ABSTRACT

Title of dissertation: ENABLING COLLABORATIVE VISUAL ANALYSIS ACROSS HETEROGENEOUS DEVICES

Sriram Karthik Badam
Doctor of Philosophy, 2019

Dissertation directed by: Professor Niklas Elmqvist
College of Information Studies

We are surrounded by novel device technologies emerging at an unprecedented pace. These devices are heterogeneous in nature: in large and small sizes with many input and sensing mechanisms. When many such devices are used by multiple users with a shared goal, they form a heterogeneous device ecosystem. A device ecosystem has great potential in data science to act as a natural medium for multiple analysts to make sense of data using visualization. It is essential as today's big data problems require more than a single mind or a single machine to solve them. Towards this vision, I introduce the concept of collaborative, cross-device visual analytics (C2-VA) and outline a reference model to develop user interfaces for C2-VA.

This dissertation covers interaction models, coordination techniques, and software platforms to enable full stack support for C2-VA. Firstly, we connected devices to form an ecosystem using software primitives introduced in the early frameworks from this dissertation. To work in a device ecosystem, we designed multi-user interaction for visual analysis in front of large displays by finding a balance between

proxemics and mid-air gestures. Extending these techniques, we considered the roles of different devices—large and small—to present a conceptual framework for utilizing multiple devices for visual analytics. When applying this framework, findings from a user study showcase flexibility in the analytic workflow and potential for generation of complex insights in device ecosystems. Beyond this, we supported coordination between multiple users in a device ecosystem by depicting the presence, attention, and data coverage of each analyst within a group.

Building on these parts of the C2-VA stack, the culmination of this dissertation is a platform called Vistrates. This platform introduces a component model for modular creation of user interfaces that work across multiple devices and users. A component is an analytical primitive—a data processing method, a visualization, or an interaction technique—that is reusable, composable, and extensible. Together, components can support a complex analytical activity. On top of the component model, the support for collaboration and device ecosystems comes for granted in Vistrates. Overall, this enables the exploration of new research ideas within C2-VA.

ENABLING COLLABORATIVE VISUAL ANALYSIS ACROSS HETEROGENEOUS DEVICES

by

Sriram Karthik Badam

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Niklas Elmqvist, Chair/Advisor
Professor Héctor Corrada Bravo
Professor Eun Kyoung Choe
Dr. Jean-Daniel Fekete
Professor Huaishu Peng
Dr. Catherine Plaisant

To my parents, Krishna Rao and Rama Devi

Acknowledgments

The past few years have given me the most defining moments of my life. These experiences tell me the kind of person I am and the kind of person I need to be. I am incredibly grateful for them. During my graduate studies, many awesome people guided me through successes and failures and helped me grow as a researcher and, more importantly, as a person.

First and foremost, I am eternally grateful to my brilliant advisor, Prof. Niklas Elmqvist, for all his support over the past seven years. I still remember the first time we met with me fumbling nervously about my interest. He was patient and encouraging since that day, and this was *the* contributing factor to my growth. He supported my interests, challenged my ideas, strengthened my resolve, fostered my skills, celebrated our successes, and softened the blow of failures. It has been a pleasure to work with you, Niklas! Thank you for being the best advisor ever!

I had the most fortunate experience of collaborating with researchers across the world. My first research trip was to Winnipeg to visit Prof. Pourang Irani's lab. I still remember Pourang picking me up from the airport late at night and treating me dinner in an Indian restaurant. The internship under Dr. Jean-Daniel Fekete in Paris helped me take a step back and gain a better perspective on my work. It also initiated the collaboration with Prof. Raimund Dachselt's group in Dresden that led to an Honorable Mention at CHI 2018. The partnership with Prof. Clemens Klokmose on Vistrates has offered a strong thread to stitch my projects together into a thesis. I am thankful to them and everyone I worked with in my

collaborations including Prof. Alex Endert, Dr. Zhicheng Liu (Leo), Prof. Roman Rädle, Prof. Jonathan Roberts, Prof. Panagiotis Ritsos (Panos), Dr. Fereshteh Amini, Emily Wall, Tom Horak, Andreas Mathisen, Eli Fisher, and Zehua Zeng. This thesis wouldn't have been possible without you.

