Eurographics Conference on Visualization (EuroVis) 2016
K.-L. Ma, G. Santucci, and J. van Wijk
(Guest Editors)

Volume 35 (2016), Number 3

Designing Multiple Coordinated Visualizations for Tablets

R. Sadana[1] and J. Stasko[1]

[1]Georgia Institute of Technology, Atlanta, US

Abstract

The use of multiple coordinated views (MCV) in data visualization provides analytic power because it allows a person to explore data under a variety of different perspectives. Since this design pattern utilizes multiple visualizations and requires coordinated interactions across the views, a clever use of screen space is vital and many synchronized interface operations must be provided. Bringing this design pattern to tablet computers is challenging due to their small display size and the absence of keyboard and mouse input. In this article, we explain important design considerations for MCV visualization on tablets and describe a prototype MCV visualization system we have built for the iPad. The design is based on the principles of maximizing screen space for data presentation, promoting consistent interactions across visualizations, and minimizing occlusion from a person's hands.

Categories and Subject Descriptors (according to ACM CCS): H.5.0 [Computer Graphics]: Information Interfaces and Presentation—General

1. Introduction

Over the last few years we have witnessed an influx of touch-sensitive devices in the consumer computing landscape. For many people, these devices have become the first and the only device they engage with. This widespread adoption has largely been facilitated by the availability of applications across a variety of domains such as desktop publishing, office suites, art & design, and education. Tablets have emerged as preferred platforms for these productivity applications as they offer an ideal combination of screen size, processing power, mobility, collaborative features and support for peripheral input (stylus, keyboard, trackpads, etc).

This shift in our computing devices has raised critical questions within the visualization domain – what does it mean for visualization tools to exist and be effective in a cursor-less world? How do visualization techniques designed over the past 30 years adapt to interfaces devoid of mouse-input? Conversely, how does touch input affect our interaction with visualization tools? Does it lead to increased efficiency or affect the way we understand data with visualizations?

Recently, researchers have attempted to address these questions by presenting several interactive visualization systems for tablets [BLC12, DFS*13, RK14, SS14]. These carefully designed systems present solutions that are accessible to novice visualization users. However, all of the systems are limited to a single visual representation at a time.

In most desktop information visualization applications, users interact with data across a diverse set of visual representations. Often, the representations are presented simultaneously in configurations known as Multiple Coordinated Views (MCV) [Rob98]. MCVs are used in situations when the data is highly complex, large in scale, and/or heterogeneous in nature. With MCVs, people can leverage the benefits of multiple representations simultaneously. The representations are often linked, and support brushing across views to help identify properties of data more rapidly.

MCVs have been the de-facto standard for visualization systems on desktops. On tablets, however, there has been a conspicuous absence of systems that support MCVs. Since MCVs thrive on larger display sizes, the challenge, in part, has been providing a rich, interactive multi-view experience on the small, constrained displays of tablets.

In this work, we address the question of whether MCV visualizations can be brought successfully to tablets, and if so, how. Our contributions include an analysis of the constraints and opportunities that tablets provide. We then develop a set of design requirements for systems in this space to follow. These requirements have driven our creation of a prototype MCV system described herein. In the system, we introduce novel interaction techniques and modify others, but the main contribution of the work is a consistent, coordinated, and accessible system design and resulting user experience. This article describes the operations that the system provides, our rationale for including the operations, and the design decisions that motivated these choices. Finally, we reflect on the overall design process and highlight a number of new principles we learned, with a goal of informing future work in this area.

2. Related Work

Multiple Coordinated Views emerged as a concept in the late 1990s [Rob98], and a plethora of MCV systems have been presented since [Ahl96,STH02,Wea04]. For a comprehensive review of MCV systems, see [Rob07]. North and Shneiderman introduced a high level taxonomy of MCVs based on focus zones and data representations [NS97]. Further, Baldonado et al. presented eight guidelines for when and how MCVs should be used [WBWK00]. While these guidelines primarily apply to desktop system design, one of our aims was to explore how well the guidelines translate to tablet systems.

Touch-based visualization research in general has only recently gained traction. Lee et al. [LIRC12] employ the terms post-WIMP and post-direct manipulation to describe the rich area of research that needs to fill this gap. Baur et al. [BLC12] were among the first to present a tablet-based multi-touch solution for stacked graphs. Sadana and Stasko [SS14] followed by presenting a gesture suite for scatterplots on tablets. Drucker et al. [DFS*13] compared WIMP and gesture-based interactions for barcharts, and found that gesture-based systems were both better performing and more preferred by study participants. Rzeszotarski and Kittur [RK14] presented Kinetica, which was a scatterplot-style visualization system that leveraged physics-based interactions. While all these systems present compelling solutions, they utilize a single representation technique. To our knowledge, MCV systems have not been addressed on mobile touch-based devices such as tablets.

Several consumer applications relevant to our work have also appeared recently. Most prominent of these is Vizable, a tablet-based visualization system by Tableau [viz15]. In Vizable, people can import their own data and view it using either a barchart or a linechart. Both techniques make heavy use of gestural interactions. The techniques, however, do not appear simultaneously and instead appear as separate modes that users switch between. Our aim is to support multiple techniques in the same view, and we believe that such systems have significantly different design challenges compared to single-visualization views.

Other applications, such as Roambi [roa14], utilize dashboards on tablets. Dashboards are similar to MCVs in how they display multiple visualizations together. However, they are directed more towards specific data types (such as financial or market data) and data reports (daily or weekly summaries). The systems, thus, are much less expressive and omit key features for attribute-level filtering and analysis.

