DATA VISUALIZATION ON TABLET DEVICES

A Dissertation
Presented to
The Academic Faculty

by

Ramik Sadana

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
May 2017

DATA VISUALIZATION ON TABLET DEVICES

Approved By:

Dr. John Stasko, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. James D. Foley
School of Interactive Computing
Georgia Institute of Technology

Dr. Rahul Basole
School of Interactive Computing
Georgia Institute of Technology

Dr. Alex Endert
School of Interactive Computing
Georgia Institute of Technology

Dr. Steven Drucker
Principal Researcher
Microsoft Research, Redmond

Date Approved: April 4th, 2017

To my family,

and Vasudhara, the love of my life!

ACKNOWLEDGEMENTS

My most heartfelt gratitude goes out to my advisor John Stasko. The past five years has been a journey of immense growth, both intellectually and personally. Through his continued support and guidance, I have felt truly valued. He has challenged me to explore things that thoroughly excite me and given me the freedom to excel at them in a manner that feels most fulfilling. For that, I am forever grateful.

Thank you also to my dissertation committee, Jim Foley, Steven Drucker, Rahul Basole and Alex Endert, for their insightful and critical feedback on my dissertation. I especially want to thank Steven Drucker for his constant words of advice and encouragement throughout my PhD.

Thank you to my good friends at Georgia Tech — my lab mates in the VIS lab, friends who I took courses with, and faculty I took courses under. I look back fondly at the time spent learning with and from you. I feel that I am a better person for it.

A sincere thanks to my family — mom, dad, brother, and sister-in-law, for their understanding and the *'He's fine, he'll figure it out!'* approach. Their love, encouragement, and support has been extremely inspiring.

Finally, all my love and gratefulness to my partner, Vasudhara. She has had the biggest influence on the work that I have done and the person that I have become. I am most excited about the new journey that begins now!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Multitouch input is now ubiquitous and the popularity of devices using it has grown tremendously in recent years. The ability to directly touch and manipulate data on the screen without using any intermediary devices is very appealing to people. This has resulted in a new generation of applications that are developed entirely for touch screens. However, one area with only a limited exposure to touch-based input is information visualization. This is in part due to the constraints of designing for touch: the absence of keyboard and mouse, typically small screen size of handheld devices, and the dependence of visualization applications on widgets such as buttons, sliders, menus, and dialog boxes.

Touch input raises interesting questions for information visualization: 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, does touch input lead to increased efficiency or affect the way we understand data with visualizations? Through my work, I address these questions and others.

Specifically, I understand, design, and develop capabilities for analyzing data within an information visualization system for tablet devices. I explore appropriate and effective visualization schemes for exposing crucial insights from data on tablets. I also develop new interaction techniques to support the rich set of operations, including advanced capabilities such as multiselection and layout, that people expect from information visualization systems. Through three evaluations, I capture the strengths of the system's utility and usability, both in short-term and long-term usage, and in comparison with a publicly available system.

CHAPTER I

INTRODUCTION

With the advances in human-computer interaction in the Post-PC era, the direct input modality of multitouch has been brought to consumers to deliver amazing new experiences. Multitouch is now ubiquitous and the popularity of devices using it for input has grown tremendously in recent years. Devices such as smartphones and tablets constitute a significant portion of all computer sales today[1]. The widespread proliferation of these devices raises the question: What made this adoption feasible? Clearly, mobility has a significant role to play. The handheld devices combine substantial processing power and considerable battery life in a form factor that is light and convenient. Moreover, unlike desktops, these devices are untethered from the constraints of fixed wires and connections.

But perhaps a big reason for a successful widespread adoption is the nature of the input — multitouch. The ability to directly touch and manipulate data on the screen without using any intermediary devices has a very strong appeal to users [12]. In particular, novices benefit most from the directness of touch screens. Touch input is easy to learn, and for certain tasks, far more effective than cursor-based input [108]. This has resulted in a new generation of applications that are developed entirely for touch screens. At the same time, a significant percentage of new users are embracing these touch-based devices as their first and only device.

A prominent effort from the past decade has been to bring the wide range of applications that people use on desktops to touch devices. These include websites,

[1]Gartner: "By 2018, More Than 50 Percent of Users Will Use a Tablet or Smartphone First for All Online Activities", http://www.gartner.com/newsroom/id/2939217

desktop publishing applications, word-processing applications, programming environments, and other similar domains. As a result, applications that gained prominence on desktops, such as Microsoft Office, Autocad, and the suite of Adobe applications, are now available for the touch platforms.

Early work in this space centered around replicating the functionality available on desktops. However, the resulting experiences were often restrictive and did not fully leverage the set of gestural interactions that touch-based interfaces provide. Subsequently, web designers and developers crafted the responsive web design paradigm that allowed websites to automatically reconfigure based on the type of device. Since the results were effective, desktop-class applications began using similar techniques, though with limited success. However, as the adoption and capabilities of touch devices progressed further, more nuanced and tailored programming styles were developed for them. Today, a vast majority of applications are developed specifically for touch devices, often before desktop applications [135] and, in several cases, only for touch devices.

One area that has only had a limited exposure to touch-based input is information visualization. This is in stark contrast to the range of systems available on desktops, including Tableau[2], Spotfire[3], Qlik[4], PowerBI[5] and Microsoft Excel. Some initial explorations do exist, including complementary applications for Tableau and Spotfire on tablets. However, these systems have largely been second-screen experiences for the desktop versions, focusing more on visualization consumption than visualization creation or analysis.

The limited development of information visualization applications for touchscreens results from the challenge of designing and implementing the interface. The size of

[2]Tableau, www.tableau.com

[3]Spotfire, spotfire.tibco.com

[4]Qlik, www.qlik.com

[5]PowerBI, powerbi.microsoft.com

handheld touch-devices is a limitation for data visualization. Perhaps more importantly, information visualization applications generally have many small visual objects to select and manipulate, and they contain many interactive widgets such as buttons, sliders, menus, and dialog boxes. Desktop information visualization applications typically make extensive use of a WIMP (window, icon, menu, pointer) interface. Translating visualization interfaces and interactive operations to a finger-directed multi-touch interface without a keyboard and mouse is a challenging problem [72].

Touch input raises interesting questions for information visualization: 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?

"Going beyond mouse and keyboard" has been identified as a a topic worthy of further research within the field of information visualization [72]. Isenberg et al. [60] discuss the visualization space specifically for touch-based surfaces, highlighting three main types of challenges: a) the technical challenges of understanding, using, and effectively combining novel displays and their interaction capabilities; b) the challenges of understanding how to best design data representations and interactions with them; and c) the social challenges associated with the use of visualization applications in novel contexts such as museums, meeting rooms, or other non-work settings.

Through this work, I address the first two challenges. Specifically, the goal of my work is to understand, design, and develop capabilities for analyzing data with visualizations on touch-based devices.

1.1 Thesis statement and contributions

My research is captured by the following thesis statement:

Through an appreciation of the opportunities and constraints of touch input, it is feasible to design an information visualization system for tablets that replicates the analytical capabilities of desktop-class visualizations, and is simple and effective for users.

My work makes three main contributions.

- I identify appropriate and effective visualization schemes for exposing crucial insights from data on a touch-device.
- I explore interactive interfaces that are effective for these visualization schemes, and coordinate their design across techniques.
- I identify and develop new interactions to support the rich set of operations on these techniques that people expect from information visualization systems.

1.2 Opportunities and Challenges with Touch

Understanding the behavior of touch-input is a complex task since touch-input is a nuanced technology. Research on multi-touch interfaces dates back to the early 1980s at IBM, Bell Labs, the University of Toronto, and other research centers [100]. These efforts produced a variety of devices that demonstrated the potential for input technologies that rely on hand and finger gestures. The research also produced several variants of the touch technology. Touch screens were introduced that differed on the number of simultaneous touch-points detected, types of input detected (touch vs. pen), precision of location returned, and hand postures detected (fingers vs. whole hand), among others.

With the commercial introduction of multitouch, the technology has matured and stabilized. Apple introduced the first mass-market consumer multitouch device in the form of iPhone. For a majority of the devices people use today, the technology is simply that — a multitouch screen that can detect more than one (often up to 10) touch

points that are indistinguishable from each other. Recently, other capabilities such as pressure-sensitive pen and touch input have also been integrated into touchscreens. However, these capabilities are currently limited to a few device models.

Considering the multitouch screens that are standard today, one faces several challenges when designing for them.

1. *Screen size and input size*: Compared to desktops, touch devices 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 [56] adds more constraints, e.g., the minimum size of a glyph, further restricting data density.

2. *Lack of Hover*: Touch interactions do not provide a hover state in the way mouse pointers do. As a result, certain features are not available, such as cursors that change to depict available actions or tooltips with data labels.

3. *Occlusion*: Occlusion of the screen caused by fingers, hands, and arms when interacting with a touch screen is a nuisance [120]. The drawbacks amplify for visualizations since the view contains glyphs that are small, and can be easily occluded.

4. *Lack of keyboard input*: In the absence of a physical keyboard, acquiring user input becomes difficult. The on-screen keyboard is an option, but it causes significant occlusion of other views. Absence of a keyboard affects performance of widgets that might need direct specification of values, such as sliders.

5. *Tactile user feedback*: The absence of tactile feedback is a big differentiator between multitouch interactions and those with mouse and keyboard. Tactile feedback can improve performance and decrease error rates [55]. Technological alternatives include using vibration and acoustic feedback, or providing visual

response at touch location. But these are ineffective substitutes that cause equal amounts of distractions.

6. *Grip*: On mobile touch-devices such as smartphones and tablets, while the dominant hand is used to operate the view, the non-dominant hand is primarily restricted to holding the tablet. Thus, while it is possible to use two-handed gestures, employing them constrains the user to place the device on a fixed surface.

Conversely, touch devices provide several opportunities for as well. Mobility is an obvious benefit, since it holds the promise of increasing the utility of these devices to new usage scenarios and locations. Touch-based gestural input also presents an interesting opportunity in 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. Additionally, while the screen size of touch devices 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, I see this as an opportunity that could lead to a wider audience.

1.3 Variations of Multitouch Devices

Consumer touch devices can broadly be categorized into three types - smartphones and wearables, tablets, and large-touch displays. The chief basis for this differentiation is the size of the displays. Smartphones and wearables typically have a less than 7" screen. Tablets extend between 7" and 15" screen sizes, with the 10" screen being

the typical display. All devices with a screen greater than 15" are categorized under large-touch displays.

Although the screen size affects the use cases associated with each device type, all three categories of devices have a unique relevance for the visualization domain. Differences in the screen sizes result in each device offering very distinct affordances. These affordances can make a particular surface more amenable to certain types of visualizations.

1.3.1 Smartphones

An exponential adoption of smartphones has occurred in the last decade. During this time, these devices also evolved tremendously, both physically and technologically. Smartphones of today are faster, thinner, and lighter than the ones from last year. People carry their smartphones at all times and in all places. They are also more powerful than before — a typical smartphone today supports a breadth of sensors, trackers, and connectivity options. Further, they now connect to a constantly increasing web of peripherals, such as smartwatches, head-mounted displays, activity trackers, and others.

The web of sensors and connectivity provide a great opportunity both for collecting data locally and accessing data from other sources. This amount of available data promotes the need for solutions to analyze it. Among others, one benefit of analyzing data in the context it was collected in is the significant reduction in latency. [1]. With this motivation, researchers have explored the area of mobile visualizations in the past. Buering et al. [16] conducted a study where they explored interaction with scatterplots on small screens devices. Karstens et al. [66] used hierarchical graphs and networks to visualize the World Wide Web on a mobile phone. Yoo and Cheon [138] use a circular radial layout to display hierarchical data on a small screen.

However, a big limitation of any smartphone for visualization related tasks is the

screen size. The form-factor and ergonomics of the phone are not capable enough to support rich visualization analysis of the type I intend to focus on in this work. This includes features such as multiple simultaneous visualizations, brushing and linking, detail-on-demand views, and filter widgets. I observed these limitations in an initial exploration of visualizations on handheld mobile devices. The aspect ratio of the screen of these devices is typically 3:2. As a result, in portrait orientation, the screen cannot accommodate a visualization in full, while in landscape orientation, the screen lacks sufficient space needed to place necessary widgets.

Thus, while it is feasible to design applications with limited analytical capability for smartphones, it is difficult to envision a system that would effectively replicate the capabilities of the desktop-class visualization systems.

1.3.2 Large-touch displays

Touch devices in this category span between screen sizes of 15 to 80 inches. Screens of this size appear either as large horizontal surfaces or as vertical flat screen televisions. These devices are currently sold by Microsoft[6], Samsung[7], and Tabler[8]. Ever since the first prototypes were introduced, the large surface area available for interaction piqued the interest of researchers across a diverse set of domains, including information visualization. The scientific visualization community has since employed the screens for application for astronomy [37], geology [117], maps and geospatial, and 3D scientific data [139]. These screens have also been explored as parts of museum and other public displays.

These devices afford two key opportunities for information visualization. First, they provide a larger, discretized display space for analysis, so users can visualize more data at a given time. Second, they allow the distribution of visualization tasks

[6]Microsoft Surface Hub, www.microsoft.com/microsoft-surface-hub

[7]Samsung 65-inch Smart Display, www.samsung.com/us/business/digital-signage-solutions

[8]Tabler Large Multi-Touch Screen TV Monitor, tabler.tv

across individuals so that they can work independently when required. These opportunities have been leveraged in the past. Cambiera [59] is a collaborative, document visualizer than runs on a multi-touch tabletop display. Group members can individually search for documents, browse through search results, and read documents, or use collaborative brushing and linking features to view where others have found or read similar documents. Similarly, Bohemian Bookshelf [118] is a tool that supports serendipitous discoveries in the context of digital book collections. The tool runs on a large vertical multitouch display and consists of five interlinked visualizations that offer unique overviews of the collection.

These examples highlight the potential of large high-resolution displays for more effective and engaging ways of interacting with visualizations. Introducing visualizations in public and collaborative settings can promote visualization use to a broad range of users beyond the traditional audience of data analysis experts.

1.3.3 Tablets

Tablets are handheld touchscreen devices with a display size between 7 and 15-inches. The history of tablets dates to early 1990s when Microsoft, Apple, and IBM introduced early versions of these devices. These versions were designed to "imitate" a notebook, and used pen-based input for interaction. This style of input continued for more than a decade, and devices such as hybrid pen+touch laptops were also introduced. The first commercial device to completely transition to multitouch input was the Apple iPad that was released in 2010. Since then, the worldwide sales of tablets has reached 195 million (2013) and is projected to reach 370 million by the end of the next year[9].

This massive adoption can in large parts be attributed to the ergonomics of the

[9]Gartner: "By 2018, More Than 50 Percent of Users Will Use a Tablet or Smartphone First for All Online Activities", http://www.gartner.com/newsroom/id/2939217

device. As with smartphones, the transition to touch encouraged a rethinking and redesigning of the interface on tablets. Consequently, the number of useful apps available to people grew tremendously. The increase in availability of these applications brought more people to the tablet platform. When compared to smartphones, tablets have a larger screen that increases the versatility of the device to more advanced tasks. Further, compared to wall-sized displays, tablets are mobile and portable, providing the opportunity for the devices to be used across locations and usage scenarios.

1.3.4 Target Device Type

The principles for designing applications for each of the three device types vary substantially and results of a design exploration on one device only partially permeate to the other devices. Thus, to narrow the focus for this work, I chose tablets as the platform to focus on. Tablets offer the right combination of computational power, screen size, types of input, and mobility. The focus of my research is to create rich and powerful capabilities for data analysis with visualizations. Because I intend for this experience to be designed for a single user, the goal is not to explore features such as multi-user collaboration and cross-device interactions. Tablets serve this use-case as they are inherently personal and intended for a single person's use.

Tablets offer one additional benefit. From my experience with building and using visualization systems on desktops, I have a good understanding of the effectiveness of these tools. Among touch-devices, tablets are most analogous to desktops and laptops because of their comparable screen-size and processing capabilities. I can leverage this similarity both as a reference for designing the features of the tablet application and to evaluate them once the design is complete.

1.4 Themes

For my exploration of visualizations on touch devices, I identified research themes to steer the design of Tangere. Similar to guidelines, these themes assisted me in taking

decisions when multiple options were feasible. However, unlike guidelines, I intended for the themes to affect the design at a much higher level. I believe that the role of these themes was vital in the design-centric research that I undertook. Because of the open-ended nature of the project, often several possible directions were feasible for the system that I was building. Themes provided one way to tie the exploration into one consistent block. They also provided a means to evaluate the system that resulted from this exercise. For this work, I leveraged three themes that I present below.

1.4.1 Novice Users

I targeted my research at people who do not have any prior experience using visualizations or visualization tools. The system needed to, thus, be designed to support uncertain, inconsistent, and distracted [17] behaviors. The topic of designing for expert versus novice users has been debated in several domains, and designers have adopted various approaches. The standard approach is to target only one of the two groups of users. Another approach is to develop two independent applications, marketed as the Basic and the Pro version. A third approach is to integrate the two in one single application as two separate modes. In Photoshop, for example, one can activate a mode where the workspace displays advanced editing features.

A fourth approach uses the same principle of supporting both novice and advanced users within one interface. However, instead of providing separate modes, the functionality is carefully fused together. The premise is that the system provides specific mechanisms for the user to systematically transition from a novice to an expert. This can also be described through the concept of threshold and ceiling: the system has a low threshold, so the users with low proficiency are able to use it effectively. Simultaneously, the system also has a high ceiling, so that users with expertise can complete a larger number of tasks too. The features, however, are designed such that

novices use only the basic functionality without being distracted or discouraged by the advanced functionality that may exist.

In desktop-based visualization applications, the prevailing trend is to target the expert users. These applications provide a rich set of operations, often at the expense of more complexity. The need for a high number of features is partly expected — analyzing data is a complex task, and a large operation set is necessary. However, current versions of the applications such as Tableau and Spotfire take the form where, although the ceiling is sufficiently high, the threshold of user expertise required to use them is fairly high as well. The application interfaces are complex and feature dense. New users are often unable to use the system right away and require considerable training to grasp the functionality provided.

The move to touch devices provides a great new opportunity to rethink the design of these applications. Rather than viewing this simply as a transition from the desktop to tablets, the visualization interface can be architected and designed in a careful manner so as to accommodate both the novice and the advanced users in equal measures. By leveraging the theme of targeting novice users, I was able to embark on a strategic path that would encourage the development of solutions which achieve this goal.

1.4.2 Simplicity

The second theme for my work is Simplicity. A simple interface is one where a user is able to achieve her goals in such a way that the interface is invisible to her [84]. Simple interfaces avoid unnecessary elements and are clear in their message. In this regard, simplicity relates closely with the previous theme.

However, simplicity is not just a function of things absent from an interface, but equally of things present on it. Simplicity is derived from how one explains the presence of an element on the interface and how she understands how to use it.

Since the interface drives our behavior, operating an interface that we do not fully understand generates a feeling of dissonance and lack of control. Complex software takes away control by forcing people into unplanned interactions, confusing pathways, and surprising outcomes.

To achieve simplicity, the designer must fully understand the usefulness of the system, and the usefulness of its every constituent[10]. In his book, Maeda describes the ten laws of achieving simplicity, which include reducing, organizing, saving time, and adding emotions [77]. Using these principles, I hope to design an interface that is *simple*. It is unclear if simplicity can be accurately measured. However, adapting best practices, such as keeping users in control by regularly surfacing system status and consistently giving then insight into what to expect, would certainly lead the design of the system in the right direction.

1.4.3 Delight

This theme caters to the role of an application to affect emotional change. The feeling of delight comes from subjecting users to an unexpected reaction. The instruments available to visualization systems are graphics and animation.

Graphics consist of the playful and friendly user of colors and shapes that afford approachability. They also provide character to the interface. One example is the design strategy employed by Android called Material Design[11]. The specification recommends the use of specific color palettes and visual effects such as shadows that give the interface a distinct identity. An interesting mix of visual feature that appear consistently across the operating system are pleasing and comforting for the user.

Animations are used for responding to user actions on the interface. Motion has been shown to be highly effective at attracting attention, facilitating transformations

[10]Buchanan [15] defines usefulness as a product's clarity of its own content and purpose.

[11]Google Material Design, material.google.com

of position, size, shape, and color, and for communicating causality and intentionality [45]. In visualizations, the role of animations has been studied in detail. Robertson et al. [98] explored the effectiveness of animation in trend analysis. Heer and Robertson [45] presented the DynaVis framework for transitioning between different visualization techniques. They also introduced key concepts on staging the characteristics of glyph motion for maximum effect. Other work has explored best practices for encoding attributes to the properties of the animation [57, 119].

However, animation plays another key role — that of emotional engagement [119, 126], engendering increased interest and enjoyment. Animations add character and fun, making a product feel unique and helping users relate to it more. However, more importantly, animations engage users. More time spent engaging with an interface is more time spent learning how to use it. In a complex design environment, such as ones visualization applications tend to be, motion and animations can be useful in attracting and connecting with users.

Graphics and animation are critical pieces of the user experience. In the system I designed, I employed these features extensively. Using these features, I intended to increase the emotional appeal of the interface and, consequently, the approachability of the system.

1.5 Organization

The rest of this dissertation is organized as follows.

In Chapter 2, I provide a detailed literature review of related work and discuss how my thesis relates to and builds on existing work. Since my work lies at the intersection of information visualization and natural user interfaces, I organize the chapter into the key themes of interaction in visualization, interaction on touch interfaces, and visualization on touch interfaces.

Chapter 3 outlines the space of possible solutions that are relevant to my research

and provides a high-level summary of the system that I have built, called Tangere. The subsequent chapters provide a detailed description of the system. The chapters map to the different stages of my research, beginning with Chapters 4 and 5 that describe my initial explorations of the Scatterplot technique and the subsequent expansion to Multiple Coordinated Views.

In Chapters 6 and 7, I present the advanced interactive features for selection and layout that I designed for Tangere. Chapter 8, 9, and 10 detail the three evaluations that I conducted to measure the effectiveness of Tangere and the degree to which it succeeds in achieving the goals I set out with. The three studies have been designed with unique objectives, and capture very different facets of the system.

Finally, in Chapter 11, I summarize my research and the contributions that I make. I also discuss future plans for Tangere and propose new research directions, including addressing the entire visualization pipeline, and integrating multimodal forms of input.

CHAPTER II

RELATED WORK

Since my research is situated at the intersection of visualization and natural interfaces, the related work for this thesis broadly covers three topics of research — interaction in visualizations, interaction with touchscreens, and visualizations on touch devices.

2.1 Interaction in Visualization

The research exercise of bringing visualizations to touch interfaces is similar to work done initially to develop visualizations for desktops and cursor-based interfaces. The method used then was to study visualizations through the three building blocks of representation, presentation, and interaction [114]. Decades of research has produced a rich understanding of the principles and challenges of representing data. Over this period, we have assembled a wide variety of visualization techniques. Heer et al. [47] present this in their survey of the commonly used visualizations.

With the switch to non-traditional inputs, however, it is interaction that becomes the critical centerpiece. The role of interaction has been well explored and researchers have presented several lists of the types of interactions one encounters in information visualization. Dix and Ellis [28] list the activities of highlighting and focus, accessing extra information – drill down and hyperlinks, overview and context, changing parameters, changing representation, and linking representation, among others. Wilkinson [130] provides the broad categories of filtering, navigating, manipulating, brushing and linking, animating, rotating, and transforming, each with specific sub-categories and examples. Other similar characterizations have also been proposed by Ward & Yang [125], and Keim [68].

In developing the notion of a "science of interaction", Pike et al. [95] categorized

interactions into two groups: 1) the low-level interactions between a user and the controls of a specific user interface and 2) the higher-level interactions between a user and the information space. They described the role of interaction as that of a reasoning aid for exploring various attributes of the data space as one develops models and better understanding of the underlying data.

Interaction has also been analyzed through the lens of people's tasks and goals. Yi et al. [137] studied a wide range of interaction techniques in information visualization research. They categorized the interactions within these techniques into seven main user intents:

1. Select: mark something as interesting
2. Explore: show me something else
3. Reconfigure: show me a different arrangement
4. Encode: show me a different representation
5. Abstract/Elaborate: show me more or less detail
6. Filter: show me something conditionally
7. Connect: show me related items

These taxonomies are relevant for my work since they serve as a starting point for designing the functionality of a visualization tool. Since touch devices provide a constrained environment, the taxonomies also help steer the prioritization of the different aspects of design. This includes guidance on interactions that are specifically germane to and particularly useful within information visualization, such as dynamic query for filtering [112], brushing and linking [20], drill-down, and various forms of selection [46], among others.

Finally, a notion particularly relevant to touch-based visualizations is *fluid interactions*. Elmqvist et al. [32] define "fluid interactions" through three components. First, interaction should promote staying in the *flow*. Here flow is a state of immersion in an activity where the participant has a high degree of involvement and concentration, is at a loss of self-consciousness, and is engaged in productive and rewarding outcomes. Second, interaction must provide direct manipulation of the objects (in this case, data) at hand. Third, fluid interaction should minimize Norman's [87] gulf of evaluation (difference between the system's state and the user's perception of that state) and gulf of execution (difference between the allowable actions for a system and the users intentions).

2.2 Interaction on Touchscreens

Interaction with touchscreens has been a topic of significant research for over two decades, though devices with touch-based input have only become mainstream in the past few years. Early work focused on studying the benefits of touch-based interaction over cursor-based interaction, with studies finding touch interaction to be faster [65, 92]. Sears and Shneiderman [108] showed that for a single target selection task, direct touch input outperformed indirect mouse input. These studies were conducted on touchscreens that detected a single point of contact. Later, DiamondTouch [26] technology pioneered multifinger and multiuser detection, and eventually culminated into the development of the Microsoft Pixel-Sense (formerly Surface) displays. These devices were instrumental in early research on the guidelines for usability on multitouch screens.

Special emphasis has also been put on guidelines for both the design of user interfaces and the use of gestures on touchscreens. Benko et al. [12] presented three guidelines for touchscreen design — (a) keeping the interface simple, (b) providing

an offset to the cursor when needed, and (c) enabling the user to modify the control-display ratio. Hinckley & Wigdor [54], in a detailed discussion on the requirements of a touch system, mention that a designer must consider the sensor, the feedback, the device, and interplay between all the interaction techniques.

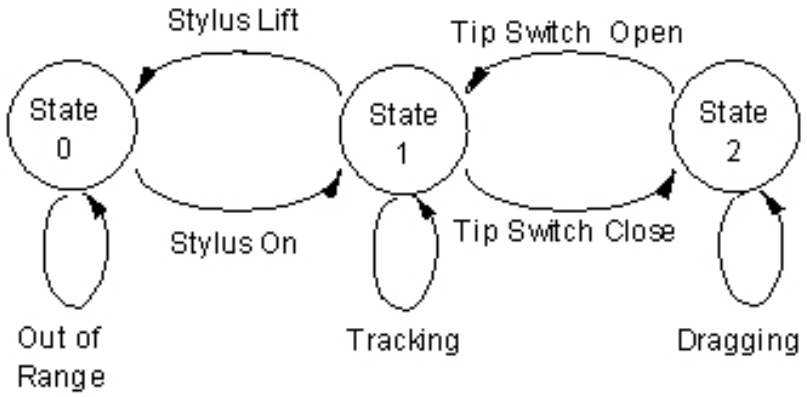


Figure 1: Buxton's three state model of graphical input [18].

For touchscreen interactions, Hinckley et al. [53] adapted the three-state interaction model (out-of-range, tracking, and dragging) for cursor-based input presented by Buxton [18] (Figure 1) to a two-state model (out-of-range and dragging) for touch screens. Wigdor & Wixon [129] provide guidelines for both gesture recognition and the use of gestures, and outline the fat-finger problem — inability of fingers to achieve the precision of a cursor. Yee [136] suggested guidelines for finding suitable gestures: systems should use appropriate metaphors, and the gestures should be easy for a system to recognize. Wobbrock et al. [133] extend the guidelines by gathering gestures from users through a guessability study. Morris et al. [80] found that participants preferred a user-created set of gestures to an expert-created set for standard tasks.

Considerable attention has also been paid to two subareas of touchscreen interaction — improving selection and expanding gesture vocabulary.

2.2.1 Issues with precise selection

Selecting objects in a dense area is a critical task in techniques such as scatterplot and linechart. However, selection in a dense area requires very precise finger action and is considerably difficult due to the inherent imprecision of touch input. The imprecision arises as a result of the fat-finger problem [56] and the occlusion that results from it [96]. These issues of imprecise selection have been well documented in the past, and researchers have presented several techniques to mitigate them. Early work included take-off [96] that placed the cursor at a fixed offset from the location of the touch, and zoom-pointing [11], where the touch area was scaled up to allow precise selection of the target. Esenther and Ryall presented DTMouse [34], a technique that positioned the cursor at the centroid of two fingers. Dragging the fingers on the screen moved the cursor, whereas tapping with a third finger toggled mouse-down mode that allowed an object to be dragged.

Over the years, several other techniques have been proposed. Moscovich [81] presented precise selection techniques for closely placed widgets by enhancing the pointer activation areas. Benko et. al. [12] performed a comparative evaluation of several unimanual and bimanual techniques. While certain techniques performed better than others in specific configurations of use, the overall results suggested that, unlike cursor-based interfaces, no ideal technique for precise selection on touch-screens existed. The effectiveness varied based on the ergonomics of the device (smartphone vs. tablets), number of fingers used, and the application context (e.g. sketching requires very precise movement).

More recently, Au et al. [5] presented the LinearDragger technique for selecting an item from a dense cluster of items, closely resembling the selection task in a scatterplot (Figure 2). Users begin by initiating a drag gesture at the position of the target object. This reveals a zoomed-in view of the target area, with the object nearest to touch location selected. As the user drags the finger away, the system sequentially scrubs

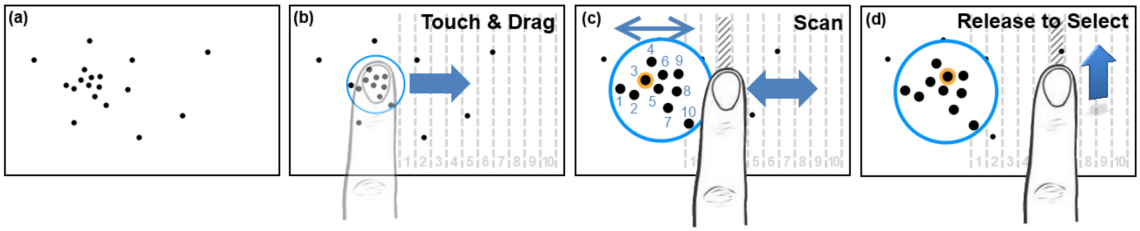


Figure 2: The LinearDragger selection technique. [5].

through each of the neighboring objects. Lifting the finger commits the selection.

Benko et al. [12] recommend that selection should be simple, since the ability to directly touch an object to select it is a very appealing aspect of touch screens. Adapting this principle is fundamental to designing appropriate selection mechanisms, specifically for techniques such as scatterplots and linecharts.

2.2.2 Limited Vocabulary for Specifying Commands

The predominant trend when designing systems for touch-devices is to adopt the standard vocabulary of gestures that people employ in their everyday use, i.e. a combination of tap, pan, pinch, and rotate gestures. While this approach has obvious benefits, namely leveraging people's familiarity with the gestures, a limited vocabulary limits the number of features that can be supported in an application. Such a limitation is particularly pronounced in the case of visualization applications due to the general feature-richness of these applications [72].

To address these limitations, researchers in HCI have explored several alternative methods for augmenting the input to the system. The methods have often aligned to one of two main categories — multitouch menus and touch overloading.

2.2.2.1 Multitouch Menus

As the number of gestures increase, the cognitive overhead of memorizing the gestures also increases. Menus help in reducing this overhead by encouraging recognition over recall. Recent literature has proposed several menu designs on touch screens. The

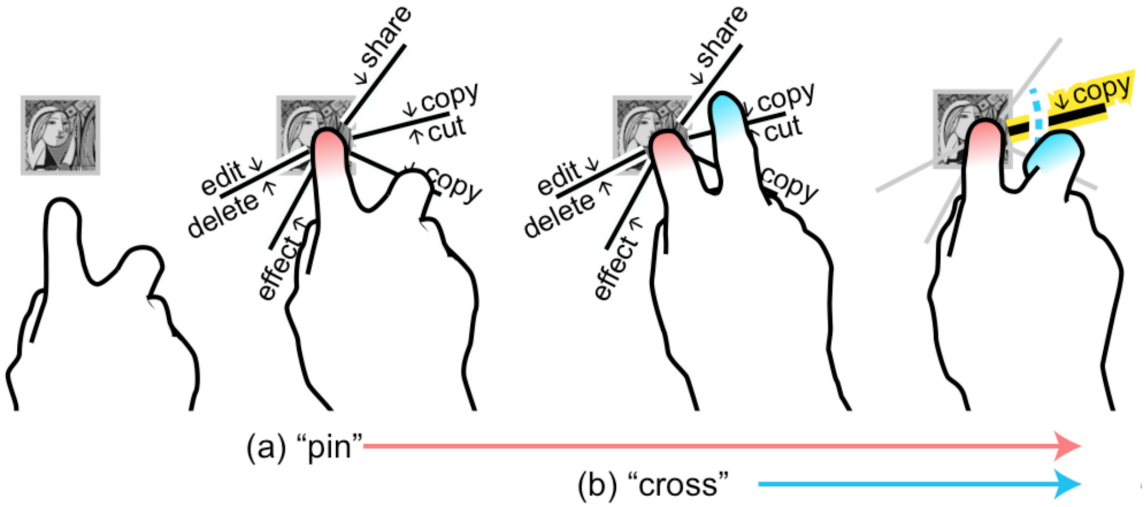


Figure 3: The Pin-and-Cross marking menu [75].

common thread between these designs is the use of an explicit gesture to reveal a menu around the location of the fingers.

Marking menus are the most common example of menu design on touch screens. Originally designed for cursor and stylus-based input, Lepinski et al. [73] adapted the technique to touchscreens, using finger-chords (combination of fingers making contact with the screen) for interaction. Bailly et al. [7] extended the design to bimanual interaction and replaced chords with number of fingers. Recently, Damaraju et al. presented Multi-Tap sliders [23], a chorded-menu design that combined the above technique to also allow parameter adjustment using index finger movement. Finally, Luo and Vogel [75] presented a unique marking-menu design wherein the user selected an option by crossing it with the nearest finger (Figure 3).

2.2.2.2 Touch Overloading

A second class of solutions for augmenting touch-input involves touch overloading: discriminating between different type of touches to enable distinct actions. This discrimination can be made based on the properties that a touch instance reveals, such as location on the screen, direction of movement, velocity, angle of finger, input

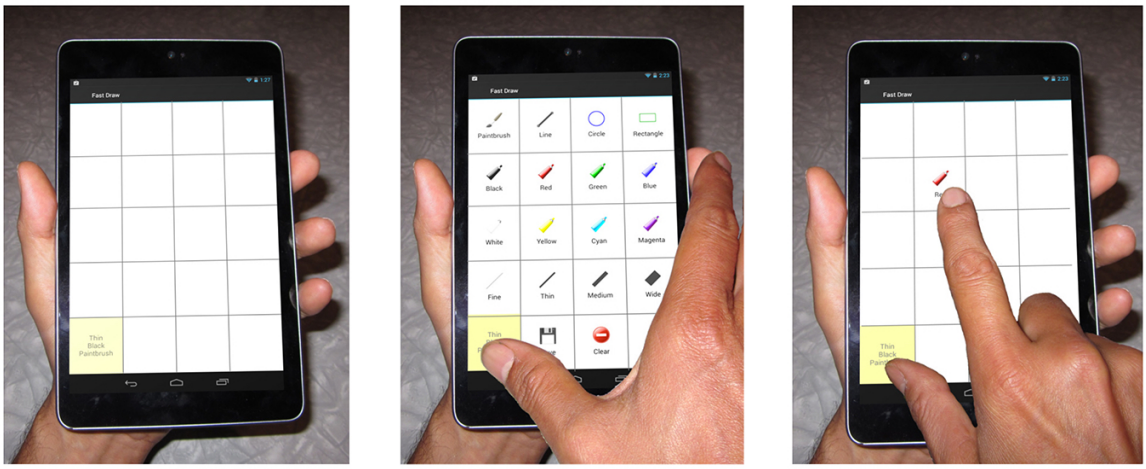


Figure 4: The FastTap menu technique [42].

type (finger vs. stylus), duration, number of simultaneous fingers, and more recently, shear and pressure. A combination of these used together can potentially result in an extensively large vocabulary of gestures.

1. Location: Although every touch instance on the screen is differentiated by its location, it is also feasible to associate high level actions to specific locations on the screen. For example, in the BiTap technique [122], buttons are positioned nearest to the location where the non-dominant hand grabs a touch-device.

2. Multiple, sequential touches: While simultaneous-touches are often used for zooming and rotating, sequential-touches have also been used as potential delimiters for command invocation. For example, Heo et al. [50] used consecutive distant taps patterns such as Ta-Tap (2 distant taps) and Ta-Ta-Tap (multiple distant taps) to trigger commands. Similarly, the FastTap technique [42] used thumb-and-finger touches to actuate commands in a spatially-stable grid-based menu (Figure 4). The technique supported the novice to expert transition since a quick thumb-and-finger tap invoked the command without revealing the menu.

3. Pressure, Roll and Shear: Researchers have successfully differentiated touch events based on pressure, roll and shear. Heo and Lee [51] differentiated strong

tap from a gentle tap using phone's existing hardware. Roudaut et al. [101] used finger roll to demonstrate 16 elementary gestures, e.g. rolling left was different to rolling right. Harrison and Hudson [43] used shear, which is a force that is tangential to a screens surface, to create five classes of advanced interactions.

2.3 Visualization on touchscreens

2.3.1 Research Tools

The importance of information visualization on touch screens has long been recognized but has only recently seen an increased emphasis. Early data visualization research for computers other than desktop PCs includes systems designed for mobile devices. Buering et al. [16] presented a zoomable user interface (ZUI) based scatterplot visualization for a PDA that used a stylus for input. In a user study comparing two scatterplot applications, one displaying both a detail view and an overview and the other displaying only the detail view, participants solved search tasks on a phone faster with the version without the overview.

