's economic crisis and inflation nightmare.

6.3 Preliminary User Feedback

We conducted a preliminary user feedback session with four participants (P1-P4), who are currently working as data analysis experts in higher education or research institutes. We started with a tutorial session, in which each participant and the experimenter (the first author) went through our video and a demonstration of our tool. In this session, the experimenter answered any questions the participant had, and they also discussed any interesting points raised in free conversation. We then conducted the "pair analytics" evaluation [3], in which the experimenter takes the control of the system and each participant asks to perform specific actions. In this process, each participant and the experimenter generated dialogues on the usability and utility of the tool for their data analysis tasks. Since the participants were experts in data mining and visualization, they were also able to comment on various aspects of the technique in comparison to other non-interactive dimension reduction techniques. We summarized their perspectives on the strengths and the weaknesses of the tool below.

All participants thought that the *AxiSketcher* technique is an intuitive and creative method for exploring data. P2 liked having the ability to construct new axes that combine multiple attributes beyond the given data attributes. P4 expressed that the tool helps users understand which features contribute most when creating and editing axes.

Participants believed that *AxiSketcher* is very useful because it is a user-driven way to change the layout of a visualization. P3 reported that this technique is very useful because it reflects the user's intention by drawing, which is an intuitive and user-directed method of dimensionality reduction. More specifically, she felt that she did not have to use a fixed algorithm hidden in a black box to output results (projection). Instead, her drawing action produced results, and she could always discover new things about the data from the new results. In addition, P4 reported that he could learn the relative importance of features (attributes) in the data set by viewing the Axis Rainbow. The axis building process was considered data exploration.

Participants appreciated the Axis Rainbow, Comet View, and Data Basket because they increase the transparency of the axis transformation process. First, P4 appreciated the initial projection from the *t-SNE* results because the initial layout showed the similarity between data points based on overall attributes, which helped him decide where to draw a line. He could identify groups of similar points and draw lines connecting those in order of preference. Then, participants could use the Axis Rainbow, Comet View, and Data Basket to view and iteratively edit their axes. P1 liked the Data Basket because it helped him track several items of interest. He used it to draw initial lines and observe how other items were positioned. Subsequently, he could understand the patterns of other items in comparison to the items in the basket. Similarly, he liked Comet View because it traced how the data points moved from an old axis to a new one. P2 expressed that the Axis Rainbow helped him interpret, choose, and create new axes. P3 created multiple axes, each of which contained a unique combination of attributes that could be easily compared by showing two Axis Rainbows on x- and y-axes. P4 also liked editing the once-drawn axis by directly changing the attribute values on Axis Rainbow.

Many participants expressed concern over visual clutter, especially when the data set contained many features and entries. P1 thought that overplotting could become an issue as the number of attributes and data points increase, because it is difficult to quickly understand the dominant pattern from aster plots with many wedges. However, P1 also believed that the aster plot is more intuitive than other available visualizations, such as the table or parallel coordinates views. P3 preferred using the simple dot glyph for the data points over the aster plot, because other features, including the Axis Rainbow and the position of a point, already showed sufficient information. P4 also thought that the interpretability of the aster plot and Axis Rainbow would decrease as the number of attributes exceeded a particular level, because it will exhaust colors for dimensions. To solve such an issue, he suggested including the ability to prioritize or aggregate dimensions/features.

Participants also expected that there would be a learning curve for non-experts to understand and use the tool. P1 felt that the tool requires some mathematical, statistical, and visualization background to understand 1) the mapping concept and axis construction process, 2) how to read the Axis Rainbow and understand its meaning, and 3) how to read the aster plot. P2 thought it would be difficult to learn, because there are many options to choose from in the dashboard. In particular, P2 also thought that the opacity (e.g., uncertainty) may not make sense to users unless they learn the concept in detail. They believed that users need to go through a well-structured tutorial session to become familiarized with the tool and to fully utilize all of it functionalities.

Interestingly, all the participants wanted to categorize or filter the entire data set by particular criteria and to construct axes over a subset of them. P1 wanted to categorize and filter cars based on certain attributes and draw on and analyze the subset. P2 thought that filtering by attributes would increase the usability of the tool. P4 said that as he constructs and reconstructs axes over the data set, he understands more about the data, as well as about the attributes. Then, he naturally wanted to drill down to a small subset based on his domain knowledge to test more specific hypotheses. Thus, he also believed that it would be more useful if the tool let users define criteria by themselves and filter or classify data points by these criteria. For instance, users could categorize cars by the manufacturer's origin country (such an attribute was not given in our data set).