3. Touch Tablet Constraints and Opportunities

Tablet devices present unique design challenges and opportunities that distinguish them from other platforms such as desktop computers. Below we highlight constraints that directly influence the design of visualization systems on tablets. Here, we define tablets to be hand-held, touch-sensitive devices with a 7 to 13-inch screen size. Further, we assume a generic use case — our tablet user is mobile and does not have access to peripherals such as keyboards or stylus.

1. *Screen size and input size*: Compared to desktops, tablets have both a smaller screen and larger sized input — a finger. For visualization applications, this restricts the space for presenting data views. The imprecision of touch [HB11] adds more constraints, e.g., the minimum size of a glyph, further restricting data density.
2. *Grip*: On tablets, while the dominant hand is used to operate the view, the non-dominant hand is primarily restricted to holding the tablet. Employing two-handed gestures constrains the user to place the tablet on a fixed surface, a limitation we seek to avoid.
3. *Hover*: Touch interactions do not provide a hover state in the way mouse pointers do. As a result, features such as cursors that change to depict available actions or tooltips with data labels are not available.

Conversely, tablets provide several opportunities for MCV systems as well. Mobility is an obvious benefit, since it holds the promise of increasing the utility of MCVs to new usage scenarios and locations. Touch-based gestural input also presents an interesting opportunity by being more direct and natural compared to cursor-based input. That is, gestures can be more expressive. Thus, a system leveraging touch in an effective manner could potentially support a wider variety of features with fewer UI elements than possible on desktops. For MCVs, such features could relate to layout controls or cross-vis interactions. Additionally, while the small screen size is a constraint, it is also an opportunity for visualization systems. In general, consumer applications designed for tablets tend to be much simpler than their desktop counterparts. This change is achieved by stripping down the functionalities of these applications to a minimum, resulting in applications that are much more approachable for novice users. For complex applications such as visualization, we see this as an opportunity that could lead to a wider audience.

4. MCV System Components and Objectives

The goal of our work is to understand whether, and if so, how MCV can successfully be brought to tablet computers. We seek to enable novice users to easily and rapidly explore multivariate data on tablets via multiple coordinated views. By examining desktop tools that have similar goals [Ahl96, SGL08, STH02], we were able to identify the components that should make up a tablet MCV system.

At the core of an MCV system is a canvas component containing the visualization views. A second component, specific to systems that provide flexibility in choice of visualizations (also called Level 2 systems [NS00]), is a *view-gallery* widget for selecting the view to add to the canvas and for removing views already added to the canvas. Two other components, although not unique to MCV systems, are frequently found in them. The first is a *tools* menu containing a set of dynamic query filter controls that allow a person to interactively change the data being shown. The second is a *details-on-demand* component that shows a tabular list of data items and their attributes. This set of components is present in many widely used desktop visualization systems, including commercial tools such as Tableau and Spotfire.

This component-level separation is especially crucial for our work due to the constrained screen sizes of tablets. To design an

effective MCV system, one must balance features and screen real-estate between these components. Beyond system components, workspace layout plays a critical role in knowledge discovery and efficiency within a MCV system. It is for this reason that desktop tools such as Photoshop offer a variety of layout options, each optimized for a specific type of task. Since MCV systems show multiple views at a time, they typically provide features to control characteristics of each view such as position, size, visibility, and order in the hierarchy. These features are often present in existing MCV systems for desktops. However, supporting them has consequences on the overall user experience since it is difficult to support them without adding specific UI (or WIMP) elements to the view.

With respect to layout issues, we have identified four goals for MCV systems on tablets:

1. Maximize the size of each visualization on the canvas
2. Minimize occlusion of visualizations by tools or fingers
3. Strive to keep all visualizations in view (prevent scroll, pagination, or tabs)
4. Exclude any need for end-user customization of layouts

Goals 1 and 2 directly emerge from the screen and input size constraints that tablets impose on visualization, as we described earlier. The two goals seek to counter these limitations by ensuring that each view is usable. Goals 3 and 4 help limit the complexity of the interface, furthering our objective of targeting novice and non-expert users.

5. System Design

Our design process was driven by the constraints and objectives discussed above. We sought to create a system that would be complex enough to investigate the potential of MCVs on tablets, but would also be a manageably-scoped prototype to build. Including a large suite of visualization techniques as found in commercial desktop systems such as Tableau or Spotfire would not be practical.

We employed a rapid prototyping process, considering several possibilities for the various features of our system. For the different components of the system, we iterated over multiple design choices and received feedback from informal trial usage by colleagues. Through this exercise, we also made several observations about the nuances of touch interactions with visualizations, limitations of MCVs on tablets, and the complexity of identifying a set of consistent interactions.

All MCV systems provide multiple visualization techniques for presenting different perspectives on data. We selected four techniques to include in our prototype system that offer variation in terms of representation and interaction: scatterplots, barcharts, linecharts, and parallel coordinates plots. The techniques typically represent the same general type of data — combinations of categorical and quantitative attributes. It was necessary for us to include visualizations built for the same type of data since we intend to design coordinated views that support brushing and linking. This precluded the use of other types of data, such as networks, trees, and text in this initial prototype. Additionally, we chose these techniques because at least three should be familiar to many people: scatterplot, barchart, and linechart. Those three share many visual properties, often appear together in dashboards, (e.g. Roambi [roa14]) and support a rich set of interactive operations.

The Parallel Coordinates Plot (PCP) offers an important visual and interactive diversity to the group. PCP is a substantially more dense representation technique than the other three. This has important implications for the sizing and layout of each view. PCP also raises interesting questions regarding the within-vis and cross-vis interactions in our system. We discuss these in detail in the following sections.

5.1. Layout of Visualizations in the Canvas View

With Goal 1 above in mind, we decided to allocate the entire screen to the canvas view and push the secondary views off-screen. We use the left-edge for the gallery, the bottom-edge for DoD view, and the right-edge for the tools view. For each of these, we employ edge swipe-in gestures to bring them into view. Figure 1 shows the interface of our prototype system.