As multitouch devices became more common, several research efforts targeted

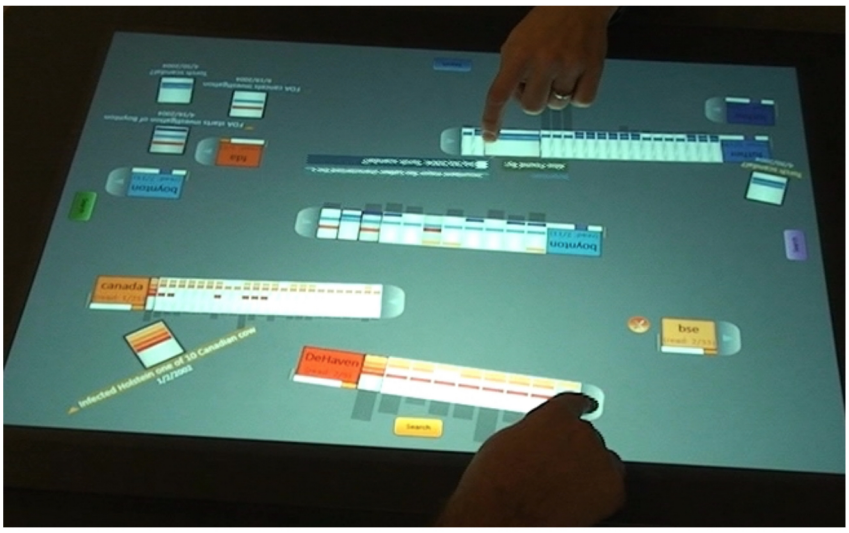


Figure 5: Cambiera: Collaborative document analysis on a tabletop [59].

visualizations on touch-based tabletops, especially the collaboration aspects. Cambiera [59] is a visualization system that supports collaborative brushing and linking of search results (Figure 5). Heilig et al. [48] describe ScatterTouch, a two-dimensional scatterplot visualization technique that used the concept of multi-focus regions to support simultaneous, multi-user interactions. Schmidt et al. [107] explore ways to use multi-touch for link manipulation and graph interaction, while North et al. [88] compared how people manipulate node-link diagrams on a touch-based tabletop and a mouse-based desktop. Frisch et al.'s study [36] used the *guessability* study approach [133] to obtain user-elicited pen and touch gestures for manipulating node-link diagrams on a tabletop.

Figure 6: Using pen and touch for data exploration on whiteboards [123].

SketchVis is a system that allows users to sketch visualizations on a whiteboard [14]. The user draws representative visualization constituents such as the coordinate system, labels on axes, and some initial glyphs; the system responds by filling in the chart correspondingly. Walny et al. [123] investigated the use of both pen and touch for data exploration on whiteboards (Figure 6). Their study explored the interaction styles people employ, in particular the role of pen and touch for representations and

linking.

Within information visualization focused on small-to-medium size screens, only a few systems have appeared. Baur et al. [8] examined the stacked graph for their TouchWave system, and developed a new set of multi-touch gestures for scaling, scrolling, providing context, extracting layers, and many other activities. They noted that the four design goals of supporting kinetic manipulation, creating integrated interactions, avoiding complex gestures, and considering the viability of the interaction set drove their interaction designs.

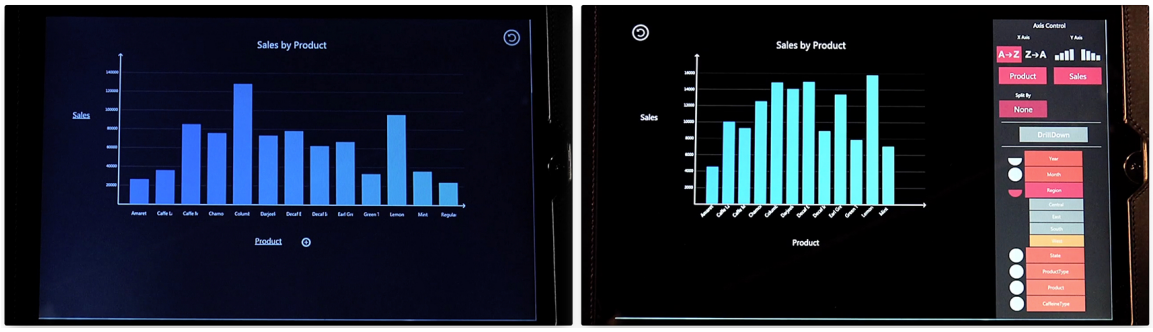


Figure 7: Gesture-driven and WIMP-driven interfaces for analyzing data with barcharts [29].

Drucker et. al. [29] highlight the difference between gesture-driven and WIMP-driven interactions for information visualization applications. They compared two different interfaces that use a barchart to present the same data (Figure 7). Both interfaces provided the same functionality. However, the first interface (WIMP) used interface elements such as menus and buttons for interaction while the second interface (FLUID) used gestures. In their experiment, participants performed a series of tasks using both interfaces. Results showed that the FLUID interface performed better than WIMP. Participants were faster at performing tasks using gestures and also demonstrated fewer errors. A majority of participants also expressed preference for the gesture interface.

Rzeszotarski and Kittur [102] present Kinetica (Figure 8), a scatterplot visualization system for tablets that employs zoom lens and razor filter interactions. The

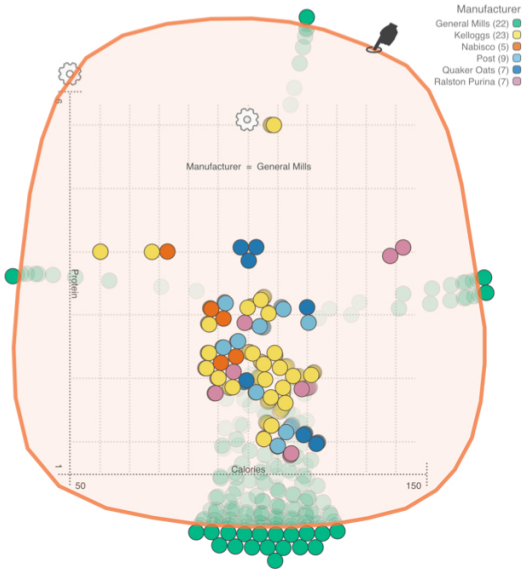


Figure 8: Kinetica: naturalistic multi-touch data visualization. [102]

authors enhance the playfulness of the interface using physics-based interactions and gravity. They compare the performance of the tablet application with Microsoft Excel for basic data exploration and sensemaking tasks. Study results indicate that their tablet-based tool was more satisfying for participants, and led to a more holistic understanding of the data they were exploring. The focus is more on exploring the physics-based affordances and interactions metaphor than on the broader design space of multi-touch interactions for visualizations.

Recently, interest in understanding the theoretical foundations for visualizations on non-desktop based interfaces has also grown. In an article that explores existing research and outlines directions for future work, Lee et al. [72] advocate strongly for moving past mouse and keyboard interactions for information visualization. The authors survey multiple forms of post-mouse/keyboard interaction, not just touch and gesture. They specify the interaction design considerations for the new modalities, eliciting four principle dimensions: the individual, the technology, social aspects of interactions between people, and the interspace between a person and the technology. They specifically identify "going beyond mouse and keyboard" as an opportunity and

a topic worthy of further research.

Isenberg et al. [60] discuss the visualization space specifically for touch-based surfaces, highlighting the technical, design and social challenges in supporting visualization on touch devices. For touch interaction in particular, they outline the creation of a gesture vocabulary that is both global to various visualization types and low in complexity as a central research topic. Jansen and Dragicevic [61] describe a modification of the infovis pipeline to accommodate beyond-the-desktop visualization systems. They suggest unifying the infovis pipeline and instrumental-interaction model [9] for post-WIMP interfaces that include touch-based interfaces as well as physical, fabricated visualizations.

2.3.2 Commercial Tools

The space of visualization on tablets has also been populated by commercial applications, available on operating systems such as Android and iOS. The target domain for these applications is business analytics, and thus they typically provide dashboard-based solution. Both Spotfire [1] and QlikView [2] provide second-screen companion apps for their desktop clients. Users can view their data in standard techniques such as linechart, barchart and scatterplots. Domo [3] and Roambi [4] are other similar standalone tools made only for tablets and smartphones. While the techniques within these tools are interactive, they support only low-level data analysis, targeting users who view fixed-format data in a static dashboard representation.

An exception to this is Vizable by Tableau [5]. Vizable is a data analysis application that runs on an Apple iPad. In Vizable, users can import their own data and

[1] Spotfire, spotfire.tibco.com

[2] QlikView, www.qlik.com

[3] Domo, www.domo.com

[4] Roambi, www.roambi.com

[5] Vizable, vizable.tableau.com

view it using two visualization techniques — linechart and barchart. The two techniques appear in two separate containers titled CategoryWorld and TimeWorld. In CategoryWorld, data is displayed with bar charts, with data aggregated using sum, average, minimum, maximum, count unique, or percent of total. In TimeWorld, one can compare data points across days, weeks, months, and years.

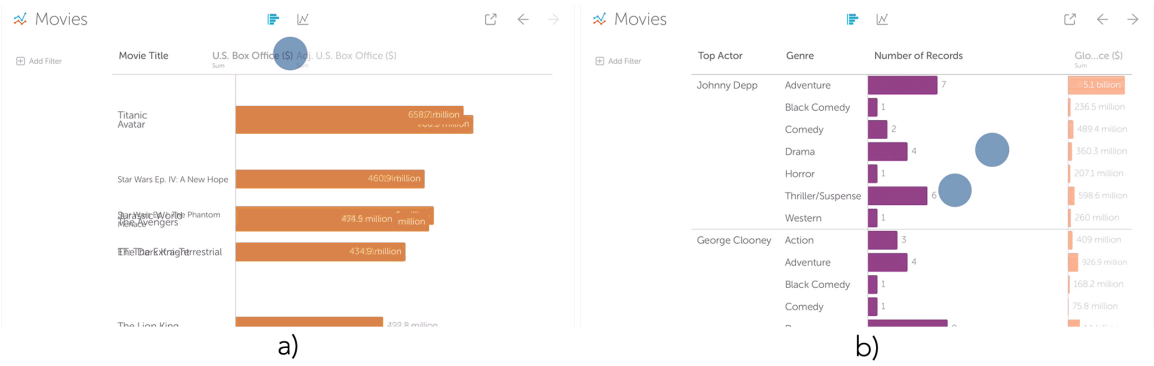


Figure 9: Vizable interface a) Swiping horizontally changes the attribute. b) Two-finger pinch adds another attribute.

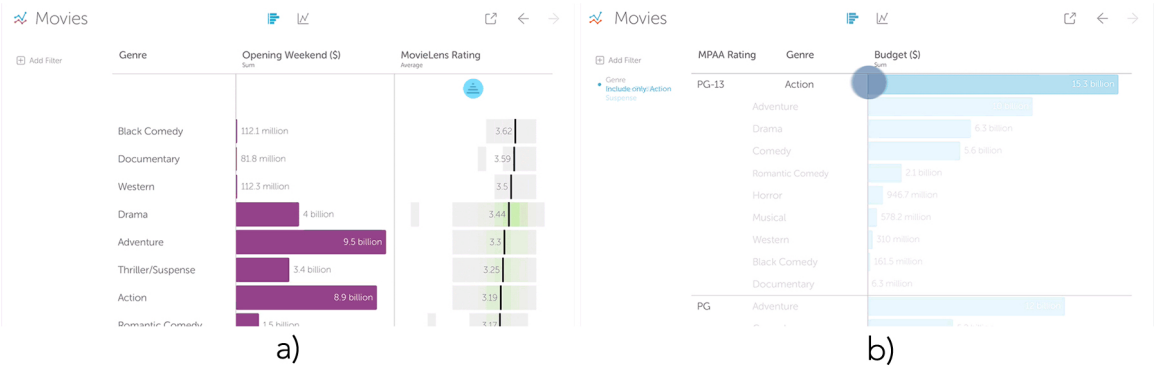
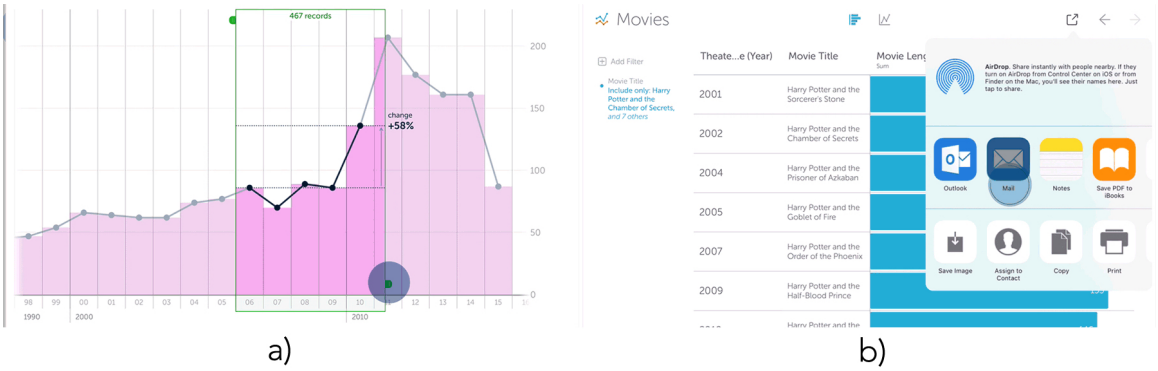


Figure 10: Vizable interface a) Swiping down on a barchart sorts the bars. b) Swiping left on a bar filters it out. Swiping right filters all other bars out.

The application makes heavy use of gestures and animations. In the CategoryWorld, the quantitative or the categorical attributes can be changed by dragging them horizontally (Figure 9a). Dragging vertically changes the type of aggregation used (sum, average, median). Tapping on the attribute shows a list of all attributes on the left. Attributes can also be selected from within this menu. Additional attributes

(both categorical and quantitative) can also be added by using a pinch gesture (Figure 9b). Pinching out adds an additional attribute while pinching in removes it. Finally, quantitative attributes can be converted into categorical attributes by dragging the corresponding attribute from the right and placing it on the left.

The bars can be sorted by swiping down on them (Figure 10a). Dragging a bar towards the left filters it out, while dragging right filters all the other bars out (Figure 10b). The filtered items appear in a list on the left, from where they can brought back in with a swipe. Tapping on them reveals a filter menu where the properties of the filter operation can be configured.



a) b)

Figure 11: Vizable interface a) Tapping on the linechart reveals a selection window. The window can be extended using the handles. b) Snapshot of the view can also be exported.

In the TimeWorld, the data appears as a continuous linechart. The chart can be zoomed in with a pinch gesture. Zooming is semantic, so the data aggregation switches from years to months to, ultimately, individual dates. Tapping on a bar selects the bar and reveals a selection window. The size of the window can be increased or decreased using handles on either sides (Figure 11a).

Considerable emphasis has also been placed on features such as importing, exporting, printing, and sharing the visualization views (Figure 11b). Various features of the system cater specifically to that.

CHAPTER III

TANGERE: A TABLET SYSTEM FOR INFORMATION VISUALIZATION

3.1 Introduction

In this chapter, I present Tangere, an information visualization system that I built for tablet devices. When I commenced this research, tools for information visualization on tablets were entirely absent. This strongly influenced my design process as my goal was not to build on existing research to improve the contribution, but instead to construct an experience from the grounds up. My aim, thus, was to design a new interface that supported intuitive, powerful, and useful suite of touch-based interactive operations on data visualizations. I intended for these operations to be housed within a tablet system that was similar in concept and outcome to ones we use for data analysis on desktops.

In the following sections, I provide a high-level summary of the system that I have built. I begin with presenting the motivations for targeting an analytical tool over other potential visualization tools. Subsequently, I present Tangere and describe its different parts. This description helps guide the conversation that follows in the subsequent chapters, where I provide a detailed account of the different stages of my design process.

3.2 Envisioning a visualization tool for tablets

What need does a visualization tool for tablets serve and how to target that need? The domain of visualization-based data analysis tools is very broad. Existing tools vary extensively in the analytical tasks they serve and on the data types, data sizes,

user types, and contexts of use they target. Each category of tools serves a particular purpose and is, thus, relevant for specific scenarios. However, given the general usage of tablets and the profile of the users using them, certain tools are more relevant than others on these devices.

To better identify these tool types, below I list their different categories. The tools are differentiated based on two factors – usage scenario and data type.

Usage Scenario: In terms of usage scenario, visualization tools can broadly be classified into three types.

1. Visualization building: The focus of these tools is to allow people to create a visual representation of their data, with the goal of ultimately exporting or sharing the representation. Over the course of constructing the representation, the user may gain new insights. For the most part, however, the user is versed with the insight s/he wants to represent. Extra emphasis, consequently, is placed on features such as styling and exporting the visualizations.

   Examples: Microsoft Excel[1], Google Charts[2], Plotly[3]

2. Dashboard - These tools focus primarily on providing quick overview of fixed format streaming data or data that is constantly updating. Stock market or company sales numbers are good examples. The dashboard representations are fine-tuned to be very effective for summarizing data for a time period, but is not designed for deep analysis.

[1]https://products.office.com/en-us/excel
[2]https://developers.google.com/chart/
[3]https://plot.ly

Examples: Google Analytics[4], PowerBI[5], Domo[6]

3. General purpose visualization tools - These tools serve the entire pipeline of data analysis tasks, from importing and cleaning data to exporting findings. However, the primary emphasis is on enabling people to identify insights through a diversity of representations and interaction mechanisms. Some advanced tools in this category can also be used for visualization building and dashboards.

   Examples: Tableau[7], Spotfire[8], Jigsaw [115]

Dataset: Visualization tools can also be differentiated based on the data they are designed for, with differences existing both in the type and the size of the dataset that they work with. On one end are niche tools designed to visualize very specific types of data formats or visualization types, e.g. Gephi[9] & Ploceus [74] for network visualizations and Parvis [44] for parallel coordinates. On the other end are general purpose tools, such as Tableau and Spotfire, that accept data in a multitude of formats. However, the predominant format that these tools are optimized for and excel at is tabular data, existing as spreadsheets or within relational databases.

3.2.1 Identifying the right configuration

From among the possible options of types of visualization tools for tablet devices, the one I chose was the *tabular data analysis* tool. The choice was fairly straightforward as it was the most challenging option. Analysis tools are more complex than the other options and require considerably more user interaction, which is a challenge to

[4]https://www.google.com/analytics
[5]https://powerbi.microsoft.com/en-us/
[6]https://www.domo.com
[7]https://www.tableau.com
[8]http://spotfire.tibco.com
[9]https://gephi.org

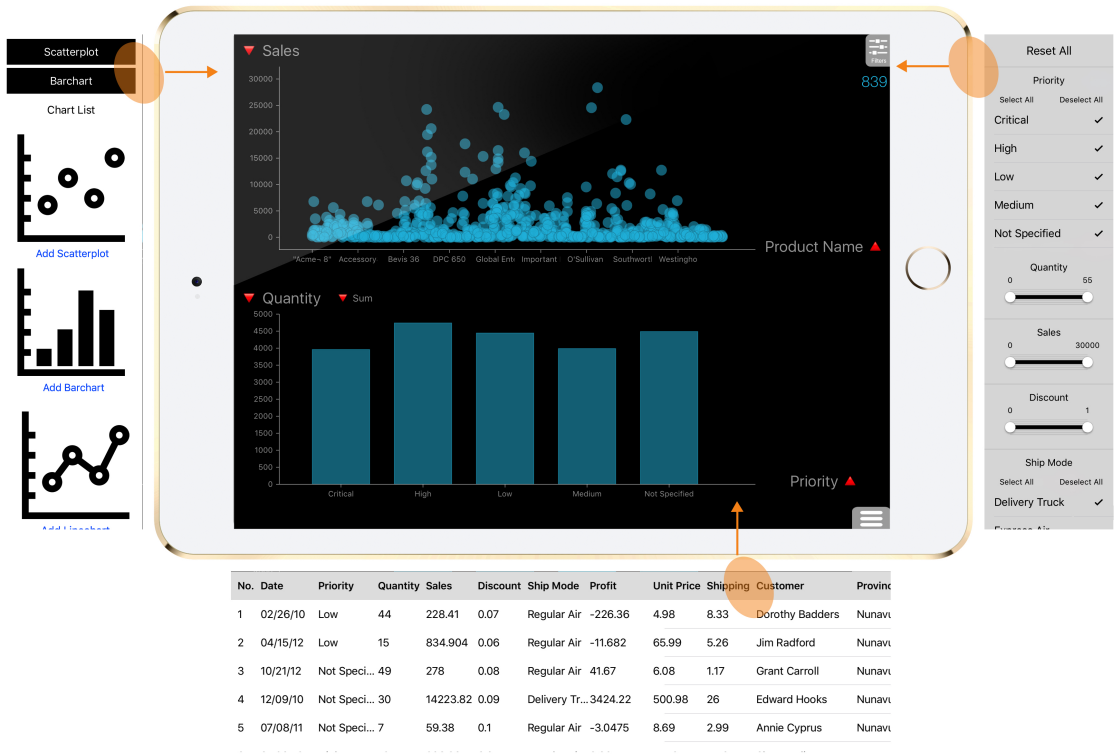


Figure 12: The Tangere application. On the interface, the canvas is the central component. The panel for adding views is to the left, data table is at the bottom, and the filter menu is to the right. The panels stay beyond the screen and can be brought in with a gesture.

get right on handheld-touch devices. Consequently, the problem is tougher to solve.

Although visualization building is also an interesting problem, the focus of the problem, and hence the challenge, is different. For analysis tools, a vast body of literature exists that investigates how people use representations to analyze data. Designing tools for visualization creation, however, builds less on such literature. Instead, the general principles of HCI are more relevant. Lately, there has been considerable interest in this space, including explorations that target touch-based devices.

Finally, I did not target a dashboard system as I consider dashboards to be a subset of the more general analytics tools. Dashboards offer more constrained work environments, particularly in terms of interactions. There are several differences as

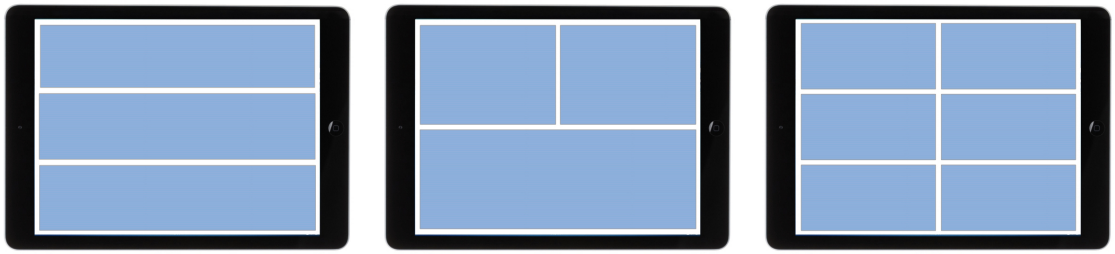


Figure 13: A subset of possible layout configurations. The canvas is a 2x3 grid and can fit upto 6 views.

well. For example, since dashboards are typically used for streaming data, a stronger emphasis is placed on highlighting the new data and differentiating with the older data. However, besides a few differences, features within dashboard tools are a subset of those in analytics tools.

Designing an analytics system serves a broader need. While iPads and other tablets have not yet gained the prominence in the professional workspace, they are ideal for personal use. People are also increasingly consuming a large amount of personal and small-scale data, either generated through personal tracking on wearables or personal portals such as point of sale terminals. Such use cases serve as a fertile ground for exploring visualization applications as the need for advanced and deeper analysis is low, datasets are typically smaller, and the analysis can be conducted with only a few basic visualizations. Consequently, systems can be designed that contain a restricted set of features initially, but mature over time to add more advanced features and serve complex use-cases.

3.3 System Overview

In my research, I used the above argument to design a tablet-based visualization system for analyzing tabular data. I call this system Tangere[10]. Tangere is aimed at novice users unfamiliar with both touch-based interfaces and data visualization tools. The tool provides a set of intuitive controls to represent and manipulate data.

[10]Tangere in latin translates to *'to touch'* and *'to elicit an emotional response'*.

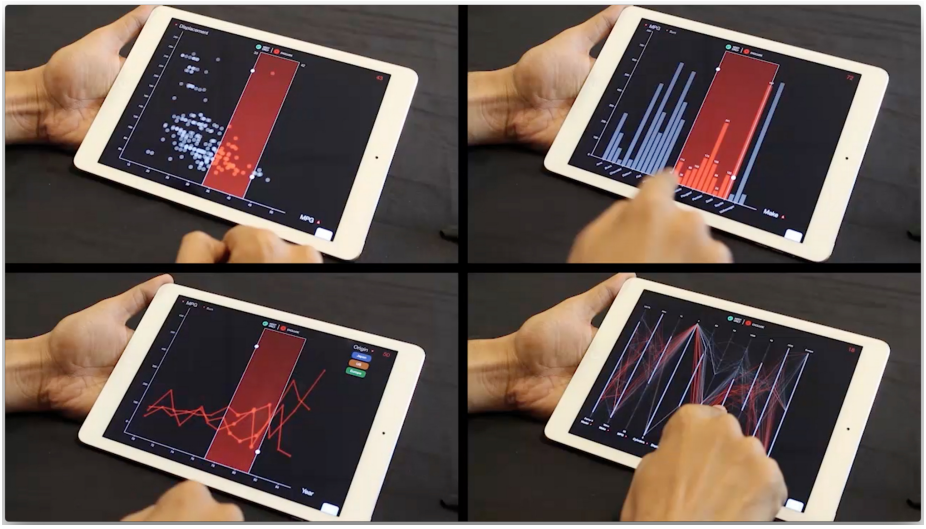


Figure 14: Several interactions are common to all visualizations. Here, axis-based selection is performed on the four chart types.

Figure 12 provides a snapshot of the system. The system consists of an array of components that work together to support an analytical process. At the core of the system is a canvas component that contains the visualization views. Four types of views can be added to the canvas - *Scatterplots*, *Barcharts*, *Linecharts*, and *Parallel Coordinate views*. At a time, the canvas can contain one or more of these views, with up to six views maximum. It consists of a 3x2 grid, where each slot can house a view, resulting in a total of 14 possible view layouts (e.g. Figure 13), and more than 400 different view configurations.

The visualization views support a range of interactive operations such as selection, zooming, filtering, and brushing & linking. For several of these, Tangere provides multiple methods to achieve the desired effect. The methods differ on the instrument used for specifying the operation and the number of glyphs that are affected as a result. For operations that are common across visualizations, the interactions employed are the same. Therefore, selecting and filtering glyphs in barcharts behaves in the same manner as in scatterplots, linecharts, and parallel coordinate views (Figure 14).

Figure 15: Attribute (green arrows) and sub-attribute (red arrows) dropdown controls. Attribute dropdown presents the options of variables that can be plotted on the axis. Sub-attribute dropdown presents options for the level of a hierarchical attribute to use as the attribute, or the type of aggregation to use (sum, average, or count).

Each visualization exposes dropdown controls to change the attributes being visualized. For hierarchical attributes such as date, an additional control presents subcategories such as Month, Day of Week, etc. The additional control is also present in charts that present aggregated data, such as barchart and linechart. This control lets the user change the type of aggregation used between sum, average and count (Figure 15).

Apart from the views, the canvas presents a count label at the top right depicting the number of data items being visualized. When glyphs are selected, the canvas presents 'Keep-only' and 'Remove' buttons at the top for filtering the selected glyphs. If data is filtered in this manner, filter-badges representing each filter operation show up on the bottom right of the canvas. The badges are colored green and red for 'keep-only' and 'remove' operations, respectively (Figure 16). The badges are also interactive — tapping on a badge enables and disables it (and the corresponding filter operation), and swiping on it towards the right removes it from the view.

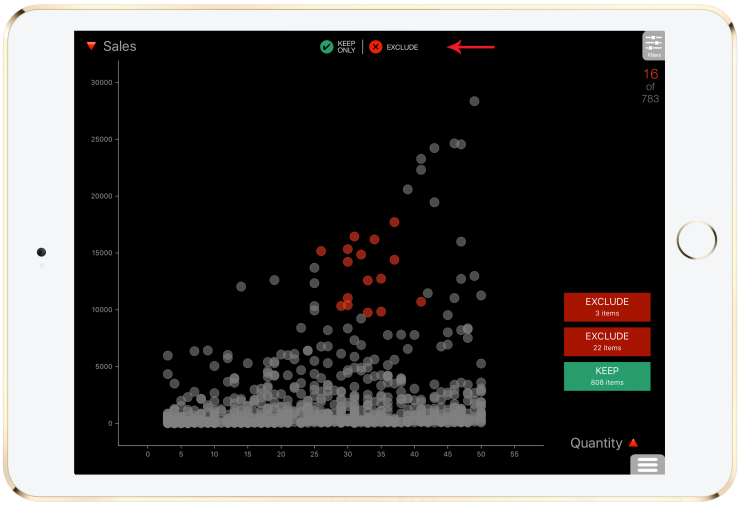


Figure 16: Keep-only & exclude controls for filtering selected glyphs. Activating a filter presents a filter badge on the right. The badge can be tapped to toggle the filter, or swiped towards the right to remove the filter.

A second component, specific to systems that provide flexibility in choice of visualizations (also called Level 2 systems [90]), is a menu widget for selecting the view to add to the canvas (Figure 12). The four chart types are present as icons that can be tapped to add a chart. When the application is first launched, the menu is visible on the screen. When a chart is added, the menu closes and places itself outside the view area. It can be brought back into view by right-swiping on the left edge. For each chart that is added to the canvas, a chart-badge with the name of the chart appears at the top of the menu. When multiple charts are present on the canvas, the layout of the menu resembles that of the views on the canvas (Figure 17). The badges can be tapped to highlight the corresponding chart, and swiped towards the right to remove the chart.

I included two other components that are frequently found in visualization systems. The first is a set of dynamic query filter controls that allow a person to interactively change the data being shown. These filter options are different from the 'keep-only' and 'remove' controls as these filters are data-driven and present options to filter the data based on the range of attribute values. The filter options are housed

in a menu that is placed outside the right edge of the screen. A draggable handle on the top right of the screen can be used to bring the control onto the screen.

The other component is a table presenting the details of the underlying data being visualized. If no glyphs are selected, the table presents details for all unfiltered data. In case glyphs are selected, the table presents the data for only the selected glyphs. The columns of the table can be sorted in ascending or descending order by tapping on the column header. Tapping on a row highlights the glyphs that correspond to that row. Similar to the filter pane, the data table can be dragged into view using the handle placed at the top right of the view. Alternatively, swipe-up and swipe-down gestures can also be used to summon or dismiss the menu.

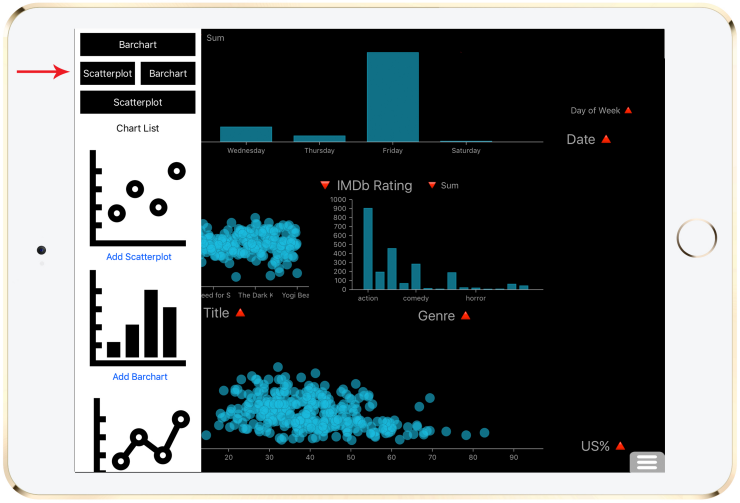


Figure 17: Layout badges in the add-view panel resemble the layout of views added on the canvas. The badges can be swiped left or right to remove the corresponding views from the canvas.

One operation that I have explored in depth in the context of visualizations is selection. Across all the chart types, glyphs can be selected in three ways — directly tapping the glyph, enclosing the glyph within a lasso selection, and selecting a range of values on the axis[11]. But these selection operations provide a basic selection mode where performing new selections overrides existing selections. An advanced state of

[11] A fourth, indirect mode of selection through brushing and linking also exists.

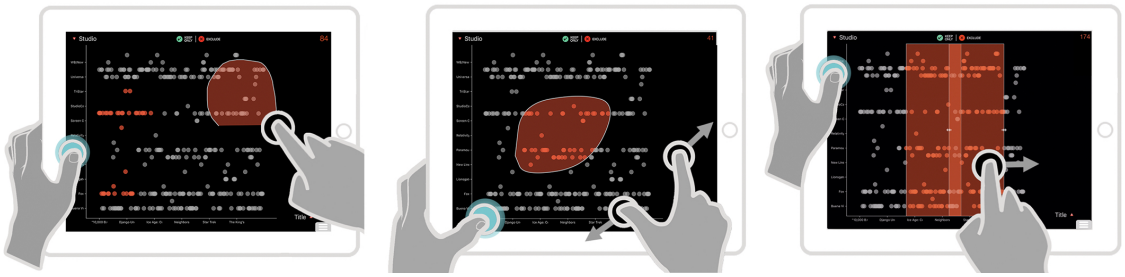


Figure 18: A subset of advanced selection actions. The left thumb clutches while the right hand performs selection operations.

selection, one that permits modes such as add-to-selection and remove-from-selection, also exits within the system. The interactions for these advanced selections are the same as basic selection, but each interaction is preceded by a modifier action called Clutch. Clutch is the action of placing a finger from the non-dominant hand on an edge of the screen (Figure 18).

3.4 System Summary

In summary, Tangere contains four components (Figure 12).

C1. Canvas (primary): The canvas contains the visualizations. Initially empty, it can be populated with one or more views, up to six.

C2. View-gallery (secondary): The gallery shows the available views that can be added to the canvas. It also indicates views already added to the canvas, and allows for removing a view from the canvas.

C3. Tools (secondary): The tools component contains features such as "dynamic query" filter widgets, and options for scaling axes and assigning color.

C4. Details-on-demand (DoD) (secondary): The DoD component presents the underlying data in a tabular form.

The system I described above is a culmination of a series of research explorations. Since tablet-based tools for data analysis were non-existent at the time I commenced

this research, several research challenges existed, stemming primarily from touch as input (Section 1.2). I have addressed these challenges by phasing them as different stages of my research.

Overall, I conducted my research in four stages. Each stage presents solutions to critical portions of the overall problem and provides a platform for the subsequent stages to be addressed. In the first stage of my research, I focus on a single visualization technique and explore it in depth on tablet devices. In the second stage, I extend Tangere to include a broader set of visualization techniques. The key challenge of this stage of research is to identify interactions that are consistent across visualization types. In the third and forth stages of my research, I extend my work to include advanced features and operations, leveraging complex and novel interaction techniques. In the following chapters, I describe the four stages of my research in detail.

CHAPTER IV

SCATTERPLOT

4.1 Scatterplot

For the first stage of Tangere's design, I identified scatterplot as the technique to focus on. A salient feature of a scatterplot's representation is that it displays every data item in the view individually. As a result, scatterplots excel at highlighting clusters, outliers, and trends. Decreasing the size of the glyphs and employing opacity to manage overlap helps achieve a high data density. Small glyph sizes work reasonably well for mouse-based interaction because a precise cursor is able to address and identify individual glyphs. However, this precise resolution is not feasible in the case of finger-based interaction (the so-called "fat finger" problem [56]), which was one of the primary challenges for scatterplots on tablets.

Since I focused on a single technique at this stage, I omitted the view-gallery (C2, Figure 12) component in this stage of the system. To help construct a feature-complete implementation of scatterplots on a tablet, I examined two widely used visualization systems available on desktop computers: Spotfire and Tableau. The purpose of this examination was to identify a set of interactive tasks/operations that support exploration with scatterplots. Analyzing the two systems generated a list of 35 tasks, that I pruned to a more concise set of 9 "primary tasks".

Primary tasks are those that are central to data exploration with scatterplots. Fundamental tasks such as selection and filtering are included, while others which are useful for some analysis but not used as commonly are excluded, such as showing trend lines and highlighting clusters. I also excluded other operations such as "swap variables on axes" that could be achieved using a short set of primary operations.

Table 1: Primary Interaction Tasks and their categorization

Task	**Interaction Intent [137]**	**Categorization**
Assign x and y	Encode	Data-centric, Essential
Assign color	Encode	Data-centric, on-demand
Assign size	Encode	Data-centric, on-demand
Find detail	Abstract/Elaborate	View-driven
Select	Select	View-driven
Zoom	Abstract/Elaborate	View-driven
Filter on points	Filter	View-driven
Filter on values	Filter	Data-centric, on-demand
Change axis scale	Reconfigure	Data-centric, on-demand

The resulting set of primary tasks is shown in Table 1. I categorized each task using the visualization interaction intent framework [137] to ensure that most aspects of interaction are covered and redundancy across the tasks is minimized.

4.1.1 Classifying Tasks

The design space of interactions for the set of primary tasks can follow one of two styles — WIMP-based and gesture-driven. The former consist of elements such as menus, buttons, and toolbars on the touch screens. These elements are familiar and hence easily discoverable to the users, but occupy valuable screen space on the device. Conversely, gesture interactions provide direct access to the objects of interest [60] and free up screen space, but do not support feature discovery or learnability as well as WIMP elements.

Ultimately, a useful solution is one that combines these interaction styles as they are both appropriate for specific category of tasks. The mapping of these styles can be better understood by classifying the tasks into two categories based on their context: data-centric and view-driven.

1. *Data-centric tasks* are motivated by users' need to change (aspects of) the underlying data and are independent of the visualization. Examples of these tasks

are filtering and changing the axis attribute. To perform these tasks, the user typically needs to pick an option from a set of options or a range. For instance, to filter data by an attribute, the user needs to view the entire range of values as well as the currently active range. Similarly, to change the attribute of an axis, the user needs to view all the attributes to pick one. Since these tasks require presentation of options, they are natural candidates for WIMP-style interactions.

2. *View-driven tasks* are motivated by users' desire to interact with the visualization or modify it, and do not affect the underlying data. Zoom, sort, and selection are examples of view-driven tasks. The selection task, for instance, is independent of whether the data attributes being selected are quantitative or nominal. Since these operations do not require a presentation of options, gestures can be used to implement interactions for them.

A subcategorization of data-centric tasks further divides them based on frequency of use into essential and on-demand tasks. This categorization helps position the interactions and their interface elements onto either the main view or in menus & submenus. Essential tasks are frequently used and there is value in making them available to users on the main view. Examples are selection, changing attribute on an axis, and showing data details. Conversely, on-demand tasks are applied by users infrequently and do not need interaction elements on the main view. Examples are changing the axis scale from linear to log and adding jitter to the glyphs.