My advisory committee—Prof. Héctor Corrada Bravo, Prof. Eun Kyoung Choe, Dr. Jean-Daniel Fekete, Prof. Huaishu Peng, and Dr. Catherine Plaisant—deserves a special mention. Their feedback gave me a concrete direction to improve my dissertation. Catherine was also the first reviewer for many of my research projects. I am very grateful to Catherine for sharing her precious time to provide the early feedback that improved my work substantially.

My past and present colleagues played a significant role in keeping me sane. Senthil Chandrasegaran has been a frequent collaborator during my graduate studies and my best friend over the past few years. His support and friendship mean a lot to me. I also cannot forget my friends in UMD: Adil Yalçin, Deok Gun Park, Zhenpeng Zhao, Zhe Cui, Andrea Batch, Lalitha Poluri, Harsha Yalamanchili, Meethu Malu, Brenna McNally, Matt Mauriello, Jonggi Hong, Manaswi Saha, Liang He, Alina Striner, Joohee Choi, Leyla Norooz, Kotaro Hara, Sana Malik, and Fan Du. Thank you for all the conversations and for making my grad life more fun.

Finally, the unconditional support of my family gave me strength and confidence during my graduate studies. I am very thankful to my parents, Krishna Rao and Rama Devi, and my sister, Keerthana, for their unwavering optimism in my ventures. It is a happy twist of fate that I also met the love of my life during this time. Pranathi, I am fortunate to have you by my side.

List of Included Research Papers

- **S. K. Badam,** A. Mathisen, R. Rädle, C. N. Klokmose, N. Elmqvist. Vistrates: A Component Model for Ubiquitous Analytics. In *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)*, 25(1): 586—596, 2019. [PDF] [VIDEO]

- T. Horak,* **S. K. Badam,*** N. Elmqvist, and R. Dachselt. ***Equal contribution from first two authors.** When David meets Goliath: Combining Smartwatches with a Large Vertical Display for Visual Data Exploration. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, p. 19. 2018. **(Awarded Best Paper Honorable Mention)** [PDF] [VIDEO]

- **S. K. Badam,** A. Srinivasan, N. Elmqvist, and J. Stasko. Affordances of Input Modalities for Visual Data Exploration in Immersive Environments. In *Workshop on Immersive Analytics at IEEE VIS 2017*. [PDF]

- **S. K. Badam,** Z. Zeng, E. Wall, A. Endert, and N. Elmqvist. Supporting Team-First Visual Analytics through Group Activity Representations. In *Graphics Interface* (GI). 2017. [PDF] [VIDEO]

- **S. K. Badam,** N. Elmqvist. Visfer: Camera-Based Visual Data Transfer for Cross-Device Visualization. *Information Visualization*, 18(1): 68-93. 2019. [PDF] [VIDEO]

- **S. K. Badam,** F. Amini, N. Elmqvist, and P. Irani. Supporting Visual Exploration for Multiple Users in Large Display Environments. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 1-10. 2016. [PDF] [VIDEO]

- **S. K. Badam,** E. Fisher, and N. Elmqvist. Munin: A Peer-to-Peer Middleware for Ubiquitous Analytics and Visualization Spaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(2): 215-228. 2015. [PDF] [VIDEO]

- **S. K. Badam,** N. Elmqvist. PolyChrome: A Cross-Device Framework for Collaborative Web Visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces (ITS)*, pp. 109-118. 2014. [PDF] [VIDEO]

- J. C. Roberts, P. D. Ritsos, **S. K. Badam,** D. Brodbeck, J. Kennedy, and N. Elmqvist. Visualization Beyond the Desktop—The Next Big Thing. *IEEE Computer Graphics and Applications (CG&A)*, 34(6): 26-34. 2014. [PDF]