A further consideration was identifying the layout of views within the canvas. For these layouts, several options made sense. The views could appear stacked - either vertically or horizontally. Another option would be to use zoomable user interface (ZUI) techniques, particularly since the touch input supports rapid direct manipulation. With ZUIs, the user could scale the entire canvas to zoom into specific sections of the canvas. Alternatively, each individual chart could be shrunk or expanded on demand, while the canvas size remained consistent. All these alternative layout configurations can be classified into four styles - *juxtaposition*, *superimposition*, *overloading*, and *nesting* [JE12].

With the layout design goals in mind, in our prototype system, we decided to utilize the simplest and most common of these styles — juxtaposition. In this style, views appear side by side and each view is fully visible to the user. We considered two alternative layout styles of juxtaposed views — vertical stacking and grid-based layouts. The benefit of grid-layouts are that they result in compact side-by-side views when the glyph count in views is low, thus fitting more views on screen. However, to be truly effective, they also require end-user controls for adjusting the width and height of the views, which conflicts with Goal 4 above. Thus, we constrained the system to a vertical layout.

When the system starts, the user is presented with a blank canvas. The view-gallery is initially open and contains a vertically scrollable list of available chart types. Tapping on a chart's icon adds it to the canvas. If the canvas already contains charts, they compress vertically and the newest chart is inserted at the bottom. Adding a chart also adds a corresponding badge at the top of the view-gallery (Figure 1). To remove the chart, the user can slide this badge horizontally towards the left. As the corresponding chart exits the view, the other charts expand to fill up the remaining space. Examples of these interactive operations can be viewed in an accompanying video.

After some initial trial use, we chose to limit the number of charts that can be added to the canvas to three in order to ensure that all charts are always visible (Goal 3). Given the space constraints, moving beyond three charts on the canvas makes each chart too

Figure 1: The MCV system interface. The panels are placed beyond the screen and can be brought in with a gesture.

small to be useful. Furthermore, we limit this number to two if one of the charts is a PCP since that visualization requires even more vertical space. This ensures that the view is tall enough to be usable.

5.2. Brushing and Linking

Brushing and linking features are crucial to effective MCV applications. In our system, brushing and linking between charts operates as one would typically expect for visualization applications. Selections made in one view are automatically reflected in other views. For example, selecting individual data items in a scatterplot shades the portion of bars they represent within the barchart.

Filtering data in any view updates all other views, including the dynamic query filters in the tools panel. Similarly, the dynamic query filters affect all views on the canvas and not just a single view. Views are not coordinated on navigation, however, and view-specific changes such as zooming or panning are not percolated to other views [NS97].

5.3. Interactions in Views

A final component of our system involves the design of appropriate interactions for operating each visualization technique. We followed an approach similar to [SS14]: we examined past systems that employ these visualization techniques and identified operations that are central to each. As a first step, we adapted the key design results from that earlier research on scatterplots for tablets.

All four visualization techniques we support include operations that are unique to that view. This is particularly evident for parallel coordinate plots, which differ substantially from the other techniques and include operations such as angular brushing and axis flipping. Many interactive operations, though, were common to all four techniques, e.g. selection, zooming, and filtering. Therefore, it was vital to identify interactions for these operations that would be usable and consistent across the techniques.

5.3.1. Improving Usability

Earlier work on developing a tablet-based scatterplot application [SS14] used the *axis pan* operation to perform value-based selection on a scatterplot. To select data items between two values on an axis, the user touched the axis at the position of one value and dragged towards the other value. Lifting the finger selected all values within this range and a rectangle depicting the selection appeared on the view. The selected range could be moved by dragging the rectangle from the inside.

One drawback of this method was that selecting a precise range was often inaccurate. The inaccuracy was a result of the gesture-recognition delta — the distance moved by the finger before the system correctly identifies the movement as a pan gesture. As a result of this delta, the selected range was always smaller than the

desired range. Our first attempt to handle this issue had two components (Figure 2a). First, we added a label to both ends of the selection rectangle. As the user dragged on an axis, the labels displayed the current range of the selection, providing user with a continuous feedback. Second, to allow editing the range of a selection that the user has already made, we added thumbs on the edges of the rectangle. The user could drag these thumbs to increase or decrease the size of the selection. In this form, the feature resembled marquee-based selection on desktops.

Together, the two modifications made it easier to select a precise range. However, even with a generous touch area, it was often cumbersome to place a finger directly on a thumb due to its small size. To eliminate the need to touch inside a small zone, we further iterated on the design of this interaction. One assumption we made here was that while the user had an active range selection, any action on the visualization mapped only to the selection and not to any other feature. In other words, while in the selection state, the user can only either manipulate the selection or dismiss it. Making this assumption allowed us to extend the interaction area from around the handles to the entire view. We identified two techniques to edit the selection in this new configuration:

1. Pinch to edit (Figure 2b) - Since the rectangle has two manipulatable edges, we map the movement of each finger in a pinch operation to the translation of an edge. The two fingers independently control an edge using either their x or y movement, depending on the axis used for selection. The mapping from a finger to an edge is one-to-one, i.e. the control-display ratio is one. This results in very fine and precise control. Moreover, the direct mapping allows the gesture to be position agnostic. Within the bounds of the visualization, the user is free to perform the gesture anywhere away from the rectangle.
2. Exterior pan to edit (Figure 2c) - To edit the position of an edge, we use the entire area adjacent to the edge outside the rectangle as a trackpad. Similar to *pinch to edit*, the x or the y movement of the finger is used to translate the edge. Since the effective size of the target area becomes very large compared to the handles, there is a considerable performance improvement.

