Visualization by Demonstration: An Interaction Paradigm for Visual Data Exploration

Bahador Saket, Hannah Kim, Eli T. Brown, Alex Endert

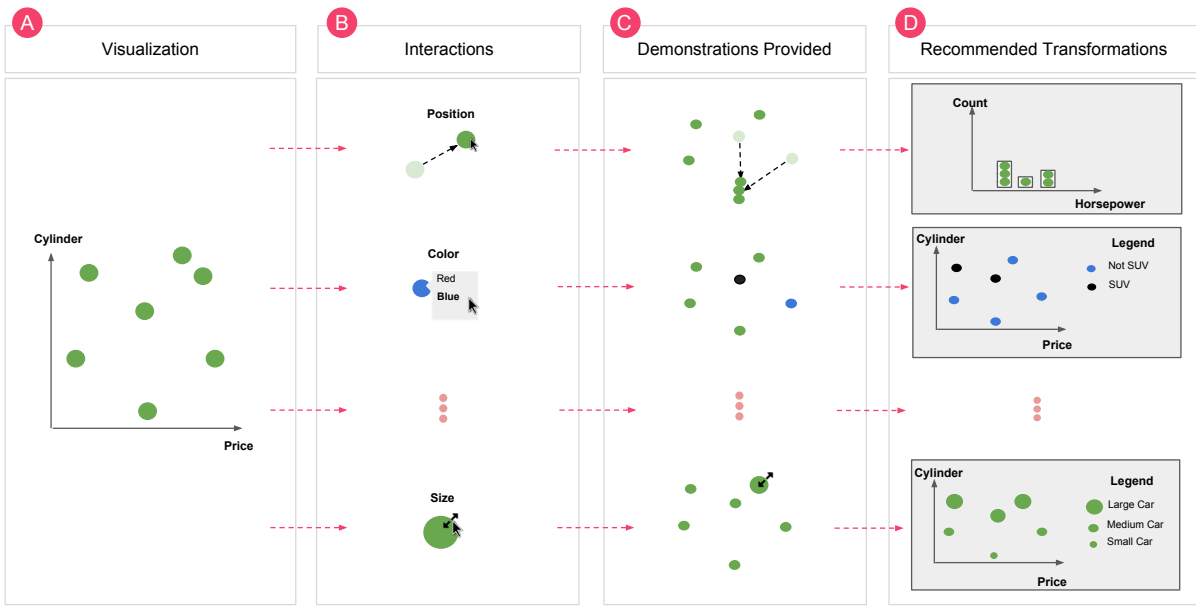


Figure 1: An overview of the Visualization by Demonstration paradigm (illustrated using a car dataset). A) Interactive visualization displayed. B) Users can manipulate spatial and graphical encodings directly *(e.g., coloring data points)*. C) Users provide visual demonstrations of incremental changes to a visualization. D) Using these demonstrations, the system estimates the intended results and recommends possible transformations.

Abstract—Although data visualization tools continue to improve, during the data exploration process many of them require users to manually specify visualization techniques, mappings, and parameters. In response, we present the *Visualization by Demonstration* paradigm, a novel interaction method for visual data exploration. A system which adopts this paradigm allows users to provide visual demonstrations of incremental changes to the visual representation. The system then recommends potential transformations *(Visual Representation, Data Mapping, Axes, and View Specification transformations)* from the given demonstrations. The user and the system continue to collaborate, incrementally producing more demonstrations and refining the transformations, until the most effective possible visualization is created. As a proof of concept, we present VisExemplar, a mixed-initiative prototype that allows users to explore their data by recommending appropriate transformations in response to the given demonstrations.

Index Terms—Visualization by Demonstration, Visualization Tools, Visual Data Exploration

1 Introduction

Visualization researchers and practitioners continue to develop a wide range of interactive visualizations which allow users to explore and make sense of their data. One widely-used interaction paradigm for these visualizations is direct manipulation through control panels and other graphical widgets [28]. This includes controls and selections such as view specification, filtering, assigning data attributes to visual encodings such as color or size, and others. Such methods require users to specify their visualization techniques *(e.g., selecting a scatterplot)*, mappings *(e.g., assigning size to a data attribute)*, and parameters *(e.g., assigning data attributes to axes)* in order to generate and modify visualizations to explore their data. Alternatively, more recent work in visualization recommender systems has given users the ability to select data attributes of interest, from which visualizations are generated [33, 4]. The recommendations are generated based on the computed data characteristics and the user input about which data attributes are of interest.

There also exist prior studies that show the effectiveness of letting people create spatial representations of data points manually, without the need to formalize the mappings between the data and the spatial constructs created. For example, Andrew et al. [2] found that people use the spatial environment to create layouts of information which have meaning to the person, without requiring users to specify the formal definition of the layout. For example, people create clusters of similar data points by moving similar data points closer to each other. Further, during such spatial exploration processes, some of the spatial arrangements exhibit characteristics similar to formal visualization techniques [15, 16]. For example, participants stacked data points in the shape of bars to count the number of specific items (similar to

- *Bahador Saket, Hannah Kim, and Alex Endert are with Georgia Institute of Technology. E-mail: {saket, hannahkim, endert@gatech.edu}.*
- *Eli T Brown is with DePaul University E-mail ebrown80@cdm.depaul.edu.*

Manuscript received 31 Mar. 2016; accepted 1 Aug. 2016. Date of publication 15 Aug. 2016; date of current version 23 Oct. 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2016.2598839

the well-known bar chart visualization technique). These studies show that users are effective at providing visual demonstrations of visualization techniques, mappings, and parameters.

The demonstration-based paradigm for human-computer collaboration has seen use in a variety of contexts. In computer programming, programming by demonstration [8] allows users to generate code by providing demonstrations of some intended result, often done visually. The user and the system continue to collaborate, using further user demonstrations to incrementally improve the computer-generated code. Wrangler [18] is also an instance of "by demonstration" systems. Wrangler allows users to provide demonstrations of expected results on tabular data by directly showing results in the table view *(e.g., selecting a substring of a column to generate the transformation for creating a new column)*. Other application domains that exhibit the "by demonstration" approach include query by demonstration [35], data cleaning by demonstration [21], 3D drawing by demonstration [17], and more. In this work, we show that this interaction paradigm can be extended to visual data exploration and information visualization.

We present the *Visualization by Demonstration* paradigm. The paradigm advocates for decreasing the level of formalism and fundamental knowledge required for visual data exploration. Instead of specifying which visualization technique, mappings, and parameters to generate and update a visualization, Visualization by Demonstration allows users to provide visual demonstrations of incremental changes to the visual representation from which transformations are recommended. Using these demonstrations, the system estimates the intentions and generates potential transformations *(e.g., a bar chart, mapping color to a data attribute)*. This iterative process allows users to visually explore their data without requiring direct visualization specification. That is, the goal is to balance the responsibility of data exploration between the user and the system — users provide visual demonstrations, while the system generates visualizations and defines the visualization mappings and parameters (see Figure 1).

The contribution of Visualization by Demonstration over existing work is twofold. First, the concept of Visualization by Demonstration does not require users to specify visualization techniques ahead of time. Instead, the paradigm will extract the visualization technique from the given demonstrations. Second, the paradigm also extracts visualization mappings and parameters that match given demonstrations. For example, if users stack data points vertically, Visualization by Demonstration suggests a bar chart with appropriate axes.

