Peek Brush: A High-Speed Lightweight Ad-Hoc Selection For Multiple Coordinated Views

Wolfgang Berger, Harald Piringer
VRVis Research Center, Vienna, Austria
berger@vrvis.at, piringer@vrvis.at

Abstract

Linking+Brushing is a proven concept to reveal relationships across multiple views. Defining complex selections, however, may involve a significant interaction overhead. This paper proposes Peek Brush, a point-brush that is designed to temporarily select and highlight items hovered by the user's mouse cursor. This enables quickly skimming through the data to identify relationships between different data projections within seconds. The Peek Brush serves the purpose of defining a starting point to a more focused inspection using brushes with higher complexity. In order to achieve rapid visual updates, we discuss acceleration techniques like preprocessing, threading, and layering. As a result, the Peek Brush is able to scale to datasets with millions of entries. A case study demonstrates how the Peek Brush minimizes the interaction effort required from the user. It delivers a quick overview and reduces the time needed for the initial visual analysis step from minutes to seconds.

Keywords—**Linking+Brushing, Detail-On-Demand**

1 Introduction

Linking+Brushing has become a common interaction and visualization technique in the field of Information Visualization (InfoVis) [6]. Providing means to define a selection directly in a visualization and to highlight the affected items in all views helps the user to form a mental representation of the analyzed abstract dataset [7].

Brushes are visual metaphors for dynamic queries and are designed to make the definition of such queries more intuitive for the user. Depending on their shape, size, and additional parameterizations like thresholds they can perform complex selections. Combining multiple brushes using methods like boolean or fuzzy operations [5, 15] further increases the query complexity. While such queries may require a considerable amount of user interaction for proper configuration, they are able to define interesting features in an interchangeable way. Composite brushes may thus represent an important result of an analysis.

In contrast to such complex queries with persistent semantics, deciding on the next investigation steps during an analysis requires quick and temporary information about certain visualized items. This may, for example, include skimming through different points of a scatterplot or through all categories of a bar chart. Such requests are of interest for seconds only, much in contrast to the brushing interaction described in the previous paragraph. While many systems support such quick requests, the offered information is mostly local and provided via mechanisms like tooltips. Relationships which only become visible through linking, however, cannot be revealed.

This paper introduces *Peek Brush*, a concept to extend the information offered for temporary requests to multiple different views. Triggered by simply pointing at items of interest, the Peek Brush highlights corresponding visual representations in all linked views. The goal is to provide rich information in various visualizations while minimizing the amount of required interaction effort by the user (e.g., no mouse clicks). The Peek Brush may co-exist with persistent brushes, but defines a separately handled query based on the currently displayed data items. The selection as well as the visual updates of all linked views have to be designed to happen within 100 ms or less after mouse cursor position changes in order to meet the requirement for immediate feedback defined by Shneiderman [12]. Thus it is necessary to consider view dependent optimization techniques and trade-offs which help to ensure that this high-frequency brushing technique scales to datasets with millions of entries and up to a hundred dimensions.

2 Related Work

Dix and Ellis [3] identified interaction as one of the key concepts in visualization and proposed using techniques like highlighting, drill-down or overview and context along with direct manipulation to overcome trade-offs and limitations introduced by static visualizations. The Task By Data Type Taxonomy [13] groups data interaction and visualization techniques in terms of seven general tasks and distinguishes them based on the type of data they are intended for. Yi et al. [16] suggest a different taxonomy which focuses exclusively on interaction techniques for which they