4.1.2 Design and Implementation

Using the above classification of tasks, I explored the design space of multi-touch operations for each task, examining multiple potential gestures in a prototype interactive scatterplot application for the iPad. I implemented Tangere on an Apple iPad using iOS's Cocoa Touch framework. The application is optimized for a 9.7 inch

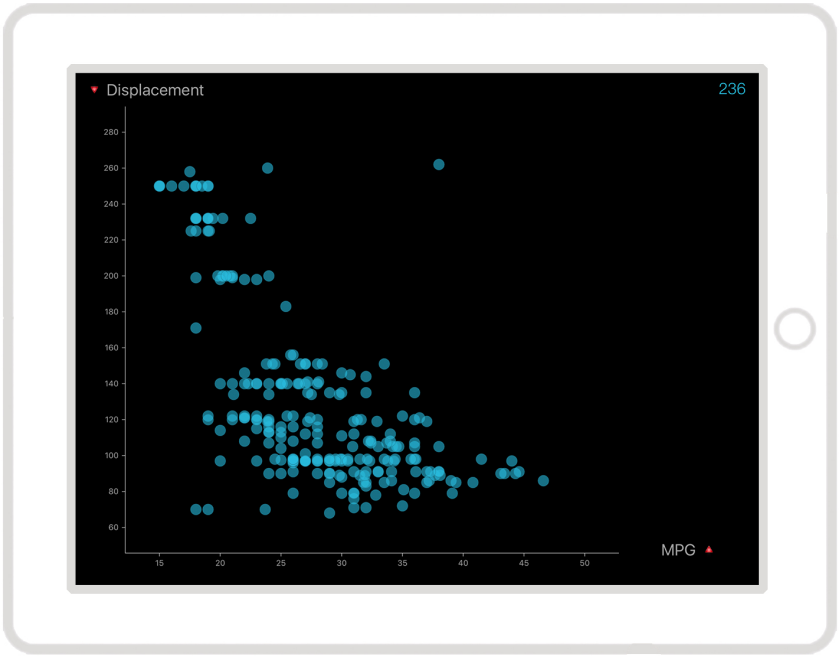


Figure 19: The scatterplot visualization system running on an Apple iPad.

screen with a 2048 x 1536 resolution. The initial view of the application presents a scatterplot using a random pair of variables from the dataset. The main view consists of data points, the two axis lines, and axis labels placed next to each line. The application reads data from a csv file where the first row contains attribute names.

Below I describe the interactions I selected for each of the 9 primary tasks. Figure 20 highlights some of these interactions

1. *Assign x and y-axis variables* - Tapping on a small down arrow icon next to the currently shown attribute displays a scrolling menu of the other variable choices (Figure 20e). The menu also includes a region to the right where a finger slide over a variable generates a preview of that variable's values in the display area.

2. *Assign glyph color and size* - Placing a finger at the right edge and swiping in reveals a control panel with touchable widgets for setting the color and the glyph size.

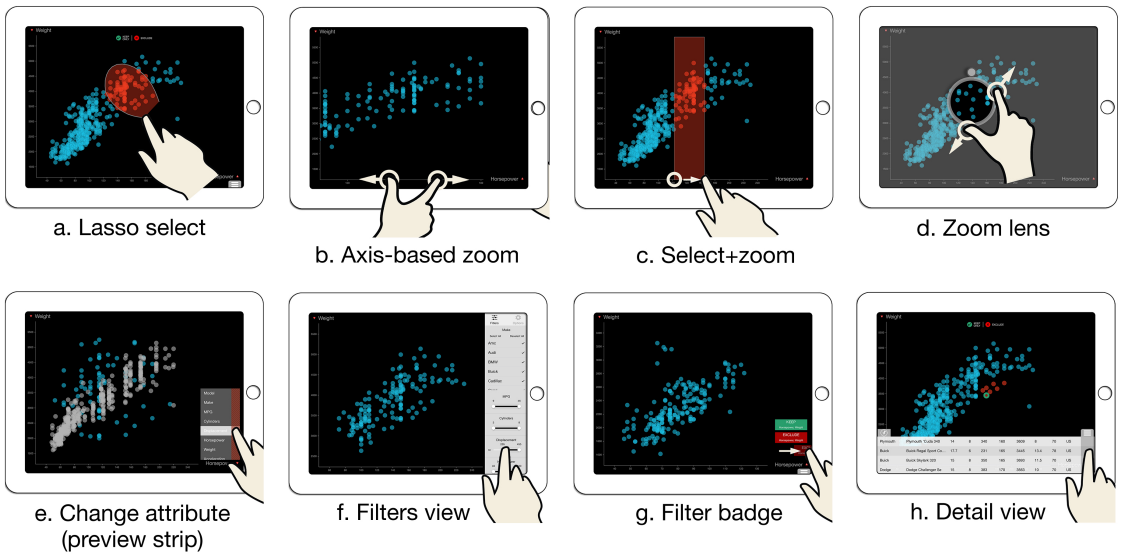


Figure 20: Interaction techniques used for primary tasks in the prototype running on an Apple iPad.

3. *Find data case details* - Swiping upwards on a handle in the lower right region of the view displays a table showing the selected data's attribute details.

4. *Select (data case)* - The application included two alternatives: A) Drawing a "lasso" around a set of data cases (Figure 20a); B) Tapping and dragging a finger to select all the data cases in the highlighted rectangular region (Figure 20c). Both of these methods also can be used when the view has been zoomed in.

5. *Zoom* - The application included four ways to zoom into the data view: A) Placing two fingers on an axis and pinching out or in to zoom in and out, respectively (Figure 20b), B) Highlighting a region along an axis via a tap-drag, then double-tapping in that region to zoom into it, C) A person performs a pinch gesture in the data area which opens a zoom lens on that area (Figure 20d), D) The user double-taps on the data display area and invokes a smart zoom that zooms into the data cases centered in that area.

6. *Filter glyphs* - When a set of data items is selected, two buttons appear toward the top of the view to "Keep only" or "Exclude" the selected items.

7. *Filter on values* - A person swipes their finger in from the right edge to reveal a filters panel with interactive widgets for selecting either quantitative or categorical values (Figure 20f).

8. *Change axis scale* - A person places their finger at the right edge and swipes in to reveal a control panel with touchable widgets for setting the axis scale.

To design these interactions, I explored the research literature to examine how similar operations were implemented in other applications. For a majority of the tasks, I designed and implemented different variations of interactions for performing these tasks and experimented using them within a custom application. Below, I briefly highlight the options that I considered for two task types — selection and zoom. I implemented most of the options in order to gain a "feel" of what they are like in practice.

4.1.2.1 Selection

Selection is a core task in any visualization system. In traditional desktop systems, selection can be realized both through hover and click-to-select actions. A selection task predominantly constitutes the following use cases:

- A user identifies a data point and wants to learn its details.
- A user wants to highlight a point and track it across views, such as a change in axis attributes.
- A user identifies a point and wants to include it in or exclude it from subsequent operations.

Each of these tasks is applicable to a single point or a set of points. Moreover, the set of points can be located close together or at diverse locations in the visualization. In the case of closely located points, the points could be non-overlapping, partially

overlapping, or completely overlapping with each other. On a touch screen, this causes considerable usability problems because large finger sizes do not match well to the typically small size of scatterplot glyphs.

Similar issues of selection on touch screens have been carefully studied in the past. Several solutions, such as enhancing pointer activation areas [81, 12] and offsetting the cursor [96] have been proposed in the past. The breadth of options previously presented suggest that there is no ideal selection technique for touchscreen interaction as there is for cursor-based interactions. The options that I explored for selection on scatterplots in my work were:

S1. Rubber band: User drags on the view to draw a rectangular area containing the point(s) of interest. This is likely faster but less precise than lasso selection.

S2. Zoom view: User taps-and-holds near the data points. This reveals a zoom lens (similar to the iOS text correction view). User taps to select the data point inside the lens and hides the lens by tapping outside.

S3. Swipe+Lens: User swipes on the intended point to select it. If system detects multiple points with the swipe, a lens opens with a magnified view to assist in precise selection [78].

S4. Off-centered pointer: User taps-and-holds the screen to reveal a cursor positioned n-pixels above and to the left of the touch location. The user drags the cursor over the intended point. Dragging over the point reveals details and lifting the finger performs selection [121].

S5. Axis Pan: User slides a finger each on the two axes simultaneously. This creates a horizontal and vertical reference line. The glyph under the intersection of the two lines is selected.

In my experimentation with these options, I identified issues with a number of them. Off-centered pointer (S4) had the limitation that certain positions on the view, such as the bottom edge, were inaccessible because of the way the cursor was placed. Axis pan (S5) necessitated bimanual input that, though feasible on tablets, required the user to first place the tablet on some surface. Zoom view (S2) and swipe+lens (S3) used a zoomed-in lens view. Selection inside a lens created issues for scatterplots since the shapes and colors of the glyphs are often the same. As a result, switching between the lens and non-lens modes caused a loss of target.

From among the list of possibilities, I found lasso (S1) to be most effective for selection since it gives users finer control during selection. To provide the user with feedback while drawing a path, I highlighted the area formed by completing the path between the start and the current point.

4.1.2.2 Zoom

Zooming is another vital operation for visualizations. The operation modifies the viewport to increase clarity of data points that lie too close to each other. Below I highlight some potential ways to perform zooming:

Z1. Pinch-to-zoom: User performs a pinch operation using two fingers. The visualization scales depending on finger movement.

   (a) Fixed aspect ratio: The visualization scales up or down uniformly in both x & y directions.

   (b) Flexible aspect ratio: The visualization scales up or down independently in the x & y directions.

   (c) Critical angle: If the angle between the x-axis and the line that connects the two fingers is less than 45 degrees, the visualization scales on X. If the angle is greater than 45 degrees, the visualization scales on Y.

Z2. Axis-based zoom: Instead of performing a gesture on the view, the user performs the pinch operation directly on the axis that s/he wants to scale.

Z3. Select + zoom: The user highlights a region on the axis or a set of points on the view. The user then double-taps to scale the view to the selected points.

Z4. Zoom lens: The user performs a pinch operation to reveal a lens containing the magnified view of the region between fingers. The user can select the data within the lens and modify its magnification [70].

Z5. Automatic zoom: The user double taps on the view to magnify the region around the touch location. The view is automatically magnified by an amount that minimizes the number of overlapping points in the region.

On touch-based devices, the pinch-to-zoom (P2Z) gesture (Z1) has been employed extensively to perform zoom across applications, such as maps, documents, and so on. The gesture employs a fixed aspect ratio constraint and maintains the aspect ratio of the underlying content being scaled. However, in a scatterplot, the two dimensions, i.e. axes, are largely independent. It is fairly common for the data to be densely packed in such a manner that zooming in only one direction is required. In such situations, a P2Z with fixed-aspect ratio is not as effective.

I modified the P2Z gesture by relaxing the fixed-aspect ratio constraint (Z1.b). However, I found this modified gesture to be fairly difficult to use as it needed fingers to move very precisely in one direction. The other variations to P2Z (Z1.c) had the downside that the default behavior of pinch-to-zoom that users have learned and expect was modified. To counter issues of P2Z applying to both dimensions simultaneously, I introduced Axis-based zoom (Z2) that permitted each axis to be scaled individually using a pinch gesture. Alternatively, the user could tap-and-drag to highlight a range on the axis and select the zoom action (select+zoom: Z3). These

options permit precise zooming into a region or range of values. Since both axis-based zoom and select+zoom were effective for zooming, I supported both in my prototype.

In lieu of the P2Z scaling, I added zoom-lens (Z4) that uses the same P2Z gesture and is particularly useful when users want to view glyphs around a dense area without losing the overall context. The lens supports repositioning, changing zoom level and selection of points inside. Finally, I also incorporated automatic zooming (Z5). On double tapping in a dense area, the system automatically calculates an appropriate zooming amount in order to minimize overlaps among the glyphs in that area.

4.1.3 Evaluation

A natural follow up to this work was a user evaluation of the prototype Tangere system. The primary research question I wanted to address with the evaluation was identifying whether a multi-touch information visualization tablet application, such as a dynamic scatterplot, can be an effective data exploration tool. I explored this question by conducting two user studies.

The first study was a qualitative examination. I demonstrated the features of Tangere to the participants and had them perform a set of data analysis tasks using it. I observed their interactions with the system, recorded any usability issues that arose, and examined their ability to successfully answer analytical questions with the system. I also gathered feedback and suggestions from the participants about the system. I then used these findings to refine the application's interface and multi-touch interactions.

In the second study, I employed the improved application in a direct comparison with an existing desktop scatterplot visualization application using a traditional GUI-style interface with mouse and keyboard. I conducted a within-subjects study in which participants performed a series of data analysis tasks using the two systems. I recorded task correctness, performance time, and errors made. I also carefully

Table 2: Scores from qualitative questions in Study 1.

Question	Average Rating	Min. Rating	Max. Rating
Fluidity of the interface (1: Very clumsy; 7: Very fluid)	6.1	5	7
Learnability of interactions (1: Very difficult to learn; 7: Very easy to learn)	5.5	4	7
Overall rating, on a scale of 7	6.3	6	7

observed how participants performed the tasks to gather further qualitative input about the multitouch interactions that were employed.

4.1.3.1 Study 1: Participants, Tasks, and Results

The goal of the first evaluation was to refine the interactions in the scatterplot technique. I wanted users to perform the different operations in the application and provide qualitative feedback on how well the interactions worked for them. I recruited 11 participants (2 Female) - 10 graduate students and 1 faculty member. All had some experience using visualization system such as Tableau or Spotfire, thus ensuring that their performance was primarily a measure of the quality of the interaction.

During the study, participants were first trained on the various features in Tangere. Subsequently, they performed a set of tasks independently without any assistance. The tasks were based on those used by Drucker et al. [29] in their comparison of WIMP-based and gesture-based interfaces for barcharts on tablets[1]. At the end, they answered subjective questions about their overall impressions of Tangere. They identified the features that they liked in the system along with ones they either did not like, found missing, or considered superfluous. Finally, they rated the system on a likert scale for three dimensions – the fluidity of the system, difficulty of the interactions, and overall rating of the system.

[1]Appendix A.1 presents a snapshot of tasks used for a movies dataset

Results: Participants generally viewed Tangere favorably. On a scale of 1 to 7, participants rated the system **6.3**. Other ratings from the study are presented in Table 2. Some of the qualitative comments provided by the participants include:

> P2: *I think the preview with scrolling was a very good use of the gesture. With touch, you just scroll past something to preview it, which is nice. With mouse, moving and clicking each option is tedious.*
>
> P7 (UX-designer): *While you were giving a demo, at least twice I felt "Woah, that's exactly how I would have built it". That obviously felt great.*
>
> P1: *I felt that the interface was very direct. I can directly choose everything I care about. It's almost the same on desktops, yet somehow this felt direct.*
>
> P6: *The interactions were not obvious in some cases. But I think it wouldn't have taken too long for me to discover them.*

4.1.3.2 Study 2 - Desktop Application Comparison

For the second stage of evaluation, I evaluated the effectiveness of the scatterplot technique within Tangere by comparing it to an existing, well known commercial visualization system – TIBCO Spotfire. The goal was to measure if a Tangere can achieve a comparable level of performance and utility as the desktop system. Spotfire provides a large number of visualization features and controls. For the purpose of the study, I only utilized Spotfire's scatterplot visualization technique.

A total of eight people (four of each gender) participated in the study. The study was implemented as a within-subjects design; each participant used both interfaces and with different datasets for each. The entire study took 75 minutes for the fastest participant and 110 minutes for the slowest. At the end of the second session, participants answered questions regarding their overall impressions of the two systems,

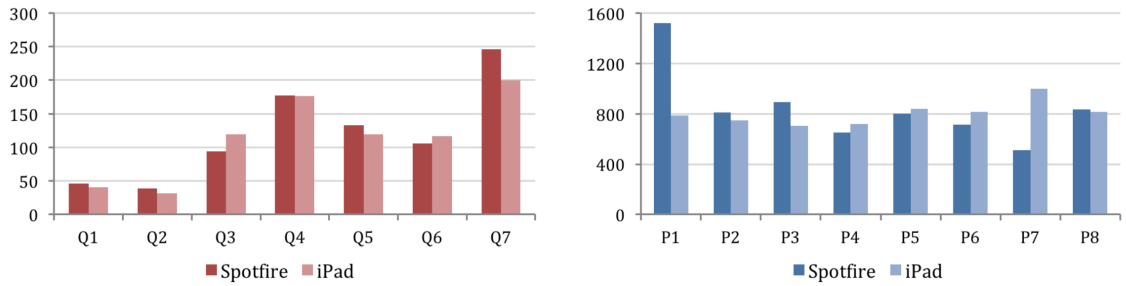


Figure 21: Time taken by participants to perform terms. A) Broken down by each question. B) Broken down by each participant.

differences between the two systems, features that they found better in one over the other, and their preference between the two.

Results: Broadly, I found that participants' performance was similar on the two interfaces. On average, participants took **803** seconds to answer all questions using Tangere and **840** seconds with Spotfire. Figure 21 presents this data, broken down by each question as well as each participant. **Five** of the eight participants expressed that, for a similar set of tasks, they would use the Tangere over Spotfire. Two users expressed preferring Spotfire while one did not have a preference. Below I explore different dimensions of comparison in more detail.

Control Three of the eight participants mentioned that they felt more in control with Tangere. The participants expressed that on the tablet, they felt more connected with the data.

> P5: "*In general, [I] don't prefer touch interfaces, but in this case I felt comfortable and in control, maybe because it was easy to match and associate interactions*"

> P4: "*I am quite surprised at how well it worked and felt. Very different from how I had expected it to be.*"

Expectations Participants expressed that Tangere felt easy to use because many of the features worked in an expected manner.

P4: "*The interactions felt very natural. Even before you showed me a demo of swiping the badge, that was exactly what I was planning to do.*"

Visibility Participants voiced conflicting opinions regarding the visibility of features. Three participants found the Spotfire interface to be intimidating and preferred Tangere's simplicity and aesthetics. Conversely, two participants found dragging in the menu and table in Tangere to be cumbersome and preferred the always-available features of Spotfire.

Precision For slider-based filters, all participants unanimously preferred Spotfire's version to Tangere's. They found Spotfire's slider control to be more precise, particularly the manual text entry that is supported for specifying exact values.

4.1.4 Modifications to Tangere

Although the results of the two studies were encouraging, both studies were admittedly small and relatively informal. It is, thus, difficult to draw many significant quantitative inferences from them. However, both studies uncovered a rich set of qualitative insights about the scatterplot technique within Tangere and the touch gestures provided in it. Below I highlight the changes I made to the system based on my observations from the study and feedback received from the participants.

C1. *Tap-and-pan*: Participants had difficulty using the tap-and-pan gesture and it took them a significant number of attempts to become accustomed to it. Thus, I replaced the tap-and-pan gesture with a direct pan gesture. Panning on either axis activated the rectangular-selection.

C2. *Tap-to-select*: In the study, participants employed the lasso-selection tool extensively and found the continuous feedback of the path's texture to be very beneficial. However, for selecting a single point, participants preferred to simply

tap instead of panning around it. As a result, I implement tap gesture for selecting single points. For taps on densely populated regions, the operation selects the single data item that is closest to the touch location.

C3. *Data-table*: Another requested feature was to have a way to sort each column in the data table. I support this feature by making the column labels interactive. Also, in the previous version of the tool, the data table opened to a fixed height irrespective of the number of selected items. In the current version, I allow the user to drag and control the height to any intermediate level.

C4. *Filter-menu feedback*: Filtering with the "Keep only" or "Exclude" options resulted in a persistent badge depicting the filter action. However, there was no such persistent feedback if filter menu widgets were used and the menu was dismissed. In certain situations, this led to participants beginning a new task without realizing that the data had been filtered. To prevent such situations, I now present a 'Filters On' label on the main view when the menu is hidden.

C5. *Object Count*: Many participants requested a simple way to view the count of selected items. Previously, they had to drag the detail table on the view, and scroll to the bottom to read the index of the last row. I now display a label at the top right of the screen. This label shows the number of selected points or the total number of points, based on whether points are selected or not.

Rectangular Selection The rectangular selection feature had a drawback 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.

To counter this issue, I added two components (Figure 22a) — a label to both

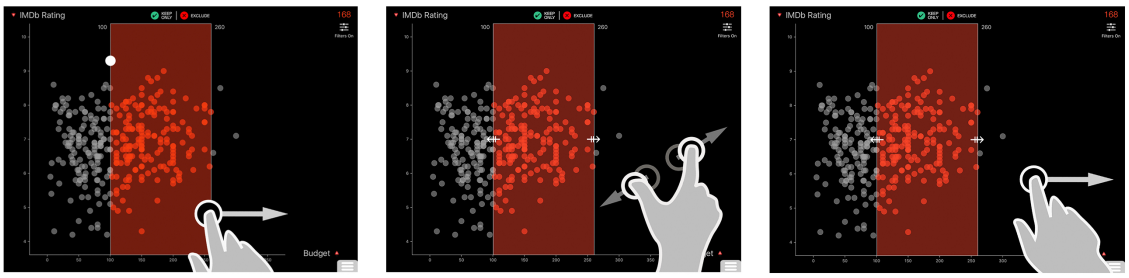


Figure 22: The evolution of the edit-selection feature. a) The selection rectangle with circular handles. The handles can be dragged to edit the edges. b) Each finger in a pinch gesture controls the movement of an edge of the selection rectangle. Here, movement of the fingers in x-direction controls the edges. c) The entire space adjacent to an edge outside the rectangle controls the edge.

ends of the selection rectangle displaying the range of the selection and thumbs on the edges of the rectangle to allow editing the range of a selection. 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, I further iterated on the design of this interaction and implemented two techniques to edit the selection in this new configuration:

1. Pinch to edit (Figure 22b) - Since the rectangle has two manipulatable edges, the movement of each finger in a pinch operation is mapped 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 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 22c) - To edit the position of an edge, the entire area adjacent to the edge outside the rectangle is used 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.

CHAPTER V

MULTI-COORDINATED VIEWS

5.1 Multi-Coordinated Views

Designing the scatterplot technique gave a glimpse into a rich interaction space. Positive user response to the system also created a platform to further the work by incorporating more visualization techniques into the system. Every visualization technique serves a special role in data analysis and tools rarely exist using a single isolated technique. Adding more techniques to Tangere was, thus, a natural next step.

To achieve this goal, the system needed to provide people the flexibility to use one or more visualization techniques in a coordinated configuration. For these configurations, identifying the layout of the workspace is critical as the layout plays a central role in knowledge discovery and efficiency within a multi-coordinated view (MCV0 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. With respect to layout issues, I identified four goals for MCV systems on tablets:

G1. Maximize the size of each visualization on the canvas

G2. Minimize occlusion of visualizations by tools or fingers

G3. Strive to keep all visualizations in view (prevent scroll, pagination, or tabs)

G4. 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. 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 the objective of targeting novice and non-expert users.

5.1.1 Visualization Techniques

All MCV systems provide multiple visualization techniques for presenting different perspectives on data. To select the other techniques to include in Tangere, I considered them along three dimensions: rich support for interactive operations, familiarity to users, and potential for cross-vis interactions with brushing and linking. I ultimately identified three techniques that offered rich variation in terms of representation and interaction:

1. *Barchart*: A canonical visualization technique that has been used extensively across a wide variety of domains. Barcharts are very effective at representing a categorical variable on one axis and a quantitative variable on the other.

2. *Linechart*: Another canonical, widely used technique. Linecharts excel at representing a quantitative variable on one axis and an ordinal, ideally temporal, variable on the other.

3. *Parallel coordinates*: A well-known and an often used technique that uses parallel axes and polylines for displaying multivariate data. A defining property of this representation is that it encapsulates all attributes of every data-item in a single view.

The techniques typically represent the same general type of data — combinations of categorical and quantitative attributes. It was necessary to include visualizations built for the same type of data since I intended 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, I 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 [1]) 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 the system.

Figure 23: The Tangere system interface. The panels are placed beyond the screen and can be brought in with a gesture.

5.1.2 Layout of Visualizations in the Canvas View

To maximize the size of each visualization, the canvas view spans the entire screen with the secondary views placed off-screen. Since the view contained multiple techniques, the view-gallery (C2) component is also present in the system. A further consideration was identifying the layout of views within the canvas. These layout

[1] Roambi, www.roambi.com

configurations can be classified into four styles - *juxtaposition*, *superimposition*, *overloading*, and *nesting* [62].

With the layout design goals in mind, I utilized 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. Within juxtaposition, two alternative layout styles are possible — vertical stacking and grid-based layouts. While grid-layouts result in compact side-by-side views, they also require end-user controls for adjusting the width and height of the views, which conflicts with goal G4 above. Thus, I constrained the system to a vertical layout.

When the system starts, the canvas is empty. The view-gallery is initially open and contains the 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 23). To remove the chart, the user can slide this badge horizontally towards the left or right. As the corresponding chart exits the view, the other charts expand to fill up the remaining space.

The number of charts that can be added to the canvas is constrained to three in order to ensure that all charts are always visible (goal G3). Given the space constraints, moving beyond three charts on the canvas made each chart too small to be useful. Furthermore, this number is constrained 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.

5.1.3 Interactions within Views

All four visualization techniques support rich and varied interactive operations, several of which 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, are 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. Consistency is critical as it feeds directly into the usability and learnability of the system, with inconsistent interactions adversely affecting user performance [124, 129].

Table 1 presents the operations common to all four visualization techniques and the interactions employed to implement them. Achieving cross-visualization compatibility was challenging. 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 [29]. For scatterplots, however, this gesture was used for a different operation — axis-based selection [104]. In fact, I adapted this interaction on linecharts and parallel coordinates as well.

To ensure barcharts are consistent with other charts, I did not use the axis-swipe gesture on barcharts for sorting, and instead used axis-swipe to perform the same operation as in other charts — selecting items in a range. For sorting, I employed the long-press + swipe gesture — the user places a finger on an axis and holds for a short period of time before swiping in the direction of sort. I selected this 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 [104]. Tapping on a scatterplot is complex since it results in ambiguous selections when tapping in a

dense area. However, for consistency I extended tap-to-select to scatterplots as well as linecharts and parallel coordinates.

5.1.4 Interaction Across Views

Brushing and linking features are crucial to effective MCV applications. In Tangere, 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 glyphs 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 [89].

5.1.5 Handling Occlusion

A significant number of design decisions made during the prototyping phase were directly impacted by issues of occlusion. For example, even though the expected location of labels in a parallel coordinates plot is at the top, I placed them at the bottom since any interaction with them would otherwise occlude the entire view. For other occlusion-related issues I encountered, I examined the utility of the solutions previously presented in HCI literature[121, 127], but found their applicability for Tangere to be limited. This raised the question whether occlusion in visualization interfaces is different from occlusion in regular, non-visualization user interfaces. I speculate that the answer to this question is yes. I explain this below using the ideas of the instrumental-interaction model [9].

Low degree of indirection In the instrumental interaction model, Beaudouin-Lafon introduces the concept of domain objects and instruments [9]. 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.

Most gestures described above have a low degree of indirection, which I speculate is largely due to the underlying domain and not just a result of the design decisions I took. The same observation can made for the interactions used in TouchWave [8], or the gesture-based interface by Drucker et al [29]. Information visualization operations lend themselves appropriately for direct interactions with the glyphs, with the instrument and the domain-object tightly coupled with each other. Unfortunately, this also means that for a majority of gestures, the instrument (a finger) overlaps with the domain-object, thus occluding it. Since the movement of the fingers are highly dependent on the underlying glyphs being manipulated, the gestures becomes very inefficient.

Many solutions for countering these types of issues suggest translating the obstructed views to locations that are not hidden [121, 127]. 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.

Occlusion-aware continuous gestures As I encountered issues of occlusion, I developed strategies for modifying interactions to counter them. 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. With

these gestures, fingers stay on the view for longer than discrete gestures, increasing the possibility of occluding views. To minimize this occlusion, I employed two strategies.

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

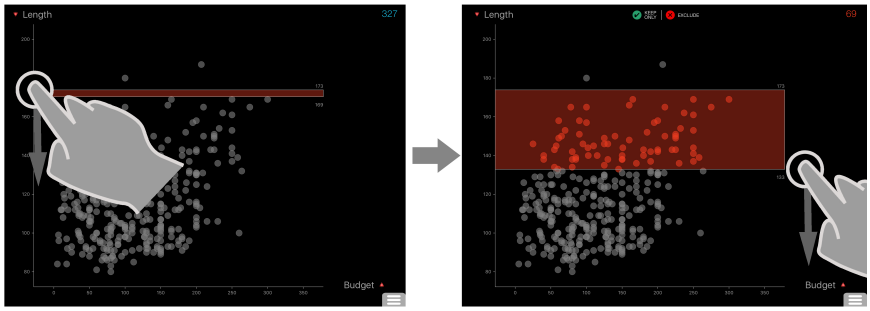


Figure 24: User begins a drag gesture on the y-axis to begin selection and subsequently moves away from the axis while continuing to control the selection.

   - Pan gesture (Figure 24): 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, 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 25): I 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

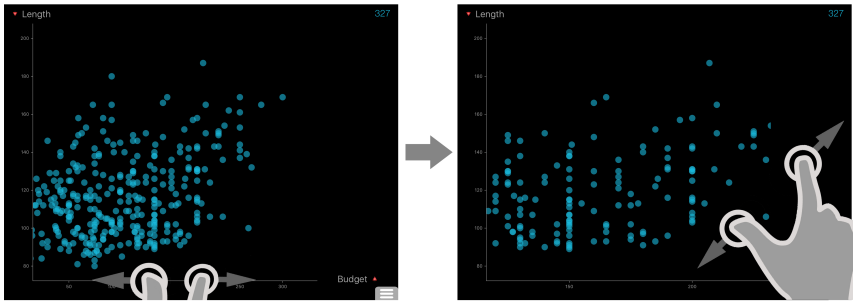


Figure 25: User begins a two finger pinch on the x-axis and then moves away. Notice that user's fingers do not need to stay horizontal.

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, I 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 [76]. Both the *pinch to edit* and *exterior pan to edit* operations I described earlier (Section 4.1.4) 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. In future work, it is important to revisit the topic within the context of tablets and small displays.

CHAPTER VI

ADVANCED SELECTION

6.1 Advanced Selection

Selection is a fundamental operation in interactive visualization applications [137]. It is a vital intermediary step for several important tasks such as filtering, brushing, and details-on-demand. As a result, having effective methods to specify selections to the system is central to any visualization application.

While basic techniques for selection such as clicking (or tapping) and lassoing (dragging a curved boundary around a set of items) are robust and employed across domains, making a selection often requires expressing more complex and detailed queries to the system than is possible with these basic techniques. A common example is the group-selection set of operations—add to selection, remove from selection, intersect with selection, replace selection, and toggle selection [132]. Wills [132] perhaps best captures the complexity of the selection operation in the domain of data visualization. Another style of complex selection, often represented with the query "select items like these," is generalized selection [46].

Performing complex selections such as these often requires the use of specialized instruments in a system. On desktop computers, such aids could be the modifier keys (such as Ctrl, Shift, and Alt) or modes (which applications such as Photoshop employ extensively). Desktop-based visualization tools such as Tableau and Spotfire, consequently, extend the use of these instruments to visualizations, thus making complex selections possible across different visualization techniques.

In Tangere, I thus far omitted methods to perform complex selections. In fact, they are noticeably absent across other tablet-based visualization systems as well,

such as Kinetica [102], Vizable [1], and TouchWave [8]. One potential reason for such an absence is the difficulty of identifying appropriate interactions for performing the operations. These systems predominantly restrict the interaction set to only the standard vocabulary of touch gestures — a combination of tap, pan, pinch, and rotate, extending it to only include gestures such as tap-and-pan and long-press. However, advanced selection operations are a cornerstone of interaction within visualization systems.

Thus, to support these operations, I addressed the constraints introduced by a limited vocabulary of interactive gestures and introduced a new approach for providing more powerful selection capabilities within Tangere.

6.1.1 Techniques for Advanced Selection

The advanced selection operations that I target in this work can be classified into three categories based on their behavior:

- Modify existing selection
- Add to selection
- Duplicate selection

The relevance of any of these categories depends on the state of the existing selection within a system. The state captures characteristics such as the type of selection (view-based vs data-driven) and the number of selected glyphs. Depending on the state, one or more of the categories may be applicable.

To integrate these three categories of selection into a system, one needs to identify appropriate interactions for each. Instead of adapting entirely unique and complex interactions, I favored extending the vocabulary via incremental modifications to existing gestures. I achieved this by leveraging the concepts of quasimodes for tablets.

[1] Roambi, vizable.tableau.com

Quasimodes on Tablets Quasimodes [97], or *transient modes*, are temporary states a system enters wherein all user actions pertain to a specific category of tasks. Unlike persistent modes that stay in effect until cancelled or changed, quasimodes stay in effect only as long as the user maintains the action required to activate the mode. On desktops, most common examples of such actions are modifier keys, such as Shift, Ctrl, and Alt. Quasimodes provide an easy and reliable way to return to the default application state and have shown to significantly reduce mode errors as compared to persistent modes [109].

I sought to replicate the behavior of the desktop-based quasimodes on tablets. The key concept here is that by introducing a unique modifier action to activate the quasimode, I am able to employ a standard gesture in the execution stage of the quasimode. The use of the modifier, however, differentiates the standard gesture performed without the modifier from the gesture with the modifier.

6.1.2 The Clutch Modifier Technique for Tablets

Because tablets do not have keyboards with shift and control keys, a different modifier is needed. I leverage the use of a person's non-dominant hand in this work. On tablet devices, the non-dominant hand is often restricted to holding the tablet. As a result, the range of movement available to the non-dominant hand is severely constrained. Based on the way the tablet is held, people can only perform basic tap, hold, and drag gestures at the edges of the screen using either the thumb or the fingers.

Although these gestures are inadequate for performing complex tasks, they are sufficient for behaving as modifier actions. Since modifiers only act as precursors to the primary action, only a single bit of information is needed—whether the modifier was performed or not. The tap, hold, and drag gestures can adequately provide this information.

Employing the non-dominant hand for modifier actions has several benefits. First,

it conforms with the division of labor model—the role of the non-dominant hand is that of framing the detailed action that the dominant hand performs. Second, shifting the modifier to the non-dominant hand relieves the dominant hand from needing to perform a complex sequence of actions. Finally, the non-dominant hand provides additional, albeit limited, degrees of freedom that can potentially be leveraged to increase the expressiveness of the interaction.

I define ***Clutch*** as the action of the non-dominant hand performing a hold (or long-press) gesture on the screen. The action can be performed by placing and holding a finger within a 100 pixels wide zone running on all four edges of the screen. The Clutch action is feasible for people holding tablets in any grip or configuration [122]. When a Clutch is detected, a bright blue halo image placed beneath the clutching finger provides visual feedback denoting the change in mode.

I added the Clutch action to the Tangere system described earlier. Using Clutch as a modifier, I reused all the primary gestures originally employed in the system. This provided the opportunity to introduce additional new operations. It was important to carefully select the operations to support with Clutch since using gestures with a common modifier for disparate tasks had the potential to be confusing to users. Thus, I applied Clutch-modified gestures to a single category of operations: *advanced*

Table 3: Summary of the advanced selection operations and their corresponding gestures. The selection operations are add-to-selection (ATS), modify existing selection (MES) and duplicate selection (DS)

Operation	**Type**	**Gesture**
Select/Deselect Single Glyph	ATS	Clutch + Tap
Select Neighboring Glyph(s)	ATS	Clutch + Drag
Precisely Select Neighboring Glyph	MES	Clutch + Drag + Pinch
Duplicate Rectangular Selection	DS	Clutch + Drag
Add Another Rectangular Selection	ATS	Clutch + Drag
Increase/Decrease Selection	MES	Clutch + Pinch

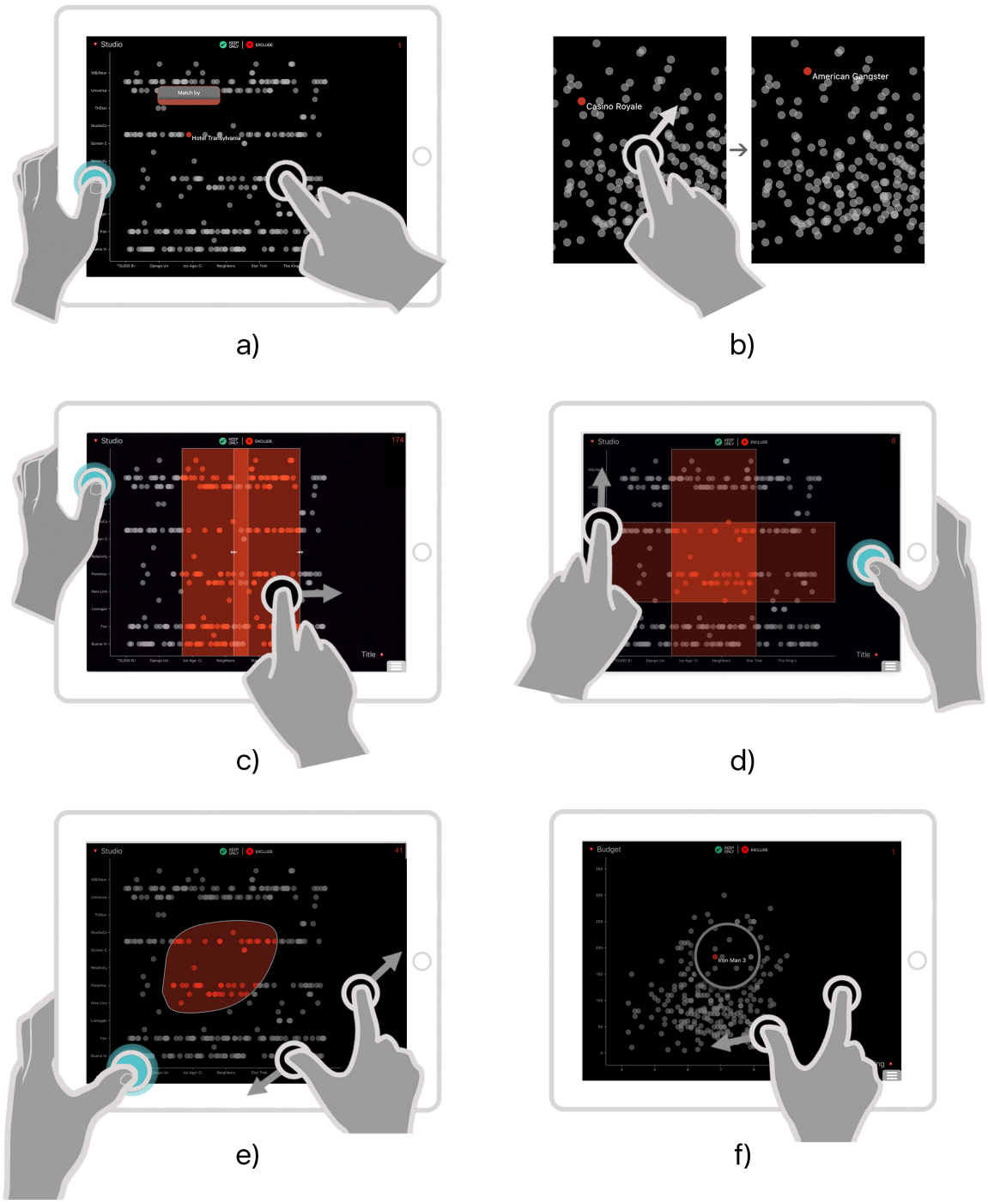


Figure 26: Clutch-modified gestures and their mapping to operations. The clutching action is highlighted using a blue halo. a) Clutch + tap gesture activates the generalized selection operation. b) Clutch + drag for single selections scrubs the selection on neighboring points. c) Clutch + drag inside a rectangular selection creates a duplicate. d) Clutch + drag on the axis creates a new rectangular selection that intersects with existing selections. Notice the color difference in selected and non-selected portions. e) Clutch + pinch for multiple glyphs grows or shrinks the are of selection. f) Clutch + drag + pinch for single selections reveals a lens with zoomed in region.

selection. Table 3 summarizes the specific operations and the gesture used to invoke each.