As minor points, P1 recommended removing ticks and labels on axes, because they do not mean anything to users, and to move the Axis Rainbow closer to the axes (or replace it completely). P2 recommended removing some of the options on the dashboard.

7 Discussion

In this section, we explore the implications of our approach from several different perspectives and discuss its limitations and related future work.

7.1 User Interfaces for Interpreting and Steering Complex Models

Although complex models have superior capabilities for flexibly representing a user's intent, it is generally difficult for the user to interpret and steer them. We faced the same challenge when building our system, in which we adopt a complex but capable, nonlinear model for representing a user-driven axes. Interpreting the resulting axis is difficult, and often leaves the user with an insufficient understanding to refine it further.

Therefore, in order to effectively utilize complex models in visual analytics, one must carefully design user interfaces so that users can properly communicate with them in order to interpret system outputs and steer them appropriately. In our approach, we tackled the issue of communicating with our nonlinear model by enriching new axes with a ThemeRiver-style visualization, called Axis Rainbow. The Axis Rainbow not only visualizes the varying attribute values along the axis

but also provides a steering handle with which a user can refine the meaning of the resulting axis in an intuitive manner. On its own, the Axis Rainbow is not just restricted to our particular nonlinear model, but can potentially be utilized as an effective communication technique for other nonlinear dimension reduction methods, including isometric feature mapping [45] and locally linear embedding [39]. In this sense, more research is needed on designing user interfaces that can help bridge the communication gap between a user and a complex model.

7.2 Sketch-Style Interactions: Including People in Model Building

To interactively steer the complex model-building process, a non-trivial amount of model specification is typically required. To fully support this process, people often build complex interfaces to allow users to fully specify the model in full detail. Such complex dialogs and commands, however, may be cumbersome to users, limiting the ability of the model to reflect the their intent.

In the case of a piecewise linear model, one possible interface allows users to specify exact feature values for each point along a piecewise line in order to construct a high-dimensional model. However, we chose to adopt a different approach for model specification that requires less human effort. The sketch-style interactions in *AxiSketcher* require a lower level of formalism, allowing users to quickly form and refine complex models by drawing polylines or free-form curves.

Furthermore, the sketch-style interactions that we propose can be viewed as a kind of observation-level interaction [15], which allows users to perform interactions directly with visualized data items rather than separate, detached user interface controls. In this respect, sketch-style interactions alleviate the undesirable cognitive load during the model building process.

We found inspiration for our techniques in demonstrational user interfaces, with which users can directly show examples of their intended actions on a few objects. Such systems then generalize the action to apply it to all data objects [34]. Demonstrational techniques create a balance between ease of use and computational power. A variety of properties of such techniques in user interfaces have been investigated [29]. Myers et al. [35] substantiated the demonstrational technique to produce basic charts based on user drawings.

7.3 Dimension Extrapolation

Dimension reduction techniques transform high-dimensional data into a lower-dimensional representation. This is inherently lossy, but has the benefit of compressing data and makes it easier to visualize meaningful relationships among complex high-dimension data items. In addition to dimension reduction, one of the primary capabilities provided by our *AxiSketcher* technique can be viewed as dimension extrapolation. We first use dimension reduction techniques to visualize high-dimensional data in a two-dimensional scatterplot. Users then interact with data points by selecting points or drawing a line in a 2D scatterplot. Afterwards, we interpret these interactions to infer a high-dimensional line. However, this presents a new challenge. A user's drawing in a 2D scatterplot does not have a unique interpretation in a high-dimensional space. Alternatively, users could interact with the data in a high-dimensional space by specifying parameters for each dimension; however, that can be a laborious and unintuitive task. Further, this sort of specification requires a high-level of formalism that tends to be difficult to learn and understand. We chose to allow users to interact with the data by clicking and drawing at the expense of explicit interpretability. However, the challenge of understanding the balance between lossy-yet-flexible interactions and lossless-yet-precise interactions remains.

7.4 Limitations

Our preliminary user feedback session revealed several of the limitations of our approach. The main concern lies in the scalability of the approach against the high dimensionality of data. The Axis Rainbow and aster plot representations may not work well with greater than 20 dimensions. As one participant suggested, we can consider implementing the capability to prioritize which dimensions to visualize. Users could preselect a subset of the most important features to initially construct axes. Alternatively, the system could aggregate similar dimensions.