To show the feasibility of the Visualization by Demonstration concept, we implemented VisExemplar. VisExemplar is a mixed-initiative data exploration prototype that allows users to explore their data using Visualization by Demonstration. VisExemplar allows users to provide visual demonstrations of incremental changes to the visualization by directly manipulating the visual representation *(e.g., moving one data point on the top of another, changing the color of a select set of point, etc.)*. VisExemplar uses a recommendation engine that generates transformations in response to the given demonstrations using a set of intent functions. To help users understand the results of different recommended transformations before they commit, VisExemplar contributes novel methods for presenting recommended transformations.

The remainder of the paper is organized as follows. Section 2 first discusses differences between interaction methods used in existing visualization tools and the Visualization by Demonstration paradigm. It then describes some of the studies that inspired the Visualization by Demonstration paradigm. In the light of previous work, Section 3 describes Visualization by Demonstration in more detail. Section 4 provides a usage scenario and the system design of VisExemplar. In Section 5 we discuss the potential value of Visualization by Demonstration and possible research avenues for continued research.

2 Related Work

2.1 Visual Data Exploration

Visual representations are one of the fundamental components of any visualization tool [34]. A central component of visual representations is the mappings from data values to graphical representations [7]. Visual representations are constructed using a combination of different visual encodings *(e.g., length, position, size, color, etc.)* [6]. Thus, interactivity in visualization tools is often designed to change one or more mappings between the data and the visual encodings. A popular method for visualizing data is using pre-existing interactive visualization tools *(e.g., SpotFire [29] and Tableau [30])*. Such visualization systems allow users to specify direct mappings between their data and the visual representation without requiring users to have programming skills. However, such systems require some amount of fundamental knowledge about the data, the domain, and of visualization techniques. For instance, to create a scatterplot, users must specify the technique, and then which of the data attributes to map onto the x and y axes.

Figure 2: Constructing a visualization with tokens. Figure from [16] used with permission.

There also exist mixed-initiative systems [14] which aim to balance the responsibility of data visualization between the user and the system. One category of such systems is visualization recommendation tools. Many visualization recommendation systems have been developed to assist users to visualize their data *(e.g., [4, 12, 22, 23, 33])*. They suggest alternative views (visual encodings, data attributes, and data transformations) based on user-specified data of interest and computed characteristics about the data. Voyager [33] is mixed-initiative system that couples faceted browsing with visualization recommendation to support visual data exploration based on user-specified data attributes of interest. VizAssist [4] generates visualizations by requiring users to specify both desired data attributes and tasks.

Instead of requiring users to specify the visualization technique, mappings, and parameters a priori (or specify data attributes of interest), Visualization by Demonstration extracts this information from created visual demonstrations. Visualization by Demonstration can be used independently or to augment existing visualization tools to increase their functionality in concert with direct manipulation controls.

2.2 Flexibility of Spatial Data Organization

Andrews et al. [2] showed how space was used by their participants as both an external memory aid in which the spatial constructs carried meaning. The participants formed spatial constructs *(e.g., groups, lists, clusters)* as a means to organize the information, as well as structure their analytic process. Various studies also used spatial environments as a thinking medium to allow users to construct their visualizations incrementally [27, 31, 32]. For example, Walny et al. [32] showed how pen and touch can be used to construct visualizations on interactive whiteboards. Similarly, SketchStory [20] demonstrates how users can draw visualization characteristics (e.g., axes of a scatterplot) to generate visualizations. Moreover, Schroeder et al. [26] introduced a sketching technique that enable graphic designers and artists to construct multivariate time-varying visualizations by painting on a digital data canvas, sketching data glyphs, and blending together multiple layers of animated 2D graphics. Satyanarayan and Heer [25] presented Lyra, a direct manipulation environment that allows users to create customized visualizations by without requiring coding (e.g., dragging and dropping data attributes directly onto visual glyphs). Similarly Ren et al. [24] presented iVisDesigner, an interactive environment that allows users to design visualizations interactively, without the need for coding.

Huron et al. [15] proposed a method called "Constructive Visualization", which advocates for allowing people to create visualizations by manually moving, adding, and removing physical tokens. In their study, each token represents a basic data unit. Thus, constructing a

visualization means assembling these tokens to encode the data in a meaningful way (see Figure 2). They found people use the spatial environment to construct visualizations and explore their data, where many spatial arrangements exhibit characteristics similar to formal visualization techniques. For example, people stacked data points in the shape of bars to count the number of specific items (similar to the well-known bar chart visualization technique). Inspired by these studies, the direct manipulation of spatial encodings of a visual representation is one of the methods that people can use to provide demonstrations in the Visualization by Demonstration paradigm.

Prior work also exists which allows users to directly adjust the position of data points (e.g., documents), interpret this feedback via a dimensionality reduction model to generate a new spatialization that better reflect the users understanding of the high-dimensional data [5, 9, 10]. DimpVis is one of the recent systems which applies embedded interaction as a substitution to other options *(e.g., time slider)* for querying and exploring time-varying information visualizations. The system allows users to directly manipulate the data points in visualizations to perform temporal navigation of the dataset [19].

Visualization by Demonstration advocates for a similar form of direct visual and graphical manipulation of data points. People are given the ability to reposition and re-encode visual demonstrations of data, from which the mappings are computed.

3 Visualization by Demonstration

The systems which make use of the Visualization by Demonstration paradigm allow users to provide visual demonstrations of incremental changes to the visualization as a method for user interaction. These demonstrations could be provided by direct manipulation of the spatial and graphical encodings used in a visualization. The systems then recommend potential transformations from the given demonstration. In this section we first discuss two different methods that the Visualization by Demonstration paradigm supports for providing demonstrations. We then discuss different classes of transformations supported by this paradigm. Finally, we present design guidelines that should be considered by systems adopting Visualization by Demonstration.

3.1 Methods for Providing Demonstrations

Building on the strong research foundation of previous work [2, 15, 16, 32], one of the methods by which the systems adopting Visualization by Demonstration might allow users to provide visual demonstrations is by directly adjusting the spatial layouts of data points *(e.g., users stacking data points in the shape of bars to convey their interest in a bar chart or placing two data points in desired positions along the x or y axis to demonstrate the attribute to map to the axis)*. In addition to direct manipulation of spatial encodings, the systems adopting this paradigm might allow users to demonstrate desired graphical encodings applied to data attributes by adjusting the graphical encodings used in a visualization *(e.g., changing the color or size of data points in a scatterplot, length or color of bar in a bar chart, and others)*.

3.2 Recommending Potential Transformations

Visualization by Demonstration suggests possible transformations that can be applied based on the provided demonstrations. For each demonstration, the system checks how the current state of the visualization, parameters, and mappings should be transformed to create meaningful visual representations that match the demonstration. The system then recommends these possible transformations. We categorize these transformations into four main categories, described below.

Visualization Representation Transformations change the current visualization technique to a different visualization technique *(e.g., transforming from a scatterplot to a bar chart)*. To convey interest in transforming to a new visualization technique, users can manipulate the spatial encoding to create a spatial layout similar to the intended visualization technique. For example, users can stack two or more data points vertically or horizontally in a scatterplot to demonstrate their interest of switching to a vertical or horizontal bar chart.

Data Mapping Transformations define mappings between graphical encodings and data attributes *(e.g., mapping color to an attribute)*. To convey interest in assigning a graphical encoding to a data attribute, users can manipulate the corresponding graphical encoding in the visual representations. For example, users could color one or more data points red to convey their interest in mapping color to a data attribute.