Table of Contents

List of Tables

List of Figures

List of Abbreviations

C2-VA	Collaborative, Cross-Device Visual Analytics
CSCW	Computer-Supported Cooperative Work
VA	Visual Analytics
HCI	Human-Computer Interaction
SA	Situation Awareness
GA	Group Awareness
ML	Machine Learning
IoT	Internet of Things
LD	Large Display
SW	Smartwatch

Chapter 1: Introduction

Relentless advancements in display and input technologies, mobile devices, and hardware infrastructures have changed the face of computing over the past decade. Beyond the desktop computer, many other form factors of computing devices exist better suited for specific cases in personal and professional use. In fact, modern offices now contain large, high-definition displays in conference rooms to support group meetings, while multiple tiled monitors create the personal workspace of many tech workers. Mobile devices are ubiquitously used to access information in any context ranging from navigating to your favorite restaurant to understanding your daily workouts. These prominent advances are turning computing into a more natural experience for humans. It is to the extent that we sometimes do not even realize how naturally we switch between various devices to carry out our everyday tasks. It is true that we are closer than ever to the profound vision of Mark Weiser to promote ubiquitous computing [3].

> "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*"
>
> Mark Weiser, 1991.

These advancements in device technologies help accomplish challenging goals by enabling people to work more flexibly—together with each other in a conference room, in a workspace at home, or on the go in a train. This versatility is needed when human reasoning is essential, for solving challenges in domains such as criminal justice, energy development, social networks, healthcare, financial markets, and economic development. As such, novel devices are the bridge between humans and the complex datasets arising in these domains. User interfaces for flexible analytical work on these devices should effectively utilize human knowledge and computational resources to make decisions from data.

In this direction, my research introduces the concept of **collaborative, cross-device visual analytics** (C2-VA). As defined by Heer and Agarwala [4], collaborative visual analytics is a process of "*peer production of information goods*" that include observations, questions, hypotheses, and insights generated in the analysis process as well as presentations of analysis results. Performing this process across heterogeneous device technologies—such as large displays, tabletops, tablets, and smartphones—is the vision of C2-VA. It requires not only the inherent support for group work but also solutions that leverage the affordances of the heterogeneous devices. Beyond this, new software platforms are needed to easily create the next generation of analytical interfaces that inherently support multiple analysts on heterogeneous devices.

1.1 Motivation: Big Data Contexts

Big data is characterized by its complexities in volume (large size), velocity (new data created quickly), variety (many sources and data types), and veracity (quality not always guaranteed)—the four V's. Over the past two decades, big data has revolutionized industries and government agencies. Decision making from data has shown to be more productive and profitable [5]. Human intuition, experience, and knowledge are essential to leveraging big data towards decision making.

> "*Data really powers everything that we do.*"
>
> Jeff Weiner, LinkedIn.

In criminal justice, big data can enable predictive policing by helping police officers and detectives work with crime datasets to define patrol routes, provide guidelines, and effectively protect the general public. In energy conservation, big data can help understand the production and usage patterns of energy across cities to reduce wastage and investigate alternate sources for sustainability. In tourism, big data can leverage popularity and customer emotion for businesses to support new investments and promotions that develop revenue. In healthcare, big data can help doctors work with their patients by keeping track of patient records, diagnosing them effectively, and predicting the course of action. To support these big data contexts, we need to more than a single mind or a single device to process the characteristics of big data and develop insights for data-driven decision making.

"*Data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom.*"

Clifford Stoll, Lawrence Berkeley National Laboratory.

1.2 Research Questions

The central goal of this research is to empower people in harnessing emerging device technologies as instruments for visual analytics—the science of analytical reasoning driven by visualization interfaces [6]. By using heterogeneous devices—such as large and small displays and input and sensing devices, as well as fixed, portable, and wearable form factors—heterogeneous groups of analysts can work together to consolidate their viewpoints and combine the cognitive processing power.