We implemented both these options and found both to be useful in our own initial trial use. We thus support them both in our system. We also wanted to provide visual affordances for these updated operations in the view. We considered retaining the handle as it depicts edit-ability, which is a learnt behavior from desktops. However, we believed that the handle would naturally guide the user to touch and drag it. To improve usability, it was necessary to replace the handle with an artifact that represented editability, but without affording draggability. We thus replaced the handles with modified arrows presented in Figure 2d.

We made several other modifications and additions to the operations described in [SS14] including:

- *Object Count* (Figure 2e): We display a label at the top right of the screen that depicts the total number of data points in the view. When the data is filtered, the label updates accordingly. Similarly, when data glyphs are selected, the label reflects the size of selected data, and changes color to red.
- *Filter-menu feedback* (Figure 2e): Filtering data inside the tools menu results in a “Filters On" label at the top right of the main

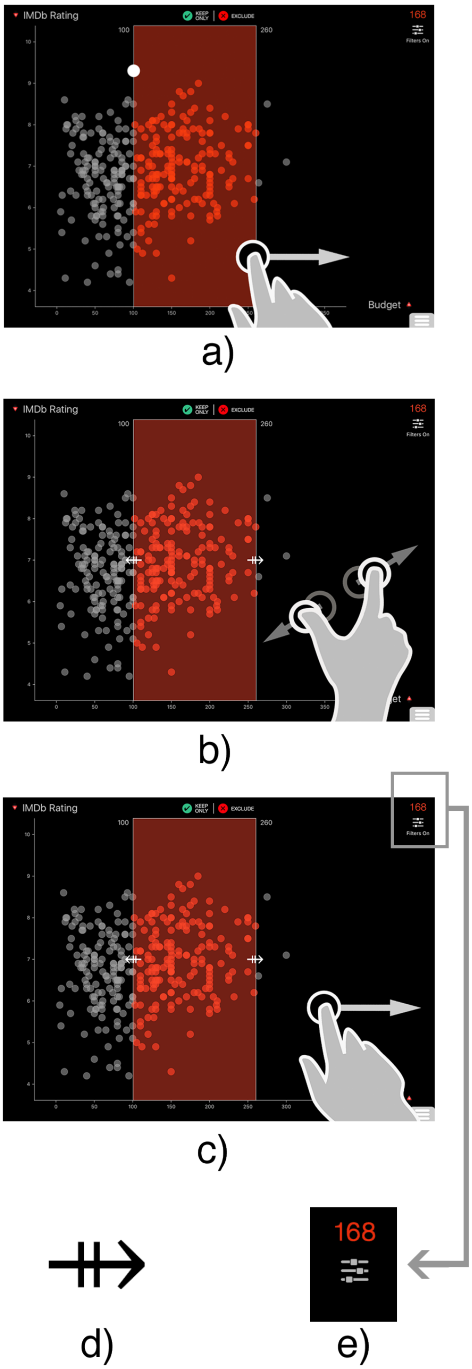


Figure 2: The evolution of the edit-selection feature. a) The selection rectangle with circular handles that can be dragged to edit the edges. b) Each finger in a pinch gesture control the movement of an edge of the selection rectangle. Here, movement of the fingers in x controls the edges. c) The entire space adjacent to an edge outside the rectangle controls the edge. Here, the finger translation increases the selection size. d) The arrow that replaces the handles on selection rectangles. e) The count label depicts the number of selected items. “Filters On" label signifies that at least one dynamic query filter is active in the tools menu.

Operation	Interaction
Select	Tap to select one glyph Lasso to select multiple glyphs Swipe on axis to select glyphs between a range of values
Zoom	Pinch on view for fixed-aspect ratio zoom Pinch on an axis for zooming in one direction
Filter	Select glyphs, then tap “keep-only” or “remove” button
View data	Select glyphs, then drag table from below

Table 1: Summary of interactions for operations common to the four visualization techniques. These interactions work in a consistent manner across all techniques.

view when the menu is closed. This provides a consistent feedback regarding the filtered state of the data, similar to the “Keep only" and “Exclude" badges.

5.3.2. Achieving Consistency

Consistency is critical as it feeds directly into the usability and learnability of the system, with inconsistent interactions adversely affecting user performance [WW11, WBWK00]. To our knowledge, our work is the first that attempts to identify a set of consistent touch interactions for multiple visualization types.

Table 1 presents the operations common to all four visualization techniques and the interactions we employed to implement them. As described earlier, for several operations we adapted relevant options developed for scatterplots in [SS14].

We faced a number of challenges in achieving cross-visualization compatibility. For instance, many gestures useful for one technique did not translate well to other techniques. A specific example of this is sorting on barcharts. A simple method for activating sort involves placing a finger on an axis and swiping in the direction of desired sort (ascending or descending). This axis-swipe gesture has been used with good results in previous work on touch-based barcharts [DFS*13]. For scatterplots, however, this gesture was used for a different operation — axis-based selection [SS14]. In fact, we adapted this interaction on linecharts and parallel coordinates as well.

To ensure barcharts are consistent with other charts, we do not use the axis-swipe gesture on barcharts for sorting. We instead use axis-swipe to perform the same operation as in other charts, selecting items in a range. This is an example of a locally suboptimal choice, because sorting in barcharts is a more significant operation than selecting bars within a range of values.

To perform sorting, we considered several options such as dragging the tallest (or shortest) bar to an extreme, employing a complex gesture such as two-finger double-tap or three-finger tap, and using sort buttons placed adjacent to the axis labels. Ultimately, we employed long-press + swipe — the user places a finger on an axis and holds for a short period of time before swiping in the direction of sort. All the options we considered for sorting were unique in that they are not used elsewhere in our system. Employing a unique gesture has drawbacks for learnability as users’ mental models of the gesture are not reinforced by use in other situations. We selected the long-press + swipe gesture since it builds on familiarity from other applications and does not require additional UI elements on the screen.