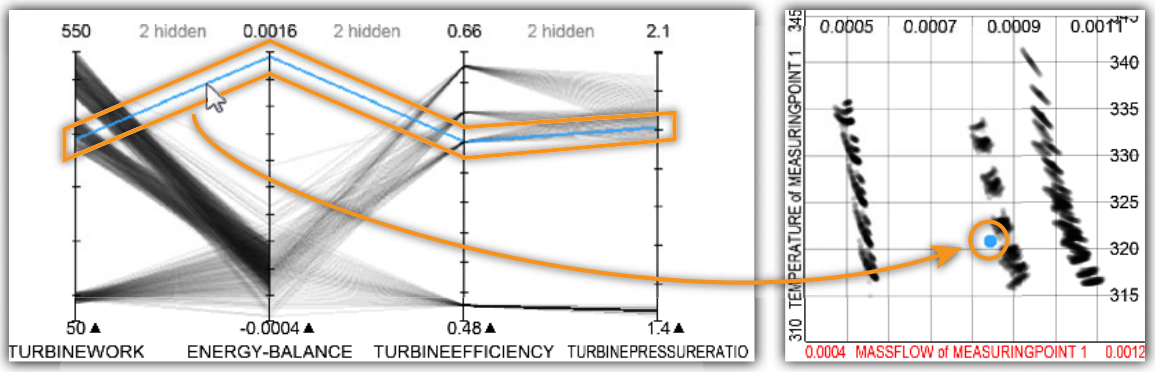

Figure 1: Hovering the outlier of "Energy-Balance" in the Parallel Coordinates with the mouse cursor temporarily selects the entry and immediately highlights the corresponding representation in the linked 2D scatterplot.

define seven high-level categories based on user intents. Both taxonomies consider selection of entries, relationship between items, and detail on demand as important aspects of interaction.

Brushing is a technique that has been proposed by Becker and Cleveland [2] and lets the user define and modify such dynamic queries directly in a visualization. While being intuitive in terms of application and expected behavior, it becomes most useful if used in combination with linking [7]. The latter describes a mechanism that connects multiple views on the same data and highlights brushed items in all visualizations. Shneiderman introduced dynamic queries as a way to incrementally move from a rough overview to a filtered data representation [12]. In contrast to direct interaction within visualizations, this early approach relies on separate query widgets.

Coordinated Multiple Views have been proposed as a term that generalizes Linking+Brushing to any action within a view that is propagated to the whole system (e.g. rotation, zoom). The goal is to aid the user in correctly interpreting the displayed data [11]. Visualizing the same dataset in multiple ways and forwarding interactions performed in one view to all other linked views contribute to a better understanding of relations and dependencies. Wang Baldonado et al. define additional guidelines for the use of multiple coordinated views which point out advantages and trade-offs for aspects like view consistency, complementarity and application of perceptual clues [14].

Munzner et al. [9] apply linked mouse-over highlighting to two tree visualizations in the TreeJuxtaposer tool in order to quickly expose similar tree nodes. Our proposal generalizes this approach to be applicable for many different types of linked views.

3 The Peek Brush

The Peek Brush as depicted in Figure 1 is a light-weight brush that is characterized by the following properties:

- It requires minimal interaction effort, i.e. no mouse clicks.
- The selection is based only on the mouse cursor position in context of the underlying visualization.
- It immediately highlights the selected subset in all views.

The Peek Brush is thus a tool to get quick information about properties and relationships for individual primitives in a visualization. Examples for such visualization primitives are bars in a histogram, points in a scatterplot, or lines in a parallel coordinate representation. Each of them represents one or more entries of the underlying dataset. The Peek Brush allows to successively investigate many primitives within a very short time. Simply moving the mouse generates an animation-like data walkthrough that characterizes entries consecutively and may help to visually identify differences between them.

It is vital that the system response time keeps up with the typically high-frequent query changes while the user is moving the mouse. Continuous visual updates within 100 ms are defined as important for data exploration [12]. Within the available response time, the affected visualization primitives have to be determined (also known as picking) and a back-projection to the associated data entries has to be performed. All linked views have to update themselves in order to visually represent the selection by the Peek Brush. Depending on the type of view, certain considerations and trade-offs have to be made, which are now discussed.

3.1 View Dependent Design Considerations

When designing a view, it is important to distinguish between *active* and *passive* support for the Peek Brush. *Passive* support refers to visualizing the data subset that has been selected by the Peek Brush. This entails the translation into the visual metaphor used by the respective vi-

sualization and the actual rendering within the available time frame. *Active* support means picking the visualization primitives at the mouse cursor position along with determining and selecting the associated data items. Strategies for active or passive support depend on the type of visualization.

3.1.1 Entry-Based Visualizations

In entry-based visualizations, every entry of the dataset has its own visual representation. Examples are scatterplots or parallel coordinates. Passive support of the Peek Brush involves varying visual attributes for the selected data items, e.g., drawing them using a different color, size, or shape. Active support refers to selecting data entries of which the visual representation is below or close to the mouse cursor.