To answer this goal, we targeted full stack support for C2-VA through interaction models and analytical platforms that showcase the power of device ecosystems for visual analytics. This research, therefore, contributes to visual analytics and human-computer interaction by answering the following research questions:

1. How to **enable multi-user interaction** in a device ecosystem?

   (a) Which interaction models are suitable for the novel device types (e.g., large displays) in a device ecosystem? In contrast to working in front of a desktop computer.

   (b) What are the interaction methods to transfer work and switch between devices in an ecosystem?

2. How to **support a visual analytics process** across multiple devices?

   (a) What is the role of each device during a visual analytics process?

   (b) How can we coordinate multiple users engaged in visual analytics?

3. How to **develop collaborative, cross-device visual analytics systems**?

   (a) What are the software primitives needed to connect devices and distribute visualizations in a device ecosystem?

   (b) How can we support the many activities common in visual analytics—from exploratory to explanatory processes [7]?

Towards the first question, we designed interaction models suitable for shared exploration of data by multiple users in a device ecosystem.[1] Firstly, we created interaction models to work in the physical environment in front of the large displays utilizing proxemics—the study of the human use of space [8]—and gestures. Beyond this, we also considered portable devices such as smartphones, smartwatches, and tablets to complement large displays by providing mobile interfaces for data [9, 10]. We designed cross-device interactions to work across devices.

In a device ecosystem, different devices can be beneficial in different ways during the collaborative exploration of data by multiple users. To answer the second research question, we focused on identifying the roles of heterogeneous devices to utilize them for visual exploration and insight management in the visual analytics

[1] A **device ecosystem** is an environment (physical or virtual) of interacting devices collaboratively used to achieve a goal.

process. We also explored user interface elements that can be added to a visual analysis interface to coordinate multiple users working across devices. We evaluated these design considerations through user studies to develop guidelines for future designers and developers of visual analytics systems.

There are many visual analytics tools currently available. However, very few of them support collaboration or analytical work across heterogeneous devices. Current commercial and research tools specialize in particular analytical activities—specific data types, visualizations, interaction methods, users, devices, etc. To address the gap in tool support for C2-VA, we answered the third research question through a new web platform called Vistrates [11, 12]. This platform provides a document-based framework of cross-cutting components to create C2-VA tools that support (1) both single-user and collaborative work, (2) the full spectrum of data analysis activities, (3) all levels of user expertise, and (4) a menagerie of devices. Vistrates is therefore essential for future research in this space. In fact, it is already being used for new research into managing visualizations across heterogeneous devices [13] and converting analysis into shareable documents for multiple users [14].

1.3 Research Contributions

The major contributions for C2-VA from this research are as follows:

1. A survey identifying the workspace configurations for C2-VA and existing research into supporting this concept (Chapters 2, 3, 4, and 5).

2. Design guidelines for interaction with heterogeneous devices (Chapter 6, 7).

- User preference suggests the combination of proxemics and gestures for interaction in large display environments.
- Diverse interaction styles elicited from users for transferring information between devices during visual analytics in a device ecosystem.

3. A conceptual framework to support the visual analysis process covering the roles of devices, interaction techniques, and visualization tasks in a device ecosystem containing large and small devices (Chapter 7).
    - Flexible user workflow and better insights from data in device ecosystems containing large displays and smartwatches (using the conceptual framework), compared to a traditional large display-only environment.

4. Design considerations to support coordination among multiple users in a device ecosystem (Chapter 8).
    - Better group performance in collaborative decision making in the context of housing search, which showcases the power of group awareness achieved through highlights and widgets added to a visualization dashboard.

5. The Vistrates platform introducing a component model for modular creation of new tools in collaborative, cross-device visual analytics (Chapter 9).
    - Middleware frameworks that work with existing visualization libraries (cf. D3 [15]) to support application developers (Chapter 5).
    - Application examples using Vistrates exemplifying C2-VA.

For all the research projects covered in this dissertation, I was a leading contributor since the project initiation and ideation. I have also led the implementation by contributing significantly to software development. In case of Vistrates (Chapter 9), the development effort was shared with collaborators—Andreas Mathisen and Clemens Klokmose. Furthermore, I have led the design, execution, and analysis of the user studies presented in this dissertation.