Another example of consistent interaction is the ‘Tap to select’ interaction. On barcharts, it is natural to use the tap gesture to select an individual bar. However, tap-to-select was missing in the original scatterplot implementation [SS14]. Tapping on a scatterplot is complicated since it results in ambiguous selections when tapping in a dense area. However, for consistency we extended tap-to-select to scatterplots as well as linecharts and parallel coordinates. We address the ambiguity by always selecting the top-most glyph at the position of tap.

5.4. Implementation

We implemented our prototype system for Apple’s iOS operating system using Objective-C and Apple’s Cocoa Touch framework. The application loads data from a csv file that contains the attribute names in the first row. Currently, we do not enforce a constraint on the size of the data, but we found the interface to be most useful when the number of data items is less than 1000. The accompanying video illustrates the system in use and includes a sample usage scenario.

5.5. Scalability

In the earlier section on constraints of tablets for visualization, we described tablets as devices with a 7 to 13-inch screen. In practice, this spread of screen sizes is fairly broad and devices lying at the two ends vary considerably in ergonomics and usability. To focus our design process, we primarily iterated on a tablet with a screen size in the middle of the range (Apple iPad 9.7 screen). This is the most-selling category of tablet devices. Moreover, targeting the mean size gave us maximum opportunity to scale our design decisions to both smaller and larger sized tablets. We have since ported our system to the 7" Apple iPad Mini and 12.9" Apple iPad Pro.

1. *7-inch tablet*: The screen resolution of the 7" iPad Mini is identical to the 9.7" iPad Air (2048-by-1536). The interface, thus, appears the same on the two devices. Nearly all of the interface elements have adequate sizes for comfortable interaction with fingers. However, due to the smaller screen size, certain elements require scaling up for effective operation, e.g. the handles of slider bars in the tools view. For an ideal experience on iPad Mini, we would modify the limit on the maximum number of charts that can be added from three to two. Simultaneously, we would allow the parallel coordinates chart to only appear by itself.
2. *13-inch tablet*: We also ported our system to the recently introduced Apple iPad Pro. Unlike the 7" tablet that has the same resolution as the 9.7" tablet, the 13" tablet has a proportionally higher screen resolution. As a result, the widgets and glyphs appear in the same physical dimensions as the iPad Air. This results in each chart spreading over a larger screen size, affording more space for the gestures to be performed. However, even

with the larger screen size, we believe that the view should be limited to a maximum of three charts.

One closely related category of devices we do not address in our work is *phablets*. These devices combine the size format of smartphones and tablets and have between 5" and 7" screen size. We believe that the design constraints for these screen sizes are substantially different from the ones we present above. Although many of our design decisions would port well to these screen sizes, it is recommended to rethink the interactions in the context of the smaller size, and the different grips and postures people employ when using these devices.

6. Discussion

In this paper, we present an initial approach to bring MCV visualization systems to tablets. Although the system is only a prototype, it makes key strides to move beyond simply porting a desktop visualization application to a tablet. Specifically, the system employs more familiar touch gestures for all interface operations. In the course of creating the prototype, we made several observations about the design of tablet-based MCV systems. We describe these below with the goal of informing future research and implementation of such systems.

6.1. Occlusion

The drawbacks in performance caused by occlusion in direct-touch input interfaces have been well established. Moreover, these drawbacks are only compounded as the number of fingers increase and the display size decreases. In a series of articles, Vogel and colleagues [VB07, VB10] address many of these concerns surrounding occlusion, and present solutions to mitigate these concerns.

As a result of occlusion we perceived in our early prototypes, we tested several solutions and made various changes to our system. For example, even though the expected location of labels in a parallel coordinates plot is at the top, we placed them at the bottom since any interaction with them would otherwise occlude the entire view. For some other occlusion-related issues we encountered, we examined the utility of the solutions previously presented in HCI literature [VB07, WBP*11]. However, we found their applicability for our system to be limited. This led us to question whether occlusion in visualization interfaces is different from occlusion in regular, non-visualization user interfaces. We speculate that the answer to this question is yes. We explain this below using the ideas of the instrumental-interaction model [BL00].

6.1.1. Low degree of indirection

In the instrumental interaction model, Beaudouin-Lafon introduces the concept of domain objects and instruments [BL00]. Users act on an instrument, which in turn affects attributes of the object-of-interest. One property of these instruments is the *degree of indirection*, which encodes the distance between the logical part of an instrument and the object it operates on. In the context of touch-screen input, the degree represents the distance between the position of a finger touching the screen and the response generated by this interaction on the interface.

Most gestures that we employ in our system have a low degree of indirection. We believe that this is largely due to the underlying domain and not just a result of the design decisions we took. The same observation can made for the interactions used in TouchWave [BLC12], or the gesture-based interface by Drucker et al [DFS*13]. By utilizing gestures for the most commonly-used operations, people can use these systems to complete a set of standard tasks without having to interact with any widgets. We think that such a tight coupling between the instrument and domain-object is very specific to data visualization. Information visualization operations lend themselves appropriately for direct interactions with the glyphs.

Unfortunately, this also means that for a majority of gestures, the instrument (a finger) overlaps with the domain-object. As a result, the probability that fingers occlude the object is very high. Since users' actions on the instrument are highly dependent on the underlying positional properties of the domain-object being manipulated, if the object is occluded, manipulating an instrument becomes very inefficient.

Many solutions for countering these types of issues suggest translating the obstructed views to locations that are not hidden [VB07, WBP*11]. There are two downsides to these solutions. First, the position of glyphs in a visualization are based on attributes of the underlying data as well as the properties of the visualization. Second, the glyphs typically look the same as their counterparts. Translating a few glyphs to a different position would sever the relationship between the glyphs and the axes, and would lead to loss of context.