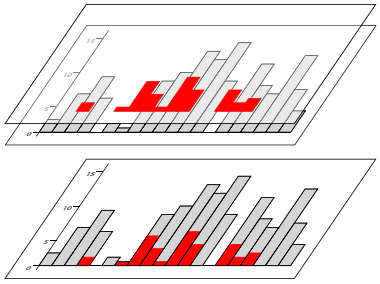

Figure 2: A background layer caches the parts of the visualization which are not affected by the passive Peek Brush representation on top of them. This layer can be reused whenever the view is updated due to Peek Brush changes.

Due to the requirement of immediate response, supporting the Peek Brush in item-based visualizations becomes increasingly challenging for truly large data sets. Naïve approaches do not scale for data sets beyond a few hundred entries. Such approaches involve processing the visualization pipeline for all data entries at every update or performing a hit-test against every item at every mouse move. As a solution, discriminating visual layers in image space as shown in Figure 2 and using multi-threaded rendering make passive support of the Peek Brush scalable up to millions of entries. Previous work describes these techniques in detail [10]. Concerning active support, preprocessing helps to significantly reduce the effort of picking data items at the cost of increased memory consumption. Such a preprocessing typically creates a data structure that enables a direct look-up of entries for a given position in image space. In order to trade off precision against memory consumption, these data structures typically bin the image space as illustrated in Figure 3.

3.1.2 Frequency-Based Visualizations

Frequency-based visualizations typically draw less primitives as they group data entries according to a certain criterion like “value” or “category”. A visual representation for each of these groups reflects the respective properties. For example, a bar in a histogram is defined by the covered value range and the number of entries within this range. Entries, as selected by the Peek Brush are subsets of the groups. As such, each subset has to be visually related to the whole group (e.g. by visualizing the relative size, see Figure 2). However, the involved calculations may take some time for large subsets. In this case, multi-threading and Early Thread Termination as described in [10] can be used to prevent blocking the whole application.

On the other hand the comparably low number of visual items in a frequency-based visualization alleviates fast picking. It is even possible to pick in constant time for visualization techniques where no overlay between items may occur (e.g. histograms, bar charts).

3.2 Implementation Examples

This section describes concrete implementation examples for views that support the active Peek Brush by means of a 2D scatterplot, parallel coordinates, and a histogram.

All views are part of VISPLORE, a system for visual exploration. In order to ensure immediate response while scaling up to millions of data entries, the views implement the architecture as described in previous work [10]. However, in order to meet the demanding requirements concerning performance of the Peek Brush, further optimization concepts have been integrated, as discussed below.

3.2.1 2D Scatterplot

Concerning passive support of the Peek Brush, the 2D scatterplot separates the visualization of the items as selected by the Peek Brush from the remaining visualization. The reason is that the contents of the Peek Brush are changing rapidly when the user is moving the mouse, while other parts of the visualization are not affected and can thus be regarded a kind of background. Technically, this background is cached in image space as a texture that is reused whenever a redraw of the Peek Brush is necessary. The items of the Peek Brush itself are drawn on top of this texture. While this involves processing the entire visualization pipeline for the selected data entries, it is still fast when the subset of items of the Peek Brush is small which is typically the case.

Active support of the Peek Brush utilizes an acceleration technique as motivated in Section 3.1.1. The basic approach is to keep an index list for each pixel of the 2D scatterplot which stores the respective associated data entries. Whenever the active Peek Brush needs to be updated due to a position change of the mouse cursor within the view, the affected bin can be determined from the x- and y-coordinates with constant effort. Depending on the size of the scatterplot points, however, one point may cover more pixels which causes an individual entry being indexed in

many different bins. This leads to a memory consumption that is a multiple of the represented data points. We therefore consider only the center of each point, so that each entry is indexed once. Consequently, the memory consumption stays reasonably low also for large data.

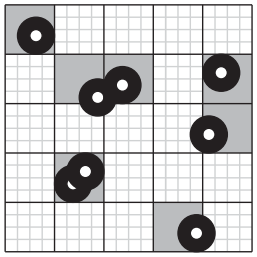

Figure 3: Each block in the image space of the 2D scatterplot maintains the indices of points that have their center located in it.

On the other hand, since the indexing is performed in image space, the amount of used memory is also dependent on the screen size of the scatterplot. In order to reduce this amount for high resolutions, we perform subsampling by using a regular grid as illustrated in Figure 3. Our implementation subdivides the image space into 4 x 4 pixel sized blocks. Our experience shows that the loss of accuracy caused by these blocks is negligible in practice for point sizes larger than two pixels.