Axes Transformations assign data attributes to axes of a visualization technique *(e.g., assigning an attribute to the x axis of a scatterplot)*. To assign new data attributes to axes, users can manipulate the corresponding graphical encoding or spatial encoding in the visual representations. For example, in a scatterplot, users could move one or more data points to a positions along an axis to demonstrate their interest in having a scatterplot in which the manipulated data point is close to the current coordinates, thus changing the attribute assigned to the axis. Alternatively, in a bar chart, users could change the length of one of more bars to demonstrate mapping a new attribute to the axis.

View Specification Transformations change the view specifications without changing the underlying technique *(e.g., aggregation, average, sorting)*. For example, users could convey their interest in sorting a bar chart by dragging the longest bar in the current bar chart to the most left or right side of the axis.

3.3 Design Guidelines

We identify the following design guidelines for applications that make use of Visualization by Demonstration. These were refined through our experiences through several design iterations.

G I: Support direct manipulation of visual representations to foster visual data exploration. Inspired by previous studies [15, 32, 20], Visualization by Demonstration should provide an environment in which users provide demonstrations by manipulating the spatial and graphical encodings of data directly in the visual representations.

G II: Balance human and machine workload in the visual data exploration process. Allowing users to construct demonstrations by manipulating the spatial and graphical encodings increases the directness of their interactions. However, manually manipulating all the data points in a visualization is time consuming. To balance human and computer's effort in this process, this paradigm advocates for a mixed-initiative approach to human-computer collaboration during the visual data exploration process – users provide visual demonstrations, and the system provides recommended transformations.

G III: Enable user interactions to drive recommended transformations. Systems adopting this paradigm should allow users to specify their intentions or desired aspects of the data by directly manipulating the data points in a visual representation. In aggregate, these interactions create visual demonstrations which serve as the primary units by which users communicate their intended changes to the system.

G IV: Enhance interpretability of recommendations. Recommended transformations should be presented in way that allow users to extract why specific transformations were shown and what would be the resulting outcome if they accept any of the transformations. This requires recommended transformations to be presented in a right position in the interface, and provide users with the rationale behind why they were recommended.

4 VisExemplar

To demonstrate the application of Visualization by Demonstration, we developed VisExemplar (see Figures 3 and 4). All components of the VisExemplar are implemented using JavaScript and the visualization modules are built using the D3 toolkit [3]. Datasets in comma-separated values format are supported. The implemented system is available at `http://bitbucket.org/bahadorsaket9/vbd`.

4.1 Usage Scenario

In this section, we motivate the design of our system and illustrate the functionality via a usage scenario. We indicate how someone can utilize VisExemplar to examine data about cars. The car dataset [13] provides specifications on new cars and trucks for the year 2004. The dataset contains 122 data points, with 18 data attributes describing each car. This dataset is used throughout the motivating examples in this paper.

Figure 3: The VisExemplar user interface consists of a ThinkBoard, Recommendation Gallery, and a Detail View panel. ThinkBoard shows each data point as a circle. The Recommendation Gallery shows visualization technique transformations. The Detail View shows data details, and also recommended data mapping transformations.

Assume Amy wants to buy a car, and wants to make a data-driven decision based on this dataset. Amy needs to find a single car to purchase that best meets her needs and preferences, and decides to do so using VisExemplar. She has limited knowledge in constructing visualizations (as well as little domain expertise about cars), but would like to use visualization to help her make her purchase.

Upon loading the data, VisExemplar shows each car as a green circle on the ThinkBoard (ThinkBoard is shown in Figure 3). Amy can interact with the cars *(e.g., move, resize, or recolor)* on the ThinkBoard, or search for a specific car or manufacturer that she is familiar with by using the search box. Two of Amy's friends drive a Toyota Prius and a Honda Civic Hybrid. She likes both of them, but is not as familiar with the attributes that define the cars, so she simply starts by searching for both. See Figure 4-(a) for more details.

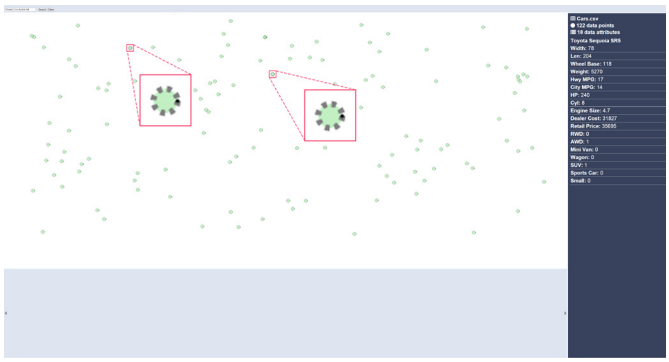
She decides to put the cars that she is interested in close to each other somewhere on the ThinkBoard. She drags the two cars close to each other (see Figure 4-(b)). Upon this interaction, the system recognizes that there are possible visual representations in which these two cars are close to each other. In this case, there exist $x-y$ axes pairs that would result in a scatterplot consistent with the two example data points placed in close proximity of each other. The system recommends placing different options for the x and y axis *(e.g., length of the cars (Len) for x axis and weight of the cars for y axis)*. See Figure 4-(b) for more details. Amy looks at different attribute options to assign to the axes. Due to her interest in a med-sized sedan, she decides to select *length of the cars (Len)* as the x axis to see how the lengths of the cars compare across the dataset.

After assigning *Len* to the x axis, VisExemplar produces the scatterplot shown in Figure 4-(c). Amy notices that both cars she initially dragged close together have a length of 175. She decides to hone her search of a car to other vehicles that have roughly this size by coloring several cars with a length of about 175 red by right clicking on them and picking the red color (as shown in Figure 4-(d)). The system automatically extracts data attributes that can be mapped to color *(e.g., cylinder (Cyl), as well as others)*. Data attributes which can be assigned to color are indicated by a brush icon (🖌) next to the data attributes in the detail panel (see Figure 4-(d)). In this case, Amy notices that the system recommended assigning color to the number of cylinders (Cyl). Intruiged, she decides to accept this mapping by double-clicking on the brush icon. Figure 4-(e) shows the resulting view, where the color mapping is shown in the legend on the Detail View panel.

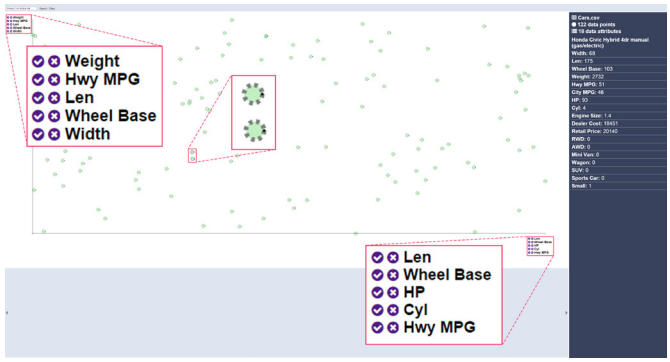
Amy notices that many of the cars with a length of 175 are 4 cylinder cars (shown in red), and asks herself "how many of the cars have a length of 175, compared to the lengths of other cars?". She stacks a few cars with the length of 175 vertically on the top of each other to group these items together and count them (Figure 4-(f)). Based on her example, VisExemplar recommends a selection of bar charts. The recommendations are based on the attributes that the stacked cars have in common. Each proposed bar chart has one data attribute in the x axis and the corresponding count of cars in the y axis. The bars are drawn as a box containing the counted cars. Amy explores different recommended bar charts by scrolling the Recommendation Gallery, and chooses one showing length as an axis labelled "Stack vertically based on **Len**." (Figure 4-(f)).