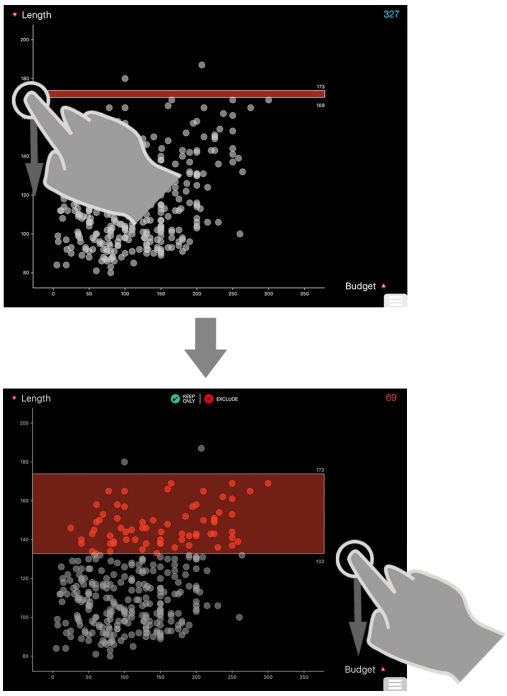
As we encountered these issues in our own work, we developed strategies for modifying interactions to counter occlusion. We present these methods below.

6.1.2. Occlusion-aware continuous gestures

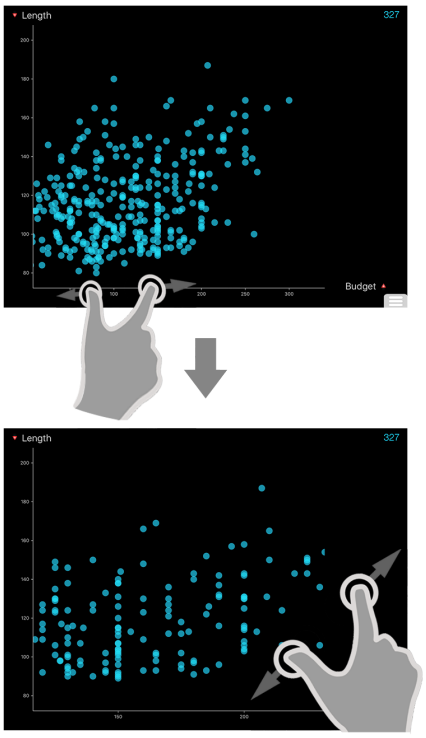
Gestures such as dragging, pinching and rotating are continuous gestures since they require a stream of event data and the response occurs while the user is performing the action. These gestures are a part of the common vocabulary of gestures [WMW09], and we use them extensively in our system. With these gestures, fingers stay on the view for longer than discrete gestures, increasing the possibility of occluding views. To minimize this occlusion, we use two strategies.

1. *Actuator-continuous gestures*: In this configuration, we define a specific location for the user to initiate a continuous gesture, but revoke this restriction as soon as the gesture is activated. The user is then free to move her fingers away while maintaining control of the operation. Examples of this from our system include:

    - Pan gesture (Figure 3a): Dragging on an axis requires the user to place her finger at a precise 2d location. However, the dragging operation itself is one dimensional. An actuated dragging of, say, the vertical y-axis allows the user to drag her finger away horizontally after the gesture is registered. The vertical position of the finger continues to control the selection. Moving the finger away horizontally, however,



(a) User begins a drag gesture on the y-axis to begin selection and subsequently moves away from the axis while continuing to control the selection.



(b) User begins a two finger pinch on the x-axis and then moves away. Notice that user's fingers do not need to stay horizontal.

Figure 3: Actuator continuous gestures.

means that the finger is not occluding the axis itself. Actuated dragging is similarly useful for moving the handles of a slider bar or filtering a parallel coordinates axis.

- Pinch gesture (Figure 3b): We also use an actuated two-finger pinch gesture. To zoom an axis, the user begins the pinch operation directly on the axis and then simply drags her fingers away. The distance between the fingers continues to map to the amount of scaling. In fact, the user does not even need to keep her fingers in a strictly horizontal or vertical configuration. Consequently, the axis being scaled is not occluded by the fingers.

2. *Location-independent gestures*: For specific states, we also relax the requirement to initiate gestures at a precise location. Instead, gestures can be initiated anywhere in a larger, unbounded space. Increasing this space increases user efficiency [Mac92]. Both the *pinch to edit* and *exterior pan to edit* operations we described earlier (Section 5.3.1) are examples of location-independent gestures.

Handling occlusion on data visualizations is one of the biggest challenge that designers face when designing for touch. Unfortunately, not enough attention has been paid to this phenomenon. This is primarily because most of the research on visualization for multi-touch has centered around large touch tables, where occlusion problems are less ardent. We feel it is important to revisit the topic within the context of tablets and small displays.

6.2. Gesture set

All the interactions we employ in the system make use of a standard set of gestures, such as tap, double-tap, pan, and pinch. Given the complexity of visualization interfaces on desktops, it is appealing to argue that an ideal solution for touch-devices would necessitate looking beyond the standard set of gestures to more novel, intricate gestures. However, to a large extent, we disagree. While it may be possible that future research reveals more effective gestures for specific operations, we believe that the strength of our system is precisely in being able to leverage the everyday gesture vocabulary for the entire breadth of operations.