6.1.2.1 Clutch + Tap

The Clutch + tap gesture selects individual glyphs located at the position of the tap (Figure 26a). Whereas non-clutched tap selects a new glyph and deselects all previously selected glyphs, clutched tap adds glyphs to the selection without deselecting other glyphs. In case the tapped glyph is already selected, clutched tap deselects that glyph without affecting the other selected glyphs.

6.1.2.2 Clutch + Drag

Clutch + drag activates different operations depending on the state of selection in the system.

Single glyph selected — In this case, Clutch + drag provides a method for rapidly alternating through glyphs located in a dense area (Figure 26b). The technique works in the following manner: with one glyph selected, the user activates the Clutch and begins dragging a finger on the view. The system detects this movement and projects it forward to identify a target glyph that is nearest to the selected glyph in the direction of the movement. Once identified, the system holds for the user to move the finger by the same amount as the distance between the two glyphs. When the magnitude of the movement exceeds the distance, the selection snaps to the other glyph, deselecting the original glyph (Figure 26b). The overall effect of the operation resembles that of using a trackpad wherein user's movements map to the movement of a cursor.

Multiple glyphs selected — When multiple glyphs are selected, the Clutch + drag gesture draws a lasso selection path and select all glyphs that lie within the path.

However, unlike the standard lasso, this gesture does not deselect the glyphs already selected. In other words, the gesture strictly provides the add-to-selection operation and not remove-from-selection.

Rectangular selection— If a view contains an active selection rectangle on any axis, dragging inside the selection with Clutch activated moves a copy of the selection along the axis, thereby replicating the selection. (Figure 26c). The glyphs that lie within either of the rectangles (i.e. union of the area) are selected. Tapping outside dismisses both the selections.

Clutch + drag on the axis — The Clutch + drag gesture performed directly on an axis creates a new rectangle selection without dismissing existing selections (Figure 26d). If a new rectangle is created on the same axis as an existing rectangle, the union of the two is used for selecting the glyphs. For the case when the new rectangle is on the other axis, the intersection operation is used and only those glyphs that belong to both the rectangle areas are selected. Color variation is used for differentiating the selected region from the unselected region (Figure 26d).

6.1.2.3 Clutch + Pinch

The Clutch + pinch gesture is utilized for increasing and decreasing the size of the selection area. If the view contains lasso selected glyphs, the Clutch + pinch gesture increases or decreases the selection by scaling the size of a convex hull that encloses all the selected glyphs (Figure 26e). If, instead, the view contains rectangle selections, Clutch + pinch scales the rectangle by mapping to the ends of the active rectangle to the movement of two fingers.

6.1.2.4 Clutch + Drag + Pinch

The Clutch + drag + pinch gesture is a special configuration of the Clutch + drag gesture, where the user begins by dragging a finger and, without lifting that finger, performs a pinch gesture using another finger. This gesture augments the single glyph selection operation presented in section 6.1.2.2. Executing a pinch in a densely packed region reveals a lens at that location with the region zoomed in (Figure 26f). With the second finger lifted, dragging the first finger cycles through the glyphs in the scaled up view. The user can close the lens by either pinching-in or lifting all the fingers.

6.1.2.5 Relaxing Clutch

In the current implementation of the Clutch-modified gestures, it is unnecessary to keep the Clutch active for the entire duration since the Clutch info is only utilized at the start of the gestures. I thus utilize a low-tension modifier [52]—users can lift the Clutch immediately after they begin the gesture with their primary hand (Figure 26f). This is similar to the actuator-continuous configuration for single-stroke gestures presented earlier. The low-tension modifier significantly reduces the discomfort that may occur in manipulating the screen with both hands simultaneously, while also supporting the tablet with one hand.

6.1.3 Generalized Selection

A different type of advanced selection operation is *generalized selection*. Generalized selection predominantly operates on the data-domain, differentiating it from the view-driven advanced selections presented so far. Here, the user identifies a target using a visual property, e.g. position or color, and subsequently intends to select other items that are similar to this item based on certain attributes of the data. For example, in a scatterplot showing *rating* vs *profit* of movies, the user selects the highest rated movie and wants to know if the other movies by the same director have done equally

well.

Heer et al. [46] described it as a "select objects like this" query, and presented a method to perform it in scatterplots on desktops. In their system, users invoke a context menu on a selected glyph and pick an attribute from a list. This selects all other glyphs in the view with the same value for the attribute as the originally selected glyph. Generalizing a selection in the absence of a specialized method such as context-menu is fairly cumbersome and involves several steps, such as switching attributes and introducing additional views. Each change of view alters the context of the application for the user, which is undesirable. Thus, I developed a specialized method to perform generalized selection in Tangere. I achieve this by designing a novel interaction technique that provides fluid access to attributes of data. The implementation expands the operations originally presented in [46] by applying it to other attribute types and adding methods for users to control the parameters of the selection.

Generalizing a selection consists of two steps: the user begins by selecting a glyph and subsequently specifies the generalization criteria to the system. The outcome of generalizing a selection varies based on the size of the existing selection and the attribute used for generalizing it. For example, for categorical attributes, the selection expands to include all data points that have the same value for the attribute as the selected glyph. (e.g., director = 'Christopher Nolan'). For attributes with hierarchical properties, selection expands to the data points that match the value of the selected glyph at the lowest level of the hierarchy. Finally, for quantitative attributes, the selection expands to data points that fall within a neighborhood around the value of the selected glyph. (e.g., profit $\in$ [90, 110]).

Design of Generalized Selection Technique For the generalized selection operation, the solution consists of a list control that appears in the periphery of the

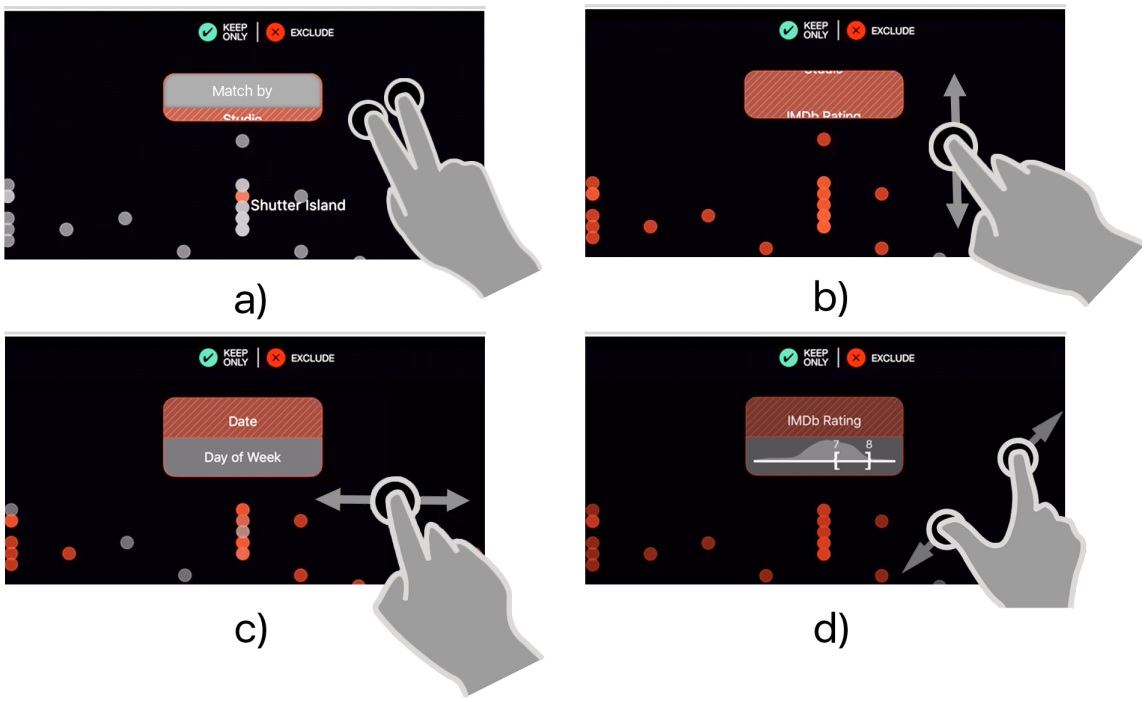


Figure 27: Generalized selection. a) User taps on the screen with two fingers to activate the generalized selection menu. b) To scroll through the list of attributes, user drags a finger vertically. c) For hierarchical attributes, user drags horizontally to access other levels of hierarchy. d) For quantitative attributes, the scented widget displays the selected and overall range of selection. Using a two finger pinch, user can modify the extents of the selection.

selected data glyph(s). The control is hidden by default, and can be introduced on demand. To circumvent the need for a persistent UI element to initialize the operation, I utilized a gesture — *two-finger tap*. If a chart contains one or more selected glyphs, a two-finger tap gesture performed anywhere within the bounds of the chart activates the operation and presents the control.

The control for the operation consists of a single label that, initially, gives a description of the operation (Figure 27a). In the background, the control contains a vertically scrollable list of options, the majority of which remains hidden from the user. To interact with the control and select a different option, I emulate trackpad input—the user can place her finger anywhere on the screen and simply drag vertically up or down to scroll through the list of options (Figure 27b). As each option appears

within the visible bounds of the control, the glyphs in the chart update to reflect the selection that is generalized with respect to the attribute the option represents. When the user identifies a suitable option, she commits the selection by simply tapping outside the list.Alternatively, to cancel the operation and return to the original state, she can either scroll to the first item in the list or simply perform a two-finger tap gesture away from the list.

The second step in generalizing a selection is controlling the parameters of the chosen attribute, which are dependent on the type of the attribute. The options for modifying the parameters are displayed in a separate control that appears below the original control (Figure 27c). This additional control only appears when the user halts on a particular option for more than 500 ms (i.e., scrolls to an option and lifts her finger). For attributes that have hierarchical properties, the secondary control presents the level of hierarchy (beginning with the lowest) that is being used for expanding the selection. The other levels appear as columns of a horizontally scrollable list that the user can switch to by swiping left or right (Figure 27c).

For quantitative attributes, the secondary control contains a slider bar that depicts the range of values of the attribute used for selecting other data points (Figure 27d). For single glyph selections, I use a 10 percent threshold around the value of the attribute. For multiple glyphs, the range extends from the minimum to the maximum values of the attribute for the selected glyphs. Further, the widget is embellished with the distribution of the values for the attribute using the scented-widgets technique [131]. The user can extend the range of selection by manipulating the position of the handles using a two-finger pinch operation (Figure 27d).

CHAPTER VII

ADVANCED LAYOUT TECHNIQUES

7.1 Introduction

Tying the design of Tangere to the goal of simplicity has provided several benefits. For example, Tangere supports only a limited set of critical operations for each technique. The restricted nature of this set has made it feasible to employ the same interactions for operations common to the different techniques. This is very useful for users' onboarding and overall experience with the system. But simplicity has also resulted in features that may be restrictive to users' workflow, providing less flexibility and freedom than they expect or need. One such feature is the layout of the views. Tangere initially constrained the views to be stacked and permits only three views to appear. However, several other layout styles are feasible that can assist the user in being more effective or productive with the interface.

In this chapter, I explore customizable, non-static layouts for visualizations that can be adapted to tablets. These advanced layouts are accompanied by operations for modifying the positions and sizes of views. Such features are currently absent from any tablet-based visualization application.

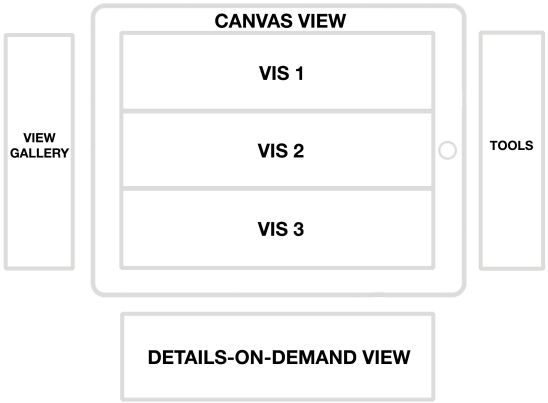


Figure 28: Components of Tangere.

Tangere initially used a simplistic, juxtaposition layout style (Figure 28). Charts were stacked vertically; new charts only appeared at the lowest position; and only three charts can be added to the view at one time. The reasons for supporting such a constrained layout were twofold. As discussed earlier, key themes for Tangere were targeting novice users and ensuring simplicity. Employing a simple layout model helped this cause by minimizing complexity. The second main reason was a restricted gesture vocabulary. Given the standard suite of gestures that I employed in Tangere, there were only a limited number of interactions available to use for layout modification tasks. A complex layout mechanism would inadvertently require advanced and more expressive interactions than were used within the application.

While these restrictions continue to hold true, well-designed layout techniques also hold the potential to vastly amplify user-performance and efficiency. And this increase in efficiency can ably compensate for the added complexity. Further, advanced interaction techniques, such as Clutch-based gestures that I described earlier, are now also available, and these offer possible interaction alternatives that can be repurposed for layout operations.

This opened a possibility for exploring two topics — 1) Identifying an appropriate layout technique to adapt to tablet devices, and 2) Integrating the layout in Tangere and designing interactions for manipulating it. Although the design exercise needed for the second topic plays a critical role, the overall solution is also highly pertinent on finding an effective layout technique for the first topic. In the following sections, I explore the layout techniques that are commonly adopted in visualization systems. Subsequently, I present the one that I incorporated within Tangere along with the design guidelines that led to the decisions.

7.1.1 Layout Options

Multiple coordinated views gained prominence as combining visualizations effectively led to new ways of representing data at a time when novel visual representations had become difficult to design. The premise of the technique was that users understand their data better if they view it and interact with it through different representations.

Making interactive, multiple visualization systems that are usable is a difficult challenge, due to the limited screen size that is typically available. Layout, thus, plays a central role in enabling the user in understanding the data. Key characteristics of a layout are the associated spatial and temporal costs of accessing and coordinating multiple views and the cognitive effort required to perceive relationship between views [62]. To balance and minimize these costs, a range of layout techniques can be implemented that permit the user to perform their tasks better. It is notable that this exploration of layout styles for MCVs on tablets resembles the earlier research on visualization layouts on desktops, and preceding that, on windowing systems for desktops.

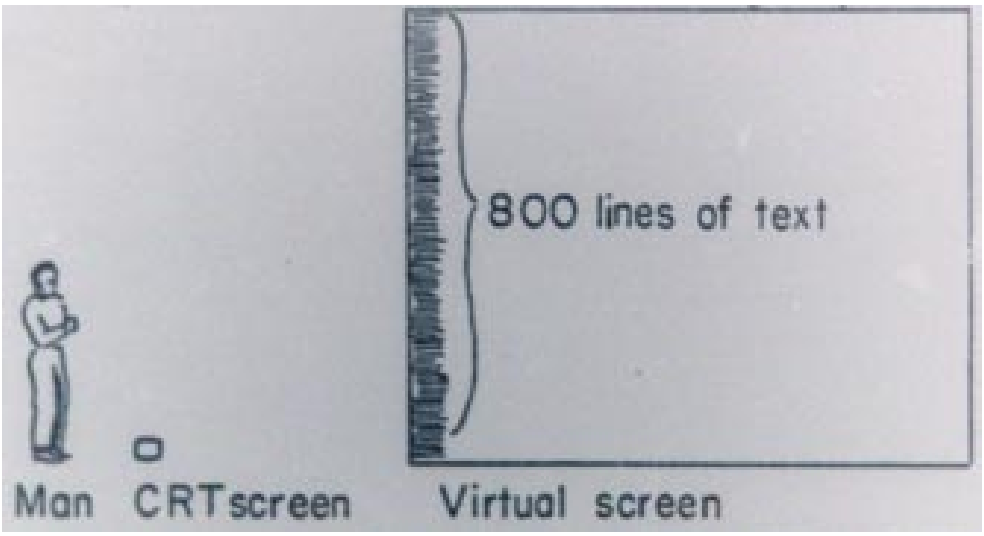


Figure 29: Desktop metaphor proposed by Alan Kay [67].

7.1.1.1 Desktop Window Management

Early versions of desktops consisted of a single window. With the introduction of window managers, separate activities could be put in physically separate parts (windows) of the computer screen. Early window managers followed either a "desktop" metaphor (proposed by Alan Kay [67], Figure 29), allowing windows to overlap each other like sheets of paper piled up on a physical desk, or used a "tiling" model (demonstrated by Engelbart [33]), arranging windows so that no overlap occurs.

The overlapping 'desktop' model allows for more freedom. Users can rearrange the windows, change the size, location and/or degree of overlap. Windows can also extend partially off screen so that only part of the window is visible on the screen. However, the desktop can become messy when the number of windows is large, and managing the windows may require effort.

In tiling systems, the system is typically in charge of managing the window placement and size. Whenever a new window is created or an old window destroyed, the system adjusts the sizes of windows based on constraints. The constraints and tiling styles can vary and windows can appear in single-column, multi-column, hierarchically, or non-hierarchically [22]. The downsides of tiling are that it takes a while for windows to finish adjusting after a change. For each window added or removed,

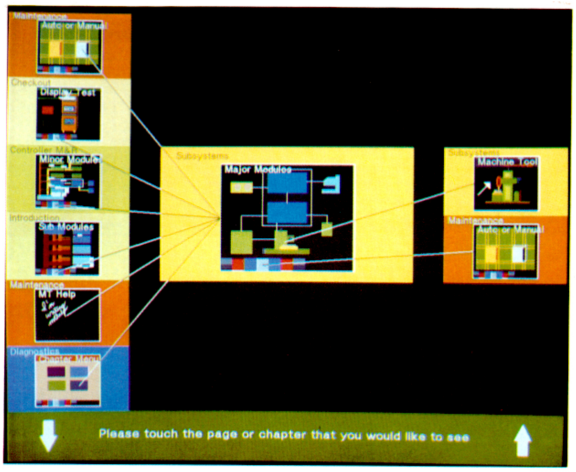


Figure 30: Digital 'book'-like interface [35]

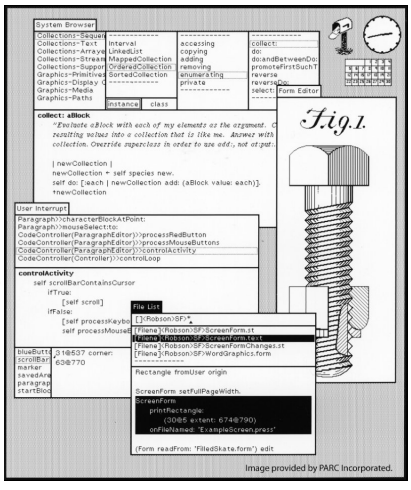


Figure 31: Smalltalk [39]

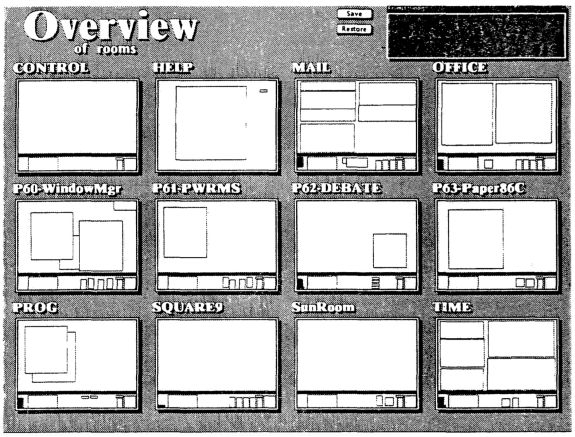


Figure 32: Rooms interface [49]

windows all over the screen need to readjust.

Bly and Rosenberg [13] compared user performance of tiled and overlapping window strategies. Their results supported tiled windows for regular tasks. For irregular tasks, however, expert performance was faster in overlapping windows, whereas novice performance was faster in tiled windows. However, by the mid-1980's, Unix and MacOS, the two main operating systems, converged on the desktop metaphor with overlapped windows [82]. Microsoft's original window managers were tiled, but eventually transitioned to the overlapping style.

A different, and popular, approach was to group windows based on tasks that were likely to be used together. One early example used a book-like metaphor for arranging projects into chapters [35] (Figure 30), while another provided hierarchical nesting of project spaces for Smalltalk [39] (Figure 31). Henderson and Card [49] presented the Rooms virtual desktop system (Figure 32). Workspaces (or rooms) were created containing windows of similar tasks. Users could transition to other rooms using *doors*. However, the main goal was to prioritize interaction between windows of the same workspace by providing mechanism for easily switching between them.

Other windowing approaches were also explored. Beaudouin-Lafon [10] presented the "tabbed windows" interactions for desktops where arbitrarily many standard windows could appear, and each was assigned a tab. Robertson et al. [99] presented the

Task Gallery (Figure 33), a 3D window manager where windows appears as artwork hung on the walls of a virtual art gallery, with the selected task on a stage. Others also explored the 3D desktop metaphor [2, 25] (Figure 34), but the metaphor did not garner mainstream acceptance, and studies found 3D desktops to be less efficient and more confusing than 2D desktops [21].

Rather than scale down windows, an alternative approach was also to scale up the workspace [1]. Rather than scale down the windows, these systems provide a large virtual workspace that could be panned (dragged), such that the user may arrange windows over a continuous area much larger than his monitor. In some cases, these systems let the user zoom out to see an overview of his larger workspace.

7.1.1.2 Layouts in Visualization Systems

While a host of desktop-based visualization systems employ MCVs, including Tableau and Spotfire, there is surprisingly little work on guidelines and principles for layout management in visualization systems. Early work by North & Shneiderman [89] mostly provided a case for using coordinated views in visualization systems. Later, Baldonado et al. [124] presented eight principles for both identifying when to use MCVs and how to use them. However, their usage principles aim primarily at managing perceptual and cognitive capabilities, such as rules of Space/Time Resource

[1] Beryl Linux, www.beryl-project.org

Figure 33: TaskGallery [99]

Figure 34: Bumptop [2]

Optimization, Consistency, and Self-Evidence.

One relevant work that explored the different layout styles was a survey on composite visualizations conducted by Javed and Elmqvist [62]. The authors presented four broad styles of layout that visualization tools employ.

1. *Juxtaposition*[2] - The most common and flexible design pattern is based on the tiled layout, with views either stacked, placed side-by-side, or both. The variation may exist in whether views are placed within the bounds of the screen or extend outside the screen. The layout manipulation can be manual or automated, though generally applications provide techniques to modify some aspects of the layout.

2. *Superimposition*[3] - This flexible layout also contains arbitrarily sized views that can, again, be placed within or beyond the bounds of the screen. Views appear above or below other views, overlapping with them either wholly or partially.

3. *Overloading* - In this layout style, one visualization (client) is rendered inside another visualization (host) using the same spatial mapping as the host. Like superimposition, the client visualization is overlaid on the host, but unlike Superimposition, there exists no one-to-one spatial linking between the two visualizations.

4. *Nesting* - Nested views are also based on the notion of host and client visualizations. In this style, one or more client visualizations are nested inside the visual marks of the host visualizations, based on the relational linking between the points. An example of this would be a scatterplot where the individual marks are barchart glyphs [79].

[2]Resembles the tiling style of window management.

[3]Resembles the overlapping style of window management.

7.2 Relevant Layout Techniques

From the range of styles I present above, ones I believe merit an exploration on tablets are juxtaposition (tiling), superimposition (overlapping), and scaled-up views [4]. Each of these styles can generate multiple layout configurations for tablets, as I discuss below. I discarded the other styles due to the limitations that I identified with them. For example, 3D layouts require complex interactions to be effective and are also inefficient with utilizing the screen space. Tabbed windows could be feasible, as evident in Vizable [5], but views in separate tabs cannot be visible at the same time, making brushing and linking between views impossible. Finally, overloading and nesting styles alter the default representations of visualizations. Supporting the standard visualization techniques was a deliberate decision because these techniques are applicable to a wide variety of data sets. The same, however, is not true for the hybrid views of overloaded and nested styles. These views are complex and may not be applicable to and effective for all datasets. Below, I explore the three selected layout styles in detail.

7.2.1 Fixed Canvas Sizes

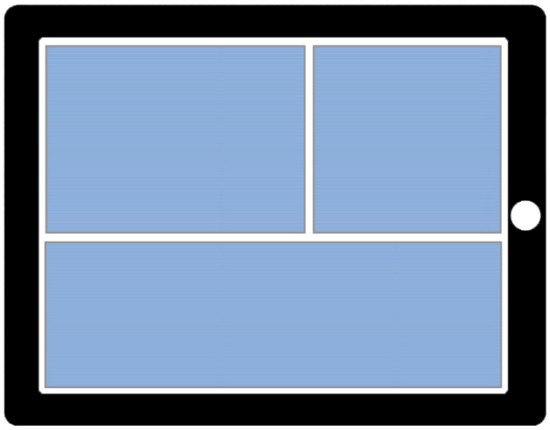


Figure 35: Tiled *juxtaposition* style.

[4]Beryl Linux, www.beryl-project.org

[5]Vizable, vizable.tableau.com

Tiled layouts (Juxtaposition) Tiled layouts (Figure 35) have a tremendous potential for tablet-based visualization systems. Within a grid, views can appear in any configuration [22]. The rows and the columns can contain an arbitrary number of views. In a typical implementation, the number of elements in a row or column is decided by the density of data the views are presenting. For example, barcharts can be compressed to small sizes since the thickness of the bars permits the view to be usable at small sizes. The space made available by compressing a chart can be allocated to other views.

The layout style is efficient, and is used in dashboard applications such as Google Analytics [6] and Roambi [7]. In these applications, the size of the data is largely fixed, and thus the layout remains predominantly static. However, in situations where the data is constantly updated, such as through use of filters, the layout requires careful use of controls and options to manipulate the layout [82]. Some of the operations that require controls are:

1. *Scaling views* - A standard operation with tiled layouts is the one for scaling the views. Since a view's size is not independent and depends on other views, the sizing control is shared among views, often appearing at the shared boundaries. Alternatively, users can expand or contract any view. Once the scaling is complete, the system sizes and repositions the other views optimally using predefined rules and constraints.

2. *Swapping views* - Controls for swapping views and moving them on the screen are also common in tile-based layouts. Since tiled layouts often cover the entire space of the screen, these operations are particularly relevant when new views are introduced or existing ones removed. Such modifications result in ineffective configurations that often require user manipulation.

[6] Google Analytics, www.google.com/analytics

[7] Roambi, www.roambi.com

3. *Connecting views* - Connecting or constraining [6] views is another behavior associated with tiled layouts. Here, the position (or size) of one view is bound to the position (or size) of another view. As a result, users can introduce constraints to ensure that a view always appears above or to the right of a view (and dragging one also drags the other), or is of the same width as another view[8].

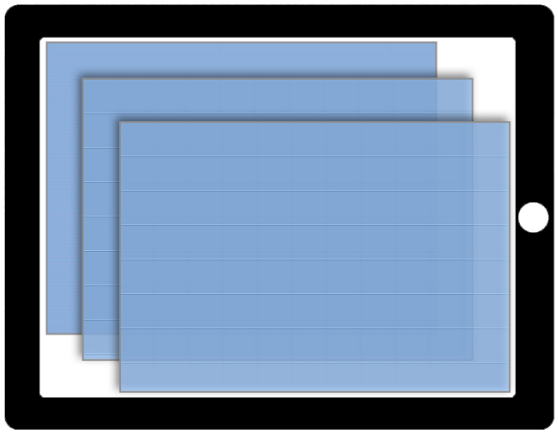


Figure 36: Overlapping *superimposition* style.

Overlapping (Superimposition) The second style I identified is overlapping (Figure 36). Views can overlap with each other, and can be scaled up or down when required. Since views do not need to be realigned to fit within a grid or a structure, the layout is highly flexible. Example operations needed within this style are:

1. *Scaling views* - Similar to the tiled layout style, operations for expanding and contracting views are important. Since the size of one view does not affect the size of any other view, views usually do not share a common boundary. Thus, scaling controls appear individually for each view. A typical approach is to provide a control at the top or bottom right corner of the view. Alternatively,

[8]Hutchings and Stasko [58] present the limitations of constraint-based layouts, including user forgetfulness, need for defining relationships every session, and unpredictability.

pinch and pan gestures can be used to change the size of views. A third option is to provide an explicit mode for layout control. When the mode is enabled, standard gestures can be used for modifying the layout.

2. *View order* - Since this layout allows views to overlap, operations are also needed for managing the relative ordering of views on the screen. A typical implementation consists of four controls — 'Bring forward', 'Send Backward', 'Bring to front' and 'Send to back'. While these controls are sufficiently expressive, there are two limitations. First, the options require the user to first select a view on the hierarchy. On touch screens, selecting views that are completely hidden and blocked by other views is non-trivial. On desktops, this is often achieved by using the TAB key, which sequentially selects all the available views. Second, when the number of views on the screen is high, controlling the order of one view can be considerably cumbersome. This issue has been addressed in previous work, and various alternates have been proposed in the scientific visualization domain [139]. However, these solutions usually consist of complex multi-handed interactions that are more feasible on the large tabletops than on small tablet devices.

7.2.1.1 Flexible Canvas Size

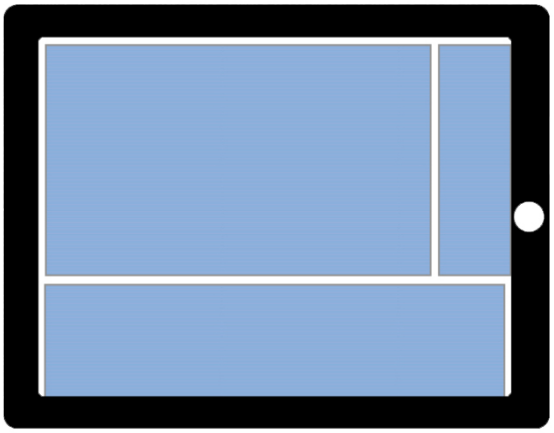


Figure 37: Tiled with scrolling.

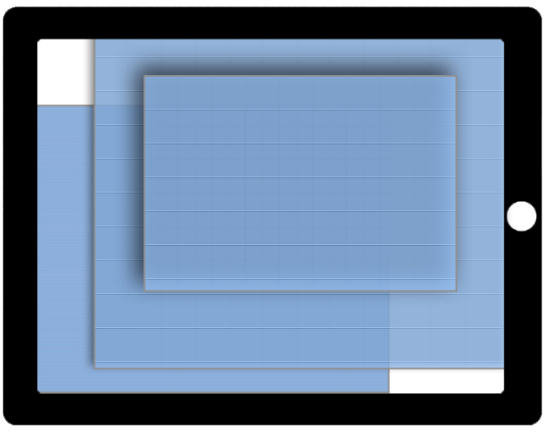


Figure 38: Overlapping with scrolling.

In this style, a part or the whole of a view may be placed outside of the screen. Also described as Zoomable User Interfaces, such configurations are dependent on easy access to operations for scaling and panning the view. Bederson and Hollan [11] presented the earlier work in ZUIs and since then the technique has been in use in various systems [19]. ZUIs are particularly relevant on touch screen since multitouch gestures are naturally conducive for panning and zooming tasks. The user is able to import a large number of views on the screen at any time and manipulate them more easily than on desktops. Views that are not relevant can be easily be placed off screen.

Within ZUIs, both juxtaposition and superimposition styles are feasible. Moreover, the advantages are compounded since the effective size of canvas increases substantially. This means that when a view is scaled up, little reconfiguration is required since the other views can simply translate their positions without needing to change their size. However, along with the advantages, there are also several disadvantages of ZUIs. Most importantly, the actions of panning and zooming take precedence over any other layout operation, such as scaling and swapping of individual views. Further, since adding and ignoring a view is fairly inexpensive, the layout of the canvas can become complicated fairly quickly. In a populated canvas, locating and identifying relevant views can become a complex task on its own. Some methods of countering this include adding overviews such as heads-up displays (HUD). But these introduce a different kind of complexity to the interface which may be equally undesirable.

7.3 Selected Layout Technique

The layout technique I ultimately adopted for Tangere is *fixed-canvas juxtaposition*. Below I highlight the reasons why I considered this technique to be the most relevant for Tangere.

1. Simplicity: Simplicity in the context of layout can be understood in two ways.

The first is simplicity of the interactions that are employed to control the configuration of the layout, such as position, size and order. Here, the juxtaposition style without scrolling requires the minimum number of controls and user intervention for effective operation.

The second is the simplicity of the physical characteristics of the views — how easy is it to understand the layout of the views? For instance, if the user opened the app and landed at a screen with several views arranged in a layout, is s/he able to comprehend the structure without interacting with the screen? With the juxtaposition style, a grid-based configuration enables easy understanding of the state of the system since there is a familiar structure to the views.

One could argue that overlapping resizable windows model is simpler, given it is a metaphor we understand from use on desktops. However, our familiarity is precisely what makes adapting overlapping model on tablets complex. This is because our expectations of the features does not change, but providing the same level of customizability we have on desktops is extremely difficult.

2. Visibility: In juxtaposition style without scroll behavior, all views are visible at all times. While this constrains the number of views that can be placed on the canvas, the benefit is that users have a comprehensive visibility into the system at all times. Placing views outside of the screen, or behind other views, requires

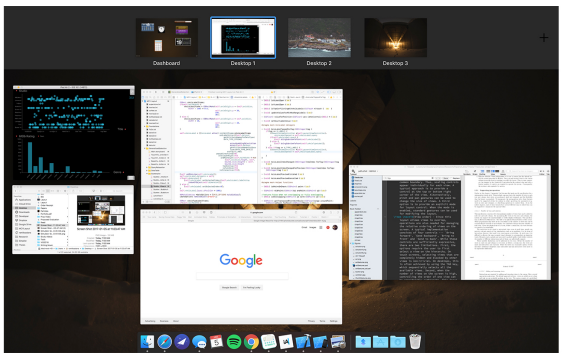


Figure 39: Mac OS Exposé

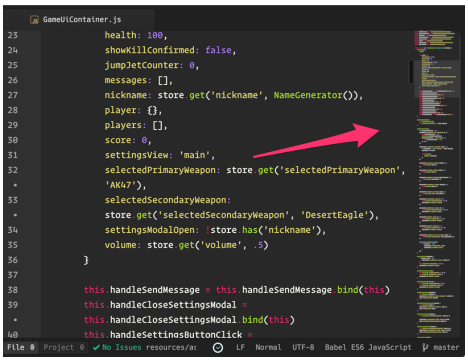


Figure 40: HUD in SublimeText

additional interactions. For example, the system has to provide cues to inform the user of all views on the screen. These include HUD or minimap available in applications such as SublimeText (Figure 40) or features such as Spotlight and Expose (Figure 39) available within the Mac OS.

3. Feasibility to Constrain: The more customization a layout affords, the more likely it is that a user configures it into an unusable state. Constraints help to ensure that the likelihood of such configurations is reduced significantly and that there is a minimum threshold to how usable the interface is. Juxtaposition supports constraints in the form of the maximum number of views that can be added to each row and column, the minimum size of each view, and relative positions of each view.

4. Adaptability & Extensibility: The juxtaposition style is extremely flexible — a grid-based layout can be adapted to any screen size. Across screen sizes, the designer only needs to reconfigure the number of views permitted in each row and column. With the views laid out, juxtaposition also more easily accommodates changes in orientation of the device. Since the views are positioned relative to each other and not relative to the edge of the screen (such as in superimposition), it is easy to locate them in the updated layout. The behavior, thus, more closely matches users' expectations. This adaptability is relevant because touch devices are available across a variety of screen sizes. For example, Apple iPad is available in three sizes today. Due to the adaptive nature of the juxtaposition style, the layout behaves in a similar manner across all the screen sizes.

5. Effective defaults: With juxtaposition, it is easy to ensure that the default layout the system generates is effective. Users can add several views to the canvas without worrying about or needing to modify the layout. This, in turn, empowers the designer as the threshold of operation for the system is low and only a

limited number of controls are required to operate the system. Consequently, the system is also operable by novices.

7.3.1 Configuring Fixed-canvas Juxtaposition Layout

Earlier in this chapter, I presented the features that make up the specification for a juxtaposition layout style. These include, among others, the number of views in each row and column, how views are positioned, sized and swapped, and the flexibility that the layout is permitted. These features need to be supported within the system by either specifying the parameters (e.g. views in rows and columns), or identifying operations to control them. Similar to the design options in the earlier chapters, each of these features have several different ways in which they can be supported. I enumerate the options below.

7.3.1.1 Number of rows and columns

This specification relates to the maximum number of views that can be added to each row and column. Maximizing views in rows and columns is relevant as techniques such as small-multiples and trellis can be supported this way. However, there are limitations to the number of views that can be accommodated. The limitations are not physical, as any number of views can be adjusted vertically or horizontally. Even at small sizes, people can perceive relevant aspects of the visualization, such as in sparklines.

It is in terms of interaction, however, that the small size of the views imposes a restriction. At such sizes, it is difficult to allocate the space around visual elements to accommodate an imprecise touch. Elements such as glyphs and axes need to be packed tightly, which further restricts the number of interactions each of them can individually support.

These usability limitations can be addressed in two ways. Either the number of views in each row and column is restricted, or below a certain size, the views are

made non-interactive and read-only. While the latter is a reasonable option, it needs interactive and non-interactive views to be represented differently, which is likely to increase the complexity of the system.

7.3.1.2 Adding and removing views

Since the number of views on the canvas is not constant, some mechanism is needed for adding and removing views from the canvas. Here, several approaches make sense. The default approach is one where views are added one at a time and take up any available position on the view. The canvas consists of a predefined number of positions, and these are ordered based on a rule, e.g. maximize the size of each view, or order views top-to-bottom or left-to-right. This approach is simple to implement and easy for users to comprehend. However, there are two downsides. First, since new views appear at fixed positions, inserting a view in a specific slot requires multiple steps and a mechanism to move or swap views. Second, the approach is less accommodating to changes in size of existing views. If the size of a view on the canvas is modified, identifying the right position for a new view to fit is non-trivial for the system and difficult for user to predict.