At this point, Amy has a visualization where the x axis of the bar chart assigned to Len and the y axis as the number of cars (Figure 4-(g)). By looking at the y axis of the bar chart Amy notices that among all 122 cars only 34 of the cars have a length between 175 and 180. Among these, 13 are 4 cylinder cars (colored red). Amy hovers over these to get more details, and finds two additional cars (Toyota Corolla CE and Honda Civic EX) which have the characteristics that she has found interesting up to this point.

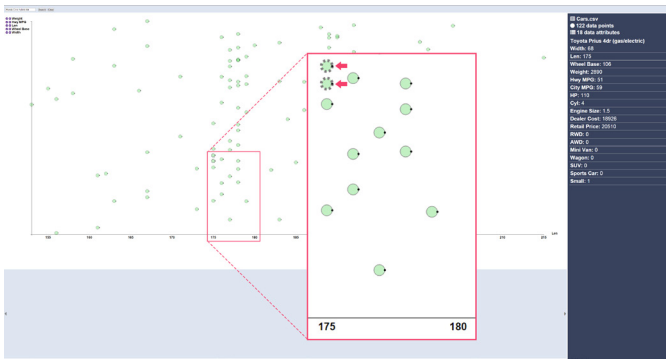
She wants to switch back to a scatterplot to see additional attributes on the axis to make a more detailed comparison. She drags these four cars out of the bars, demonstrating the intent to switch to a scatterplot. The system will again compute the $x-y$ axes pairs that would result in a scatterplot given the locations of the dragged cars. Amy starts exploring different recommended scatterplots by scrolling the Recommendation Gallery. Suddenly the label **Retail Price** in one of the thumbnails grabs her attention (Figure 4-(h)). She picks that recommendation as she realizes she has ignored price up to this point. The visualization shows that all four cars are roughly the same in price, and decides to schedule a test drive for each of them as she feels confident in her choices, and has learned a bit more about attributes of cars that define her subjective preference for cars.



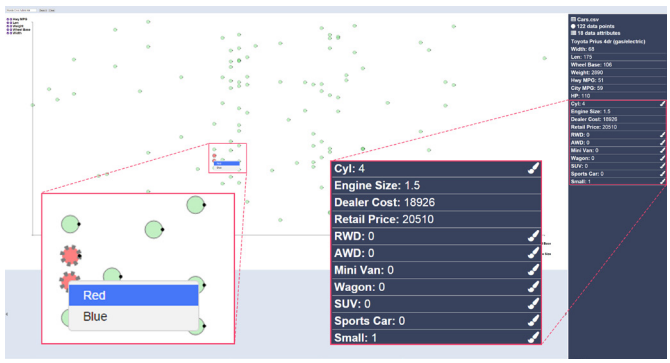
(a) Searched for two cars on the search box. Border line of the selected cars changed to dashed line.



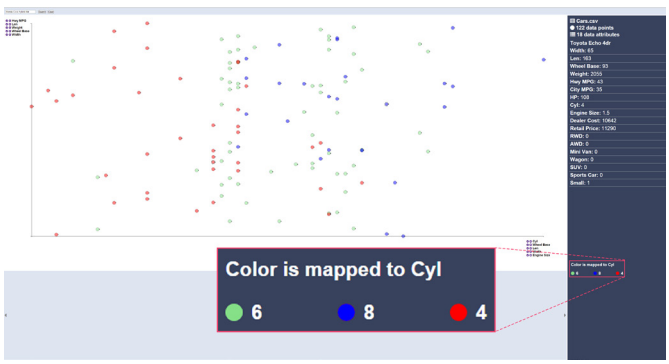
(b) Dragged two cars close to each other. The system recommended options for assigning to the x and y axis.



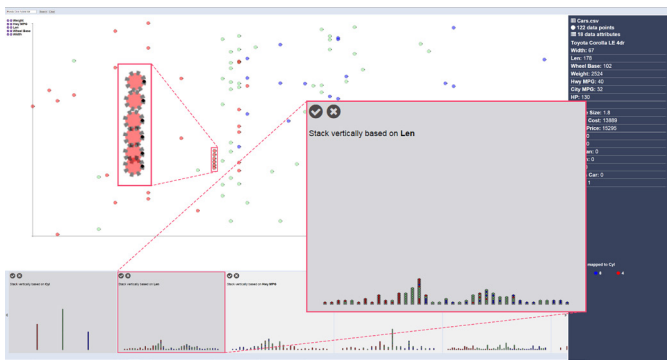
(c) Assigned Len to the x axis.



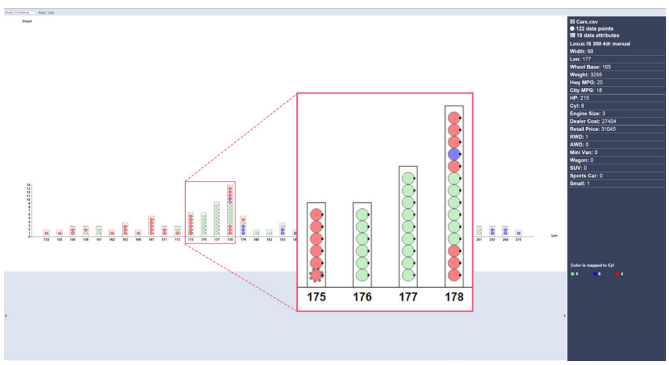
(d) Colored two cars. The system recommended data attributes that can be mapped to color.



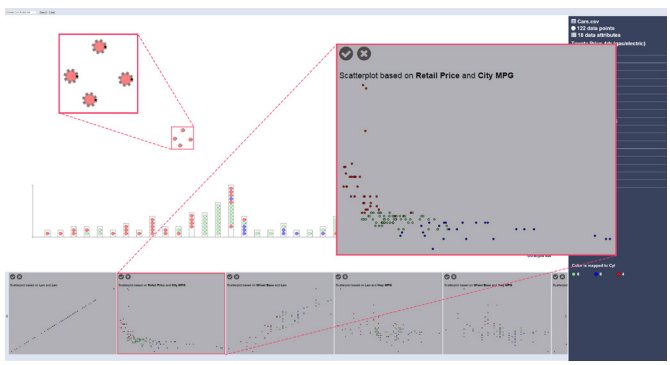
(e) Color mapping is shown. Color is mapped to Cylinder (Cyl).



(f) Stacked a few of the cars with the length of 175 vertically on the top of each other. The system recommends different bar charts.



(g) Selected a bar chart where x axis assigned to Len and the y axis as the number of cars.



(h) Selected a scatterplot where $x-y$ axes indicate Retail Price and City MPG.

Figure 4: A series of screenshots showing the usage of Visualization by Demonstration in VisExemplar.