Unlike widgets, gestural operations have a drawback that it is difficult to provide appropriate visual affordances for them. Novel gestures in particular suffer drastically from discoverability and learnability issues. Everyday gestures have the advantage of leveraging users' learned behaviors from other applications. Thus, they support a new user better and minimize the need for extensive training. Since, in comparison with other domains, visualization applications have only recently started to appear on touch devices, it is in users' best interest that we greet them with familiar and predictable features so as to minimize errors in use. As the tools mature, it would be natural to explore the edges of the interaction space for more effective techniques.

6.3. Layout

It is common for visualization systems on desktops to have feature-rich interfaces — often providing a sense of "drowning in functionality" [LIRC12]. Thus, to adapt these systems to tablets, careful

planning is required to select the features that are supported in the new configuration. This is challenging because with every added element, the interface becomes more complex. Complexity in turn has a negative impact on the approachability of the system.

Since we intended to address entry-level, novice users, making the system approachable was critical. We tried to achieve this by limiting the functionality of the system and introducing specific constraints. This is best exemplified in the design of the system's layout. We approached the design not with the intention to devise the most unique or novel layout, but rather the most functional configuration. This was possible by using specific constraints on the number of views allowed, fixing the size of each view, and defining the position of new views.

While introducing these constraints was crucial for our goals, we also believe that layout offers tremendous potential for further exploration, as several choices make sense. For instance, instead of vertical layouts, a system might use adaptive layout where views readjust to stack vertically or horizontally, depending on space required by each view. A barchart with only three bars can compress and release space to a more constrained chart such as a parallel coordinates plot. At a higher level, given the constrained space of tablet devices, zoomable user interfaces also make for worthy candidates, as we discussed earlier. In this case, the workspace can be a large grid of visualization views, and the user can zoom into specific portions of the view depending on the visualizations of interest.

7. Design Reflections

We summarize the key design principles for MCV systems on tablets that emerged from our design process:

1. *Provide views-on-demand*: Space constraints on tablets amplify the need for maximizing the size of visualizations. To achieve this, the visualizations should extend to the entire size of the screen and the secondary components can be placed off-screen (Figure 1). Those components can be called into view only when needed, either through gestures or by using drag-handles.
2. *Employ single-finger operations*: There are many tradeoffs between one and multi-finger gestures. Multi-finger gestures increase the expressiveness of a system. Conversely, single-finger gestures are better in terms of occlusion. One-finger gestures have the additional advantage of being accessible for stylus and mouse input, if available. Stylus-based input has been a feature across devices such as Microsoft Surface, Samsung Note and Apple iPad Pro.
   Due to this capability, we recommend maximizing the use of one-finger gestures in place of multi-finger ones. An important issue to note is that in the absence of multi-finger gestures, the complexity of one-finger gestures, either in terms of number of taps, steps, or time, may need to increase. An example of this is the tap-and-pan gesture [SS14].
3. *Prevent occlusion*: We observed that the effects of occlusion on direct-touch interfaces are further compounded in the case of MCV systems. Designers must pay close attention to effects of occlusion when constructing layouts and designing interactions for touch-based visualizations systems. Techniques we employ to manage occlusion are optimizing a user's action on screen (actuator-continuous and location-independent gestures) and minimizing the footprint of the input (single-finger over multi-finger gestures).
4. *Promote standard gestures and consistency*: As more visualization applications come to touch devices, it is beneficial to leverage users' existing familiarity with gestures such as tap, double-tap, pan and pinch. Using such gestures improves discoverability and learnability as well as reduce the need for training. Similarly, as much as possible, systems must ensure that these gestures always map to the same action and result in consistent feedback across techniques.

8. Conclusion

In this design-focused research, we explored the key challenges in creating a Multiple Coordinated View data visualization system for multi-touch tablet computers. The work provides multiple contributions. We addressed technique-specific interaction issues as well as inter-technique layout and coordination issues. We identified and implemented a consistent set of interactive operations for these techniques and produced a working prototype that embodies our design principles. More importantly, we identified a set of key challenges that designers will face when moving information visualization to tablets. Ultimately, we presented reflections to help guide the design of similar systems in the future.

Much further work remains. For example, we can explore techniques for altering the arrangement of views. Although omitting such techniques in the current system was a conscious design decision we made to promote simplicity, it is also clearly limiting. Similarly, we implemented only four visualization techniques in the system. While the suite of gesture actions we present works consistently across the techniques, the addition of new visualizations likely will present further challenges. Finally, user studies to examine people's abilities to learn and effectively operate the system and its interactions are essential. We employed informal formative feedback from colleagues as we iterated through design choices, but more summative evaluation now can be performed with a full prototype implemented.

Our work takes a crucial first step in providing a system for enhanced data interaction using multiple coordinated views and rich brushing & linking. We have presented an approach that leverages people's everyday knowledge of touch-devices and provides a simple and approachable system for visualizing rich and complex information. We hope that in the future, our guidelines and reflections will help generate new and more diverse approaches for exploring data on mobile and tablet devices.

9. Acknowledgements

We thank Chad Stolper, Alex Godwin, Alex Endert, and the reviewers for their suggestions on improving the article. This work was supported by a Google Faculty Research Award and by the National Science Foundation via award IIS-1320537.