An alternate approach allows more flexibility by permitting users to specify the position they want the views to appear at. There are several ways to achieve this. For instance, views can be dragged from the menu and placed at a specific position. Alternatively, in a two-step action, users can first tap on a view in the menu and subsequently pick the desired position from ones highlighted on the canvas. Another technique is to first create space in the layout and then place the desired view at the location. The approach to specify the end position, as highlighted by these methods, is more flexible than the default approach. Moreover, users have more control on the expected output. However, the action itself takes longer to complete, particularly in situations when the canvas only contains a few views.

A similar variation in options also exists for removing views from the canvas. In the existing system, the layout of the views is replicated at the top of the add-view menu. This approach can certainly scale to more views. However, other options also exist. For instance, a 'remove' icon can show up if the user performs an action, such as tap-and-hold, on the view directly. Alternatively, the view can be dragged and dropped on the edge of the canvas or on a drop zone such as a recycle bin.

7.3.1.3 Repositioning and swapping views

The capability central to an advance layout mechanism is of repositioning views on the canvas. Ad-hoc configurations in support of specific analytical queries can be generated by placing views above and below, or next to each other. Repositioning also promotes recycling and reuse as views that are less important can easily be placed at the side so as not to disrupt the usage of other views, and brought back in when needed. The methods for repositioning views are similar to those for adding views that I described above. Views can be dragged and placed at a new position or, using a mechanism to *initiate* repositioning, the user can select a position from a list of highlighted positions.

Swapping is another operation for modifying the layout. Swapping can be considered a special case of repositioning since, in most cases, views can be swapped by repositioning each individually. There are two exceptions to this, however. First, if two views are placed side-by-side they can be swapped in a single step by dragging the left view and placing it to the right of the other view. Second, if all rows and columns contain the maximum permissible number of views, the repositioning operation is inaccessible. The swapping operation is still valid, however, but cannot be achieved using two repositioning steps.

7.3.1.4 Resizing views

Similar to repositioning, the ability to resize a view improves the flexibility of a system tremendously. Views may be resized to reflect an increase or decrease in their relevance to the task being performed, or to create space for accommodating new views. There are several techniques for resizing views. On touch surfaces, the expected method involves the use of the two-finger pinch gesture. Pinching in and pinching out can scale the view down and up, either by maintaining the aspect ratio or mapping the ratio to movement of the fingers in the two directions. As one view scales, the surrounding views can respond and resize to release or take up additional space.

However, there are a few drawbacks of the pinch gesture. For instance, using the pinch gesture within the bounds of a view is not feasible because the gesture is already employed for scaling content within the views. (An alternate is to use modified pinch gestures, such as Clutch + pinch). Another limitation is the ambiguity of the pinch operation — how does the system respond if the two fingers used for pinching land on different views?

Other methods for scaling views include marquee-based resizing, where each view reveals a control with an 8 degree-of-freedom resizing behavior. This approach is useful for the juxtaposition style as the controls show up at the shared boundaries between views. Dragging the control for one view resizes multiple views at once, ensuring that there is no empty space on the canvas. In a similar approach, all boundaries between views can also be made draggable.

7.3.2 The Selected Layout Specification

From the above presented options for features in juxtaposition, I identified ones that seemed most relevant for Tangere. Below I describe the selected specification and the rationale for my decisions.

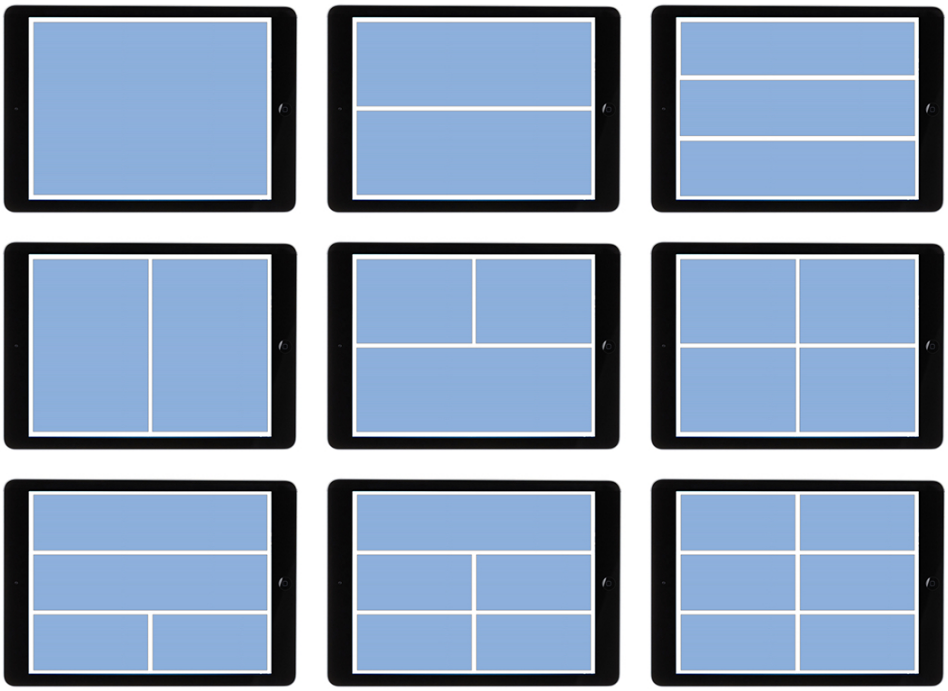


Figure 41: A subset of possible layouts for views on the canvas.

1. Grid count: The canvas now consists of a 3x2 matrix. Three rows containing views can be added to the canvas, and each row can fit two views. Figure 41 presents some possible configurations of the selected layout.

2. Adding and removing views: The views are added one at a time. If the canvas contains less than three rows, the added view appears in its own row at the bottom. If three rows are already present, the view is added to the first available slot in the second column. Views are added by tapping on the label in the menu on the left.

   The layout of the badges representing the added views resembles the layout of the views themselves (Figure 42). The views can be removed by dragging the badge sideways, either left or right.

3. Modifying layout: The views can be repositioned using a tap-and-swipe gesture. Upon initiating the gesture, a small snapshot of the view is generated and attached to the touching finger. The snapshot can be dragged across the screen.

Figure 42: Layout of tags in the Add View menu matches that of the views.

With each movement, the position of the finger is evaluated for feasibility as a drop spot. If the new position changes the resulting configuration, a blue rectangle highlights the new bounds of the dragged view, and the other views move to their new position. As a result, the feedback on the resulting layout is always provided to the user (Figure 43).

4. Resizing: I omitted features for selectively resizing the views that are placed on the canvas. Within the permutations of views on the screen, a view can appear in one of six different sizes, corresponding with two levels of width and three levels of height. However, within each configuration, the size of a view is fixed and I refrained from adding features for changing the size of an individual view.

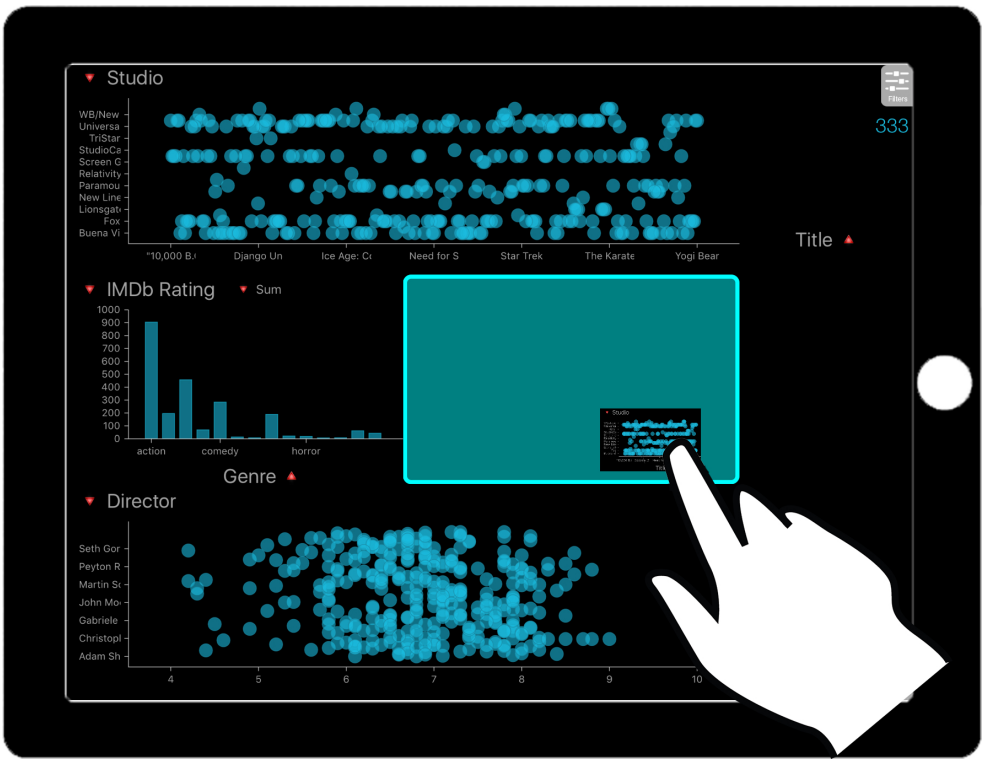


Figure 43: A view being repositioned. Green zone depicts the resulting frame if the dragged view is dropped.

7.3.3 Rationale

Although several factors played a role in the decision to adapt the above specification, the predominant feature was simplicity. The goal was to provide users with the maximum capability at the minimum interaction cost. Simplicity has been an ongoing theme in the design of Tangere, and has influenced most of the concepts I have used. Here, again, keeping interaction as simple as possible was the central tenet.

For instance, to add a view, user taps on the icon and the view appears in the canvas at a specific, predefined position. Other options I considered were drag-to-add — users drag a view from the menu and place it wherever they want, and two-tap-to-add — users tap on the icon in the menu → the system highlights possible locations for the view to be placed → user taps on the preferred location. The drawback of these options was that they made adding a view a multi-step process. For a novice users, it adds complexity and latency to the interface.

Instead, my overall goal was to ensure that the advanced layout specification did not come in the way of a user using Tangere for the basic features. In other words, if a user does not know about the layout operations, the tool should still be effective and fully usable. This rationale also motivated my decision to omit features for resizing the views. With views whose size can be modified, there is likelihood that users generate unusable and ineffective layout configurations, which is undesirable.

The advanced layout solution is, relatively, complex to operate. However, the overall solution is congruent to the existing design philosophy of Tangere. In summary, the solution can be described using the following properties.

1. *Optional* - For a user, managing layout is not mandatory. The default layout should be sufficiently usable without any user input.

2. *Feedback* - While manipulating the views, Tangere displays the resulting layout consistently throughout the period of the operation.

3. *Secondary Action* - The advanced layout operations are secondary features in Tangere. Thus, the interactions for all other primary features take precedence over those for controlling layout. Consequently, it is easier to apply filters on the glyphs than to change the layout.

4. *Simplistic* - Continuing with the overall theme of my work, controls for manipulating layout have been designed to be as simple to operate as possible, within the constrains of the possible advanced layout mechanisms.

CHAPTER VIII

EVALUATING EFFECTIVENESS OF TANGERE

8.1 Introduction

The focus of my research has been on exploring the utility, effectiveness, and usability of tablet-based tools for analyzing data using multiple coordinated visualization views. My explorations of these areas has resulted in a feature-rich tablet-based system called Tangere. In exploring these areas, I have used simplicity as a theme guiding my research by making assumptions at various stages of the design. These assumptions have pertained to the expertise of the end-users, nature of data that they would import in the tool, and the type and complexity of questions they would ask of such data. In turn, the assumptions have guided the design of Tangere, the visualization techniques it provides, and the breadth of operations each technique supports. While I made these assumptions with careful perusal and through informal feedback I received from colleagues, these assumptions have not been examined rigorously. Thus, one critical aspect that remains is a thorough and in-depth evaluation of Tangere with actual users.

In the following chapters, I describe the user studies that I have conducted to understand the effectiveness of the Tangere system. The three studies capture very different facets of the system, with each study progressively informing design alterations and refinements for the next. Although the design space of possible evaluations for Tangere is large, with these studies I assess characteristics of the system that were most reflective to my design process.

8.2 Designing Experiments to Evaluate Tangere

One of the themes for the Tangere system was targeting novice users. I define novice users as people who belong to one of the following three groups:

1. People who do not have previous experience with touch-based interfaces.
2. People who do not have previous experience with visualizations or visualization systems for data analysis.
3. People in both groups 1 and 2.

An understanding of the requirements of each of these three groups played a crucial role in specifying the guidelines important for designing Tangere. Designing for the first group suggested employing only the standard suite of gestures that can be easily learned, including through use in other applications. Although I do introduce novel and more complex interactions in the latter stages of my research, these advanced interactions are supplementary to the system and employed for behavior that can already be achieved using a combination of existing interactions.

Similarly, designing for the second group requires using the standard visualization techniques that are easy to understand and use, and applicable to a wide variety of data sets. Additionally, the explicit constraints placed in the system, such as stacking views vertically and only permitting a limited number of views at a time, further help in simplifying the interface for novice users and limiting the complexity they may face.

Finally, for both groups, once the novices are past the initial learning phase, the design of the system reflects the frequency with which I expect them to use the features in the system. While task importance and frequencies have been studied in the past, the mapping of the tasks to the controls and the layout of the controls remains unverified.

These assumptions about the users of Tangere and their usage behavior have certainly assisted in the design of the tool. However, verification of the validity of

these assumptions is clearly required. Several approaches to evaluation are relevant and can be employed. I discuss these below.

8.2.1 Evaluation Techniques

The evaluation of information visualization systems remains a deeply challenging problem. While several evaluation techniques are feasible, no one technique stands out as being the most effective. Lam et al. [71] provided a comprehensive review of 360 information visualization research papers that included some form of evaluation. They identified seven prototypical evaluation scenarios, four under the idea of "understanding data analysis" and three under the notion of "evaluating visualizations". Within each of these scenarios, the authors described three descriptive characteristics: 1) goals and outputs, 2) evaluation questions, 3) methods and examples.

At a high level, my goals for the evaluation of Tangere are aligned with the Evaluating Visual Data Analysis and Reasoning (VDAR) and the Evaluating User Performance (UP) scenarios, and to a lesser degree, the Evaluating User Experience (UE) scenario. However, for each of these scenarios, several evaluation strategies are feasible. Identifying the optimal strategy for tablet-based visualization devices remains a challenge because very little actual development of visualization systems for this platform has occurred.

One example of relevant previous work is Kinetica [102]. In evaluating Kinetica, the authors compared the performance of the tablet application with Microsoft Excel for basic data exploration and sensemaking tasks. However, adopting this approach for evaluating Tangere has several drawbacks. First, there exists a big gulf in the number of features supported by Excel and Tangere. In fact, the two applications have been designed with very different goals — Excel is primarily a tool for spreadsheet creation and not analysis. Second, it would be difficult to contextualize any observations made during the comparative evaluation. The observations could be resulting

from the difference in the techniques, types of features, or simply the difference in the platforms (desktop vs. tablet).

Other evaluation approaches may be more relevant. While designing Tangere, each decision required examining multiple choices and subsequently choosing the one that I considered ideal. Although I sought to be objective in my analysis, it is not clear if the options I chose are the actual best options. Thus, a potential evaluation strategy is to perform a comparative analysis of the different alternatives for each operation with participants. The goal is to identify if the interactions chosen by a majority of the participants match the ones I chose.

An alternate study design targets the original goal of promoting simplicity and creating a system for novices and unfamiliar users. Several of my design choices were heavily based on reducing complexity in Tangere. Identifying how successfully the system achieves this goal is another possible strategy for evaluation.

Although elaborate to perform, a long term evaluation is another relevant strategy for an application like this. Users develop proficiency with tools, particularly those with novel interactions, over time and repeated use. A longer-term evaluation, preferably conducted in a realistic setting, is an ideal method for capturing issues that participants might have with the discoverability and learnability of the tool, and their satisfaction with it over time.

Finally, a comparative evaluation is also feasible. During early stages of my research, commercial visualization tools for tablet devices were absent. However, this has changed with the availability of Tableau's Vizable tool that I described in detail in the chapter 2.3.2. The fundamental goals and targets of the Vizable system are similar to Tangere — the tool employs standard visualization techniques, and targets novice users. But the two tools take a fairly different approach towards visualization-based analysis. For instance, Vizable shows multiple techniques disjointedly instead of showing them in a coordinated configuration. The combination of a similar goal

and a difference in approach make a comparative evaluation of the two tools highly relevant.

All the evaluation strategies I mention above are feasible, with each highlighting unique insights about the system. However, to scope the evaluation process, I identified three evaluation strategies as being most useful based primarily on my design process for Tangere. These were an evaluation of the simplicity of design of the system, an evaluation to compare the performance of the system to that of another tablet-based visualization system, and an evaluation to understand how the tool supports the workflow of trained users.

8.3 Evaluation 1: Measuring Simplicity

Simplicity of a system can be captured by measuring two usability metrics — *discoverability* and *memorability*.

1. Discoverability - To what extent (in terms of the number of features) can users discover the system on their own without receiving any prior training?

2. Memorability - When users return to a system after a gap (e.g., a few days), how accurately can they recall the system and its different features?

These metrics are commonly used in HCI for evaluating the usability of systems [83]. In general, the metrics are dependent on each other — interfaces high on one tend to also be high on the others. However, the metrics are evaluated using different methods, each of which reveals specific and distinct insights about the interface.

8.3.1 Methodology

I conducted this evaluation in two phases. In the first phase, I focused on the discoverability metric. In the second phase, I evaluated the memorability of the features and captured the general effectiveness of the interactions in the tool. The second

phase was conducted at least four days after the first phase. Below, I describe the methodology in detail.

Phase One - Discoverability Discoverability of an interface can be captured by observing how people approach and use the system for the first time. If first time users find the interface relatable and are easily able to connect to aspects of it by virtue of its resemblance to other existing applications they use, it is a win for the interface. To capture participants' reaction to the interface and measure discoverability, I engaged participants in an open-ended exercise. I gave the participants an iPad with an instance of Tangere running, and asked them to spend twenty minutes exploring the application on their own. To contextualize this exploration, I gave them a brief overview of the goals of my research as well as the evaluation, i.e. my goal is to advance the usage of visualization-based analytical tools on touch-enabled devices, and the evaluation tries to capture how far this exploration takes us in that regard.

In the twenty minute period, I monitored their interactions without participating in the exploration in any direct way. My role was twofold: I kept a track of the different features they were able to discover, and I interfered in situations when Tangere encountered a bug. Although I invested heavily in quashing the bugs that existed in the system, I could not ensure that all were addressed. Olsen labels the limitation of ensuring a bug-free evaluation environment as the *fatal flaw fallacy* [91]. It is virtually impossible to examine all eventualities of a system and flaw analysis is frequently a barrier to new systems research. Since participants were unequipped to discern features from bugs, and would likely interpret the system response incorrectly, it was important to limit the negative implications. Thus, in situations when a bug was encountered during evaluation, I stepped in and addressed it by restarting the system. As a consequence of this, the number of bugs went down significantly over the course of the experiment.

The number of features participants explored in the twenty-minute period gave a measure of the discoverability of the interface. Clearly, participants' expectation strongly influenced this measure. Participants who had used other visualization tools in the past expected features such as filters and data table to be present. This influenced their exploration in two ways: first, if they expected and did not find a feature, they looked for it more aggressively; second, once they found an expected feature, they spent less time operating it and proceeded ahead in search of other feature faster compared to participants who found an unexpected feature. I captured participants past experience with visualization tools in the qualitative questionnaire at the start.

In the next stage of the study, participants completed a series of predefined tasks with Tangere. The tasks were presented to them as questions on a sheet of paper. The questions were based on the dataset loaded in the system. To find answers to the questions, participants had to perform a set of operations on the interface. To gather richer data, I asked participants to think-aloud for each question. This way I could better catch instances when they correctly identified the operation they need to use, but could not locate the operation on the screen. I captured both their intentions to perform a task ("*I want to multiselect on the table*") and their confusion in interpreting the response of the system ("*why is the top of bar chart changing?*")

It was important that the tasks required participants to access all features in the system at least once. Since participants only had a limited time, I paired several features together within a single task. For example, selecting all glyphs within a range of values (rectangular selection) and filtering to those glyphs (keep-only) was combined into a single task. Overall, I created 18 tasks, and participants had roughly 50 minutes to complete these tasks[1]. The tasks resembled those described in the study in Section 4.1.3 and were modeled on ones used by Drucker et al. [29] in their

[1]Appendix A presents a snapshot of the tasks and dataset I used in this study.

comparison of WIMP and gesture interfaces for tablets. The tasks were laid out in a random order of difficulty, but all participants were given the tasks in the same order. I estimated the difficulty of each task based on the number of steps and time needed to complete it.

Participants completed the tasks in one continuous session. I only interrupted them if Tangere encountered a bug and had to be restarted. After 45 minutes, I asked participants to complete the task they were performing and ended the session. For these sessions, I did not collect quantitative data such as time taken to complete a task, number of tasks completed, or number of tasks completed correctly. I communicated the same to the participants at the start of the study so that their interactions would not be biased by efficiency constraints.

I did not collect quantitative data, such as performance time and accuracy, since other sources that I could compare this data to were absent. The data itself would give little insight into discoverability and learnability of the system. Instead, I only made qualitative observations. This included noting the features that participants successfully discovered and were able to operate while performing the tasks. Further, participants were encouraged to think-aloud. This provided insight into how they were interpreting the application and was particularly useful for highlighting situations when their actions on the screen did not respond in a manner they expected.

I also collected a more detailed feedback at the end of the session. Each participant completed an online survey, providing feedback on their perceived ease-of-learning and ease-of-use of the system. The survey consisted of 22 likert-scale questions largely based on the questionnaire generated by Elliott [31].

Phase Two - Memorability In the second phase, I studied the memorability of the features in Tangere. For this phase, a subset of participants from the first phase were invited back after a few days. Over the course of the study, I measured how well

participants remembered the features of the system and how to operate them.

The central activity of the experiment was a case-study. The experiment was run in pairs where participants were paired based on availability. The experiment design allowed several different mechanisms to capture the memorability of the system. At the start of the study, participants were given two tasks to complete. One, they had to enumerate all the features of Tangere that they could remember from using it the last time. Two, they had to generate a series of questions based on a specific dataset that they would subsequently solve using the system. For every pair in a session, I randomly assigned one person to complete task one first and the other person to complete task two first.

For task one, I captured memorability in three steps. Initially, participants were asked to recall Tangere from the previous session and verbally describe all of its features that they could remember. Next, for the features they could not recall, the experimenter provided verbal hints to encourage recall. These included statements such as "Do you remember if there was a way to filter data?". Finally, to stimulate the visual memory, the experimenter provided participants with a snapshot of the interface. As expected, certain features were easier to recall once the interface was made available.

For task two, participants were shown a sample of the coffee dataset and asked to spend ten minutes crafting questions based on the dataset. They completed this exercise on their own. I did not give any instructions regarding the style of questions they should construct except inform them that they would solve the questions with the interface.

Half the participants completed the system recall task first, while the other half crafted the questions first. After completing the two tasks, participants spent the remaining time trying to answer the questions they had crafted. They did this in pairs and were free to structure their process as they wanted. This included controlling the

use of the iPad — participants were given their own iPads, but were free to use only one and analyze the data together. In that case, one participant would take the lead and operate the interface while the other participant observed.

Similar to phase one, I did not collect quantitative data in this phase such as the number of questions participants generated or solved in the study. Instead, I observed how participants used the features of the system over the course of the experiment. Since my focus was on measuring memorability, another useful data point was the difference between the features they could recall earlier and ones they ultimately used. For the features they used but could not recall earlier, I wanted to identify the affordances that assisted them in this recollection.

8.3.2 Participants

In the first phase of the study, a total of 14 people participated (2 pilot). For the second phase, 10 of the 14 people participated (2 pilot). The total pool comprised of 7 men and 7 women, all of whom were graduate students in Computer Science at Georgia Tech. Similar to the study described in section 4.1.3, I sought to recruit participants who were *subsequent learners* [24], i.e. people who are novices to a specific software system, but experienced with a similar system. This ensures that they not only have the required domain knowledge, but also a general understanding of which tools and functions will be available. This also guarantees that their performance is primarily a measure of the quality of the interaction and not of their subjective lack of understanding of the underlying features. The experience I sought was with systems such as Tableau or Spotfire. Thus, I recruited participants through the roster of past courses on Information Visualization taught at our school.

I asked participants to rate their knowledge and experience with visualization systems in general and tablet-based visualizations systems in specific. Five out of twelve participants stated that they had a good understanding of visualization systems

("*I am very familiar with the idea of information visualization and have used several applications*"), whereas the other seven mentioned that only had a fair knowledge("*I have used applications like Tableau, Excel, QlikView etc. occasionally (less than 10 times in the past year)*"). Further, majority of the participants (9 out of 12) had low or no experience with tablet-based visualization systems, while one participant had used Roambi [2] on an iPad in the past.

8.3.3 Results

Tangere is fairly feature dense. It contains a total of 45 different features that can be distinctly classified under the seven task categories presented by Yi et al. [137].

Discoverability Table 4 presents the discoverability classification for the key features in Tangere. Figures 44 & 45 present snapshots of these features. The discoverability of the features was dependent on several factors.

1. Interaction type: The features used either gestural interactions, interface elements, or both for operation. Interface elements clearly performed better at discoverability than gestural interactions. Gestural interactions, however, were more promising for serendipitous exploration. Accidental interactions revealed features participants did not know about or did not anticipate.

2. Past experience: As discussed earlier, participants who have previous experience using visualization systems expected certain features in Tangere. The same was true for people who had past experience with iPad and iOS. For instance, participants were versed with the 'Keep only' and 'Remove' options from using the Tableau system. Similarly, participants could discover the swipe action to remove the 'Keep only' and 'Remove' filter badges as it is a part of the vocabulary of gestures used by apps on iOS.

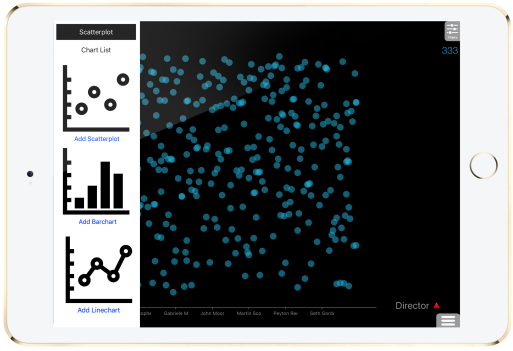
[2] Roambi, www.roambi.com

3. Workflow: Perhaps the biggest factor that led to participants discovering features was expectation and need. If a feature was not needed, participants would not seek it. For instance, the interface contains a label on the top right that depicts the number of data items in the dataset. Since this information was not relevant during the open-ended exploration, most participants missed it, even though the label occupied a prominent position.

I found the results of discoverability to be as I had expected. A majority of features achieved medium or high scores in discoverability, which was encouraging. However, it was also concerning to observe the performance of some features that achieved lower scores than preferable. Most notably, only a few participants discovered sorting on barchart and the filter menu. As I mentioned in section 4.1.4, the gesture I used for sorting (tap-and-hold) is unique both to Tangere and iOS in general. In fact, the gesture is more common on the Android operating system, and this was evident in the study too as the participants who discovered sorting were Android users.

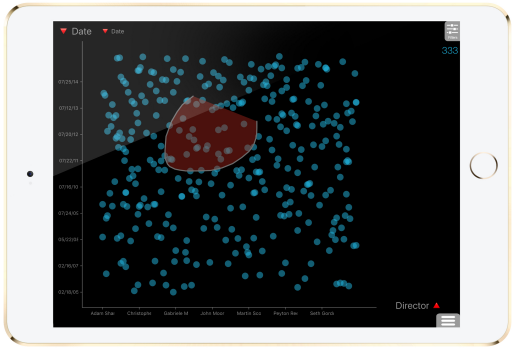
The filter menu suffered from discoverability issues due to a lack of visibility and a relevant affordance for it. This absence was particularly detrimental in light of the affordances in place for the other two side-views. The data-table view at the bottom had a handle that was always visible, and while the add-chart view did not, it was open by default. The settings view had neither. Taking cue from this result, I added a handle to the filter menu for the subsequent studies.

Memorability Table 5 presents the memorability classification from the second phase of the study for a subset of features in Tangere. Overall, I consider the performance on memorability to be promising. Over the course of the three steps through which I captured memorability, participants were able to recall most of the features they had discovered during the previous session. However, assisted recall of the second and third steps clearly helped the overall memorability, as highlighted in Table 5.

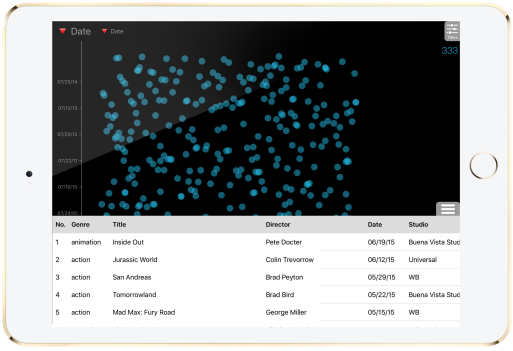


(a) Add chart panel that can be dragged in using an edge-swipe gesture.

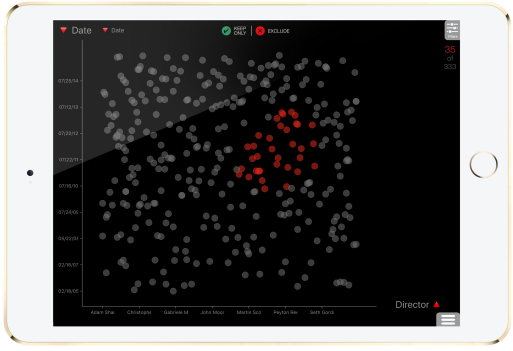
(b) Rectangular selection activated by dragging on an axis.



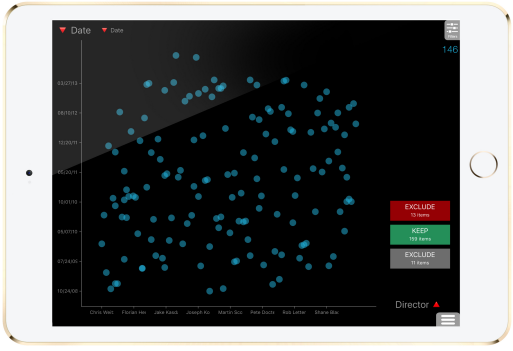
(c) Lasso selection. Drawing a path on the canvas selects glyphs within the path.



(d) Data table panel that can be dragged in view from the bottom.

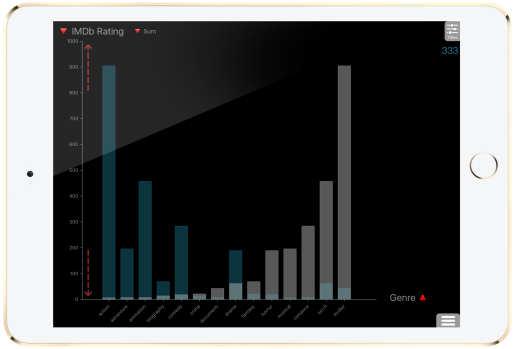


(e) Keep-only and remove options that are revealed at the top when glyphs are reselected.

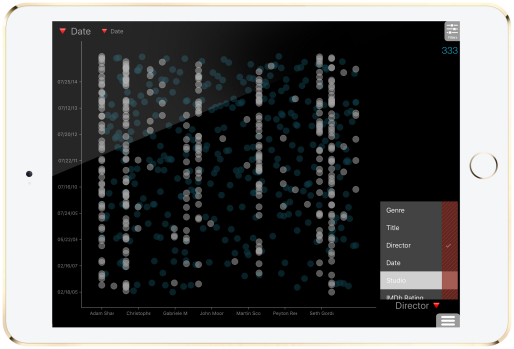


(f) Keep only and remove badges that can be enabled/disabled by tapping and removed by swiping.

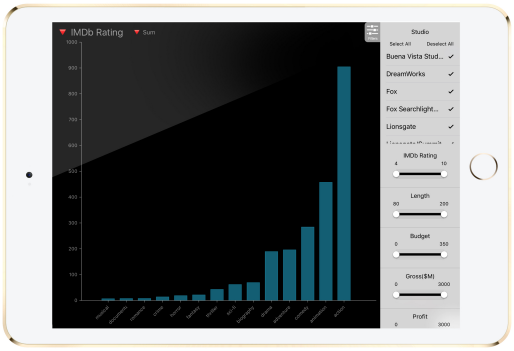
Figure 44: Tangere's features that received high discoverability scores.



(a) Bars can be sorted with a hold+drag gesture on an axis.



(b) Attributes can be previewed by dragging a finger on the red region.



(c) Filter panel contains categorical and quantitative filters. The panel can be dragged in from the right.

Figure 45: Tangere's features that received low discoverability scores.

Table 4: The discoverability of the features in Tangere, classified under high, medium, and low categories.

Discoverability score	**Features**
High (Figure 44)	1. Add Charts 2. Change X or Y attribute and aggregations 3. Select by tapping, lasso, rectangular window 4. View data table 5. Pinch to zoom chart 6. Keep/Remove filters 7. Toggle filter badge
Medium	1. Remove a chart 2. Unselect by tapping outside 3. Sort table column 4. Interpret data count 5. Remove filter badge 6. Open Filter menu
Low (Figure 45)	1. Preview X or Y axis graph 2. Pinch to zoom axis 3. Double tap to zoom out 4. Two finger drag to scroll 5. Sort barchart 6. Use filters in filter menu 7. Use reset all button

Below I highlight the interesting observations from the memorability analysis.

1. Memorable features: Features that all the participants could recall were the three visualization types present in Tangere (scatterplot, barchart, linechart), ability to place them as multiple coordinated view, data table, keep/remove filters, lasso selection, and rectangular selection.

2. Accurate guessing: Several features had low discoverability and memorability scores. This was mainly because participants did not originally expect the feature in the system. However, upon asking them to elicit the gesture they expect

Table 5: The memorability scores for Tangere's features. The scores represent the number of participants (out of 8) who could recall the feature and the average stage (out of 3) when they recalled the feature.

Feature	**# of recalls (out of 8)**	**Avg. recall stage (lower is better)**
Scatterplot, Barchart, Linechart	8	1
Add charts	8	1
Remove a chart	7	2.1
Change X or Y axis	8	1.4
Change subattribute or aggregation	6	2
Select by tapping	7	1.9
Select by lasso	7	1.4
Select through rectangular window	8	1.4
View data table	8	1.1
Sort data table column	5	2.4
Interpret data count label	5	2
Pinch to zoom chart	5	1.2
Pinch to zoom axis	2	2
Two-finger drag to scroll	3	1.3
Brushing and Linking	8	1.6
Sort barchart	0	-
Split line chart into categories	2	3
Keep only/ Exclude	3	1.3
Toggle filter badge	1	1
Remove filter badge	2	1.5
Use filters in filter menu	4	1.25
Use reset all button	1	2

Table 6: Participants' response to the ease of learning and ease of use likert-scale questions. The scale used was 1: Strongly Agree to 5: Strongly Disagree.

	Question	**Mean**	**Median**	**SD**
1.	The system was easy to learn	**3.67**	3.5	0.78
2.	I liked learning the system	4	4	0.79
3.	I felt comfortable learning the system	4	4	1.04
4.	I found the system easy to understand (higher is better)	3.5	4	0.9
5.	The ideas behind the system were easy to appreciate (higher is better)	3.92	4	1.08
6.	The system is no more difficult than other information visualization systems	3.5	4	1.09
7.	I often became confused learning the system (higher is better)	**2.83**	2.5	1.03
8.	It took too much time to learn the system (higher is better)	3.58	3	0.79
9.	The time and effort learning the system were well spent	3.92	4	0.9
10.	It was easy to use the system	**3.5**	3.5	1
11.	I liked using the system	3.92	4	0.9
12.	I had no difficulty understanding how to use the system	2.92	2.5	1.08
13.	The set of operations one needed to use were easy to remember	4.25	4	0.62
14.	It was obvious what to do next	3	3	0.95
15.	I became confused trying to complete the tasks (higher is better)	**2.92**	3	1
16.	The system made it difficult to complete the tasks	3.83	4	0.94
17.	I felt frustrated using the system (higher is better)	3.42	4	1.16
18.	I felt comfortable using the system	3.58	3.5	0.9
19.	The system was fun	**4.33**	5	0.98
20.	It took too much time to use the system (higher is better)	3.58	4	1

the operation to use, participants often accurately guessed the interaction. Examples include axis preview and barchart sorting. In both cases, participants described a hold-and-drag gesture for the operations, which is the gesture actually employed in Tangere.

3. Recall spike: In the first recall step, participants had an initial spike in how much of the system they could could recall on their own. However, since they could not track what they had recalled and what remained, they would often conclude by saying "I think that is all there was on the interface", only to remember several others as soon as the experimenter provided a hint.

   An example is removing a view. When describing how views were added, participants recalled both how to open the menu and how to add multiple views. But they did not describe how views are removed until the experimenter either provided a verbal hint ("*Can a view be removed?*") or an image of the add-view panel. For this reason, I believe that the importance of features recalled in step one should not be over-emphasized in comparison with those recalled later.

4. Feature obviousness: Some features were difficult to provide hints for. For example, when participants were asked how they would focus on a portion of the view, they always responded that they would use the pinch-to-zoom gesture. Similarly, when asked how they would order data in the table, their response was always to tap the column header. In both these cases, it did not matter if participants had attempted the operation in the first session. When given the hint, participant simply described the most obvious method or the method they remembered from other contexts.

5. Tap & Double tap: The use of tap and double-tap was particularly interesting. In both the first and the second sessions, tap and double-tap gestures were used extensively. Tap deselected any selected glyphs. Double-tap reset the zoom

if the view was zoomed in, and defaulted to deselection if the view was not. Across several participants, the use of these two interactions was frequent and fairly non-deliberate. They would attempt the gestures almost unconsciously when they were distracted with another thought or activity away from the iPad. However, in spite of the frequency of use, those participants entirely missed the interactions when recalling the systems. Participants' familiarity with and dependance on the gestures underlines the importance for the interface to support these interactions. Further, their confusion in using the two gestures highlights the need to support them in an inexpensive manner.

6. Linechart split-by attribute: In a linechart, the view could be split into multiple lines based on a categorical attribute. Most participants missed this description in the recall steps, including when the experimenter provided the hint. However, with an image, participants immediately pointed to the third attribute dropdown on the top right, irrespective of whether they used it in the first phase or not.