4.2 The VisExemplar Interface

Figure 3 shows VisExemplar's interface, consist of a ThinkBoard, a Recommendation Gallery, and a Detail View panel. ThinkBoard is a thinking medium for users and allows them to construct their demonstrations through direct manipulation of the visual representation. Moreover, possible *Axes and View Specification transformations* be shown on the ThinkBoard. *Visual Representation transformations* will be presented in the Recommendation Gallery. The primary goal of the Detail View panel is to show details of selected data points. By hovering on a data point on the interaction board, the Detail View panel will show detail information related to that data point. See Figure 3 for more details. In addition, *Data Mapping transformations* will be shown as small icons on this panel.

VisExemplar realizes the four Visualization by Demonstration design guidelines presented in Section 3 as follows. VisExemplar provides an environment similar to a spatial workspace in which users provide demonstrations by manipulating the spatial and graphical encodings used in visual representations (**G I**). In addition, to balance human and computer effort during data exploration process, VisExemplar suggests variety of possible relevant transformations in the form of *Visual Representations transformations*, *Data Mapping transformations*, *Axes transformations*, and *View Specification transformations*. Depending on the transformation type, they will be shown in different forms and locations on the interface (**G II**). VisExemplar allows users to provide demonstrations by directly manipulating the data points in a visual representation (**G III**). Finally VisExemplar uses different methods for showing recommendations to the user. First, visual representation transformations are shown as thumbnails below the ThinkBoard. These are ordered and colored based on their computed relevance to the visual demonstration. Data mapping transformations are shown as icons in the detail panel, and axes transformations on the axes. This helps users browse the possible space of transformations and interpret their result (**G IV**).

4.3 Transformations Supported in VisExemplar

VisExemplar currently supports four categories of transformations.

Visual Representation Transformation. VisExemplar currently supports transformations from bar charts to a scatterplots and vice versa. Recommended *Visual Representation Transformations* will be shown on the Recommendation Gallery. Each recommended transformation is shown as a thumbnail in the gallery. Users can explore different transformations by scrolling through gallery. Each thumbnail consists of a textual explanation describing what the visualization in the thumbnail is showing *(e.g., Stack vertically based on Cylinder)* and a visualization which gives an overview of the transformation. We decided to show this type of recommendation as thumbnails because during the design process we found it difficult to imagine the resulting changes from one visual representation to another without seeing the resulting view. Relevance of this type of transformations is dually encoded by color and position. By default, we use a light gray color as background for recommended transformations in the gallery. We show the relevance of the recommended transformations by adjusting the darkness of the background color, the lighter the background color the lower the relevance. In addition, the recommended transformations are ordered left to right based on relevant (left being highest). Figures 4-(f) and 4-(h) indicate examples of *Visual Representation transformations* in VisExemplar.

Data Mapping Transformation: The current version of VisExemplar supports color and size encodings. These types of transformations are shown as small icons on Detail View panel corresponding to the attribute which is being recommended to map to the visual demonstration. We decided to show this type of transformation on the Detail View panel since each icon is located beside corresponding data attributes. For those recommended data attributes that can be assigned to color, a small brushing icon (🖌) will appear near the data attributes on the detail panel. Similarly, the system recommends data attributes to be mapped to size by showing a small expand icon (⤢) beside the appropriate data attributes on the Detail View panel. The background color of the data attributes on hover shows the relevance. The lighter the background color the lower the relevance. Hovering on the recommended data attributes will also show a preview of resulting changes. A user can apply any of the recommended transformations by double clicking on the suggested data attribute. Figure 4-(d) shows examples of *Data Mapping transformations* in VisExemplar.

Figure 5: An example of a view specification transformation.

Axes Transformation. We show *Axes transformations* directly on the corresponding axes in the ThinkBoard. In the early stages of our design process, we noticed that it is easier to understand the meaning of these type of transformations when they are located close to the corresponding axis. For this type of transformation, the position of the data attributes beside each axis show their relevance. The higher the data attribute the higher the relevance. Figure 4-(b) shows examples of *Axes transformations* in VisExemplar.

View Specification Transformation. This type of transformation is shown on the ThinkBoard below the visualization technique. One of these transformations that VisExemplar currently supports is sorting the bar chart in ascending or descending order (See Figure 5).

4.4 Recommendation Engine

When a user performs an interaction with the visual representation and generates a visual demonstration, the recommendation engine of our system accepts the interaction as input, and produces the recommendations as outpout for transformations mentioned in Section 4.3. VisExemplar allows direct manipulation of three encodings of data points including position, color, and size (see Figure 6-(A)). Direct manipulation of each encoding will invoke a series of intent functions related to that specific encoding (see Figure 6-(C)). Based on the demonstrations provided, the intent functions determine which transformations are most relevant. VisExemplar contains seven intent functions. All related intent functions are checked against every interaction. For example, by directly re-positioning data points in a scatterplot to new x coordinates, one of the intent functions which will be invoked is the *assigning X axis* function. Considering the points that have been moved, the system then recommends potential data attributes for the *x* axis that would result in a scatterplot where the moved data points would be as close as possible to the new x coordinates (see Figure 6).

As a result of each interaction, the recommendation engine will update the recommendation table (see Figure 6-(D)). The recommendation table consists of a set of potential transformations. Each row of the table represents a potential transformation. Each transformation consists of a name *(e.g., xAxis_Cylinder)*, relevance, and location on the interface. The table will be created only once and will be updated after each interaction. The relevance value for each transformation indicates the number of times the transformation is generated. The relevance value is normalized to a range of $[0, 1]$ and the table is updated to contain the normalized relevance value for each transformation. The system only shows transformations with relevance above 0.3. Upon accepting a transformation the changes are applied and relevance values of all recommendations reset. The transformation type column dictates where each transformation is shown in the interface.

The recommendation engine then passes the recommendation table to the interface. The interface will update the visualization based on the given recommendation table; See Figure 6-(E).

4.4.1 Intent Functions

Depending on the interaction, any of seven currently-supported intent functions might be invoked (see Figure 6-(C)). For example, changing the position of a data point could invoke functions 1, 2, 3 or functions 4 or 5, depending on current state of the visualization (scatterplot or bar chart). If the current visualization is a scatterplot, then resizing a data