While our participants were able to understand the mapping of the drawing to the axis after interacting with the system, they expressed the concern that non-experts would require time to learn the functionality supported by the system and how to interpret the results. Participants found the Axis Rainbow, Data Basket, and Comet View helpful in understanding how their drawing translated onto a new axis. Thus, we believe that we need to continuously make the underlying computation process more transparent in order to help users understand it. Additionally, users would benefit from a thorough tutorial. Practitioners can also think of designing a tutorial for users to learn and practice the use of the tool, as Kwon and Lee did for Parallel Coordinates [30].

Our user feedback session also showed that participants were guided to draw lines from the initial spatial arrangement. The initial spatial arrangement of points on the scatterplot is determined by *t-SNE*, a popular 2D embedding technique. However, the choice of 2D embedding technique for arranging the points has implications in the sketch-based interaction paradigm. Patterns or clusters of points present may change according to the initial spatial arrangement, making some features more or less salient. Additionally, a user's intended sketch around particular points may yield different results according to the proximity of other nearby points, which can be further exacerbated in cases of occlusion. In order to mitigate this, the axes of a scatterplot are configurable based on any of the features of the data, the original *t-SNE* arrangement, and any previous user-defined axes. The user may change the spatial configuration of the points based on the axes to find the one that reveals an interesting pattern.

8 Conclusion

In this paper, we introduce a visual analytics technique that enables users to bring their domain knowledge to data analysis by drawing lines which then form axes on scatterplots. We provide intuitive visualization for these newly formed axes, increasing their interpretability and accessibility using an interactive, non-linear manifold learning approach. Our use cases show the benefits of such techniques in exploring multidimensional datasets on scatterplots. In future work, we aim to generalize a variety of methods to elicit users' preferences and mental models with regard to data points in other types of high-dimensional visualizations, so that the underlying model and users' knowledge can be integrated in order to generate new knowledge.

Acknowledgments

The authors wish to thank our colleagues who provided constructive feedback to improve the paper. Support for the research is provided in part by the DHS VACCINE Center of Excellence.

References

[1] C. Ahlberg. Spotfire: An information exploration environment. *SIGMOD Rec.*, 25(4):25–29, 1996.

[2] J. Alsakran, Y. Chen, Y. Zhao, J. Yang, and D. Luo. Streamit: Dynamic visualization and interactive exploration of text streams. In *Proc. the IEEE Pacific Visualization Symposium (PacificVis)*, pages 131–138, 2011.

[3] R. Arias-Hernandez, L. Kaastra, T. Green, and B. Fisher. Pair analytics: Capturing reasoning processes in collaborative visual analytics. In *Proc. the Hawaii International Conference on System Sciences (HICSS)*, pages 1–10, 2011.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[5] E. Brown, J. Liu, C. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *Proc. the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92, 2012.

[6] J. Choo, C. Lee, H. Kim, H. Lee, C. Reddy, B. Drake, and H. Park. PIVE: Per-iteration visualization environment for supporting real-time interactions with computational methods. In *Proc. the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 241–242, 2014.

[7] J. Choo, C. Lee, C. K. Reddy, and H. Park. UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization.

IEEE Transactions on Visualization and Computer Graphics (TVCG), 19(12):1992–2001, 2013.

[8] J. Choo, H. Lee, J. Kihm, and H. Park. iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Proc. the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 27–34, 2010.

[9] J. Choo, H. Lee, Z. Liu, J. Stasko, and H. Park. An interactive visual testbed system for dimension reduction and clustering of large-scale high-dimensional data. In *Proc. SPIE 8654, Visualization and Data Analysis (VDA)*, pages 1–15, feb 2013.

[10] J. Chuang, D. Ramage, C. Manning, and J. Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 443–452, 2012.

[11] W. C. Cleveland and M. E. McGill. *Dynamic graphics for statistics*. CRC Press, Inc., 1988.

[12] D. Dong and T. McAvoy. Nonlinear principal component analysis–based on principal curves and neural networks. *Computers & Chemical Engineering*, 20(1):65–78, 1996.

[13] A. Endert. Semantic interaction for visual analytics: Toward coupling cognition and computation. *IEEE Computer Graphics and Applications (CG&A)*, 34(4):8–15, 2014.

[14] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 473–482, 2012.

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

[16] A. Endert, M. S. Hossain, N. Ramakrishnan, C. North, P. Fiaux, and C. Andrews. The human is the loop: new directions for visual analytics. *Journal of Intelligent Information Systems*, pages 1–25, 2014.

[17] S. Few. *Now you see it: simple visualization techniques for quantitative analysis*. Analytics Press, 2009.

[18] H. Guo, N. Mao, and X. Yuan. WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(12):2106–2114, 2011.

[19] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[20] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. ThemeRiver: visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 8(1):9 –20, 2002.