The above factors describe the observations both in the performance on memorability and the process used to collect the data. Below, I highlight the key factors that affected the memorability scores.

1. Frequency of use - Operations that were used more frequently were remembered better. Typically, these were either low level operations such as selection, or others that were critical to the workflow, such as adding and removing views.

2. Ease of use - Features that participants found easy to operate, particularly those that matched their expected behavior, were also more memorable. Examples include selecting glyphs with lasso and filtering with 'keep-only' and 'remove' options.

3. Delight - People remember a feature or an interaction that instills an emotional connection [86]. The connection emerges as a result of finding the feature fun and novel (such as animations and transitions) or by discovering it serendipitously. This was also evident in Tangere, specifically for interactions with keep/remove filter badges and rectangular selection on the axis.

Key observations Over the course of the study, beyond learnability and memorability of the features, various other observations stood out. Below, I summarize the ones that emerged across several participants.

1. In a barchart, participants wanted to put quantitative attributes on the x-axis.
2. They were confused by the line chart only allowing time/date on the x-axis.
3. They had difficulty interpreting brushing in barcharts when only a portion of the bars was highlighted (Figure 46).
4. They searched for the undo feature, and the ability to specify exact value for the range slider filters.
5. They expected selection of multiple rows on the data table.

8.3.4 Modifications to Tangere

As a result of the filter menu's low discoverability observed in the study, I made two changes to the design of the feature.

1. Filter menu handle (Figure 47): The filter menu on right was the least discoverable feature in Tangere. To account for this, I added a handle to the view. The handle was visually similar to the one for the data table at the bottom and could be dragged or tapped to bring the view on the canvas.

Figure 46: Brushing in barcharts. Participants had difficulty interpreting the highlighted portions of the bars in the lower barchart.

2. Filter menu peek: To provide additional feedback for the filter menu, I added a *peek* animation. The first two times users try to add a chart, a portion of the filter menu animated into the view briefly before sliding out again. This notified users of the existence of an additional view to the right.

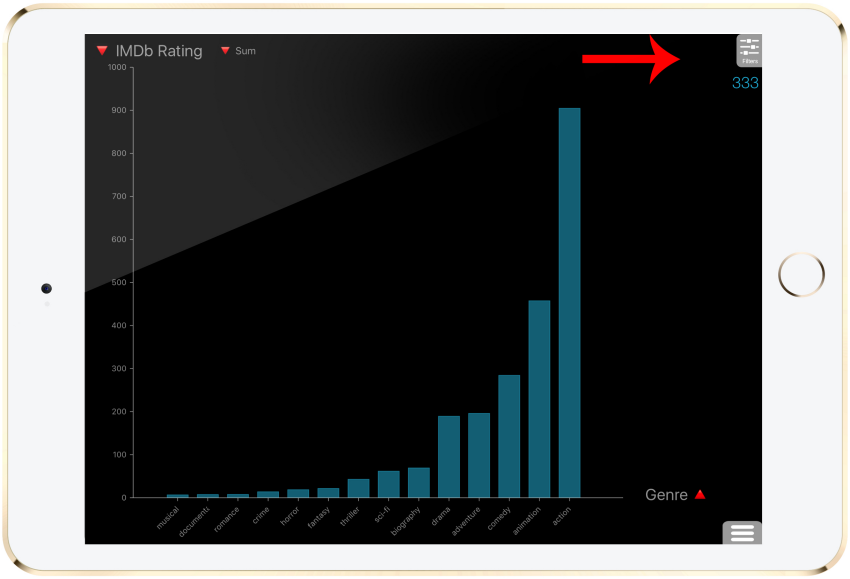


Figure 47: A handle was added to the top of the filter menu to make the menu more discoverable. The panel could be brought into view by dragging with the handle or tapping on it.

CHAPTER IX

EVALUATION 2: COMPARISON WITH AN EXISTING TOOL

In the second evaluation I conducted, I compared Tangere to an existing visualization tool for iPad — Vizable.

9.1 Designing a Comparative Study

There are several ways in which two tools can effectively be compared. The common approaches center around comparison of systems' usability, which encompasses metrics such as performance time, accuracy, utility, learnability, and safety [27]. Of these, quantitative metrics of performance, such as time and accuracy, have been used extensively in the past. Qualitative comparisons have also been made using metrics such as discoverability and learnability.

Besides usability-based evaluation, the other method is to compare systems on their user experience goals. These goals take into account how memorable, fun, enjoyable, and engaging the experience of using the system is. Recently, there has been a growing interest in using these metrics to evaluate visualization systems [106].

For comparing Tangere to Vizable, both usability and user experience evaluation approaches could be adapted. However, the relevance of each metric was dependent on the differences between the applications that were important to study and critical to highlight. By examining the two applications, I identified the following points of similarity and departure:

1. Novice users: Both the systems have been designed with novice users in mind. Consequently, being learnable and discoverable has been an important design

goal for both.

2. User flow: Both systems visualize tabular datasets and require cleanly formatted data. They both also support barchart and linechart, but Vizable omits scatterplot. Vizable allows only one visualization on the screen at a time, while Tangere supports multiple coordinated views.

3. Task support: The two systems support a wide range of visualization tasks, such as filtering, sorting and zooming. Simultaneously, there are several tasks that are also missing from both systems, e.g. neither tool provides methods for editing the data or adding attributes to it. Crucial task support differences also exist between the systems. An example is selection — Vizable does not support selection whereas Tangere provides multiple methods for selecting glyphs.

4. Feature richness: Vizable supports certain features that are critical for commercial applications, but are currently missing from Tangere. Basic examples are sharing and bookmarking of visualizations, where more significant examples include undo and search. For novices, the presence of undo can significantly alter the initial exploration and freedom with which they experiment with the system.

5. Interface and interaction design: The two systems differ considerably in terms of the interface design. Several of these differences are high level. For example, in Tangere, the emphasis is on presenting the overview. Thus, in a barchart with a very large number of categorical attributes, the bars appear extremely thin. Users have to zoom in to select an individual glyph, which adds steps. Conversely, in Vizable, bars are always of the same thickness and are placed in a scrolling view that may require substantial dragging. Therefore, ensuring usability of each glyph is preferred over providing overview.

Another example concerns the type of interactions used. Although my goal was to keep the interactions simple, some operations do require a combination of the basic interactions (e.g. drag + tap for filter or hold + drag for sort). In Vizable, however, the goal of single-step interactions is preferred over expressiveness. Thus, advanced behaviors are mostly not supported. For instance, multiple items cannot be selected and thus multiple bars cannot be filtered together in one step. Instead, each bar has to be filtered out individually.

There are several low-level differences in the interface as well. The systems employ a very different color schema, with Vizable using a white background and Tangere using a black one. These variations, while subtle, together alter users' impressions of the tool.

This examination is useful for defining the goals of the comparative study as it highlights the factors that need to be compared. Only the features that are common to both systems could be a part of the study since one system may have an unfair advantage otherwise. For instance, tasks given to the participants must not require the use of scatterplots or multiple coordinated views. Exceptions include the use of features such as undo and search that, in this case, were present in and inherent to Vizable.

Comparing the features present in both systems is a useful exercise since the systems approach the features differently. For instance, both tools provide methods for filtering and sorting data, but the implementations are fairly different. Similarly, several usability goals are also common, e.g. both systems target novice users. However, the effectiveness with which each system achieves these goals, I speculate, is different.

To address these differences between the systems, I selected the following metrics for comparison:

A. Discoverability

B. Performance time

C. Accuracy

D. Ease of Learning & Ease of Use

9.2 Methodology

The study was conducted in one phase. The study followed a within-subjects design where all participants used both the systems. The study began with each participant being provided a short overview of the goals. Subsequently, they were given an iPad with one of the two applications running. Similar to the previous study, I wanted to capture participants' natural response to the systems. Therefore, participants were not given any training on either system. Instead, participants were given five minutes to explore the system on their own.

Next, they were given a series of tasks to complete with the system. These tasks were modeled on the ones present in the previous study. The number of tasks, however, was fewer (twelve) and participants had 25 minutes to complete the tasks. Once completed, participants completed a survey capturing their perception of the interface for ease of learning and ease of use. Next, participants repeated the same steps on the second application that presented a different dataset.

At the end of the study, I asked participants to provide feedback on their experience with the two tools. Specifically, I encouraged them to elicit how the tools were similar and how they differed, and what the strengths and weaknesses of each were. I also asked them to describe scenarios in which they would prefer one tool over another.

9.3 Participants, Tasks, and Datasets

For this study, I recruited 16 participants (10 male, 6 female) with similar background and experience as the previous study. They were also recruited in the same manner

Table 7: Discoverability score for features in Tangere. The visual cue column represents if the presence of the feature is indicated by a UI control. The interaction type column represents the type of interaction the feature utilizes. Basic interactions are tap, pan, & pinch, whereas compound interactions are all gestures that use a combination of the basic interactions. The count column represents the number of participants (out of 16) who discovered the feature. OE: discovered during open-ended exploration. TA: discovered during task analysis.

Operation	**Visual Cue**	**Interaction Type**	**Count** (OE/TA)
High			
Add charts	y	basic	**16** (16/0)
Change attribute	y	basic	**16** (16/0)
Filter through menu	y	basic	**16** (16/0)
Access data table	y	basic	**15** (15/0)
Change aggregation	y	basic	**15** (9/6)
Select by lasso	n	basic	**13** (11/2)
Select by tap	n	basic	**13** (9/4)
Toggle filter badge	n	basic	**13** (9/4)
Brushing and Linking	–	–	**12** (10/2)
Filter through selection	y	basic	**12** (7/5)
Medium			
Remove a chart	n	basic	**11** (6/5)
Reset all filters	y	basic	**10** (7/3)
Remove filter badge	n	basic	**10** (2/8)
Select by rectangular window	n	basic	**9** (6/3)
Low			
Zoom	n	basic	**6** (4/2)
Data count	y	–	**5** (4/1)
Split line chart by category	y	basic	**5** (1/4)
Zoom by axis	n	basic	**4** (4/0)
Pan chart	n	compound	**1** (1/0)
Sort bar chart	n	compound	**1** (1/0)
Preview X or Y axis graph	y	basic	**0**

Table 8: Discoverability score for features in Vizable. The visual cue column represents if the presence of the feature is indicated by a UI control. The interaction type column represents the type of interaction the feature utilizes. Basic interactions are tap, pan, & pinch, whereas compound interactions are all gestures that use a combination of the basic interactions. The count column represents the number of participants (out of 16) who discovered the feature. OE: discovered during open-ended exploration. TA: discovered during task analysis.

Operation	**Visual Cue**	**Interaction Type**	**Count** (OE/TA)
High			
Access both charts	y	basic	**16** (16/0)
Open filter menu	y	basic	**14** (14/0)
Change attribute from menu	y	basic	**14** (11/3)
Change aggregation from menu	y	basic	**14** (6/8)
Medium			
Select on Linechart	n	basic	**11** (8/3)
Filter from menu	y	basic	**11** (7/4)
Undo	y	basic	**10** (9/1)
Filter with swipe	n	basic	**10** (8/2)
Zoom Linechart	n	basic	**10** (5/5)
Data count	y	–	**10 (9/1)**
Sort bars	n	compound	**7** (6/1)
Toggle filter badge	n	basic	**7** (4/3)
Low			
Remove filter badge	n	basic	**6** (4/2)
Change attribute with swipe	n	basic	**5** (3/2)
Add column with menu	y	basic	**5** (0/5)
Add column with pinch	n	basic	**4** (2/2)
Sort by multiple columns	n	compound	**3 (2/1)**
Change aggregation with gesture	n	basic	**1** (0/1)
Switch attribute type (Q $\leftrightarrow$ C)	n	compound	**0**

as the previous study and consisted of graduate students from Georgia Tech who had previously completed a course in visualization. The study took about one hour to complete average. I video recorded the sessions for later perusal. For their time, participants were compensated with a $20 Amazon gift card.

To balance the two interfaces, I used two publicly available Tableau datasets — Coffee sales and Superstore sales. I designed the study following a latin square design. I divided the participants into four groups of four and assigned each group a unique combination of dataset, interface, and order. Half the subjects experienced coffee-sales dataset with Tangere and half experienced it with Vizable; half the subjects used Tangere first, while half the subjects used Vizable first. For the two datasets, I created a set of matching, synonymous tasks. The tasks resembled those I used in the previous evaluation[1].

Similar to the first study, I also captured participants' knowledge and experience with visualization systems in general and tablet-based visualizations systems in specific. Seven out of sixteen participants mentioned that they had a good understanding of visualization systems, whereas the other nine stated a fair knowledge. In terms of the systems, most of them had used Tableau and Excel in the past year, while a few had used D3, Spotfire, and QlikSense. Finally, fifteen of the sixteen participants had low or no experience with tablet-based visualization systems.

9.4 Results

9.4.1 Discoverability

Tables 7 & 8 present the discoverability results for a subset of features for the two systems. Each table lists a count of the number of participants who discovered the different operations provided by a system. For Tangere, a majority of features achieved

[1]Appendix A presents a snapshot of these tasks.

medium or high scores. The outcome matches that from study one, with the exception of the filter menu. I added a handle to the filter menu and as a consequence, most participants could discover the menu.For Vizable, the results were similar with most of the key features achieving medium or high discoverability.

WIMP vs. gesture In examining the attributes of the design that assisted or hampered discoverability for novices, the results were fairly expected. Unsurprisingly, the primary factor to positively affect discoverability was simplicity. Operations that used basic interactions (tap, pan, and pinch) or had clear visual cues were easily discoverable. Conversely, features that were invisible (e.g. filter menu), those that used complex interactions (e.g. hold+drag), or ones that required contextual operation (e.g. drag directly on axis) were difficult to discover.

If designing for novice users, the results seem fairly obvious. However, these results conflict with those observed in another study [29], where trained participants were both better at and preferred gesture-based operations over WIMP-based. As that study's authors point out, efficiency with gestures may not translate into discoverability and learnability. An implication of these observations is the emphasis needed for designers to establish early on whether users are expected to receive training or not.

Familiarity Features adopted from existing applications fared well with discoverability. For instance, in Tangere, participants versed with Tableau could comprehend and typically attended to the 'Keep only' and 'Remove' options that appeared when glyphs were selected, while participants without such experience tended to ignore these options. Similarly, removing a view or filter by swiping on a badge was easy to discover in both systems for participants who had experience with applications on iOS.

Contextual gestures Features that required contextual actions did not perform well on discoverability. A common example for both system includes contextualized drag gestures. These gestures require touch to occur at specific locations, followed with movement in specific directions. In Vizable, people can switch the attribute in a barchart by swiping left on the column header or change the aggregation by swiping down on it. In Tangere, people can drag a finger on the axis to create a range selection. Both these features had low discoverability, and were often discovered only serendipitously. The corresponding visual elements in both systems do not contain affordances that indicate the presence of these features.

The design of such affordances is a non-trivial task, however. Further, as I observed, once discovered, the operations are easy to learn and operate, further reducing the incentive for more explicit affordance. The solution Vizable adopts is to add redundancy — both the attribute and the aggregation can be changed from within a menu. Performing this action takes more number of steps, but certainly supports the use case of novices.

Although redundancy is an option, the feature density in visualization systems makes supporting it difficult to achieve comprehensively. It would, perhaps, be more suitable to improve the design of the operations with better affordance and more appropriate feedback.

Complex action In Tangere, only a few participants could discover sorting on barchart. The gesture I used for sorting was hold-and-drag on the axis [103], and participants likely did not discover it because of its uniqueness within Tangere and to iOS in general. Instead, some participants tried to sort the bars by sorting the data table, while others attempted to manually drag the tallest bars to one end (i.e. by demonstration [105]).

Posture and grip I also recorded how participants held the tablets and operated them over the course of the session. Participants exclusively used the fingers on only one hand to perform gestures and touch operations. For much of the sessions, they used the tablet by placing it on a table. However, they also spent a significant time holding the tablet with their non-dominant hand. Both systems supported one-handed operations. In Tangere, I had selectively accounted for one-handed input and carefully designed the interactions for the same [103].

Exploration approach I observed two approaches in which participants explored the system. The first category of participants spent a majority of their time interacting with features that they anticipated would exist, and found working the way they expected. Such features included lasso-based selection and filters. The second category of participants were more experimental. They quickly moved beyond what they already knew and tried to discover the other features of the systems. While the second category ultimately discovered more of the systems, the first category provided better insights into what people expect in such a system and how they translate their existing knowledge to the new environments.

9.4.2 Performance

Table 9 presents participants' performance on each question across the two interfaces. On average, participants took slightly lesser time to answer questions on Tangere compared to Vizable. However, large individual differences emerged in the time taken by the participants, resulting in a high variance ($sd = 27.03$s) in the data. For this reason, the data cannot be analyzed at the question level to understand where the difference stems from.

Table 9: Participants' average time taken and standard deviation (in seconds) for each question across the two interfaces. While the differences stand out for a few questions (highlighted in color), the standard deviation is too high for the results to be meaningfully interpreted. V: Vizable, T: Tangere

Question	**V time**	**V std. dev.**	**T time**	**T std. dev.**	**Difference**
Q1	73.75	*73.49*	66.20	*72.26*	7.75
Q2	57.75	*37.83*	51.42	*50.93*	6.33
Q3	44.10	*28.87*	65.82	*52.33*	**-21.72**
Q4	89.40	*47.86*	98.75	*80.90*	-9.35
Q5	91.30	*47.49*	37.17	*27.87*	**54.13**
Q6	115.92	*66.32*	91.25	*49.63*	**24.67**
Q7	66.73	*73.05*	58.27	*51.99*	8.46
Q8	104.22	*77.47*	77.36	*36.26*	**26.86**
Q9	76.30	*81.28*	94.20	*36.09*	**-17.7**
Q10	30.20	*36.85*	75.22	*68.95*	**-45.22**
Q11	31.13	*38.11*	30.88	*21.61*	0.25
Q12	28.40	*35.12*	27.71	*31.39*	0.70
Average	67.42	53.65	64.49	48.35	2.93
Average when first	60.69		58.50		2.19
Average when second	70.42		68.55		1.87

9.4.3 Accuracy

I used a scoring system where I assigned one point for each task participants completed correctly, a half point for tasks attempted, and zero otherwise. Table 10 presents the number of questions participants attempted and answered correctly using the two interfaces. While the number attempted is roughly the same for the two interfaces, the number answered correctly is lower in Vizable. Also interesting are the results when the order is taken into consideration. I originally speculated that, due to learning effects, participants' performance on the interface they used second would be better than the one they used first. However, the results are to the contrary. In both orders of interfaces, the performance on Vizable was lower than that on Tangere.

Table 10: The number of questions attempted (out of 12) on each system and the number of questions answered correctly. While the number of questions attempted on both the systems were the same, participants' scores for these questions were lower on Vizable, irrespective of which system they used first

Interface	**Questions attempted (out of 12)**	**Questions correctly answered (out of 12)**
Vizable	9.3	**5.4**
Tangere	9.3	**6.7**
Vizable first	8.6	4.4
Tangere first	9.5	8.0
Vizable second	9.9	6.4
Tangere second	9.1	5.3

Table 11: Average score of participants on the 12 questions they solved with the two systems. I assigned one point for each task participants completed correctly, a half point for tasks attempted, and zero otherwise.

	Score		
Question	V	T	Δ
Which specific product type has most number of transactions with sales of 500$ or more?	0	0.72	**-0.72**
Is there a relation between the profit and sales?	0.6	0.96	**-0.36**
For the year 2014, what were the total number of transactions made?	0.63	0.89	**-0.26**
Which product was sold in the transaction that generated maximum profit?	0.48	0.73	**-0.25**
On which date did California's highest sales transaction occur?	0.35	0.59	**-0.24**
Which year was the most profitable? Which month within that year in particular? Does the same month tend to be the most profitable across all years?	0.56	0.7	-0.14
How many products are of type coffee? What are their names?	0.83	0.96	-0.13
Which product types have sold the most and the least number of times? How about in August?	0.48	0.5	-0.02
How have the sales for the different regions changed over time?	0.75	0.67	0.08
On an average, how much profit was earned from each transaction of the products of type 'herbal tea'?	0.93	0.79	0.14
Is there a product type whose sales have been increasing over time?	1	0.75	**0.25**
Is there an yearly pattern to the profits?	0.75	0.43	**0.32**

Table 11 presents the scores broken down at the question level. The table only presents the questions for one dataset; questions for the other dataset were synonymous and similarly worded. The difference in scores is notable for seven different questions. Below, I examine these questions in detail.

- *Which specific product type has most number of transactions with sales of $500 or more?* Δ = **-0.72**

  Participants did better for this question on Tangere because Vizable does not provide quantitative filters. As a result, restricting the data to only items with sales higher than 500 was considerably difficult to achieve. The current method to do this in Vizable is to convert the quantitative attribute to categorical attribute and sequentially filter out all the bars with value less than 500. In Tangere, one can either use the scatterplot to select all values higher than 500 and press 'keep-only' or use the range slider in the filter menu.

- *For the year 2014, what were the total number of transactions made?* Δ = **-0.26**

  Filtering the data to contain items for only 2014 was fairly easy for participants to achieve on both interfaces. Counting the number of items that remained in the dataset, however, was a step that caused confusion. Both interfaces contained UI labels to depict the count of items in the dataset. On Vizable, the label was presented at the bottom left (Figure 48b), while in Tangere, the label was in the top right (Figure 48a). For both systems, participants either did not see the label or did not register it. In Tangere, however, participants made use of the alternate method of count — scrolling to the bottom of the data table to use the index of the last row. Vizable did not contain such a table, and there was no other alternate for count.

- *Which product was sold in the transaction that generated maximum profit?* Δ

$= \mathbf{-0.25}$

On which date did California's highest sales transaction occur? $\Delta = \mathbf{-0.24}$

Is there a relation between the profit and sales? $\Delta = \mathbf{-0.36}$

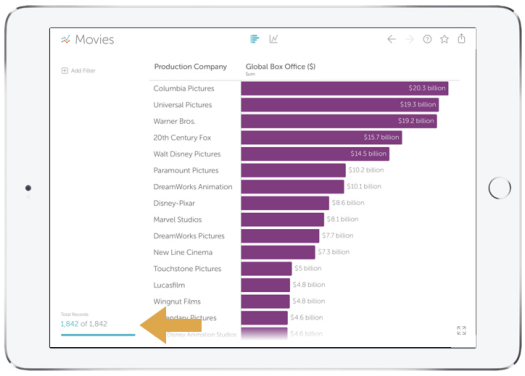
For each of these three questions, it was considerably easy to find the answer on a scatterplot present in Tangere but absent in Vizable. That said, the questions, specifically the first two, could also be solved using a barchart. Given the very typical nature of these questions, I deemed it important to include them in the study. However, Vizable was at a clear disadvantage due to a missing scatterplot, and thus I believe the emphasis on the difference in scores should be low.

- *Is there an yearly pattern to the profits?* $\Delta = \mathbf{0.32}$

  Is there a product type whose sales have been increasing over time? $\Delta = \mathbf{0.25}$

  For these two questions, the answer could be found using the linechart. The implementation of linechart in Vizable was better and considerable more usable than ours, which biased the result in Vizable's favor. I discuss this in detail in Section 9.4.6,

(a) Count label appears on the top right in Tangere.



(b) Count label appears on the bottom left in Vizable.

Figure 48: A majority of the participants could not locate or interpret the labels representing the size of the active data in the two systems.

Table 12: Ease of use and ease of learning metrics for the two interfaces (T: Tangere, V: Vizable). The table highlights the overall scores and the scores adjusted for the order effects. Scale used: 1: Strongly Disagree, 5: Strongly Agree

Questions	**Overall**		**When 1st**		**When 2nd**	
	T	**V**	**T**	**V**	**T**	**V**
Ease of use:						
The system was easy to learn	4.00	2.81	4.25	2.75	3.75	2.88
The system was easy to understand	3.88	2.63	4.00	2.50	3.75	2.75
I liked learning the system	4.25	3.25	4.75	3.50	3.75	3.00
I felt comfortable learning the system	4.06	3.25	4.38	3.25	3.75	3.25
The system is no more difficult than other visualization systems	3.19	2.94	3.25	3.13	3.13	2.75
I often became confused learning the system (higher is better)	**3.38**	**2.44**	3.75	2.38	3.00	2.50
It took too much time to learn the system (higher is better)	3.75	2.94	4.00	2.63	3.50	3.25
The ideas behind the system were easy to appreciate (High good low bad)	4.13	3.25	4.25	3.13	4.00	3.38
The time and effort learning the systemwere well spent	3.75	3.06	4.25	3.00	3.25	3.13
	3.82	**2.95**	**4.10**	**2.92**	**3.54**	**2.99**
Ease of learning:						
It was easy to use the system	3.75	3.13	4.25	3.25	3.25	3.00
I liked using the system	4.00	3.25	4.63	3.50	3.38	3.00
I had no difficulty understanding how to use the system	3.25	2.44	3.50	2.50	3.00	2.38
The set of operations one needed to use were easy to remember	4.00	3.19	4.00	3.00	4.00	3.38
It was obvious what to do next	3.19	2.75	2.88	2.63	3.50	2.88
I became confused trying to complete the tasks (higher is better)	**3.31**	**2.44**	3.63	2.50	3.00	2.38
I felt frustrated using the system (higher is better)	3.63	2.38	3.75	2.50	3.50	2.25
I felt comfortable using the system	3.88	3.00	4.13	3.25	3.63	2.75
The system was fun	**4.06**	**2.94**	4.63	3.00	3.50	2.88
It took too much time to use the system (higher is better)	3.63	3.00	3.75	3.25	3.50	2.75
	3.65	**2.84**	**3.91**	**2.94**	**3.39**	**2.74**

9.4.4 Ease of Learning and Ease of Use

Table 12 presents participants' responses to a subset of questions on the ease of learning and the ease of use of the two systems. Overall, participants seemed to find Tangere easier to use and easier to learn than Vizable. This is better evident in the comparison of the scores participants gave to the tools they used first (column 4 & 5).

From my observations of participants' usage of the two systems, the difference in scores can be attributed to two factors — how the systems respond to low-level interactions, and how they support low-level tasks. In Vizable, not every element on the screen responds to actions such as taps and swipes, which tend to be the predominant actions participants perform when experimenting with an interface. Conversely, in Tangere, interactions such as tap, double-tap and swipe are operable on most of the elements. As a result, the interface feels more responsive and accessible.

Similar differences also exist between how the tools support low-level tasks. For instance, Vizable does not support selection of glyphs. Vizable's primary view is a barchart, but since selection is not supported, the only interactions possible on the bars is swipe. Consequently, most of participants' time and interactions were spent on the filter menu on the left. This impressed a higher relevance of the filter menu to the workflow than the tasks demanded, i.e. the interaction design assigned an importance to the features that may not match the actual relevance of the features. Further, with support for simpler operations missing, such as tap and double-tap, the system felt more complex to use.

9.4.5 System Preference

At the end of the study, I collected data on how participants compared the two tools. Table 13 presents their preference on ease of learning, use, and complexity. Overall, participants preferred Tangere to Vizable, finding it to be both easier to learn and

Table 13: Participants' preference for the two systems. The value in the cells represents the number of participants who answered Tangere or Vizable for the specific question.

Question	**Tangere**	**Vizable**
Which system was easier to learn?	14	2
Which system was easier to use?	13	3
Which system was easier to understand and interpret?	11	5
Which system has a more complex interface ?	6	12

easier to use.Participants primarily differentiated the system on intuitiveness and aesthetic appeal, as illustrated in the comments below.

> P13: "*I liked Tangere better because it was easier to use and I understood faster how to use it. Vizable was prettier but wasn't very useful. It was also less intuitive than Tangere.*"
>
> P1: "*I felt like (Vizable) compromised aesthetics for function. Often times I wasn't sure what I could or should do next.*"
>
> P9: "*I liked the intuitive interface and flexibility of Tangere. I was able to quickly learn how to navigate the features, and was able to find a way to answer all the questions in the task.*"

9.4.6 Qualitative Observations

Hacking discoverability A key factor that affects discoverability is familiarity. Familiarity leads to people seeking an expected operation or attempting a known interaction in a new environment. I typically use this property to replicate common behavior. However, it can also be used as a "hack" to force a desired outcome. For instance, in Vizable, the designers opted to use a pinch-to-zoom gesture to add columns to a barchart. Such a use of the gesture is fairly unconventional, since the degree of compatibility between the pinching action and add-column behavior is very

low [9]. I argue that the resulting design suffers from low usability, as it lacks any affordance or indication that the pinch gesture would result in the addition or removal of a column of bars. Participants' confusion with the operation was apparent. P5 mentioned "*The zoom-in/zoom-out feature for barchart is really weird. I understood it is used to get more insight on sub-column (category → subcategory) but sometimes I got new unwanted columns"*. It is notable that the participant describes the operation as 'zoom-in/zoom-out' instead of 'pinch gesture'.

Ultimately, Vizable attempts to leverage people's familiarity with the pinch interaction. By combining it with an outcome that should be fairly comprehensible, the goal is to result in a feature that can be serendipitously discovered. Given sufficient operation, the behavior can also be learnt. However, it is difficult to argue for this style of design. The use for add-column is both different from the form of the actual gesture (the pinch action), and from the familiar and accepted use of the interaction.

Consistency Consistency is often cited as a theme essential to good design [85]. However, Tangere and Vizable adopt very different approaches to consistency. For Tangere, maintaining consistency between techniques was fundamental to my design process [103]. The three visualization techniques in the system support the same set of operations, and use the same interactions throughout. Vizable's approach to consistency is considerably different. The two techniques, barchart and linechart, exist independently in separate tabs and do not share any common operations or interactions. While linechart supports selection and zoom through tap & drag and pinch, the barchart does not support either of these operations and instead provides filtering, sorting, and adding a column or bars using a similar combination of gestures.

Naturally I were interested in assessing if the two approaches resulted in a different experience. My analysis suggests that the lack of consistency within Vizable negatively affected participants' perception of the system. It was also one of the key

Table 14: Order effects in the discoverability metric. The value represents the number of participants (out of 8) who discovered the feature when the system was used first or second.

	Order	
	First	Second
Vizable		
Filter with swipe	3	7
Remove filter badge	1	5
Toggle filter badge	2	5
Change attribute with swipe	1	4
Tangere		
Select by tap	8	5
Select by lasso	8	5
Interpret data count	5	0

reasons participants found the interface to be complex, non-intuitive, and difficult to learn (P14: "*Learning period in the Vizable system is a lot. Some of the features are not apparent immediately and intuitively.*").

Inconsistency also has other drawbacks besides complexity and reduced learnability. An inconsistent behavior introduces a perception of dissociation or disjointedness between the fragments of the system, wherein people might understand the views as separate subsystems. For instance, participants were unclear if operations such as selection or filtering translated across the views. Such limitations would only amplify when the system is expanded to more techniques, or views that are colocated and coordinated, such as in Tangere.

Order effects In capturing discoverability, I observed interesting order effects. Table 14 presents features that were discovered by at least three (∼20%) additional participants in one order condition. Results indicate that the difference occurred exclusively in the condition when Tangere was used first. In other words, using Tangere

first improved the number of features participants discovered in Vizable. Conversely, using Vizable first reduced the number of features a participant discovered in Tangere.

The difference, in all likelihood, emerges from the change in user expectations. Using Tangere first primed participants to expect certain features and interactions within Vizable. For instance, the filter badges in Tangere were interactive and could be tapped and horizontally swiped to toggle and remove the filter, respectively. Consequently, more participants attempted these interactions on the filter badges in Vizable. In general, Tangere more broadly supports interactions directly on glyphs. I hypothesize that this led to participants more easily discovering features such as filtering and changing attributes by swiping on interface elements. Using Vizable first similarly affected scores for Tangere. Since Vizable does not support selection, fewer participants attempted to select glyphs within Tangere, negatively affecting its discoverability.

The change in discoverability scores underscores the need for the features to be designed better, and with more clear affordances. While we can leverage people's familiarity with existing interfaces, it would be detrimental to assume it. The responsibility of priming a user to expect a feature, instead, lies with the interface itself.

Undo One of the biggest differences that I could perceive in participants' experience with the two systems stemmed from the presence of the undo operation in Vizable. As results suggest, participants found Tangere to be easier to understand and operate. The design of Vizable is fairly different from the visualization tools we use on desktops such as Tableau and Spotfire. This difference was apparent in participants' initial unfamiliarity with Vizable.

However, it was interesting to observe that this complexity of interface rarely translated into frustration for the participants. A big contributor to this was the

presence of the undo operation. Participants used undo at will, mostly when the interface did not operate the way they expected it to. Undo easily fixed all the gaps that existed in their comprehension of the system. Tangere did not have that benefit since undo was not available. As a result, each accidental activation of a feature required careful actions to undo its effects.

In an earlier discussion on the need to support undo, one argument I had made was that if the operations are designed such that they are truly reversible (as stated in the definition of Direct Manipulation [111]), the need for undo is substantially reduced as users can simply reverse their actions. This model fails, however, for novice and untrained users. To get people trained on the interface, the interface needs to support exploration by encouraging mistakes. Undo helps with that tremendously.

Application specific observations Below I present other lower-level qualitative observations I made about the two systems.

- Vizable

    1. Participants were confused between Vizable's filter panel and change attribute panel. They appeared at the same location and with the same slide-in animation, used the same visual style, and contained the same list of attributes. Thus, it was easy to mistake one for the other (Figure 49).
    2. The stability and polish of Vizable certainly helped with the overall impressions. The system rarely crashed, even with very large datasets. The experience was fluid and responsive. Tangere clearly missed the polish, and this was particularly apparent in observations of participants who used Tangere second.
    3. In the linechart, Vizable uses bars as glyphs for representation, with the line in the chart connecting the top of the bars (Figure 50b). The bars in

the linechart, however, confused several participants as there already was a bar chart.

- Tangere

    1. Participants appreciated being able to brush between views, even though the questions were designed such that they did not need to use the feature, given it was missing in Vizable.
    2. Since the filter panel on the right had a handle, participants expressed a need for a similar handle for the 'add-chart' panel on the left. Although they could operate the panel using the swipe-in gesture, they found the handle missing.
    3. Some participants found removing a chart to be cumbersome (Figure 51). While roughly half of the participants did not discover the feature, of those that did, several expressed that the operation could be made simpler, for example by using a gesture directly on the chart.
    4. A subset of the participants also complained about the choice of colors in Tangere, expressing that they would prefer a white background over a black one (Figure 50). This feedback was missing in the first study, so was clearly influenced by the Vizable system.
    5. The implementation of linechart in Vizable was different from the implementation that I had in Tangere. In Vizable, the linechart used semantic zooming, thus the view at each state displayed aggregated values. In Tangere, the linechart displayed each event individually, aggregating only at the level of each day. Thus, the view was too noisy, and ended up being less useful for participants. Vizable's implementation is certainly more user-friendly, even if it is difficult to observe the noisiness of the data in the overview (Figure 50).

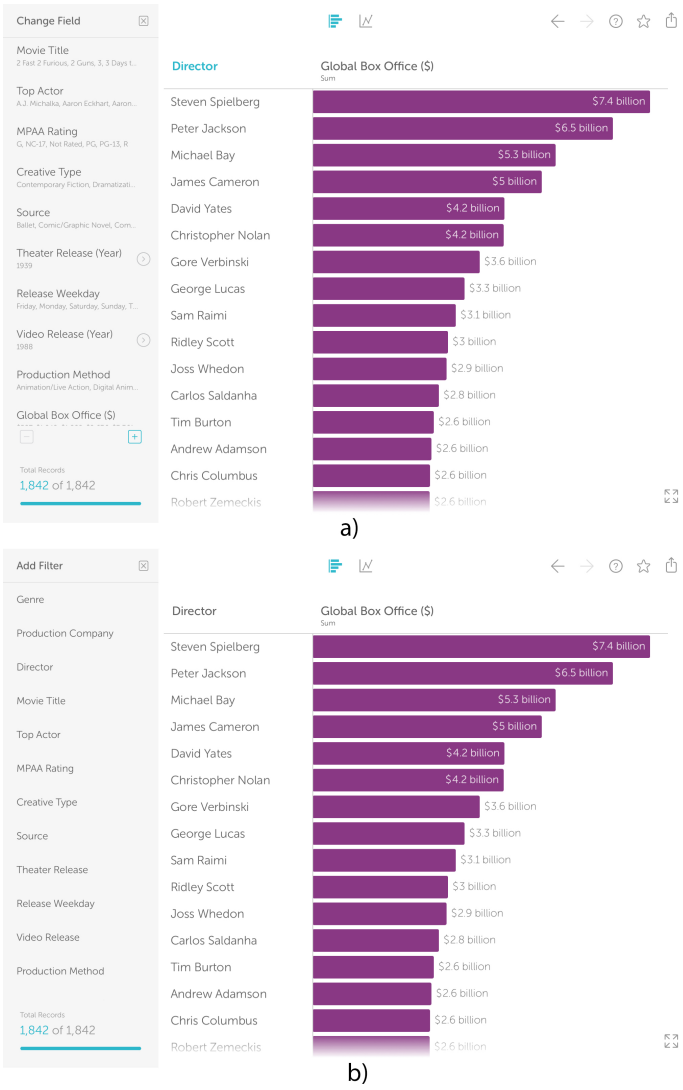


Figure 49: Similarity between Vizable's (a) filter panel and (b) change attribute panel. Both panels appear in the same location on the left and with the same slide-in animation. They also use the same visual style, and contain the same list of attributes. Thus, participants often mistook one for the other.