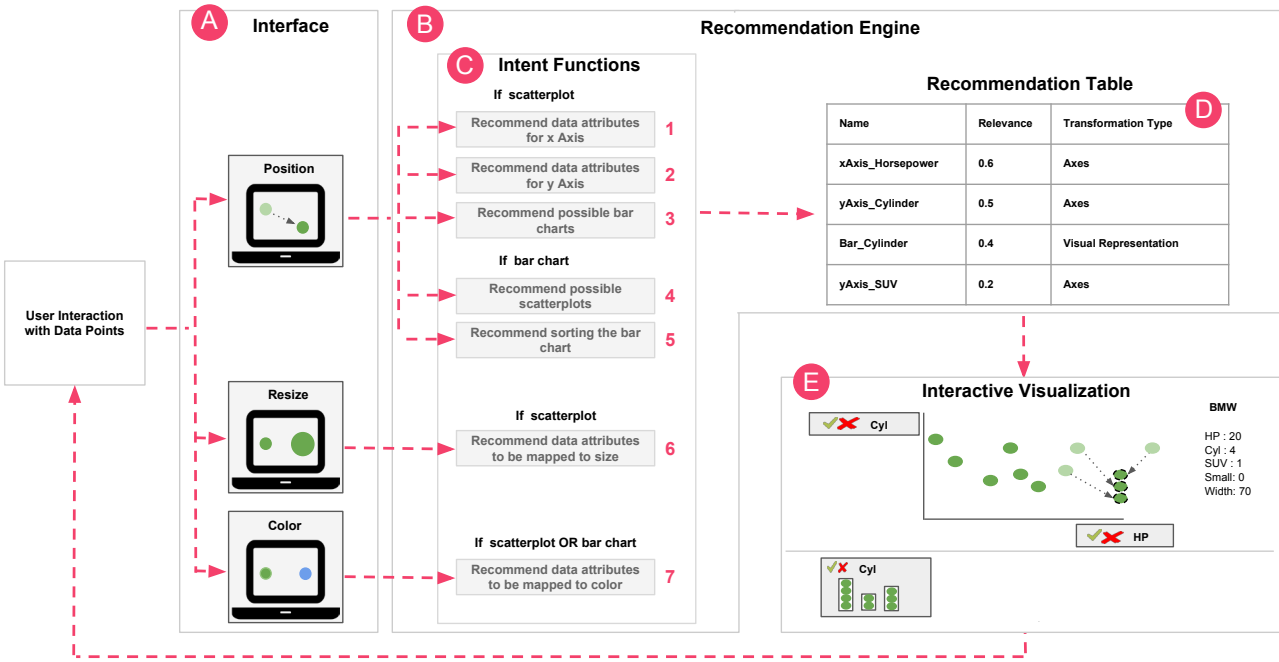


Figure 6: VisExemplar's Low-level Architecture. A) Recommendation engine takes user interactions as input. B) A series of intent functions drive the recommendation table. C) Direct manipulation of each encoding will invoke a series of intent functions related to that specific encoding. D) Recommendation Table will be updated after each interaction and stores a ranked list of potential transformations. E) The updated recommendation table feeds the recommendations in the user interface.

point invokes intent function 6. Recoloring a data point will invoke intent function 7 regardless of the current state of visualization. Below we explain how each of these functions work. Full support of Visualization by Demonstration will require additional intent functions and our system design supports this extensibility.

The notations used in this section are summarized in Table 1. We refer to the data generally in normalized form, i.e. scaled into the interval $[0,1]$ by attribute:

$$\tilde{d}_{ij} = \frac{d_{ij} - \min(\mathbf{d}_{\cdot j})}{\max(\mathbf{d}_{\cdot j}) - \min(\mathbf{d}_{\cdot j})}, \tag{1}$$

where $\min(\mathbf{v})$ and $\max(\mathbf{v})$ indicate the smallest and largest element respectively of vector $\mathbf{v}$.

Position: Depending on the type of the current visualization and user interactions, upon changing the positions of data points, intent functions 1, 2, or 3 (for a scatterplot), or functions 4 or 5 (for a bar chart) are triggered. In this section, we describe how each of these five functions will be triggered after moving the data points.

If the current visualization is a scatterplot, upon the movement of the point, the system either recommends changing the Axes (changing the attributes assigned to x or y-axis) or changing the visual representation to a bar chart.

Intent Function 1 (Figure 6-(C)-1): After a position-changing interaction, the system searches for data attributes to assign to the axes in a scatterplot based on the positions of the moved data points. For example, in Figure 7, the user starts with a scatterplot whose x-axis represents miles-per-gallon (MPG). When the user moves a data point (black arrow in Figure 7 (a)), the x coordinates of the scatterplot no longer map to MPG. Rather, their position is better aligned with the length attribute (Figure 7 (b)). In this case, the system recommends assigning length to the x-axis.

In detail, we linearly normalize coordinate vectors $\mathbf{x}$ and $\mathbf{y}$ into $\tilde{x}$ and $\tilde{y}$ in the same way as Eq. 1, so that they are in the range of $[0,1]$. As a result, coordinate values and data attributes values are in the same scale. Then, we find an data attribute that minimizes the sum of squared differences between normalized coordinate values and data attribute values of data points. In other words, the system recommends attribute j^* to be assigned to x-axis such that

$$j^* = argmin_j \left\| \tilde{\mathbf{d}}_{\cdot j} - \tilde{\mathbf{x}} \right\|_2^2.$$

Table 1: Notation used in this paper.

NOTATION	DESCRIPTION
n	The number of data points
m	The number of data attributes
$\mathbf{d}_i$	The i-th data point
d_{ij}	The j-th attribute value of the i-th data point
$\mathbf{d}_{\cdot j}$	The j-th attribute vector
$\mathbf{x}$, $\mathbf{y}$	The vector of plotted coordinates in the x (or y) axis
x_i, y_i	The plotted value of the i-th data point in $\mathbf{x}$ or $\mathbf{y}$
c_i	The plotted color of the i-th data point
a_i	The plotted area of the i-th data point
$\tilde{v}$	The normalized value of vector v into the interval $[0,1]$
$\|S\|$	The number of elements in a set S

Intent Function 2 (Figure 6-(C)-2): In the same way, the system also recommends data attribute(s) for the y-axis of a scatterplot.

Intent Function 3 (Figure 6-(C)-3): If a position-changing interaction results in more than three data points lined up in a row, the system then checks if the three data points are within a specified distance from each other. If so, the system detects user interest for transforming the visualization into a bar chart. In other words, the system interprets the visual demonstration of "stacked data points" as a user's interest in transition to a bar chart. The function then computes common data attributes shared by the aligned data points. Finally, it updates the recommendation table. For instance, if a user places four data points that represent SUVs with six cylinders on top of each other, the system recommends a bar chart by car type *(e.g., SUVs, sports cars, etc)* and a bar chart by the number of cylinders.

If the current visualization is a bar chart, users can move the data points inside each bar or move a bar itself within the bar chart. Note that each bar is shown as a visible box which contain a set of corresponding data points.

Intent Function 4 (Figure 6-(C)-4): If the user drags a data point out of a bar in a bar chart, the system interprets it as a demonstration of changing the visual representation to a scatterplot. After the user drags out two or more data points, the system searches for data attributes that will be assigned to axes of a new scatterplot (using a similar method to the Intent Functions 1 and 2). Based on the current x and y coordinates of the moved points, the system recommends potential x and y axes so that the new scatterplot representations of the moved points would be similar to the current user-defined positions. The recommendations with previews will be shown in the Recommendation Gallery.

Intent Function 5 (Figure 6-(C)-5): Users can drag and drop any of the bars shown in a bar chart. If the user drags the longest bar in the bar chart to the left most side of the bar chart the system recommends sorting the bar chart descending. If the user drags the longest bar in the bar chart to the right most side of the bar chart the system recommends sorting the bar chart in the ascending order.

Resizing: Users can resize data points any time during the data exploration process. Users can adjust the size by dragging a small handle (tiny black circle) on the perimeter of the data point.

Intent Function 6 (Figure 6-(C)-6): When a user resizes a data point in a scatterplot, the system interprets it as the user's interest in encoding a data attribute to the demonstrated sizes of data points. In order to provide enough information to the system for recommending a mapping from a data attribute to data point sizes, the user has to resize two or more data points. The system shows recommended transformations that are above a developer-defined threshold by showing a expand icon (⤢) beside those attributes on the Detail View. Specifically, we first normalize the sizes of data points the user has adjusted in a way that reflects the fact that the minimum is nonzero and that we are seeking changes in the same direction (bigger or smaller) by setting the default drawing value to 0.5. We calculate a scaled value $\tilde{a_i}\ \forall i \in S$ (the set of modified points) as follows:

$$\tilde{a_i} = \begin{cases} \frac{1}{2}\left(1 + \frac{a_i - a_0}{\max(a) - a_0}\right) & \text{if } a_i > a_0 \\ \frac{1}{2}\left(\frac{a_i - \min(a)}{a_0 - \min(a)}\right) & \text{if } a_i < a_0 \end{cases},$$

where a_i is the current plotted size of the i-th data point, a_0 is the default plotting size, and the max and min functions return the predetermined maximum and minimum drawing sizes for points in the visualization. The system recommends attribute j^* for size encoding such that

$$j^* = argmin_j \sum_{i \in S} (\tilde{d_{ij}} - \tilde{a_i})^2,$$

where S is the set of resized data points.