1.4 Dissertation Outline

Chapter 2 overviews the modern workspace configurations that contain multiple devices and many users working together towards analytical problems. It includes descriptions of the device ecosystems in example application contexts. This chapter provides an outline of where, when, and why C2-VA is essential.

Chapter 3 introduces the theoretical backend for C2-VA—a visualization pipeline for the collaborative, cross-device visual analysis compared to the traditional visualization pipeline [16]. It also introduces analytical activities, tasks, and device modalities [17, 18] based on Chapter 2. This captures what C2-VA is about.

Chapter 4 provides a detailed review of the related work within C2-VA. The related work is also summarized in Tables 4.1 and 4.2 in terms of their support of heterogeneous devices, collaboration styles, tasks, and users. Therefore, this chapter describes who is the target for C2-VA and how it was supported in related work.

Chapter 5 introduces early work in my dissertation research into software middleware for C2-VA called PolyChrome [19] and Munin [20]. It focuses on the

software primitives to support C2-VA. It also introduces design considerations to answer how to create visualization interfaces across multiple devices.

Chapters 6 and 7 introduce our interaction models for visual analysis in device ecosystems [21–23]—containing large displays and portable devices. These chapters also help in understanding the roles of individual devices in a device ecosystem and how large displays and portable devices can be brought together to complement their capabilities.

Chapter 8 presents methods to help multiple users coordinate across heterogeneous devices. It also introduces InsightsDrive [24] to explore how to represent group activity, to promote awareness of the group during collaboration, along with user studies showcasing the power of this idea.

Chapter 9 showcases the culmination of this dissertation research in a platform called Vistrates [12], which supports the development of new applications of C2-VA. Vistrates, an open source, scales the concept of C2-VA to many analytical activities in data science. Together, chapters 6-9 illustrate how to support C2-VA.

Chapter 10 reiterates a summary of the research contributions and provides suggestions for future research into C2-VA.

Chapter 2: Modern Analytic Settings

Where, when, and why C2-VA is essential.

The face of the modern workspace has changed drastically in the past decades. A simple desk and chair setup is no longer the norm for the modern workforce, especially in computing work. In this chapter, I will present some unique workspace configurations that incorporate multiple devices and multiple people working together to reach their target outcome. They are focused on analytical work to understand information and make decisions from the data. Furthermore, they utilize heterogeneous devices in their device ecosystems; thus exemplifying the target settings for collaborative, cross-device visual analytics (C2-VA).

2.1 The Cockpit

In modern workspaces, a cockpit refers to an enclosed space with compartments of instruments/devices for control. A cockpit is therefore compact with every object accessible to the users while they are sitting down or standing close (Figure 2.1). Cockpits can incorporate one or two people in front of them. Compact as they are cockpits pack a lot of information and interaction surfaces into space. It includes multiple displays, tablets, and input devices such as a keyboard and mouse.

Figure 2.1: Cockpit examples: (A) An analyst's workstation with multiple monitors from Andrews et al. [25] and (B) A commercial Bloomberg trading desk aiding stock traders in investment management with real-time data (photo credits: Chris Ratcliffe/Bloomberg).

A traditional version of a cockpit can contain tiled displays together with keyboard and mouse (cf. space to think [25]), while a modern version of a cockpit can incorporate small and moderate-sized touch displays in an ad hoc manner (cf. surface constellations [26]). Further, they allow the users to distribute the information in space and strategically place instruments in reach of the users. This nature leverages distributed cognition by attaching meaning to space and relies on proprioception.

The users within a cockpit are often engaged in a focused task, barely moving around physically in space. As an individual workspace, the cockpit excels at providing abilities to gather and distribute information across devices. As a collaborative workspace, it promotes quick decision making as the users focus on the same devices and have a common view that supports coordination. There are many variations of a cockpit as it reappears in many forms in the modern world.