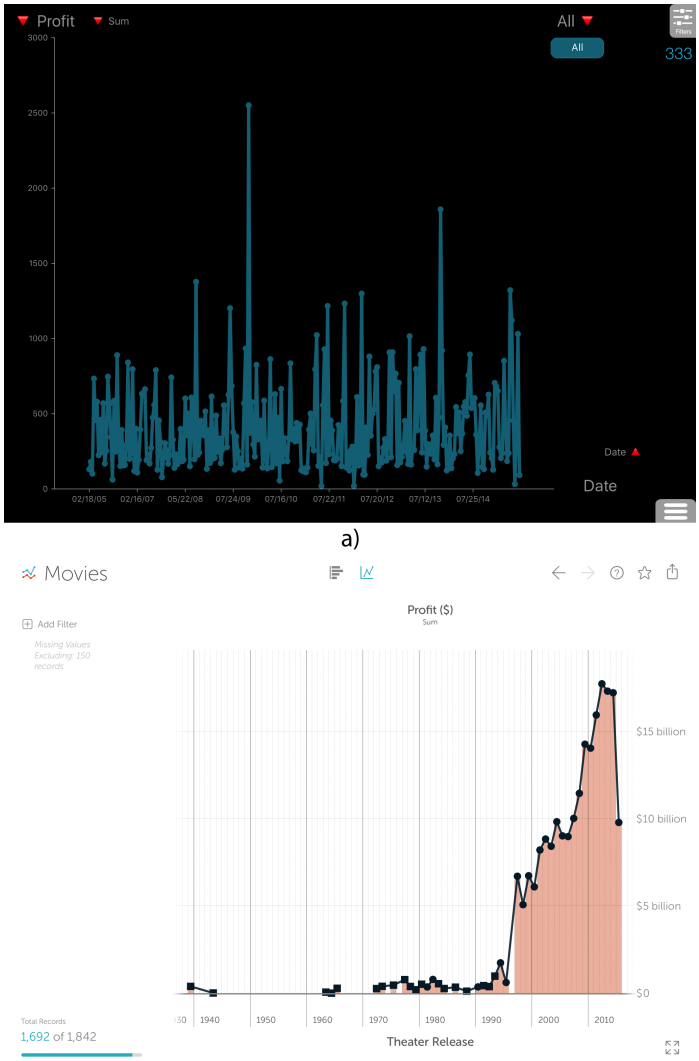


Figure 50: Linechart view in a) Tangere b) Vizable. In Vizable, the linechart uses semantic zooming, thus the view at each state displays aggregated values. In Tangere, the linechart displays each event individually, aggregating only at the level of each day. Thus, the view is noisy, and ended up being less useful for participants.

Figure 51: A chart can be removed by dragging the corresponding badge in the add-chart panel towards the right or left. Participants did not find this interaction to easily discoverable

CHAPTER X

EVALUATION 3: TRAINED USER WORKFLOW ANALYSIS

In the first two studies, my goal was to evaluate the design and functionality of Tangere by capturing how novice users would approach the tool. In the final study, my goal was to understand how a user who is trained on Tangere uses it for analyzing data. As people gain experience, a well-designed interface often ceases to exist individually and instead blends seamlessly into the workflow. Conversely, poorly designed interfaces hinder this smooth functioning. With this study, I wanted to capture how well the system aids in letting users complete a workflow smoothly. A user profile that I intended to target was that of an early adopter - someone who willingly and excitedly agrees with the motivation of the tool, gains proficiency on it, and buys into using it despite the flaws that may exist.

To this end, I designed a study where I captured people's usage of the system several times over separate days. My objectives were threefold. First, I wanted to understand the workflow of a user operating the system and determine the aspects of the system that make or break it. This analysis of the workflow is relevant because it helps me validate my assumptions regarding people's use of the system. For instance, when designing the system I assumed that the user would be trained and familiar with the tool. In the earlier study that I conducted, this was not ensured. In fact, users were specifically not given any training. As a result, the observations I made were devoid of any insight into how the behavior of a user changes after spending a significant amount of time with the system.

Second, I want to better define the immediate next steps for the design of the

system. Here, I was specifically not seeking new features to add. Instead, my goal was to extract behaviors that already exist within the tool, but are cumbersome to perform. For instance, if a task currently requires several steps to complete, and the user performs the task several times in one session, there is value in simplifying that workflow and optimizing it.

Third, by observing how participants incorporate system's features for completing analytical tasks, I wanted to capture the memorability of the system for a trained user. A three-phase study with phases spread across several days helps with capturing this detail. Unlike the previous studies, participants in this study would be given ample training on the system. Thus I expected the amount of learnt behavior and recallability to differ compared to the previous studies.

10.1 Participants and Data

I loosely modeled this experiment on the MILC evaluation technique described by Shneirderman and Plaisant [113]. The study consisted of three sessions and between 3-4 hours spent with each participant. Since significant time was to be spent with each participant, I elected to have a small number of them. Unlike the previous studies, at the end of this study, participants were expected to deliver an artifact — a report summarizing their findings.

One other difference from previous studies was the type of data used. In the first two studies, I chose the dataset (i.e. Movies, Coffee sales, and Superstore sales) and used the same across all the participants. The datasets were generic so that their contexts were familiar to the participants. However, the relevance of the data to the participants was low. This, in turn, affected the quality of analytical questions participants asked of the data.

For this study, since participants had to spend several hours working with a

dataset, it was important that the data was relevant to them. Thus, I asked participants to bring their own datasets. For a better likelihood of participants having a dataset they are familiar with, I recruited participants from the business and analytics programs at Georgia Tech.

At the time of my initial correspondence, I shared sample data files with the participants and prompted them to send me options of datasets they would prefer to work with ahead of time. When we agreed on the dataset to use, I advised them to spend time with it before the first session to gain familiarity with it.

Overall, five people participated in the study (3 men and 2 women). Three participants were students in the MBA program, while the other two were enrolled in the MS in Analytics program. Participants had past experience with handheld touch devices, with all of them having used either iOS or Android-based smartphone as their main device for at least five years. As a result, they were versed with the basic interactions that these operating systems provide. They also had previous experience with visualization and analytics systems. All five participants had previously used Excel for plotting graphs. Three of the five had also used Tableau in the past.

The datasets they used included public health, public education, crime, human development, and video games. All the datasets had more than ten attributes, with at least four categorical and five quantitative attributes. The size of the data ranged from minimum 500 rows to maximum 1000 rows. Table 15 provides a summary of the type and size of each dataset.

10.2 Deliverable

The deliverable of this study was a report that participants had to create by the end of the final session. At a high level, the report summarized the key insights, findings, and takeaways from their analysis of the dataset within Tangere. Participants were free to use any form of documentation — they could enumerate high-level insights or

Table 15: The five datasets that participants brought for analysis with Tangere. The second column represents the number of rows in the dataset.

Dataset	**Size**
Public Health	886
Crime	968
Public Education	481
Human Development	527
Video Games Sales	921

write short summary sentences with additional detail.

My goal with the study was not to gather feedback on the design of Tangere. I also did not intend to compare the effectiveness of the tool to other existing tools, or assess whether it is an accessible 'introduction to data analysis' tool. In other words, I was neither evaluating the system, nor measuring participants' effectiveness in using it. Having a report as a deliverable helped as it communicated to the participants that their performance was not being evaluated. The report acted as a tool to encourage earnest participation as it framed participants' approach to constructing and answering questions. I did not, however, intend to use the reports for any quantitative or qualitative measurement.

I expected the participants to generate relevant insights for the report over the course of the three sessions. However, since there was no means for saving or bookmarking the state of the system, the insights across sessions had to be manually recorded in a document. I informed the participants about the discontinuity between the sessions at the beginning.

At the time of designing this study, I considered incorporating screenshots as a part of the report participants generated. For each insight that emerged, participants could take a screenshot of the view and attach it with the description within the report. However, on further exploration, I identified that the shortcut for taking screenshots on an iPad (i.e. pressing the power and home button simultaneously)

can be problematic because uncoordinated button presses lock the device. Further, the shortcut requires changing how the tablet is held, which is disruptive. Another challenge with screenshots was annotation. While it might be feasible to take screenshots, just a collection of images are not useful to a person unless they are able to differentiate them. That requires annotating the screenshots with notes or shapes. However, elaborate snapshot annotation would require a detailed feature on its own, and using the feature would also disrupt the flow of the existing system. Thus it seemed sensible to omit screenshots from the final report.

10.3 Methodology

The agenda of the sessions broadly mapped to *training*, *analysis*, and *reflection*. I describe the three sessions of the study below.

First session (Day 0) The aim of the first session was to get participants versed with the goals of the study and familiarize them to the interface. The session began with me providing the participants with an overview of the study plan and the expected outcome. Subsequently, I gave participants a thorough walkthrough of Tangere. The walkthrough consisted of detailed demonstration of each feature. After each feature, I gave participants a chance to operate the feature. They were free to ask any question they had of the interface or the feature.

Once all the features were demonstrated, participants were given time to operate the system on their own. At this stage, I used a dummy dataset. I observed how they were adapting to the system, ensuring that by the end, they had tested all the features.

In the remaining time in this session, participants used Tangere to explore the dataset they brought. This was an open-ended exploration with no predefined tasks or goals. They were free to explore the data in any depth they preferred. I encouraged participants to take notes any time they found an insight.

I interrupted participants in situations where I observed them performing an operation incorrectly or if I felt confidently that they did not remember a feature. In such situations, I demonstrated the correct behavior in the system to them. However, I did not indulge participants in any dialogue or discussions on their process and the design of the system.

Second session (Day 4-6) The second session was designed for participants to mostly spend time analyzing their data. At the start of the session, I gave participants the iPad with Tangere running and asked them to recall everything they remembered about the system. To ensure they recalled the features accurately, I nudged them to talk out loud what they expected each action to result in. At the end, if they had skipped a feature, I brought it to their attention and asked them if they could remember how to perform it.

To help structure and streamline their exploration of the dataset in the rest of the session, participants spent five minutes crafting a series of questions of the dataset that they would want to answer. I did not direct this process, but I expected their experience with Tangere from the first session and the feature-recall exercise to have a bearing on the type of questions they generated.

For the next 45 minutes, participants used Tangere to answer the questions they created. The exploration session was, again, open-ended, and participants were free to approach it in any manner they wanted. I also answered any questions they had of the interface. Participants typically asked questions regarding specific features, such as trend lines or search, that they sought but could not find in Tangere.

Third session (Day 9-15) The final session was centered around creating the report. The session began with a similar recall exercise as the one at the start of the second session. Next, I encouraged participants to spend fifteen minutes perusing their notes from the earlier sessions and answering any questions that remained.

Subsequently, they compiled their results as a report. For the final twenty minutes, I gathered feedback on participants' impressions of Tangere and the process of using it for analysis. Specifically, I gathered qualitative feedback on how their analytical process with the system differed from their process with other applications such as Excel and Tableau. I also asked them to highlight aspects of the system that they liked, did not like, and found missing.

10.4 Results

The overall outcome of the study was fairly positive. Participants could successfully use Tangere to answer the vast majority of the questions they crafted. Those that they could not were primarily ones that had the relevant functionality missing from the system.

Participants also provided an encouraging feedback at the end. They expressed a clear liking for the tool. Compared to the other tools they had experience with, such as Excel, Tableau, and Spotfire, participants appreciated the simplicity and low-barrier to entry of Tangere.

10.4.1 Data Analysis

I encouraged participants to initially define as many questions as they could. They did so while perusing the data spreadsheet that was open in front of them on a computer. Overall, each participant generated more than 16 questions. Table 16 summarizes this data for each participant.

The questions they generated were similar in nature to the ones I designed for participants in the first two studies. They contain a mix of both low-level and high-level tasks, including retrieve value, filter, compute derived value, find extremum, sort, determine range, and identify clusters. Figure 52 presents a snapshot of the type of tasks one of the participants created.

Over the course of the study, participants were able to answer the vast majority of

Table 16: The number of questions created and answered by participants over the course of the study. P3 decided against constructing questions and instead explored the data in an open-ended manner.

Participant #	Questions Created	Questions Answered
1	20	20
2	19	14
4	16	14
5	17	16

the questions they created. Table 16 presents the summary of the number of questions each participant answered. For those questions that were not answered, the leading reason was a lack of support for specific features in Tangere. For example, both P2 and P4 had questions that looked to identify correlation or trend between specific attributes ("*What is the correlation between type of crime and hour of day*"). P2's data also contained geographical information, but Tangere did not provide any geo-views. Thus, P2 was also unable to accurately answer questions that were based on latitude and longitude positioning ("*What is the distribution of crimes by latitude*

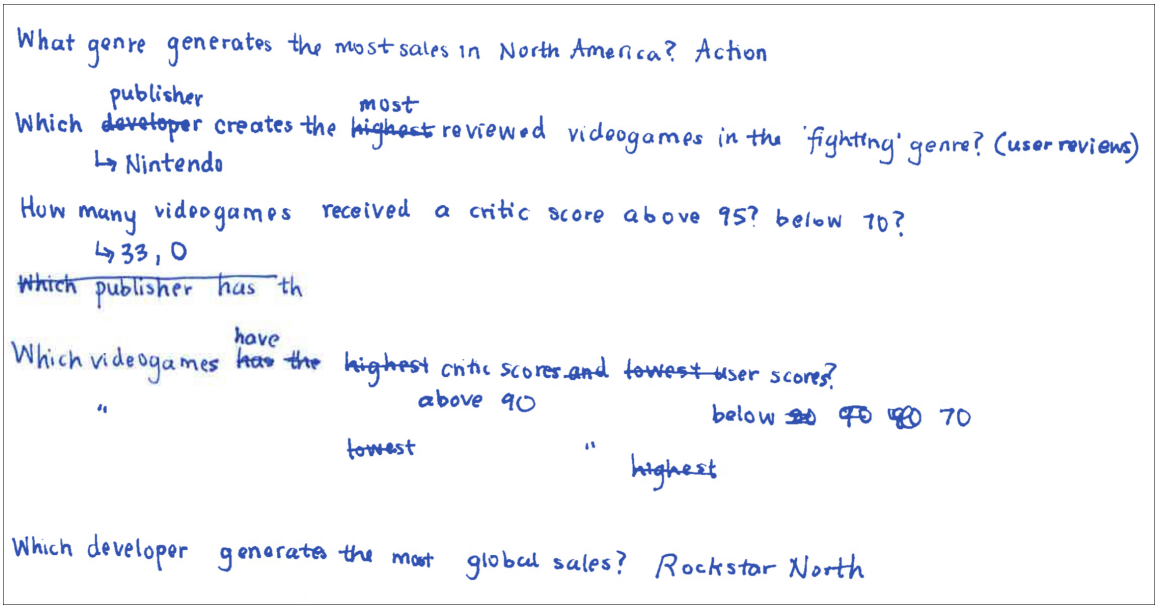


Figure 52: A snapshot of the questions created by P5 for the video games sales dataset.

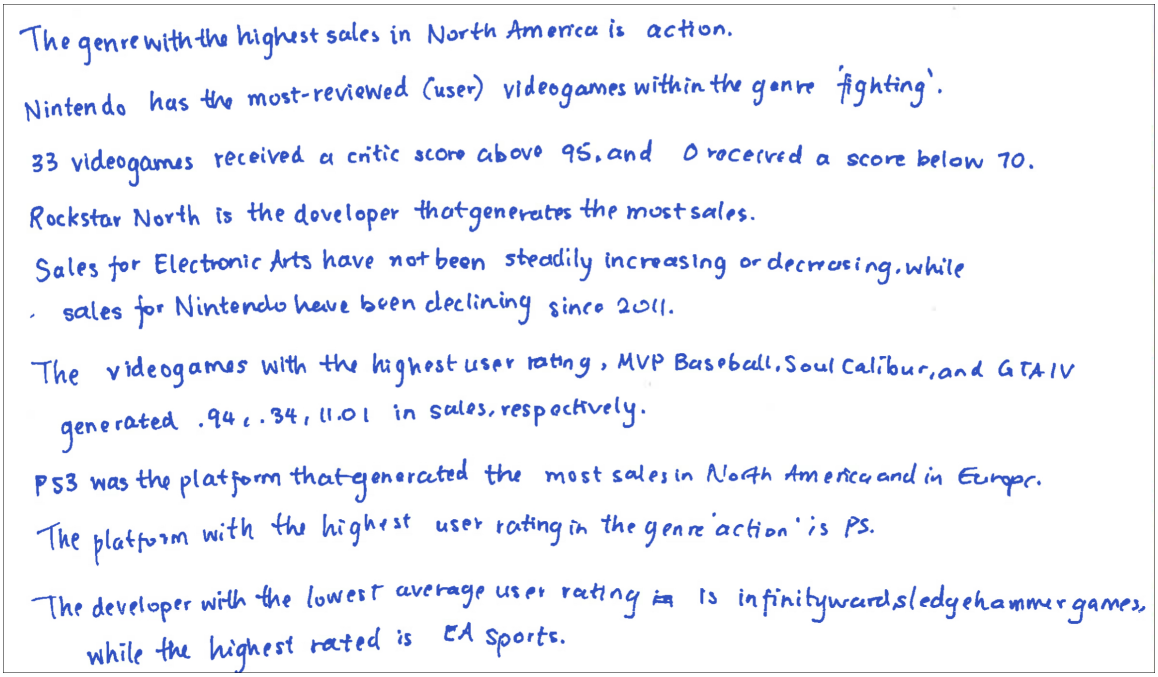


Figure 53: A snapshot of insights identified by P5 on the video games sales dataset.

and longitude").

Ultimately, participants collected all the insights and summarized them in a report. Figure 53 presents a snapshot of the report created by a participant.

10.4.2 Memorability

Perhaps the most encouraging observation from this study was the performance of features of Tangere on the memorability metric. I collected memorability scores for the different features in the first study (Section 8.3). In this study, however, participants were given training on the system. I expected training to have an effect on memorability since the exposure participants received to features was precise and accurate. Conversely, when there is no training, participants are tentative about the features they discover and verifying each feature requires repeated trials. In several cases, the repeated trials simply do not occur over the course of an open-ended exploration.

Ultimately, with training, the features performed really well on memorability.

Table 17 presents the scores for a subset of features. Most notable is the performance of the features that did not gain a high memorability score in the first study (Table 5). Critical features that received low scores earlier were sort in barchart, rectangular selection, and data count label. All of these features performed well in this study, with participants successfully recalling them in the second and third sessions.

The improvement in memorability is notable for two reasons. First, there are clear advantages of training in a system such as this. Not only does training accelerate the comprehension of the system, participants also remember the system better. This might certainly be limited by the complexity of a system, i.e. beyond a certain number of features, participants might not recall the system even with training. But, for a system as complex as the one I designed, training clearly has benefits.

Second, the features that did not do well earlier but did well in this study can certainly be designed better. I assert this based on the observation that features were easier to remember simply because participants could comprehend them better. We can aspire to achieve that level of comprehension without training, and simply through a better designed feature.

The differences in the performance of the features across the two studies can also be explained using Norman's gulf of evaluation [85] — the difference between what the user thinks happened in the system and what actually happened. In the first study, through accidental activation, participants were able to discover many features in Tangere. However, often they were unable to comprehend exactly which part of their action caused the operation to occur, and what the resulting changes to the interface denoted. For instance, rectangular selection by dragging on the axis was often discovered by accident, usually when participants tried to drag panels into view from the left or below. Since this was an unexpected behavior, with the resulting output being difficult to comprehend, rectangular selection was often not remembered as well, even if it was discovered by several participants. With training, however, the

Table 17: Memorability scores of a subset of features in the second and third sessions. Scores are the number of participants (out of 5) who could recall the feature.

Feature	**2nd session (out of 5)**	**3rd session (out of 5)**
Add Charts	5	5
Change X or Y attribute and aggregations	5	5
Select by tapping and lasso	5	5
View data table	5	5
View filter menu	5	5
Remove a chart	5	5
Brushing and linking	5	5
Keep only/Exclude filters	5	5
Rectangular selection	5	5
Sort column	5	5
Split line chart for categorical variables	5	5
Zoom	4	5
Sort barchart	4	5
'Reset all' in filter menu	4	4
Toggle filter badges	3	5
Pinch to zoom axis	3	4

confusion in comprehension was almost eliminated and the gulf of evaluation was reduced. As a result, participants could also remember the feature better.

10.4.3 Qualitative Observations

Design of barchart sort gesture Since participants were given training in this study, I was able to understand how well the barchart sort interaction performed. This was not the case in the previous studies as participants could not discover the operation.

The barchart sort gesture uses a hold+swipe gesture. To sort bars based on height, the user has to place a finger on the y-axis and hold it for 500 ms. Once past this threshold, the system enables the sort mode. Swiping the finger up sorts the bars in ascending order and swiping it down sorts them in descending order. The

sort feedback is given using gray, translucent preview bars that animate to the final positions.

At the start of the study, I found that participants learnt the gesture fairly easily and quickly. They were also able to recall it in subsequent sessions. However, interesting patterns emerged when they used the gesture for their exploration. While the hold+swipe gesture sorted the bars, swiping alone activated rectangular selection (Figure 54). I observed that participants used the two gestures interchangeably. Irrespective of the intended action, they would often try the basic swipe gesture first.

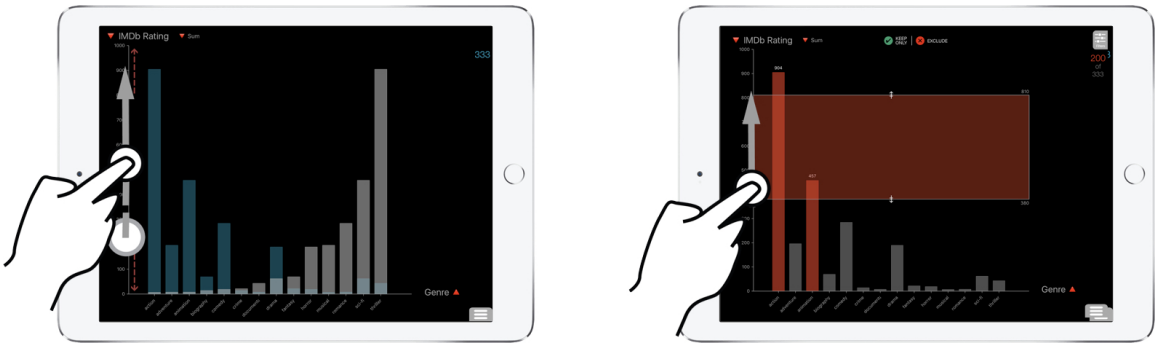


Figure 54: The two axis-based operations that used swipe. Hold + swipe activated the sort operation, while simply swiping activated the rectangular selection.

The similarity between the swipe and hold+swipe interactions, which is also the drawback of the design choice, is obvious. Swiping alone is clearly the fallback interaction in the two. This combination of gestures has been successfully employed in applications in general, particularly on Android OS. It was interesting to note, however, that the participants did not bring this out in their qualitative feedback. Even though they faced the problem repeatedly, they did not express a discomfort or suggest that either the gestures be changed or one operation be preferred for the basic swipe over the other.

Filter Menu The filter menu was a feature that saw considerable usage (Figure 55). Although participants used it repeatedly, it also garnered substantial feedback, both positive and negative. Participants appreciated the access to the filter widgets, both

for managing the data and for viewing the range of values for each attribute. There were several concerns too, however. Specifically, the sliders were difficult to operate. For quantitative attributes with a wide data range, accessing a specific value on the slider was extremely cumbersome. Some gesture-based techniques have been employed in other applications to make access to specific values feasible. However, participants overwhelmingly preferred to have a numerical input available on the control for specifying the exact value.

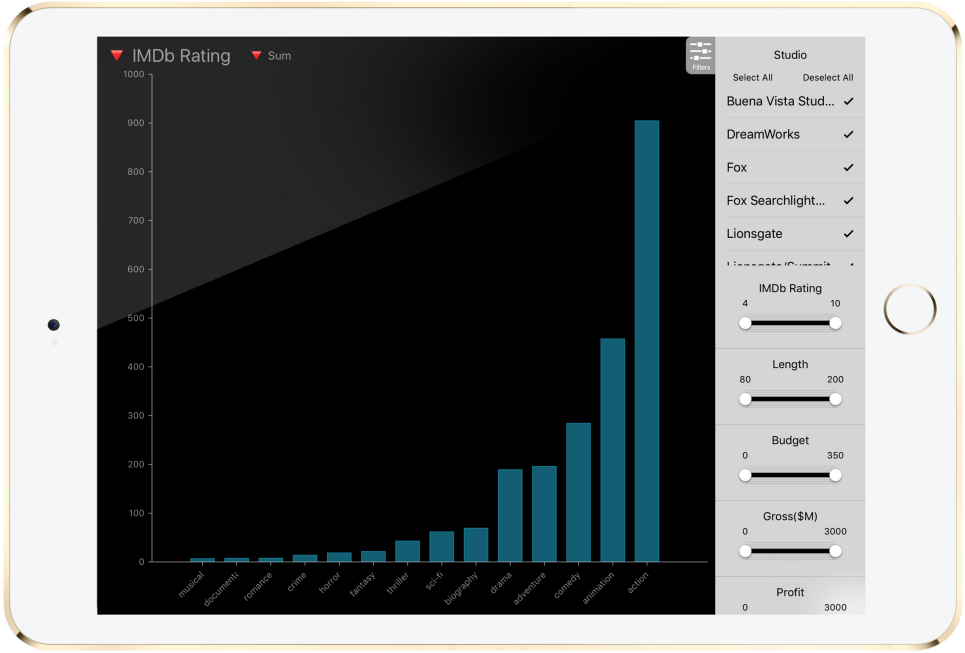


Figure 55: Filter menu. Participants were concerned with the usability of the menu. For quantitative sliders, the handles were difficult to move to a specific value. For categorical values, it was cumbersome to find one option from the large list of options.

A similar concern also arose for categorical attributes that had a large number of values. Since the values were listed individually in a vertically scrolling list, it took considerable effort to find one specific value. In such cases, filtering one value by selecting it on the chart and using 'keep-only' and 'remove' filters was often faster than using the filter widgets.

Another concern was regarding the usability of the filter menu as a whole when the

number of attributes was large. This resulted in a very long panel, and searching for an attribute in the panel took a while. Similarly, identifying which filter was currently active also required considerable scrolling. There are several methods to optimize this behavior. One method is to adopt the design used within Vizable. There, the filter menu is condensed to initially show only the list of attributes. The filter controls are instead shown when an attribute is tapped. Clearly, filtering takes more touches, but the usability of the menu is improved.

Brushing and Linking Participants made considerable and clever use of brushing and linking (B&L) between coordinated views. B&L is a feature that was difficult to test in the previous studies. Those were controlled experiments where participants had limited time to learn and spend on the interface. To fully utilize B&L, one needs to gain proficiency with the workflow of the system. After proficiency with the system, the feature can be used for a multitude of effects, e.g. quick selection and filtering. These behaviors are infused into the usage only over time.

This study gave participants sufficient time with Tangere to be able to do that. They were also given training initially during which I spent some time highlighting the feature and encouraging its importance. Ultimately, participants used B&L in more effective ways than I had expected. In several scenarios, participants used the features serially to narrow their exploration to a selective point. (e.g. to answer questions such as 'Which is the highest grossing movie from the most profitable director of action films?').

Latency with large datasets In situations when the dataset being visualized contained a large number of rows, the response time of the system was affected. This was specifically true for features that operated on the glyphs, e.g. lasso selection. This is because with each movement of the finger, the system had to reevaluate the touch position with respect to every glyph on the screen. For a very large number

of glyphs, processing the response took longer than the 30ms needed for real-time feedback.

In such situations, presenting visual feedback, such as tap ripples [128], would be beneficial. Per-contact feedback can be supported at a hardware level, or only within the application. The feedback may be distracting in situations when it is not needed, particularly since it is visually similar to the glyphs in a chart. However, the user currently receives no feedback at all, which has drawbacks when the system is not responsive.

10.5 Reflection

10.5.1 What does Tangere excel at?

To design Tangere, I began with a hypothesis on who the target user is and scenario in which the system is useful to them. Over the course of the three studies, I was able to test and refine this hypothesis. Since participants were subjected to a predefined scenario in each study, insights on the hypothesis emerged from two sources — observations to identify the exact characteristics the system excels at, and remarks made by participants in the qualitative feedback at the end of the sessions.

Looking back, the properties of Tangere that stood out for the participants, and which in turn help define the ideal usage scenario, are:

1. Speed of access - Tangere excels at the 'pick-up-and-use' scenario. The threshold of effort required at the start is low, and people are quickly able to dive right into data. Opening the application places the user in the middle of the analysis. Although this rapid access assumes that the dataset to load is known and cleanly formatted, the ease and simplicity of system's start up was particularly appealing to the participants.

   The rapidness of the experience is clearly in line with the behavior expected on mobile operating systems such as iOS and Android. Most iPad applications,

including Vizable, provide a similar experience. However, while expected, the reduced latency improves visualization application experience.

2. Quick overview - In reflecting on their experience with Tangere, several participants acknowledged that they would use the system in the first stage of a data analysis workflow. For instance, if they had a dataset that they were not familiar with, exploring the data in this system would be a good starting point. The tool provides most of the features necessary for an initial exploration.

   One feature in specific that participants appreciated was the data table. A data table is very useful for getting familiarized with the data, but is often missing in visualization tools, including Vizable. In others that include it, such as Tableau and Spotfire, the view is not very accessible.

3. Open-ended exploration - Participants often cited that exploring data with the tool was fun and gave them ideas while they were using it. Such a feedback loop, where the interface drives progress by providing cues for the next steps, is beneficial for open-ended exploration.

   Aiding open-ended analysis is a characteristic of any tool that is designed for data exploration. However, the benefit of touch-based input is that the exploration seems more immersive and direct. In line with the conversation of low degree-of-indirection of touch-based visualization interfaces (Section 5.1.5), users can feel more involved and more in control of the interface than with cursor-based systems.

4. Infrequent usage - While certain features in Tangere could be better designed, e.g. barchart sorting, a majority of the critical features needed for exploring data are robust and fairly easy to discover and operate. The accessibility of these features is a strength for people who would use the system only infrequently.

Since participants found most aspects of Tangere to be intuitive, there is less expectation for them to remember the system long term.

10.5.2 What would a redesigned system look like?

Over the course of the studies, I was able to address several issues with the design of Tangere. Thus, the system in study 2 was more stable than the one in study 1, and the one in study 3 was more improved than the one in study 2. However, there were several other observations that were more difficult to integrate since the design of the Tangere had already matured to a certain degree.

All the feedback serves the question — what would a redesign of Tangere look like? Which features would change and which would not? Which other scenarios would the system be able to address that it currently does not and how? Here are my thoughts on how I would redesign the system if I had to.

Aspects of Tangere that work well and I would keep

1. Three interface components: In a system with the four visualization types, the three components (canvas, filter menu, and data table) encapsulate a large number of features and offer an experience that is easy to comprehend.

2. Selection: Tap-to-select, lasso selection, and rectangular selection are robust and fun to operate, and cover the vast majority of the (basic) selection scenarios well.

3. Keep only & exclude filters: Filtering with 'Keep only' and 'Exclude' is very accessible. It drives the exploration with glyphs alone. Used in conjunction with selection, it reduces the usage of the data-driven filters in the filter menu.

4. Data table interaction: Easy access to data in a spreadsheet is very handy, particularly since selection and filtering brushes to the table view.

Aspects of the system that need improvement/revision

1. Filter menu: Although the component is useful, interaction with the widgets within the menu needs to be rethought. Specifically, three situations that currently are not well supported need to be addressed — precise selection of values from quantitative attributes, faster access to values of categorical attributes, and improved usability if the number of attributes is large.

2. Adding and removing charts: Tangere provided good affordances for adding charts. Consequently, participants could perform the operation fairly effectively. The same, though, was not true for the remove-chart operation. However, while the remove operation clearly needs to be rethought, I believe that the add-chart feature could also be refined further. The number of steps required can be reduced and the workflow can be tightened. Currently, there is also a disconnect between the views on the canvas and the views in the panel. When the user selects a view to add, there is no feedback on which location and in which configuration the view would be appear.

3. Zooming: Although participants could discover and operate the feature, the ease of learning and use was primarily built on past knowledge of the pinch-to-zoom interaction. In practice, though, the zooming feature saw very little usage. This is true for both zooming on the chart and zooming on the axes. Part of the reason was that the experience was jarring. On desktop-based systems, zooming is often performed with buttons. The scaling can, thus, be controlled and anchored around the center of the visualization. With touch-based pinch-to-zoom, however, the view scales around the centroid of the touch points.

   Scaling symbolic shapes such as scatterplots and linecharts is considerably different from scaling iconic content, such as pictures and maps. We are less effective at the former for two reasons. First, choosing an appropriate position

to zoom around is difficult. Only a small portion of the total area of a chart is typically covered by content such as lines, dots, and labels. Moreover, to minimize occlusion, we typically place fingers in areas that are empty. Consequently, zooming-in tends to result in mostly blank views. Reorienting the views requires a combination of panning and zooming out, which is slow and undesirable.

Second, when the view scales up, it is very easy to lose context of the areas of interest. This is because the size of the glyphs do not change, but the distances between them increase and, thus, the noticeable patterns that glyphs collectively form are lost. The situation is further exacerbated when the zooming is semantic and the view switches entirely.

To fix zoom, additional smarts need to be integrated within the interaction. Scaling the view can be made contextual such that in lieu of simply scaling the view based on user's finger movement, user's intention can be modeled and the view can be scaled based on an understanding of which content is critical and needs scaling.

4. Linechart: As discussed in Section 5, the linechart needs to be redesigned with the goal of making it more useful and usable. Currently, the chart does not provide as rich an experience as scatterplot and barchart provide. In fact, the implementation within Vizable is considerably more effective and should certainly influence the redesign of linechart within Tangere.

10.6 Conclusion

Overall, the three studies generated useful insights on the design of Tangere, highlighting specific features that were well-received and others that need further thought for future revisions. The combination of quantitative and qualitative metrics, coupled with a post-hoc analysis of the sessions, revealed the strengths and weaknesses of the

system. Overall, I believe that the results on discoverability, memorability, and the ease of use and learning from the three studies reflect fairly positively on the overall usability of Tangere.

10.6.1 Reflection on the design of the studies

Over the course of the studies, I also made several observations regarding the design of the studies themselves.

Implications of Training In reflecting on the studies, both approaches to training resulted in useful findings. While studying the usage patterns of trained users was beneficial in modeling the workflow that the system affords, understanding how untrained users approach the system generated better insights regarding the design of the system. Ultimately, gathering data using both approaches gave me the opportunity to benchmark the results against one another. Training improved usage of Tangere by providing the benefit of both accelerated comprehension and improved recall. Without training, participants were tentative about some features they discovered and verifying what each feature did required repeated trials. With training, they interpreted the features accurately without needing to verify.

It is important to consider, however, that neither of these conditions resemble how people typically receive training in the wild. For publicly available applications, people usually are not personally trained by an expert. Instead, training is provided through interface annotations on the application at first launch, or through how-to videos in the help section[1]. Leveraging such methods is important for evaluations since results would better map real-world situations. However, the challenge would be to standardize the quality of the tutorial content, given the content would differ across vendors and would be difficult for experimenters to generate on their own.

[1]Previous work [69] has explored how to make this content effective.

Ultimately, however, it is important to understand the implications of training in light of the prevailing practices in visualization system design. For desktop-class systems, often it is assumed that the end-user would receive some form of training. This has resulted in solutions that novice users often find difficult to operate and interpret [40]. In building systems for tablets, we must agree on whether a similar approach should be replicated. However, if novice users are the target audience, a promising goal for tablet-class applications, our evaluation practices need to reflect the same.

Discoverability A challenge I faced in measuring discoverability was agreeing on cues to use to mark a feature discovered. Often, features were discovered serendipitously either through accidental or imprecise touches on the screen. For example, one could drag within a chart to activate lasso selection, drag on the axis to enable axis-based selection, and drag on the edges to bring in side-view panels. Since the activation regions were connected, participants would often accidentally activate one operation when attempting another.

In such situations, it was unclear if I should mark the accidentally activated feature as *discovered*. I initially considered requiring an acknowledgement from the participants, such a reattempting the feature or through verbal confirmation that they recognize the feature. This appeared to be only semi-effective, however. It was useful for unfamiliar features, such as axis-based selection or sorting, for which participants would often reattempt the operation. For familiar features such as tap-to-select or lasso-select, however, participants would not necessarily reattempt the action, but it was difficult to ascertain if that was due to familiarity with the operation or lack of acknowledgment of it. Ultimately, I marked a feature as discovered either when participants consciously acknowledged it after the first use, or used it again over the course of the exploration.

Memorability In the third study, I captured memorability by asking participants to perform all the operations within the system running on an iPad. In the first study, I had used a different tactic to measure memorability — verbal recall with hints ((Section 8.3). In reflecting on the results from the first study, verbal recall highlighted features that participants seemed to remember existed in the system, but it did not provide insight on whether they remembered how to operate them. Participants' memory and description was mostly feature-level as they would make statements such as ("*I could select glyphs between values*" or "*there was brushing between views*"), as opposed to interaction-level description ("*I could swipe on the axis to select glyphs between values*").

I found that the in-system recall surfaced the interaction-level memory in a much better way. Observing participants operate the features provided a clear indication on the extent to which they remembered the system. The downside of in-system recall, however, is that it measures participants' memory of the system that is conflated by the cues and affordances that the system provides. However, regardless of this restriction, I found that in-system recall offered a better methodology than verbal recall as it more closely mapped to the need for memorable features in the real world, i.e. people using a system after a gap of some time.

CHAPTER XI

CONCLUSION

11.1 Summary

The goal of this dissertation was to provide a novel visualization solution to explore multidimensional tabular data on a multitouch-enabled tablet device. In Chapter 3, I discussed the space of possible visualization systems that are feasible on tablet devices. Subsequently, I provided a walkthrough of the system that I have built.

In the subsequent chapters, I provided a detailed description of the Tangere system. Addressing the different stages of my research, I presented my initial explorations of the scatterplot technique and the subsequent expansion to multiple coordinated views. In the following chapters, I presented the advanced interactive features for selection and layout modification that I designed for Tangere.

Finally, chapters 8, 9, & 10 detailed the three evaluations that I conducted to measure the effectiveness of the system. The studies qualitatively capture how well the design of the system performs individually in short and medium-term usage, and in comparison to a publicly available tool.

To conclude, the goal of this work was to design an information visualization system for tablets that replicates the analytical capabilities of desktop-class visualizations, and is simple, effective, and delightful for users. During this process, I made the following contributions to the information visualization research community:

- A classification of the opportunities and challenges for visualization on touch-based tablet devices.
- Identification of a set of appropriate and effective visualization schemes for exposing crucial insights from data on a touch-device.

- Exploration of the design space of multi-touch interactions for data-driven operations in visualizations.
- Implementation of an iPad application for novice users for analyzing data using multiple coordinated visualizations, designed to deliver simplicity and delight.
- Design of novel Clutch-based interactions for performing advanced selection on visualizations in a tablet system.
- Examination of the key differentiating features affecting user performance on tablet-based and desktop-based visualization system, and those affecting discoverability, learnability, and memorability on tablet-based visualization systems.
- Exploration of the design space of layout configurations for tablet-based multi-view visualization systems.

11.2 Future Work

11.2.1 Visualization pipeline

In their seminal work on scoping the field of Information Visualization, Card et al. [19] defined a Visualization Pipeline, describing the steps that data undergoes as it is trans- formed from raw formats into final visualizations. The input data exists in four states and undergoes four types of transformations to eventually result in a visual image. Figure 56 presents the different stages of the pipeline.