[21] X. Hu, L. Bradel, D. Maiti, L. House, C. North, and S. Leman. Semantics of directly manipulating spatializations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 19(12):2052–2059, 2013.

[22] D. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. iPCA: An interactive system for pca-based visual analytics. *Computer Graphics Forum (CGF)*, 28(3):767–774, 2009.

[23] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[24] B. Kegl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(3):281–297, 2000.

[25] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Proc. the International Conference on Information Visualization*, pages 9–16, 2006.

[26] A. Kerren, J. Stasko, J.-D. Fekete, C. North, D. Keim, G. Andrienko, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In *Information Visualization*, pages 154–175. Springer, 2008.

[27] H. Kim, J. Choo, H. Park, and A. Endert. InterAxis: Steering scatterplot axes via observation-level interaction. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 22(1):131–140, 2016.

[28] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[29] B. C. Kwon, W. Javed, N. Elmqvist, and J. S. Yi. Direct manipulation through surrogate objects. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 627–636, 2011.

[30] B. C. Kwon and B. Lee. A comparative evaluation on online learning approaches using parallel coordinate visualization. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 993–997, 2016.

[31] B. Lee, R. Kazi, and G. Smith. Sketchstory: Telling more engaging stories with data through freeform sketching. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 19(12):2416–2425, 2013.

[32] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[33] S. C. Leman, L. House, D. Maiti, A. Endert, and C. North. A bi-directional visualization pipeline that enables visual to parametric interaction (V2PI). Technical report, Georgia Institute of Technology, 2011. FODAVA-10-41.

[34] B. A. Myers. Demonstrational interfaces: A step beyond direct manipulation. *Computer*, 25(8):61–73, Aug. 1992.

[35] B. A. Myers, J. Goldstein, and M. A. Goldberg. Creating charts by demonstration. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 106–111, 1994.

[36] M. Natali, T. G. Klausen, and D. Patel. Sketch-based modelling and visualization of geological deposition. *Computers & Geosciences*, 67:40–48, 2014.

[37] D. Sacha, H. Senaratne, B. C. Kwon, G. Ellis, and D. A. Keim. The role of uncertainty, awareness, and trust in visual analytics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 22(1):240–249, 2016.

[38] D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. Ellis, and D. A. Keim. Knowledge generation model for visual analytics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 20(12):1604 – 1613, 2014.

[39] L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research (JMLR)*, 4:119–155, 2003.

[40] D. Schroeder, D. Coffey, and D. Keefe. Drawing with the flow: A sketch-based interface for illustrative visualization of 2d vector fields. In *Proc. the Sketch-Based Interfaces and Modeling Symposium*, pages 49–56, 2010.

[41] J. Schumann, T. Strothotte, A. Raab, and S. Laser. Assessing the effect of non-photorealistic rendered images in cad. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 35–41, 1996.

[42] E. Shen, S. Li, X. Cai, L. Zeng, and W. Wang. Sketch-based interactive visualization: a survey. *Journal of Visualization*, 17(4):275–294, 2014.

[43] D. F. Swayne, D. Temple Lang, A. Buja, and D. Cook. GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43:423–444, 2003.

[44] Tableau. *version 8.1*. Tableau Software, Seattle, Washington, 2014.

[45] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[46] J. J. Thomas and K. A. Cook. *Illuminating the path*. IEEE Computer Society, 2005.

[47] T. Tsandilas, A. Bezerianos, and T. Jacob. SketchSliders: Sketching widgets for visual exploration on wall displays. In *Proc. the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 3255–3264, 2015.

[48] J. M. Utts. *Seeing through statistics*. Cengage Learning, 2014.

[49] B. Wang, P. Ruchikachorn, and K. Mueller. SketchPadN-D: WYDIWYG sculpting and editing in high-dimensional space. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 19(12):2060–2069, 2013.

[50] J. Wei, C. Wang, H. Yu, and K.-L. Ma. A sketch-based interface for classifying and visualizing vector fields. In *Proc. the IEEE Pacific Visualization Symposium (PacificVis)*, pages 129–136, 2010.

[51] M. Williams and T. Munzner. Steerable, progressive multidimensional scaling. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 57–64, 2004.

[52] J. Wood, P. Isenberg, T. Isenberg, J. Dykes, N. Boukhelifa, and A. Slingsby. Sketchy rendering for information visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18(12):2749–2758, 2012.

[53] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & Magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005.

[54] J. Zhao, C. Collins, F. Chevalier, and R. Balakrishnan. Interactive exploration of implicit and explicit relations in faceted datasets. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 19(12):2080–2089, 2013.