Figure 2.2: A Cockpit Configuration: Flight deck of Boeing 787 with its flight instrument system containing many displays and controls for the pilot and co-pilot.

2.1.1 The Flight Deck

Traditionally, the term "cockpit" refers to a flight deck, where the pilots control flight instruments and monitor the flight condition to fly the aircraft. An electronic flight instrument system consists of primary navigation displays to show flight information—airspeed, altitude, heading, vertical speed, etc.—to provide situation awareness, along with alerting mechanisms in case of unusual behaviors [27]. Beyond that, it contains displays to overlay information such as a route plan or weather data on a map or a chart to look at it in context. The control units allow users to provide flight plan, control speed, and navigation. Focused on the primary activity

of managing flight control, monitoring the visual information for situation awareness is the analytical activity performed by the pilots. Therefore, it should require minimal but focused attention from the flight crew. Ergonomics and design play a significant part in the construction of a flight deck for this reason [28]. Tangible controllers and multi-sensory aids in this cockpit help ensure critical information is monitored for situation awareness during the flight.

From a human factors perspective, (1) the displays should ease interpretation, (2) the layout of the controls should be easy to reach and appropriately positioned, and (3) the controls should be natural to use and the components standardized [28]. While they were traditionally analog, digital versions of flight decks, called glass cockpits, with interactive LCD displays are common now (Figure 2.2).

2.1.2 The Tiled Workspace

Tiled-monitor displays are the cockpits of the modern computer worker. They form a modern information workspace present in personal offices across the world. The displays support multitasking and spatial organization of information. Space has a meaning in these multiple monitors. Within visual analytic work, these tiled displays have been found to promote distributed cognition [29]. They can support exploratory analytics activities of large amounts of data, through multiple visualization views, to collect evidence. Bradel et al. [30] performed an evaluation with such a display space. The tiled displays can act as organized storage for information, along with the persisted information and provenance during the analytical process.

Territoriality and integrated workspace patterns happen when multiple people work in this space. The tiled display spaces used by Bradel et al. [30] contain a 2×4 monitor setup leading to a very high-resolution workspace, but the interaction on this large space was using mouse and keyboards. There is a lot of potential for advanced natural input mechanisms. A modern version of this workspace can be made up of tabletops [31, 32], tablets [33], projector displays, and with touch input [31]. A commercial example of a tiled workspace is the Bloomberg trading desk[1] created for stock traders to track the market patterns and make investment decisions.

2.1.3 The Gaming Deck

A more casual version of a cockpit appears in gaming and entertainment. This cockpit is oriented towards visual and sensory immersion to engage the user in the casual activity. In contrast to the others, there can be advanced input devices to enable the immersion in these setups such as a car wheel for racing games or a muscle controller for athletic games. The curvature of the displays contributes to immersion by indulging the peripheral vision of the user. Such a workspace has utility as a presentation space in analytical work—for instance, as Endert et al. [34] point out the choice of metaphor can include a presentation.

[1] https://www.bloomberg.com/professional/solution/bloomberg-terminal/

Figure 2.3: A Control Room: NASA's Flight Control room in Houston, Texas [35].

2.2 The Control Room

A control room is a large physical space with computing devices where a service can be observed and controlled by many people situated in the space [35] (Figure 2.3). Significantly larger than a cockpit, a control room spans the size of a room or sometimes an even larger space, depends on the number of people expected to engage in the process. This setting is meant to control processes from the real world. It is rich with devices that support both personal workspaces (aka cockpits) and shared workspaces through large wall-sized displays. As such, they are powerful for tackling complex analytical activities. They enable activities covering monitoring, exploration, and also the communication of information. Control rooms

are common in large-scale service enterprises such as transportation, aerial flight or mission control, network operations, power plant monitoring, and military facilities. Research into control rooms is prevalent and actively funded by these enterprises.