References

[Ahl96] AHLBERG C.: Spotfire: An Information Exploration Environment. *SIGMOD Rec. 25*, 4 (Dec. 1996), 25–29. URL: http://doi.acm.org/10.1145/245882.245893, doi:10.1145/245882.245893. 2

[BL00] BEAUDOUIN-LAFON M.: Instrumental Interaction: An Interaction Model for Designing post-WIMP User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2000), CHI '00, ACM, pp. 446–453. URL: http://doi.acm.org/10.1145/332040.332473, doi:10.1145/332040.332473. 7

[BLC12] BAUR D., LEE B., CARPENDALE S.: TouchWave: Kinetic Multi-touch Manipulation for Hierarchical Stacked Graphs. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces* (2012), ITS '12, ACM, pp. 255–264. URL: http://doi.acm.org/10.1145/2396636.2396675, doi:10.1145/2396636.2396675. 1, 2, 7

[DFS*13] DRUCKER S. M., FISHER D., SADANA R., HERRON J., SCHRAEFEL M.: TouchViz: A Case Study Comparing Two Interfaces for Data Analytics on Tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), CHI '13, ACM, pp. 2301–2310. URL: http://doi.acm.org/10.1145/2470654.2481318, doi:10.1145/2470654.2481318. 1, 2, 6, 7

[HB11] HOLZ C., BAUDISCH P.: Understanding Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), CHI '11, ACM, pp. 2501–2510. URL: http://doi.acm.org/10.1145/1978942.1979308, doi:10.1145/1978942.1979308. 2

[JE12] JAVED W., ELMQVIST N.: Exploring the design space of composite visualization. In *2012 IEEE Pacific Visualization Symposium, (PacificVis)* (Feb. 2012), pp. 1–8. doi:10.1109/PacificVis.2012.6183556. 3

[LIRC12] LEE B., ISENBERG P., RICHE N., CARPENDALE S.: Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Dec. 2012), 2689–2698. doi:10.1109/TVCG.2012.204. 2, 8

[Mac92] MACKENZIE I. S.: Fitts' Law As a Research and Design Tool in Human-computer Interaction. *Hum.-Comput. Interact. 7*, 1 (Mar. 1992), 91–139. 01051. URL: http://dx.doi.org/10.1207/s15327051hci0701_3, doi:10.1207/s15327051hci0701_3. 8

[NS97] NORTH C., SHNEIDERMAN B.: *A Taxonomy of Multiple Window Coordinations*. Tech. Rep. CS-TR-3854, University of Maryland, Department of Computer Science, 1997. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.2424. 2, 4

[NS00] NORTH C., SHNEIDERMAN B.: Snap-together visualization: can users construct and operate coordinated visualizations? *International Journal of Human-Computer Studies 53*, 5 (Nov. 2000), 715–739. URL: http://www.sciencedirect.com/science/article/pii/S107158190090418X, doi:10.1006/ijhc.2000.0418. 2

[RK14] RZESZOTARSKI J. M., KITTUR A.: Kinetica: Naturalistic Multi-touch Data Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), CHI '14, ACM, pp. 897–906. URL: http://doi.acm.org/10.1145/2556288.2557231, doi:10.1145/2556288.2557231. 1, 2

[roa14] Roambi, 2014. URL: http://www.roambi.com. 2, 3

[Rob98] ROBERTS J. C.: On encouraging multiple views for visualization. In *1998 IEEE Conference on Information Visualization, 1998. Proceedings* (July 1998), pp. 8–14. doi:10.1109/IV.1998.694193. 1, 2

[Rob07] ROBERTS J. C.: State of the Art: Coordinated Multiple Views in Exploratory Visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV '07* (July 2007), pp. 61–71. doi:10.1109/CMV.2007.20. 2

[SGL08] STASKO J., GÖRG C., LIU Z.: Jigsaw: Supporting Investigative Analysis through Interactive Visualization. *Information Visualization 7*, 2 (June 2008), 118–132. URL: http://ivi.sagepub.com/content/7/2/118, doi:10.1057/palgrave.ivs.9500180. 2

[SS14] SADANA R., STASKO J.: Designing and Implementing an Interactive Scatterplot Visualization for a Tablet Computer. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces* (2014), AVI '14, ACM, pp. 265–272. URL: http://doi.acm.org/10.1145/2598153.2598163, doi:10.1145/2598153.2598163. 1, 2, 4, 5, 6, 9

[STH02] STOLTE C., TANG D., HANRAHAN P.: Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (Jan. 2002), 52–65. doi:10.1109/2945.981851. 2

[VB07] VOGEL D., BAUDISCH P.: Shift: A Technique for Operating Pen-based Interfaces Using Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2007), CHI '07, ACM, pp. 657–666. URL: http://doi.acm.org/10.1145/1240624.1240727, doi:10.1145/1240624.1240727. 7

[VB10] VOGEL D., BALAKRISHNAN R.: Occlusion-aware Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2010), CHI '10, ACM, pp. 263–272. URL: http://doi.acm.org/10.1145/1753326.1753365, doi:10.1145/1753326.1753365. 7

[viz15] Vizable by Tableau, 2015. URL: vizable.tableau.com. 2

[WBP*11] WIGDOR D., BENKO H., PELLA J., LOMBARDO J., WILLIAMS S.: Rock & Rails: Extending Multi-touch Interactions with Shape Gestures to Enable Precise Spatial Manipulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), CHI '11, ACM, pp. 1581–1590. 00042. URL: http://doi.acm.org/10.1145/1978942.1979173, doi:10.1145/1978942.1979173. 7

[WBWK00] WANG BALDONADO M. Q., WOODRUFF A., KUCHINSKY A.: Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (2000), AVI '00, ACM, pp. 110–119. URL: http://doi.acm.org/10.1145/345513.345271, doi:10.1145/345513.345271. 2, 6

[Wea04] WEAVER C.: Building Highly-Coordinated Visualizations in Improvise. In *IEEE Symposium on Information Visualization, 2004. INFOVIS 2004* (2004), pp. 159–166. doi:10.1109/INFVIS.2004.12. 2

[WMW09] WOBBROCK J. O., MORRIS M. R., WILSON A. D.: User-defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2009), CHI '09, ACM, pp. 1083–1092. URL: http://doi.acm.org/10.1145/1518701.1518866, doi:10.1145/1518701.1518866. 7

[WW11] WIGDOR D., WIXON D.: *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*, 1st ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011. 6