3.2.2 Parallel Coordinates

For active support of the Peek Brush, Parallel Coordinates employ a similar acceleration structure like the 2D scatterplot. In contrast to the 2D scatterplot, however, the pixels covered by a single data entry are not close together, but spread across the whole width of the view. It is therefore inevitable to refer to the same data entry in multiple locations. In order to overcome this possibly excessive memory consumption, we limit the number of entries per location to a fixed value. If this limit is reached, the index list is disposed and the corresponding location is treated as if it was empty for the Peek Brush since it covers too many overlapping items. This thresholding could be seen as a major trade-off, but it ensures that the active Peek Brush in the parallel coordinates scales to millions of entries while having no accuracy impact on representations of smaller datasets.

Our implementation subsamples the image space by blocks of 4 x 4 pixels as described for the 2D scatterplot. Rasterizing the polyline of each entry into these blocks, however, makes the necessary preprocessing more expensive than for the 2D scatterplot.

The passive support of the Peek Brush subdivides the visualization into layers and renders them using textures as described in Section 3.2.1.

3.2.3 Histogram

The histogram serves as an example for the implementation of the Peek Brush in a frequency-based visualization. The regular positioning of the visualization primitives without overlap allows for picking a histogram bar within constant processing time. As for the 2D scatterplot and the Parallel Coordinates, quick back-projection of the selection requires storing the entry indices per bar during the binning process. The additional memory that is required for keeping these indices scales linearly with the number of data items and is therefore (as in the case with the 2D scatterplot) an acceptable trade-off.

Regarding passive support of the Peek Brush, the necessary binning of the selected items might become the bottleneck during an update. Since the contents of a Peek Brush selection can be regarded as volatile, accelerating the binning through the use of precomputations is not possible. However, we observed that in most cases the number of entries selected by a Peek Brush is small enough to allow for binning and drawing within the available time. Otherwise support for the Early Thread Termination pattern [10] ensures responsiveness of the view.

4 Results

We now demonstrate how the Peek Brush accelerates a visual exploration by means of an application scenario. The analyzed dataset contains the results of 4160 simulation runs, which have been performed during the design stage of a four cylinder turbodiesel car engine by AVL [1], a company in the field of powertrain simulation, measurement and design. The outcome of a single run predicts properties like torque and fuel consumption at defined measuring points. In order to evaluate several design options with different engine states (e.g. engine load and speed), this high number of different runs is necessary. Matković et al. [8] describe an extensive analysis of such data.

The goal of the presented example is to analyze the behavior of the engine's turbo charger at the simulated engine speeds. Moreover the associated general temperature and massflow characteristics at the first measuring point of the exhaust pipe are investigated, as they impact the efficiency of the turbo charger.

As a first step, all values for engine speed (measured in rotations/minute) that occur in the dataset are assigned to a categorical visualization as distinct classes. Each class is represented as a box as seen in the leftmost images of Figure 4. The width of such a box is scaled depending on the relative frequency of the respective category. Furthermore massflow (kg/cycle) and temperature (K) are assigned to the axes of a scatterplot. As the third visualization, Parallel Coordinates contain axes for four different operating characteristics of the turbo charger.