Recoloring: Users can recolor a data point by right clicking on it and picking a color from the pop-up menu. VisExemplar currently supports three colors: red, blue, and green (default).

Intent Function 7 (Figure 6-(C)-7): We now describe the intent function triggered by changing the colors of data points. For coloring interactions, we consider categorical attributes only and ignore numerical attributes. We define an attribute as categorical if the number of unique values present is fewer than ten and also fewer than half of the number of data points. That is, attribute j is categorical if and only if

$$|\text{unique}(\mathbf{d}_{\cdot j})| \leq \min(10, \frac{n}{2})$$

If the user changes the color of one data point, the system makes a recommendation for each categorical attribute, suggesting applying the same recoloring to all other points sharing the value. For example, if the user changes the color of an AWD sedan with 6 cylinders to red, the system recommends three options: coloring all AWD vehicles red, coloring all sedans red, or coloring all 6-cylinder cars red.

When a user colors two or more data points, two conditions are checked to find the appropriate mapping. The first checks for positive correspondence between a data attribute and the assigned color. The condition is satisfied for an attribute if all points given the same color also have the same value for that attribute. The second condition tests

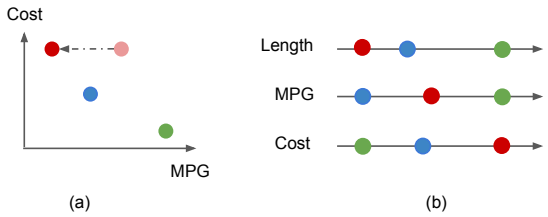


Figure 7: Position-changing interaction. (a) A scatterplot with MPG as x-axis and cost as y-axis. A user moves a red data point. As a result, the system recommends assigning length attribute to x-axis. (b) A visual representation of data distributions for potential data attributes.

that whenever two points have different colors, they have different attribute values. Given a set of indices of k modified points $i_1, \ldots, i_k$, the two conditions on attribute j are, for all pairs $(k, p) : k, p \in i_1, \ldots, i_k$ and $k \neq p$:

$$\text{Condition I: If } c_k = c_p, \text{ then } d_{kj} = d_{pj}$$

$$\text{Condition II: If } c_k \neq c_p, \text{ then } d_{kj} \neq d_{pj}$$

If all the user-colored data points have the same color, the system checks every categorical data attribute to see if its attribute values of the colored data points are the same using Condition I. The system then recommends all data attributes that meet Condition I. For example, suppose the user colors an AWD sedan with 6 cylinders and a non-AWD sedan with 6 cylinders red. Since both cars are 6-cylinder sedans, two attributes, car-body-type (which includes sedan) and number-of-cylinders, satisfy Condition I. The system recommends two options: coloring all sedans red or coloring all 6-cylinder cars red.

If the data points are colored with two or more colors, the system uses both conditions to evaluate attribute mappings. The two or more colors specify not only the mapping of color to an attribute, but the assignment of values of that attribute to one specific color. First, Condition I is applied across each subset of the re-colored points that have been assigned the same color. This discovers the data attributes shared by each colored group of the modified points. Second, Condition II is applied with the attributes revealed by Condition I to find which attributes can account for the differences across color groups. For example, suppose the user re-colors three data points: two representing cars with attribute values given by the tuples (AWD, sedan, 6 cylinders) and (FWD, sedan, 6 cylinders) are colored red; the third, (AWD, sedan, 4 cylinders), is colored blue. In the red group, both body-type *(e.g., sedan)* and number-of-cylinders satisfy Condition I. However, when Condition II is checked, we see that only the cylinder attribute satisfies both conditions. The system recommends mapping the cylinders attribute to color by coloring 6-cylinder cars red and 4-cylinder cars blue. A brush icon (🖌) beside each of the candidate attributes on the Detail View shows the recommendation to the user.

5 Discussion and Future Work

5.1 Interoperability with Direct Manipulation

There is an inherent tradeoff between the flexibility of Visualization by Demonstration and the loss of formality and expressiveness. A given demonstration could imply multiple transformations and multiple demonstrations might imply the same transformation. Of course this many to many relationship between given demonstrations and transformations raises technical challenges in translating given demonstrations into meaningful transformations. Early on during the process of providing demonstrations, mapping the given demonstrations to possible transformations might be more ambiguous but as users continue completing their demonstrations and providing more evidence and training data to the system, the number of possible transformations would decrease (and ideally become more accurate with respect to the user's goals). This is similar to interactive machine learning [11], in which more examples lead to more accurate decisions. For instance, by coloring a single data point blue, there may be many recommended transformations because the system has less accuracy

about the user intentions. However, coloring a few more data points would increase the accuracy of the recommended transformations.

Visualization by Demonstration can be used independently or augment the interaction design of existing visualization tools. For example, the interaction design of SpotFire [1] can be enhanced by Visualization by Demonstration. In this case, expert users could create a bar chart by specifying the data attributes for $x-y$ axes and a visualization technique from the control panel. In addition, non-expert users could benefit from the Visualization by Demonstration approach by providing visual demonstrations to the visualization incrementally and letting the system compute possible transformations.

An important avenue for continued research is conducting a study to compare Visualization by Demonstration with interaction paradigms applied in other existing visualization tools. This requires a separate in-depth study utilizing both qualitative and quantitative techniques to measure the impact of Visualization by Demonstration compared to other interaction methods, using various usability *(e.g., time and error)* and user experience *(e.g., engagement)* metrics. We anticipate that using the Visualization by Demonstration paradigm will increase user engagement, but this remains to be formally studied.

5.2 Generalizing Visualization by Demonstration

We developed VisExemplar to show the feasibility of Visualization by Demonstration. The current version of VisExamplar supports two types of visualization techniques (bar chart and scatterplot) and direct manipulation of three graphical encodings (position, size, and color). It supports interactions which are recognized to be meaningful by previous work. We view the current version of VisExemplar as the first step towards exploring the Visualization by Demonstration paradigm. Multiple avenues for future work lie in improving the VisExamplar interface, as well as enriching the Visualization by Demonstration idea space. We envision expanding VisExamplar by including other visualization techniques *(e.g., linecharts)* and graphical encodings *(e.g., angle and volume)*, and working towards a generalizable interaction framework for visualization.

Generalizing Visualization by Demonstration requires support for providing demonstrations to imply more sophisticated *analytic operations* and *visualization techniques*. For example, how can users indicate their interest in data grouping or aggregation? This requires demonstrations that trigger analytic operations on the data, and show the results visually. For example, users could draw regions around specific data points to demonstrate their interest in executing a clustering algorithm. Additionally, computing standard error to support error bars in bar charts can be demonstrated by drawing the error bars directly. Using demonstration techniques as a medium for performing analytic operations can make complex computation more accessible to a broader set of users.