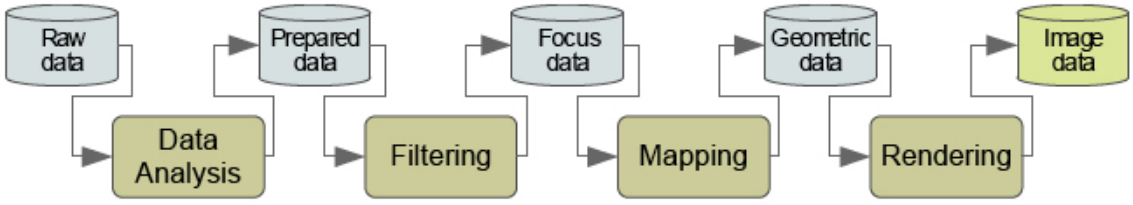


Figure 56: The visualization pipeline describing the (step-wise) process of creating visual representations of data.

In Tangere, the first two stages are currently missing. The system expects the input data to be formatted in a clean tabular structure. Missing data values are not

managed and substructures in spreadsheets, e.g. subtables and reference cells, are ignored. However, in the scenario that I hope to address with the tool, i.e. semi-professional or business users looking to analyze data on the go, expecting cleanly formatted data is a limiting requirement to have. Thus, methods for cleaning data before importing it are clearly needed. And for the experience to be streamlined, methods for cleaning data should be available on the tablet itself.

Tasks involved in cleaning include selecting appropriate attributes, identifying and correcting erroneous or missing values, and selecting subsets of larger data sources. Traditionally on PCs, much of this process is done in spreadsheet tools such as Excel, OpenOffice, etc. The same process, however, is not streamlined in the case of tablets for two reasons. First, spreadsheet tools on tablets have not matured enough to be as proficient as their desktop counterparts. Second, tablet operating systems do not provide the fluid access to application multitasking that is often needed by users to make incremental changes. Applications tend to exist in silos where they operate on independent data sources. A data solution would, thus, need to integrate within the visualization application to be effective.

In acknowledging this need, one goal for the future is to design a methodology for reformatting data that integrates directly into my visualization application. By integrating directly within the application, people would have ready access to the tool when they need it, even at later stages of visualization (addressing reason 2). Also, by scoping the feature set specifically to include only those that are central to visualization tasks, the design problem of interacting with spreadsheets with touch can be more effectively solved (addressing reason 1).

The analyst can be expected to use the tool either at the start of the application before moving to the visualization stage, or during the visualization stage to edit properties of data. Moreover, it might be feasible to integrate the tool within the details-on-demand data table view.

Process steps Integrating data cleaning in the current system would require three steps:

1. List of transformations: The first step is to identify the key tasks users perform on raw data files/tables before importing them into the next stage. This step would involve surveying any previous research that enumerates the data transformations and/or understand the usage of existing tools such as Data Wrangler [64] and Tableau.

   The goal is the identify a list of primary transformations conducted on a spreadsheet that are specifically relevant to visualization. Potential transformations include:

   a. Edit Row
   b. Delete Row/Column
   c. Assign Row as
   d. Filter on values
   e. Select Subtable
   f. Join subtable
   g. Create Aggregations
   h. Adding randoms
   i. Combine sheets
   j. Generalized Selection

2. Identify key transformations: Rank the tasks in order of importance and how appropriate they are for a tablet system. Once ranked, identify a subset of the most relevant tasks to eventually support in the tool. Identifying the right feature set is critical for two reasons:

a. Interactions: Only a limited set of features can be optimized in terms of interactions, given the restrictions of the interface. Not all features are necessary, but finding interactions that "blend in" with the feature density/design of the rest of the application is crucial.

b. Complexity: Certain tasks on spreadsheets require analysis that is not appropriate for tablets. These might be tasks that require too many steps (e.g. combining sheets by matching columns) or those that require higher processing power, screen space or multi-application dependency.

3. Design interactions: This stage would eventually resemble the process I followed in building Tangere. For each task that is identified, interactive operations would need to be designed and prototyped. Broadly, the tool might adopt the look and feel of a spreadsheet. If, however, the workflow is integrated within an existing tool, the interaction design can simply be adapted from the existing tool.

11.2.2 Multi-modal input

Integrating multiple forms of input is another piece of future work that would be extremely relevant to explore. The input in Tangere currently is strictly touch-based. The devices, however, are capable of accepting several forms of input, including voice-based, motion-gesture based, and camera-based. Each has been explored in the past, with voice-based (or natural language based) input gaining particular momentum on handheld devices [94] and in the context of visualization [38, 110].

Several efforts exploring the usage of speech for interacting with visualizations are currently underway. The platforms being explored mainly are desktops and large touchscreen displays, with limited exploration of small, handheld touchscreen devices. The exploration on large screens differs from that on small screens since display size affects how speech is integrated with touch input, just as it would even if the input

was only touch-based.

More specifically, the division of labor between touch and speech on a large-display touchscreen would be different to the division on a small-display touchscreen. Due to the lack of screen space on small screens, a larger share of tasks can be performed using speech. Conversely, due to the size of large displays, and thus the need for the user to view the screen from a distance, the need for speech-based input is more paramount.

Consequently, even though speech input is being explored already, exploring it selectively within the context of small tablet devices is important. A fully functioning speech-based interface is certainly challenging to build. However, using an engine that builds on a vocabulary of possible choices may be a good first step. The scenario below highlights what such an interaction might resemble.

Dave receives a data file from his office that contains the company's sales information for the past month. Dave imports the data into the tablet visualization system. He places a scatterplot onto the screen and configures it to show profit versus quantity. To compare the weekend sales to all sales, Dave says "highlight items whose sell date was a Saturday or Sunday".

Dave taps on the screen to cancel the selection and subsequently plots a linechart with inventory on the y-axis. He taps on the line at the point showing a steep drop. A tooltip displaying the date of the event shows up. Dave says "Change the label to show the day of sell date". The label now displays 'Tuesday'. Dave says "Highlight other points with sell date 'Tuesday'".

With all Tuesdays selected, Dave taps on the 'Keep only' button. He adds a barchart and maps the x-axis to store. Two stores stand out — Midtown and 32nd Street. He lasso selects the two bars and brings the data table into view. To view all details about the Midtown store, Dave says "Export a linechart with sales on y axis and store as Midtown". A share screen with the image of a linechart shows up. Dave

chooses the mail app and emails the image to his team.

The above scenario presents capabilities that can be targeted initially. It also highlights a way in which tasks can be distributed among touch-based and speech-based input. Here, the distinction is based on ease of performance, number of steps required, and context.

1. Ease: Tasks that are otherwise straightforward to perform should be performed with touch. The accuracy of speech detection may be low. Therefore, the drawbacks of inaccurate detection are high. For example, selecting glyphs with tapping or lasso is simple to perform. Communicating the intention accurately to the system through words is nontrivial, however.

2. Number of steps: Tasks that require considerable number of steps to complete can be substituted with speech, granted natural language expression is unambiguous. An example is the command "Highlight other points with sell date 'Tuesday'".

3. Context: Some operations may not be very difficult to perform, but may switch user's context considerably. Ideally, such operations should execute in the background with little distraction to the user. Speech-based commands are ideal for these operations because they require minimum interaction. Once an unambiguous command is provided to the machine, all processing can occur in the background and simply the result can be presented to the user. An example is "Export a linechart with sales on y axis and store as Midtown".

The division of labor between speech-based input and direct manipulation (DM) input, including touch-based, has been explored in the past [93]. In the summative study for Eviza [110], the authors identified user preference for natural language interfaces for three reasons — the DM interface required many clicks to complete the task, the user did not know how to do the task or where to find the control in the DM

interface, or the user did not know where to find the control for the function. While these reasons differ slightly from the three highlighted above, these emerge from user preference for one interface over another. Conversely, the ones listed above emerge out of the designer's requirements for why natural-language input must be integrated.

In summary, considerable promise exists in exploring alternate forms of input to supplement touch-based input on a tablet. The most promising is speech-based natural language input that builds on top of the ever-improving architectures of Siri and Google Now. A sufficiently advanced version of natural language interface should allow most visualization tasks to be completed with speech alone. However, for an initial exploration, speech could be integrated within the interface designed for touch. By identifying an appropriate division of roles, the multimodal interface consisting of speech and touch based interactions should be more efficient than three interface with touch-input only.

11.2.3 Other extensions

1. *Integrating additional techniques* — The four visualization techniques Tangere currently supports are a subset of a much larger set of techniques that people typically use. Other commonly used techniques include pie chart, treemap, histogram, geo-maps, choropleth map, node-link diagrams, and others. Comprehensive analytical tools such as Tableau support a majority of these techniques. Since each technique excels at highlighting unique patterns in the data, a broader set of supported techniques result in a more powerful system.

   The challenges are the same as the ones discussed in Chapter 5 — introducing other representations raises concerns regarding existing gestures and interactive operations transferring from the currently supported techniques to the new ones. Adding other techniques would, perhaps, require revisiting some of the design choices I have already made within the system. However, a rigorous design

process that includes modifying some of the previous design decisions, although tedious, would ultimately enrich the experience within the system while ensuring that usability is maintained.

2. *Cross-platform and cross-device* — I designed Tangere specifically for a tablet running the iOS operating system. Several design decisions I took along the way were directly influenced by these two factors. For instance, I used gestures that are common to and typically found on other iOS applications. Similarly, the requirement to use only one-handed interactions was influenced by the tablet form factor and the goal to not require participants to place the device on a table.

   Both the decisions, however, reveal interesting directions for future work. How does the design of Tangere change if the operating system is changed? Do the results of discoverability and memorability differ on an Android-based device? I partly captured this in the three studies as several participants were using Android devices as their main device. Testing the system with them did not reveal significant performance differences compared to others who used iOS. However, if the app I designed is housed within an Android environment, I speculate that several aspects of the design would need adjusting to better map to the guidelines stipulated by the operating system.

   Similarly, another important direction for future work is to explore the implications on the design if the platform is switched from tablets to smartphones or large-sized displays. Clearly, the constraints of these platforms are considerably different. In that regard, the underlying question is not how well the system fares when adapted to the other platforms. Instead, the more pertinent question is — how is the system redesigned to provide people with the most consistent experience across devices. A prominent trend in consumer devices

in recent years has been to seamlessly transition a user's scenario and activity across devices. For example, if a user is drafting an email on a desktop and opens her smartphone, she is able to continue drafting the email within the email app on the phone.

It is valuable to explore how such an experience would behave for the domain of visualization. Several challenges exist —

- How does each visualization scale up or down to adjust to different device sizes while maintaining the same general design and interaction capabilities?
- Do the constraints of the device affect the configurations being used? For instance, do views show up in a coordinated configuration across all devices or only those with a minimum display size? Further, does the number of views allowed in a row or column change with screen size?
- When transitioning from one device to the other, how is the current state of the system, such as selection and filtering, maintained?
- How does the system behave if it is operated upon on multiple devices simultaneously?

All these questions point to the richness of the solution space and the need to conduct an in-depth exploration. However, given the pace at which we are transitioning to a ubiquitous presence of computing devices around us, the need for these solutions might arise sooner than we think.

APPENDIX A

STUDY DATASET AND TASKS

A.1 Sample Tasks from Study 1

1. Which movie has grossed the most money at the box office?
2. What is the average length of horror movies?
3. How many movies has Steven Spielberg made? What are their names?
4. Is there a relation between the budget and length of the movies?
5. What was the budget and profit of The Dark Knight?
6. How many movies were released in 2010?
7. Do the top rated movies (IMDb ratings >8) come from a specific genre? What about the most profitable movies (profit % >2000 or profit >$2000M)?
8. Which genre is the most profitable in terms of $? Is it different for US markets and overseas markets?
9. Which year was the most profitable? Which month within that year in particular? Does the same month tend to be the most profitable across all years?
10. Do the profits follow a trend every year?
11. Is the movie budget going up over time?
12. Between 2006 & 2010, which genre had maximum movies rated higher than 8 on IMDb?

13. What was the most money making and least money making movie in US and outside US?

14. Does the runtime of a movie have an effect on the profit or the rating?

15. Do the highest rated movies (>8 IMDb rating) stand out in any other category?

16. I'm interested in watching a big-budget action movie. But I would also like it if it had a higher IMDB rating. How many options do I have? Are there more options from recent years than earlier years?

A.2 Sample Dataset

Title	Director	Genre	Studio	Release Date	Gross ($M)	Overseas ($M)	US ($M)	Budget ($M)	Profit ($M)	Runtime	IMDb Rating
Avatar	James Cameron	sci-fi	Fox	12/18/09	2788	2027.5	760.5	237	2551	162	7.9
Titanic	James Cameron	drama	Paramount	12/17/97	2186.8	1528.1	658.7	200	1986.8	194	7.7
Avengers	Joss Whedon	action	Buena Vista	5/4/12	1518.6	895.2	623.4	220	1298.6	143	8.2
Furious 7	James Wan	action	Universal	4/3/15	1511.6	1160.6	351	190	1321.6	137	7.6
Frozen	Chris Buck	animation	Buena Vista	11/22/13	1274.2	873.5	400.7	150	1124.2	102	7.7
Jurassic World	Colin Trevorrow	action	Universal	6/12/15	1245.9	745.5	500.4	215	1030.9	124	7.5
Iron Man 3	Shane Black	action	Buena Vista	5/3/13	1215.4	806.4	409	200	1015.4	130	7.3

Figure 57: Sample movies dataset

REFERENCES

[1] Abowd, G. D., "What Next, Ubicomp?: Celebrating an Intellectual Disappearing Act," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, (New York, NY, USA), pp. 31–40, ACM, 2012.

[2] Agarawala, A. and Balakrishnan, R., "Keepin' It Real: Pushing the Desktop Metaphor with Physics, Piles and the Pen," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, (New York, NY, USA), pp. 1283–1292, ACM, 2006.

[3] Ahlberg, C., "Spotfire: An Information Exploration Environment," *SIGMOD Rec.*, vol. 25, pp. 25–29, Dec. 1996.

[4] Amar, R., Eagan, J., and Stasko, J., "Low-level components of analytic activity in information visualization," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pp. 111–117, Oct. 2005.

[5] Au, O. K.-C., Su, X., and Lau, R. W., "LinearDragger: A Linear Selector for One-finger Target Acquisition," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, (New York, NY, USA), pp. 2607–2616, ACM, 2014.

[6] Badros, G. J., Nichols, J., and Borning, A., "SCWM: An intelligent constraint-enabled window manager," in *Proceedings of the AAAI Spring Symposium on Smart Graphics*, 2000. 00034.

[7] Bailly, G., Mller, J., and Lecolinet, E., "Design and evaluation of finger-count interaction: Combining multitouch gestures and menus," *International Journal of Human-Computer Studies*, vol. 70, pp. 673–689, Oct. 2012.

[8] Baur, D., Lee, B., and Carpendale, S., "TouchWave: Kinetic Multi-touch Manipulation for Hierarchical Stacked Graphs," in *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '12, (New York, NY, USA), pp. 255–264, ACM, 2012.

[9] 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*, CHI '00, (New York, NY, USA), pp. 446–453, ACM, 2000.

[10] Beaudouin-Lafon, M., "Novel interaction techniques for overlapping windows," in *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pp. 153–154, ACM, 2001.

[11] Bederson, B. B. and Hollan, J. D., "Pad++: A Zoomable Graphical Interface System," in *Conference Companion on Human Factors in Computing Systems*, CHI '95, (New York, NY, USA), pp. 23–24, ACM, 1995.

[12] Benko, H., Wilson, A. D., and Baudisch, P., "Precise Selection Techniques for Multi-touch Screens," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, (New York, NY, USA), pp. 1263–1272, ACM, 2006.

[13] Bly, S. A. and Rosenberg, J. K., "A comparison of tiled and overlapping windows," in *ACM SIGCHI Bulletin*, vol. 17, pp. 101–106, ACM, 1986.

[14] Browne, J., Lee, B., Carpendale, S., Riche, N., and Sherwood, T., "Data Analysis on Interactive Whiteboards Through Sketch-based Interaction," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, (New York, NY, USA), pp. 154–157, ACM, 2011.

[15] Buchanan, R., "Good design in the digital age," *AIGA Journal of Design for the Network Economy*, vol. 1, no. 1, pp. 1–5, 2000.

[16] Buering, T., Gerken, J., and Reiterer, H., "User Interaction with Scatterplots on Small Screens - A Comparative Evaluation of Geometric-Semantic Zoom and Fisheye Distortion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 829–836, 2006.

[17] Burigat, S. and Chittaro, L., "Geographic Data Visualization on Mobile Devices for Users Navigation and Decision Support Activities," in *Spatial Data on the Web Modelling and Management*, Springer, 2007.

[18] Buxton, W., "A Three-state Model of Graphical Input," in *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, INTERACT '90, (Amsterdam, The Netherlands, The Netherlands), pp. 449–456, North-Holland Publishing Co., 1990.

[19] Card, S. K., Mackinlay, J. D., and Shneiderman, B., eds., *Readings in Information Visualization: Using Vision to Think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[20] Chen, H., "Compound Brushing Explained," *Information Visualization*, vol. 3, pp. 96–108, June 2004.

[21] Cockburn, A. and McKenzie, B., "Evaluating the Effectiveness of Spatial Memory in 2d and 3d Physical and Virtual Environments," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, (New York, NY, USA), pp. 203–210, ACM, 2002.

[22] Cohen, E. S., Smith, E. T., and Iverson, L. A., "Constraint-based tiled windows," *IEEE computer graphics and applications*, vol. 6, no. 5, pp. 35–45, 1986.

[23] Damaraju, S., Seo, J. H., Hammond, T., and Kerne, A., "Multi-tap Sliders: Advancing Touch Interaction for Parameter Adjustment," in *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, (New York, NY, USA), pp. 445–452, ACM, 2013.

[24] Davis, S. and Wiedenbeck, S., "The effect of interaction style and training method on end user learning of software packages," *Interacting with Computers*, vol. 11, no. 2, pp. 147–172, 1998.

[25] Deaton, K. and Gedeon, S. A., "Method and system for creating and distributing collaborative multi-user three-dimensional websites for a computer system (3d Net Architecture)," Sept. 2006. 00077 U.S. Classification 715/836, 715/850, 715/848, 707/E17.111; International Classification G06T19/00, G06F3/048, G06F3/033, G06F13/00, G06F17/30; Cooperative Classification G06F17/30905, G06F17/30873, G06T19/00, G06F3/04815, G06T2219/024; European Classification G06F17/30W9V, G06T19/00, G06F3/0481E, G06F17/30W3.

[26] Dietz, P. and Leigh, D., "DiamondTouch: A Multi-user Touch Technology," in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, (New York, NY, USA), pp. 219–226, ACM, 2001.

[27] Dix, A., *Human-computer interaction*. Springer, 2009. 05758.

[28] Dix, A. and Ellis, G., "Starting Simple: Adding Value to Static Visualisation Through Simple Interaction," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '98, (New York, NY, USA), pp. 124–134, ACM, 1998.

[29] Drucker, S. M., Fisher, D., Sadana, R., Herron, J., and 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*, CHI '13, (New York, NY, USA), pp. 2301–2310, ACM, 2013.

[30] Dusan, S., Gadbois, G. J., and Flanagan, J. L., "Multimodal interaction on PDA's integrating speech and pen inputs.," 00031.

[31] Elliott, G., Jones, E., and Barker, P., "A grounded theory approach to modelling learnability of hypermedia authoring tools," *Interacting with Computers*, vol. 14, no. 5, pp. 547–574, 2002.

[32] Elmqvist, N., Moere, A. V., Jetter, H.-C., Cernea, D., Reiterer, H., and Jankun-Kelly, T. J., "Fluid interaction for information visualization," *Information Visualization*, vol. 10, pp. 327–340, Oct. 2011.

[33] Engelbart, D. C., "Conceptual Framework for the Augmentation of Man's Intellect," 1963. 00000.

[34] Esenther, A. and Ryall, K., "Fluid DTMouse: Better Mouse Support for Touch-based Interactions," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, (New York, NY, USA), pp. 112–115, ACM, 2006.

[35] Feiner, S., Nagy, S., and Van Dam, A., "An experimental system for creating and presenting interactive graphical documents," *ACM Transactions on Graphics (TOG)*, vol. 1, no. 1, pp. 59–77, 1982.

[36] Frisch, M., Heydekorn, J., and Dachselt, R., "Investigating Multi-touch and Pen Gestures for Diagram Editing on Interactive Surfaces," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, (New York, NY, USA), pp. 149–156, ACM, 2009.

[37] Fu, C.-W., Goh, W.-B., and Ng, J. A., "Multi-touch Techniques for Exploring Large-scale 3d Astrophysical Simulations," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 2213–2222, ACM, 2010.

[38] Gao, T., Dontcheva, M., Adar, E., Liu, Z., and Karahalios, K. G., "Datatone: Managing ambiguity in natural language interfaces for data visualization," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pp. 489–500, ACM, 2015.

[39] Goldberg, A. and Robson, D., *Smalltalk-80: the language and its implementation*. Addison-Wesley Longman Publishing Co., Inc., 1983. 06631.

[40] Grammel, L., Tory, M., and Storey, M.-A., "How information visualization novices construct visualizations," *IEEE transactions on visualization and computer graphics*, vol. 16, no. 6, pp. 943–952, 2010.

[41] Gunn, T. J., Zhang, H., Mak, E., and Irani, P., "An Evaluation of Onehanded Techniques for Multiple-target Selection," in *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, (New York, NY, USA), pp. 4189–4194, ACM, 2009.

[42] Gutwin, C., Cockburn, A., Scarr, J., Malacria, S., and Olson, S. C., "Faster Command Selection on Tablets with FastTap," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, (New York, NY, USA), pp. 2617–2626, ACM, 2014.

[43] Harrison, C. and Hudson, S., "Using Shear As a Supplemental Two-dimensional Input Channel for Rich Touchscreen Interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 3149–3152, ACM, 2012.

[44] Hauser, H., Ledermann, F., and Doleisch, H., "Angular brushing of extended parallel coordinates," in *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002*, pp. 127–130, 2002.

[45] Heer, J. and Robertson, G., "Animated Transitions in Statistical Data Graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1240–1247, Nov. 2007.

[46] Heer, J., Agrawala, M., and Willett, W., "Generalized Selection via Interactive Query Relaxation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, (New York, NY, USA), pp. 959–968, ACM, 2008.

[47] Heer, J., Bostock, M., and Ogievetsky, V., "A tour through the visualization zoo," *Commun. ACM*, vol. 53, pp. 59–67, June 2010.

[48] Heilig, M., Huber, S., Demarmels, M., and Reiterer, H., "ScatterTouch: A Multi Touch Rubber Sheet Scatter Plot Visualization for Co-located Data Exploration," in *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, (New York, NY, USA), pp. 263–264, ACM, 2010.

[49] Henderson Jr, D. A. and Card, S., "Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface," *ACM Transactions on Graphics (TOG)*, vol. 5, no. 3, pp. 211–243, 1986.

[50] Heo, S., Gu, J., and Lee, G., "Expanding Touch Input Vocabulary by Using Consecutive Distant Taps," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, (New York, NY, USA), pp. 2597–2606, ACM, 2014.

[51] Heo, S. and Lee, G., "Force Gestures: Augmenting Touch Screen Gestures with Normal and Tangential Forces," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, (New York, NY, USA), pp. 621–626, ACM, 2011.

[52] Hinckley, K., Baudisch, P., Ramos, G., and Guimbretiere, F., "Design and Analysis of Delimiters for Selection-action Pen Gesture Phrases in Scriboli," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, (New York, NY, USA), pp. 451–460, ACM, 2005.

[53] Hinckley, K., Czerwinski, M., and Sinclair, M., "Interaction and Modeling Techniques for Desktop Two-handed Input," in *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, (New York, NY, USA), pp. 49–58, ACM, 1998.

[54] Hinckley, K. and Wigdor, D., "Input Technologies and Techniques," in *The Human-Computer Interaction Handbook Fundamentals, Evolving Technologies and Emerging Applications, Third Edition* (Jacko, J., ed.), Taylor & Francis, 2012.

[55] Hoggan, E., Brewster, S. A., and Johnston, J., "Investigating the effectiveness of tactile feedback for mobile touchscreens," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 1573–1582, ACM, 2008.

[56] Holz, C. and Baudisch, P., "Understanding Touch," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, (New York, NY, USA), pp. 2501–2510, ACM, 2011.

[57] Hudson, S. E. and Stasko, J. T., "Animation Support in a User Interface Toolkit: Flexible, Robust, and Reusable Abstractions," in *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, UIST '93, (New York, NY, USA), pp. 57–67, ACM, 1993.

[58] Hutchings, D. R. and Stasko, J. T., "New operations for display space management and window management," 2002. 00011.

[59] Isenberg, P. and Fisher, D., "Collaborative Brushing and Linking for Co-located Visual Analytics of Document Collections," *Computer Graphics Forum*, vol. 28, no. 3, pp. 1031–1038, 2009.

[60] Isenberg, P., Isenberg, T., Hesselmann, T., Lee, B., von Zadow, U., and Tang, A., "Data Visualization on Interactive Surfaces: A Research Agenda," *IEEE Computer Graphics and Applications*, vol. 33, no. 2, pp. 16–24, 2013.

[61] Jansen, Y. and Dragicevic, P., "An Interaction Model for Visualizations Beyond The Desktop," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2396–2405, 2013.

[62] Javed, W. and Elmqvist, N., "Exploring the design space of composite visualization," in *Visualization Symposium (PacificVis), 2012 IEEE Pacific*, pp. 1–8, Feb. 2012.

[63] Jeffrey M. Rzeszotarski and Kittur, A., "Kinetica: Naturalistic Multi-touch Data Visualization," in *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, (New York, NY, USA), pp. 897–906, ACM, 2014.

[64] Kandel, S., Paepcke, A., Hellerstein, J., and Heer, J., "Wrangler: Interactive Visual Specification of Data Transformation Scripts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, (New York, NY, USA), pp. 3363–3372, ACM, 2011.

[65] Karat, J., McDonald, J. E., and Anderson, M., "A Comparison of Menu Selection Techniques: Touch Panel, Mouse and Keyboard," *Int. J. Man-Mach. Stud.*, vol. 25, pp. 73–88, July 1986.

[66] Karstens, B., "Presenting large and complex information sets on mobile handhelds," 2005. 00007.

[67] Kay, A. C., "The reactive engine," 1969. 00120.

[68] Keim, D. A., "Information Visualization and Visual Data Mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 1–8, Jan. 2002.

[69] Kim, J., Nguyen, P. T., Weir, S., Guo, P. J., Miller, R. C., and Gajos, K. Z., "Crowdsourcing step-by-step information extraction to enhance existing how-to videos," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 4017–4026, ACM, 2014.

[70] Kser, D. P., Agrawala, M., and Pauly, M., "FingerGlass: Efficient Multiscale Interaction on Multitouch Screens," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, (New York, NY, USA), pp. 1601–1610, ACM, 2011.

[71] Lam, H., Bertini, E., Isenberg, P., Plaisant, C., and Carpendale, S., "Empirical Studies in Information Visualization: Seven Scenarios," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 1520–1536, Sept. 2012.

[72] Lee, B., Isenberg, P., Riche, N., and Carpendale, S., "Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2689–2698, 2012.

[73] Lepinski, G. J., Grossman, T., and Fitzmaurice, G., "The Design and Evaluation of Multitouch Marking Menus," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 2233–2242, ACM, 2010.

[74] Liu, Z., Navathe, S. B., and Stasko, J. T., "Ploceus: Modeling, visualizing, and analyzing tabular data as networks," *Information Visualization*, vol. 13, pp. 59–89, Jan. 2014.

[75] Luo, Y. and Vogel, D., "Pin-and-Cross: A Unimanual Multitouch Technique Combining Static Touches with Crossing Selection," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, (New York, NY, USA), pp. 323–332, ACM, 2015.

[76] MacKenzie, I. S., "Fitts' Law As a Research and Design Tool in Human-computer Interaction," *Hum.-Comput. Interact.*, vol. 7, pp. 91–139, Mar. 1992.

[77] Maeda, J., *The Laws of Simplicity*. The MIT Press, 2006.

[78] Mankoff, J., Hudson, S. E., and Abowd, G. D., "Interaction Techniques for Ambiguity Resolution in Recognition-based Interfaces," in *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, (New York, NY, USA), pp. 11–20, ACM, 2000.

[79] McDonnel, B. and Elmqvist, N., "Towards Utilizing GPUs in Information Visualization: A Model and Implementation of Image-Space Operations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 1105–1112, Nov. 2009.

[80] Morris, M. R., Wobbrock, J. O., and Wilson, A. D., "Understanding Users' Preferences for Surface Gestures," in *Proceedings of Graphics Interface 2010*, GI '10, (Toronto, Ont., Canada, Canada), pp. 261–268, Canadian Information Processing Society, 2010.

[81] Moscovich, T., "Contact Area Interaction with Sliding Widgets," in *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, (New York, NY, USA), pp. 13–22, ACM, 2009.

[82] Myers, B. A., "A taxonomy of user interfaces for window managers," *IEEE Computer Graphics and Applications*, vol. 8, no. 5, pp. 65–84, 1988.

[83] Nielsen, J., "Enhancing the explanatory power of usability heuristics," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 152–158, ACM, 1994.

[84] Norman, D. A., *The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution*. 1998. 00013.

[85] Norman, D. A., *The Design of Everyday Things*. New York, NY, USA: Basic Books, Inc., 2002.

[86] Norman, D. A., *Emotional design: Why we love (or hate) everyday things*. Basic books, 2004. 05321.

[87] Norman, D. A. and Draper, S. W., *User Centered System Design; New Perspectives on Human-Computer Interaction*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1986. 00007.

[88] North, C., Dwyer, T., Lee, B., Fisher, D., Isenberg, P., Robertson, G., and Inkpen, K., "Understanding Multi-touch Manipulation for Surface Computing," in *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*, INTERACT '09, (Berlin, Heidelberg), pp. 236–249, Springer-Verlag, 2009.

[89] North, C. and Shneiderman, B., "A Taxonomy of Multiple Window Coordinations," tech. rep., 1997.

[90] North, C. and Shneiderman, B., "Snap-together visualization: can users construct and operate coordinated visualizations?," *International Journal of Human-Computer Studies*, vol. 53, pp. 715–739, Nov. 2000.

[91] Olsen, Jr., D. R., "Evaluating user interface systems research," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, (New York, NY, USA), pp. 251–258, ACM, 2007.

[92] Ostroff, D. and Shneiderman, B., "Selection Devices for Users of an Electronic Encyclopedia: An Empirical Comparison of Four Possibilities," *Inf. Process. Manage.*, vol. 24, pp. 665–680, Nov. 1988.

[93] Oviatt, S., "Multimodal interfaces for dynamic interactive maps," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 95–102, Acm, 1996.

[94] Oviatt, S., Cohen, P., Wu, L., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J., and others, "Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions," *Human-computer interaction*, vol. 15, no. 4, pp. 263–322, 2000.

[95] Pike, W. A., Stasko, J., Chang, R., and O'Connell, T. A., "The Science of Interaction," *Information Visualization*, vol. 8, pp. 263–274, Dec. 2009.

[96] Potter, R. L., Weldon, L. J., and Shneiderman, B., "Improving the Accuracy of Touch Screens: An Experimental Evaluation of Three Strategies," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, (New York, NY, USA), pp. 27–32, ACM, 1988.

[97] Raskin, J., *The Humane Interface: New Directions for Designing Interactive Systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. 00003.

[98] Robertson, G., Fernandez, R., Fisher, D., Lee, B., and Stasko, J., "Effectiveness of Animation in Trend Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1325–1332, Nov. 2008.

[99] Robertson, G., Van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., and Gorokhovsky, V., "The Task Gallery: a 3d window manager," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 494–501, ACM, 2000.

[100] Ross, D., "A Personal View of the Personal Work Station: Some Firsts in the Fifties," in *Proceedings of the ACM Conference on The History of Personal Workstations*, HPW '86, (New York, NY, USA), pp. 19–48, ACM, 1986.

[101] Roudaut, A., Lecolinet, E., and Guiard, Y., "MicroRolls: Expanding Touch-screen Input Vocabulary by Distinguishing Rolls vs. Slides of the Thumb," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 927–936, ACM, 2009.

[102] Rzeszotarski, J. and Kittur, A., "Kinetica: Naturalistic Multi-touch Data Visualization," (Toronto, ON, Canada), 2014. 00000.

[103] Sadana, R. and Stasko, J., "Designing Multiple Coordinated Visualizations for Tablets," *Computer Graphics Forum*, vol. 35, pp. 261–270, June 2016.

[104] Sadana, R. and 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*, AVI '14, (New York, NY, USA), pp. 265–272, ACM, 2014.

[105] Saket, B., Kim, H., Brown, E. T., and Endert, A., "Visualization by Demonstration: An Interaction Paradigm for Visual Data Exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, pp. 331–340, Jan. 2017.

[106] Saket, B., Endert, A., and Stasko, J., "Beyond Usability and Performance: A Review of User Experience-focused Evaluations in Visualization," in *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, BELIV '16, (New York, NY, USA), pp. 133–142, ACM, 2016.

[107] Schmidt, S., Nacenta, M. A., Dachselt, R., and Carpendale, S., "A Set of Multi-touch Graph Interaction Techniques," in *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, (New York, NY, USA), pp. 113–116, ACM, 2010.

[108] Sears, A. and Shneiderman, B., "High Precision Touchscreens: Design Strategies and Comparisons with a Mouse," *Int. J. Man-Mach. Stud.*, vol. 34, pp. 593–613, Apr. 1991.

[109] Sellen, A. J., Kurtenbach, G. P., and Buxton, W. A. S., "The Prevention of Mode Errors Through Sensory Feedback," *Hum.-Comput. Interact.*, vol. 7, pp. 141–164, June 1992.

[110] Setlur, V., Battersby, S. E., Tory, M., Gossweiler, R., and Chang, A. X., "Eviza: A Natural Language Interface for Visual Analysis," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 365–377, ACM, 2016.

[111] Shneiderman, B., "Direct Manipulation: A Step Beyond Programming Languages," *Computer*, vol. 16, pp. 57–69, Aug. 1983.

[112] Shneiderman, B., "Dynamic queries for visual information seeking," *IEEE Software*, vol. 11, pp. 70–77, Nov. 1994.

[113] Shneiderman, B. and Plaisant, C., "Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies," in *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pp. 1–7, ACM, 2006.

[114] Spence, R., *Information Visualization: Design for Interaction (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007. 00000.

[115] Stasko, J., Grg, C., and Liu, Z., "Jigsaw: Supporting Investigative Analysis through Interactive Visualization," *Information Visualization*, vol. 7, pp. 118–132, June 2008.

[116] Stolte, C., Tang, D., and Hanrahan, P., "Polaris: a system for query, analysis, and visualization of multidimensional relational databases," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 52–65, Jan. 2002.

[117] Sultanum, N., Somanath, S., Sharlin, E., and Sousa, M. C., ""Point It, Split It, Peel It, View It": Techniques for Interactive Reservoir Visualization on Tabletops," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, (New York, NY, USA), pp. 192–201, ACM, 2011.

[118] Thudt, A., Hinrichs, U., and Carpendale, S., "The Bohemian Bookshelf: Supporting Serendipitous Book Discoveries Through Information Visualization," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 1461–1470, ACM, 2012.

[119] Tversky, B., Morrison, J. B., and Betrancourt, M., "Animation: Can It Facilitate?," *Int. J. Hum.-Comput. Stud.*, vol. 57, pp. 247–262, Oct. 2002.

[120] Vogel, D. and Balakrishnan, R., "Occlusion-aware Interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 263–272, ACM, 2010.

[121] Vogel, D. and Baudisch, P., "Shift: A Technique for Operating Pen-based Interfaces Using Touch," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, (New York, NY, USA), pp. 657–666, ACM, 2007.

[122] Wagner, J., Huot, S., and Mackay, W., "BiTouch and BiPad: Designing Bimanual Interaction for Hand-held Tablets," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 2317–2326, ACM, 2012.

[123] Walny, J., Lee, B., Johns, P., Riche, N., and Carpendale, S., "Understanding Pen and Touch Interaction for Data Exploration on Interactive Whiteboards," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2779–2788, 2012.

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

[125] Ward, M. and Yang, J., "Interaction Spaces in Data and Information Visualization," in *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, VISSYM'04, (Aire-la-Ville, Switzerland, Switzerland), pp. 137–146, Eurographics Association, 2004.

[126] Wattenberg, M. and Kriss, J., "Designing for Social Data Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 549–557, July 2006.

[127] Wigdor, D., Benko, H., Pella, J., Lombardo, J., and 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*, CHI '11, (New York, NY, USA), pp. 1581–1590, ACM, 2011.

[128] Wigdor, D., Williams, S., Cronin, M., Levy, R., White, K., Mazeev, M., and Benko, H., "Ripples: utilizing per-contact visualizations to improve user interaction with touch displays," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pp. 3–12, ACM, 2009.

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

[130] Wilkinson, L., *The Grammar of Graphics*. New York, NY, USA: Springer-Verlag New York, Inc., 1999. 00006.

[131] Willett, W., Heer, J., and Agrawala, M., "Scented Widgets: Improving Navigation Cues with Embedded Visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1129–1136, Nov. 2007.

[132] Wills, G. J., "Selection: 524,288 ways to say ldquo;this is interesting rdquo;," in *Proceedings IEEE Symposium on Information Visualization '96*, pp. 54–60, 120, Oct. 1996.

[133] Wobbrock, J. O., Morris, M. R., and Wilson, A. D., "User-defined Gestures for Surface Computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1083–1092, ACM, 2009.

[134] Wongsuphasawat, K., Moritz, D., Anand, A., Mackinlay, J., Howe, B., and Heer, J., "Voyager: Exploratory analysis via faceted browsing of visualization recommendations," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 1, pp. 649–658, 2016.

[135] Wroblewski, L., *Mobile first: Prface de Jeffrey Zeldmann*. Editions Eyrolles, 2012. 00129.

[136] Yee, W., "Potential Limitations of Multi-touch Gesture Vocabulary: Differentiation, Adoption, Fatigue," in *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, (Berlin, Heidelberg), pp. 291–300, Springer-Verlag, 2009.

[137] Yi, J. S., Kang, Y. a., and Stasko, J., "Toward a Deeper Understanding of the Role of Interaction in Information Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1224–1231, Nov. 2007.

[138] Yoo, H. Y. and Cheon, S. H., "Visualization by Information Type on Mobile Device," in *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*, APVis '06, (Darlinghurst, Australia, Australia), pp. 143–146, Australian Computer Society, Inc., 2006.

[139] Yu, L., Svetachov, P., Isenberg, P., Everts, M. H., and Isenberg, T., "FI3d: Direct-Touch Interaction for the Exploration of 3d Scientific Visualization Spaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1613–1622, 2010.