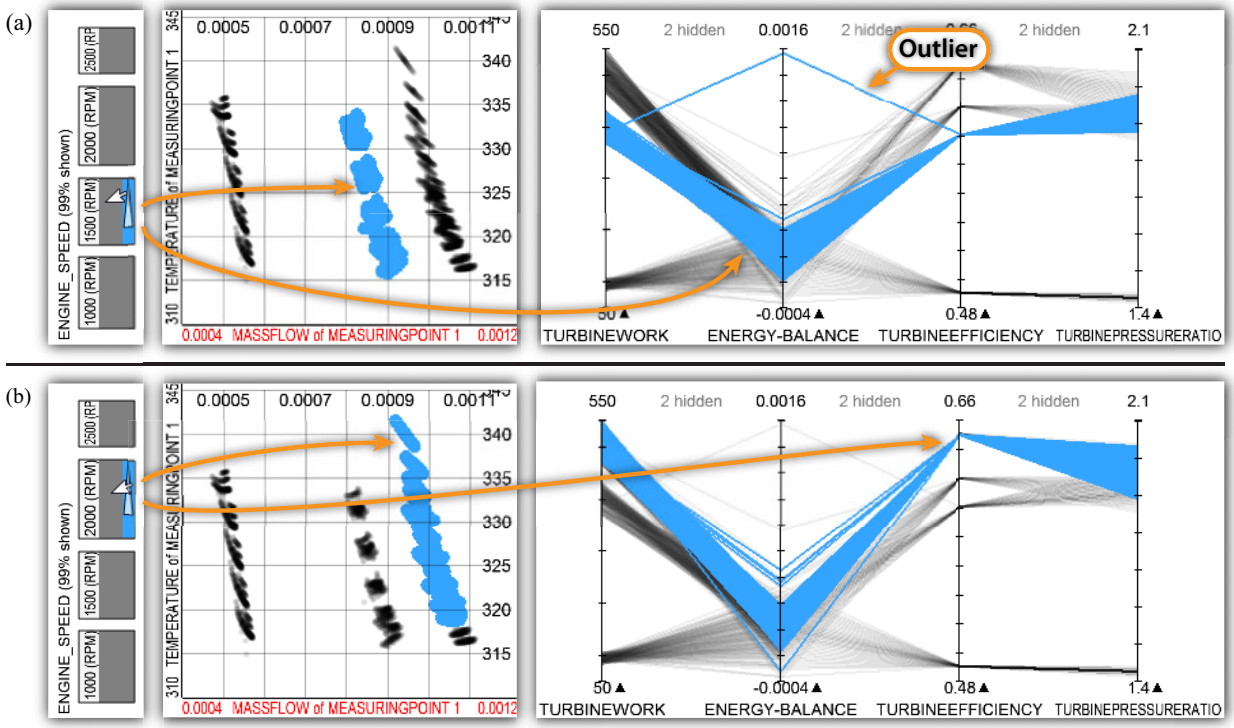

Figure 4: Hovering across the four simulated engine speeds in the categorical visualization (left) immediately highlights the associated entries in all other views. (a) shows the Peek Brush for speeds of 1500 rpm, where "Energy-Balance" contains an outlier as seen in the Parallel Coordinates (right). (b) highlights simulations for 2000 rpm, where Parallel Coordinates show that the turbine of the turbocharger peaks in terms of work, efficiency and pressure ratio. The exhaust temperature, which is visualized in the 2D scatterplot, reaches its maximum as well.

Hovering across the four simulated engine speeds immediately highlights the associated simulation runs within the other two plots. Moving the mouse cursor from 1000 rpm to 1500 rpm (Figure 4 (a)) shows a general rise of massflow. The turbocharger starts unfolding its capacity, as the turbine work increases significantly along with the efficiency and pressure ratio. When hovering to 2000 rpm (Figure 4 (b)), it is instantly visible, that the turbocharger reaches its maximum in terms of work, pressure ratio and efficiency, which has a value of 65 %. The relation to the exhaust characteristics is reflected in the linked 2D scatterplot which highlights points of high temperature and massflow. At 2500 rpm the performance ratings decline along with exhaust temperature. Massflow reaches its maximum values due to the high engine speed.

The Parallel Coordinates show an interesting outlier for the attribute "Energy-Balance" at a speed of 1500 rpm. In order to inspect the associated point in the 2D scatterplot, the respective line representation is simply hovered as shown in Figure 1 and immediately highlighted in all views. As we can see, also the temperature and massflow values reflect a slight deviance, since the respective point is located outside of the cluster that is formed by simulation runs at the same engine speed. As a result, this single run could either become subject to more detailed investigation or excluded from the following data analysis.

All insights described in this section have been gathered within fifteen seconds of interaction (excluding setting up the three views), which fortifies the Peek Brush's potential to save the user a significant amount of time by reducing complexity and interaction effort to a minimum.

5 Discussion and Future Work

We believe that the Peek Brush functionality can be successfully integrated in many different types of visualizations. Currently, our InfoVis system VISPLORE implements it for 2D scatterplots, Parallel Coordinates, time series visualizations, function graphs, histograms, and categorical representations. Previous work [10] has shown providing immediate visual feedback is possible even for millions of entries by utilizing multi-threaded rendering, visual layering, and view specific optimizations as described in Section 3.