Further, how can users indicate their interest in transforming to visualization techniques that use a different graphical encoding to encode data points *(e.g., while data points in a scatterplot are encoded using $x-y$ position in a Cartesian space, parallel coordinates encode these data points using lines across multiple parallel axis representing data attributes)*. Implying this type of transition requires new strategies for providing demonstrations. For example, users could connect two data points in a scatterplot using a line to demonstrate their interest in switching to a line chart. Additionally, users could draw vertical parallel lines on a scatterplot to show their interest in switching to a parallel coordinates. We believe that Visualization by Demonstration can generalize to many visualization techniques, although additional forms of interaction may be needed (e.g., sketching and visual authoring).

Generalizing Visualization by Demonstration should be properly evaluated. There exist multiple methods to demonstrate specific transformations. Some are likely more easy to use than others. One research direction might be to observe different strategies people use to demonstrate their interest in more sophisticated operations, extract the common strategies used, and adapt them to expand the Visualization by Demonstration paradigm.

5.3 Consistency of Visual Mappings

Not all the visual mappings used in a visualization technique can be maintained throughout the exploration process. Similar to the existing tools, when transforming from one visualization technique to another, we are required to reset some of the encodings to their initial values. For example, if size is mapped to a data attribute in a scatterplot and then the user changes the technique to a bar chart, the system resets size because it is inaccurate to have circles with different sizes where the y axis indicates the number of data items. However, we can envision creating hybrid visualization techniques and transformations in the future that allow automated swapping of data mappings to other valid visual encodings when switching techniques.

5.4 Recommendation Presentation and Timing

Although exploring different ways and timings of representing transformations is not our main focus in this work, during the design process of VisExemplar, we examined different ways of presenting recommended transformations.

Similar to many existing tools *(e.g., [4, 12, 22, 23, 33])*, we first decided to show all the recommended transformations as small thumbnails in the Recommendation Gallery. We tried this method for *Visual Representation transformations* and it worked well. However, by adding other types of transformations to the Gallery, we faced two challenges. First, we found exploring the Gallery consisting of different types of transformations shown in one place confusing. Interpreting what the recommended transformation changes from the current view to the recommended view was not clear. The second challenge was the large number of potential recommendations shown in the Gallery made it less readable. We thought it might be a better idea if we located different types of transformations in places which are still meaningful to users and easy to understand. We tried different places and finally decided to put the *Axes transformations* and *View Specification transformations* on the ThinkBoard, and *Data Mapping* transformations in the Detail view. One possible follow-up research direction includes exploring methods of presenting recommendations in visualizations and evaluating their effectiveness.

In the current version of VisExemplar, the recommended transformations will be updated in the interface whenever the recommendation table in the recommendation engine gets updated. However, one interesting research direction includes understanding the impacts of interruptions caused by incoming recommendations and investigating methods for minimizing the interruption caused by incoming recommendations. For example, one way could include presenting recommendations upon pressing a specific button on the interface. Alternatively, the system could observe the cadence of user interaction with the system and propose recommendations at a less active time.

6 Conclusion

This paper introduces Visualization by Demonstration, a user interaction paradigm for visual data exploration. The paradigm advocates for mapping visual demonstrations provided by users to meaningful visual transformations and recommending them to users. Users are able to provide demonstrations of intended changes to an existing visualization, and the system computes the appropriate transformations to accommodate the desired change as closely as possible. In order to demonstrate the technical feasibility of this paradigm, we developed a prototype called VisExemplar. VisExemplar allows users to explore their data via Visualization by Demonstration. After users provide visual demonstrations, the system recommends possible transformations based on the given demonstrations. VisExemplar computes these recommendations through a set of intent functions used to compute a best fit from the provided visual demonstrations and resulting transformations. While this paper has taken initial steps to introduce the concept of Visualization by Demonstration, there exist several exciting avenues for continued research.

7 Acknowledgements

We would like to thank the reviewers and GT Visualization Lab members for their feedback.

REFERENCES

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

[2] C. Andrews, A. Endert, and C. North. Space to think: large high-resolution displays for sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 55–64. ACM, 2010.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[4] F. Bouali, A. Guettala, and G. Venturini. Vizassist: an interactive user assistant for visual data mining. *The Visual Computer*, pages 1–17, 2015.

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

[6] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984.

[7] W. S. Cleveland and R. McGill. Graphical perception and graphical methods for analyzing scientific data. *Science*, 229(4716):828–833, 1985.

[8] A. Cypher and D. C. Halbert. *Watch what I do: programming by demonstration*. MIT press, 1993.

[9] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 473–482. ACM, 2012.

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

[11] J. A. Fails and D. R. Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003.

[12] D. Gotz and Z. Wen. Behavior-driven visualization recommendation. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 315–324. ACM, 2009.

[13] H. V. Henderson and P. F. Velleman. Building multiple regression models interactively. *Biometrics*, pages 391–411, 1981.

[14] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166. ACM, 1999.

[15] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer. Constructive visualization. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 433–442. ACM, 2014.

[16] S. Huron, Y. Jansen, and S. Carpendale. Constructing visual representations: Investigating the use of tangible tokens. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2102–2111, 2014.

[17] T. Igarashi and J. F. Hughes. A suggestive interface for 3D drawing. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pages 173–181, New York, NY, USA, 2001. ACM.

[18] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *ACM Human Factors in Computing Systems (CHI)*, 2011.

[19] B. Kondo and C. M. Collins. Dimpvis: Exploring time-varying information visualizations by direct manipulation. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2003–2012, 2014.

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

[21] J. Lin, J. Wong, J. Nichols, A. Cypher, and T. A. Lau. End-user programming of mashups with vegemite. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 97–106. ACM, 2009.

[22] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1137–1144, 2007.

[23] D. B. Perry, B. Howe, A. M. Key, and C. Aragon. Vizdeck: Streamlining exploratory visual analytics of scientific data. 2013.

[24] D. Ren, T. Höllerer, and X. Yuan. iVisDesigner: Expressive interactive design of information visualizations. *IEEE transactions on visualization and computer graphics*, 20(12):2092–2101, 2014.

[25] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. *Computer Graphics Forum (Proc. EuroVis)*, 2014.

[26] D. Schroeder and D. F. Keefe. Visualization-by-sketching: An artist's interface for creating multivariate time-varying data visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):877–885, 2016.

[27] F. M. Shipman III and C. C. Marshall. Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems. *Computer Supported Cooperative Work (CSCW)*, 8(4):333–352, 1999.

[28] B. Shneiderman. Dynamic queries for visual information seeking. *Software, IEEE*, 11(6):70–77, 1994.

[29] SpotFire. http://www.spotfire.com, 2015.

[30] Tableau. Tableau software, http://www.tableau.com/, 2015.

[31] J. Walny, S. Carpendale, N. H. Riche, G. Venolia, and P. Fawcett. Visual thinking in action: Visualizations as used on whiteboards. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2508–2517, 2011.

[32] J. Walny, B. Lee, P. Johns, N. H. Riche, and S. Carpendale. Understanding pen and touch interaction for data exploration on interactive whiteboards. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2779–2788, 2012.

[33] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2015.

[34] J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1224–1231, 2007.

[35] M. M. Zloof. Query by example. In *Proceedings of the May 19-22, 1975, national computer conference and exposition*, pages 431–438. ACM, 1975.