The use of image space precomputations for accelerated picking, however, can come at the cost of high additional

memory consumption. In Parallel Coordinates, for example, a data entry representation covers many pixels. Thus the necessary caches may become very large, as each entry is indexed multiple times. As a result, trade-offs have to be introduced as described in Sections 3.2.2 and 3.2.1.

Investigating the support for the Peek Brush in visualizations for other types of data like graphs and hierarchies is part of future work. Furthermore, combining it with spatial distortion techniques like the Fisheye View [4] may be a promising approach to increase picking accuracy by reducing potential overlaps between visualization primitives.

6 Conclusions

This paper introduced the Peek Brush as a way to select data in a visualization with minimal interaction effort and to display it immediately in all linked views. We discussed performance considerations which have to be made when designing a view that supports the Peek Brush. In this context we discriminated between frequency- and entry-based visualizations and identified different optimization requirements for supporting the active and the passive Peek Brush. As demonstrated, this temporary ad-hoc brush allows for quickly skimming through the data and is able to reveal relations and properties within seconds.

7 Acknowledgments

This work has been supported by the Austrian Funding Agency (FFG) within the scope of the projects AVISOM (No. 818060) and IVAn as well as by our industry partner AVL.

References

[1] AVL List GmbH. http://www.avl.com.

[2] R. A. Becker and W. S. Cleveland. Brushing Scatterplots. *Technometrics*, 29(2):127–142, 1987.

[3] A. Dix and G. Ellis. Starting simple: adding value to static visualisation through simple interaction. In *AVI '98: Proceedings of the working conference on Advanced visual interfaces*, pages 124–134, New York, NY, USA, 1998. ACM.

[4] G. W. Furnas. Generalized fisheye views. *SIGCHI Bulletin*, 17(4):16–23, 1986.

[5] H. Hauser, F. Ledermann, and H. Doleisch. Angular Brushing of Extended Parallel Coordinates. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, pages 127–130, Washington, DC, USA, 2002. IEEE Computer Society.

[6] D. A. Keim. Information Visualization and Visual Data Mining. In *IEEE Transactions on Visualization and Computer Graphics*, volume 8, page 18. IEEE Computer Society, 2002.

[7] R. Kosara, H. Hauser, and D. L. Gresh. An interaction view on information visualization. In *State-of-the-Art Proceedings of EUROGRAPHICS (EG 2003)*, pages 123–137, 2003.

[8] K. Matković, M. Jelović, J. Jurić, Z. Konyha, and D. Gračanin. Interactive Visual Analysis and Exploration of Injection Systems Simulations. In *IEEE Transactions on Visualization and Computer Graphics*, pages 391–398. IEEE Computer Society, 2005.

[9] Tamara Munzner, François Guimbretière, Serdar Tasiran, Li Zhang, and Yunhong Zhou. TreeJuxtaposer: Scalable Tree Comparison using Focus+Context with Guaranteed Visibility. *ACM Transactions on Graphics*, 22(3):453–462, 2003.

[10] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A Multi-Threading Architecture to Support Interactive Visual Exploration. In *IEEE Transactions on Visualization and Computer Graphics*, volume 15, pages 1113–1120, Piscataway, NJ, USA, 2009. IEEE Computer Society.

[11] J.C. Roberts. Waltz - An Exploratory Visualization tool for Volume Data, using Multiform Abstract Displays. In R. F. Erbacher and A. Pang, editors, *Visual Data Exploration and Analysis V - Proceedings of SPIE*, volume 3298, pages 112–122, 1998.

[12] B. Shneiderman. Dynamic Queries for Visual Information Seeking. *The Craft of Information Visualization: Readings and Reflections*, 11(6):70–77, 1994.

[13] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–343, Washington, DC, USA, 1996. IEEE Computer Society.

[14] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *AVI '00: Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 110–119, New York, NY, USA, 2000. ACM.

[15] C. Weaver. Conjunctive Visual Forms. In *IEEE Transactions on Visualization and Computer Graphics*, volume 15, pages 929–936, Piscataway, NJ, USA, 2009. IEEE Computer Society.

[16] J. S. Yi, Y. Kang, J. T. Stasko, and J. A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. In *IEEE Transactions on Visualization and Computer Graphics*, volume 13, pages 1224–1231, Piscataway, NJ, USA, 2007. IEEE Computer Society.