2.2.1 The Bridge

A bridge in the nautical sense is a room in a ship to monitor the sea and command the ship. It contains many officers taking control of the various aspects of the ship. Bridges are the size of a small room with most of the view covering the water, along with displays on the peripheries and controls for navigation, communication, cargo handling, and resource management. Over time, similar to flight decks, displays have replaced analog sensors in this space for ease of use and effective training. It, therefore, contains many displays and standard input devices such as keyboard and mouse. There is a lot of potential for research including studying decision making in this space. This workspace configuration has also appeared in popular science fiction for space craft control (cf. Star Trek [36]) (Figure 2.4a).

2.2.2 The Command Center

In military operations, command centers provide a centralized command towards a purpose. They contain large wall displays along with many personal computers for a functioning space. Within law enforcement and policing, command centers help monitor the high-crime areas, investigate criminal activity, and protect the community. They bring together an advanced video wall system with surveil-

Figure 2.4: Control Room configurations: (A) A sci-fi bridge of the Enterprise from J.J. Abrams' 'Star Trek', (B) Miami Gardens Police Department using a CineMassive visualization system for crime fighting [37], and (C, D) a smart room called Allosphere at UC Santa Barbara [38].

lance tools and data on a visualization platform including CCTV networks that span the city. Building command centers is a priority in modern law enforcement to monitor the crime and manage the resources of the police force [37] (Figure 2.4b).

Similar centers are popular in traffic management, where multiple analysts watch over the general traffic patterns through CCTV cameras and real-time data, while individual analysts and engineers on desktops deal with tools for historical data. Research in this space has been towards creating visual solutions (cf. CATT lab[2]) and developing better interaction and collaboration techniques. For instance,

[2]CATT lab: https://www.cattlab.umd.edu/

Prouzeau et al. [39] introduced a prototype for interacting with real-time and simulated traffic data for road traffic management on a large wall-sized display.

2.2.3 The Smart Room

Building on the idea of a command center, smart rooms contain advanced sensing technologies to enable effective input and sensing in these room-scale spaces. In a way, they are the futuristic version of current command centers (Figure 2.5). Smart rooms are equipped with movement trackers, microphones, vision, and other sensors to aid the users in their activities proactively. They interpret what people are doing and how to support them in their tasks. As MIT researchers describe them [40], "they are like virtual butlers." In an analytical task, they offer immersion through display and input, where each object and each action has a meaning and technologies blend into the background to support the user goals naturally [3]. They are futuristic at this stage since an actual smart room requires an advanced artificial intelligence that is trained to develop a broad understanding of human activities. Existing smart rooms target specific activities in a living space [41] or navigating an information space (e.g., through an immersive web browser [42]). Among deployed setups, Allosphere [38] is close to a smart room by immersing the users through audiovisual output along with natural interaction (Figure 2.4c,d).

Figure 2.5: A futuristic Control Room designed by Schwarz et al. [43] through contextual inquires of six control rooms. The future of the control room is rich with interactive surfaces and users. It is a target configuration for C2-VA.

2.3 The Telepresence Office

Not all office work is situated in a single location or space. In the modern age of computing, big data analytics needs people from across the world to work together to solve big problems together. An obvious implication of this has been the evolution of the office into a telepresence space. This vision has been of interest since the beginning of this century [44].

An augmented office creates a mixed reality environment where multiple people from different physical locations can be "placed" in a single environment through technological interventions. It supports a seamless hybrid collaboration (co-located + distributed teams). Similar to control rooms, the telepresence office contains many heterogeneous devices to work in an environment (rather than just at a desk).

Over which, this space is augmented to enable collaboration seamlessly by creating a presence for each user and an awareness of each other's activities. While modern offices have not reached this point entirely, augmented reality technologies are making this vision closer than ever. In visualization research, mixed-presence analytical spaces have been explored through tabletop displays for collaborative visual exploration of data (cf. Hugin [45]). In commercial systems, devices such as Facebook's Portal[3] provide a seamless presence to a remote participant by tracking them within the scale of a room. In healthcare, specific equipment is used to perform remote surgeries and diagnosis to enable telepresence [46]. There are two exciting ideas to create a telepresence office.

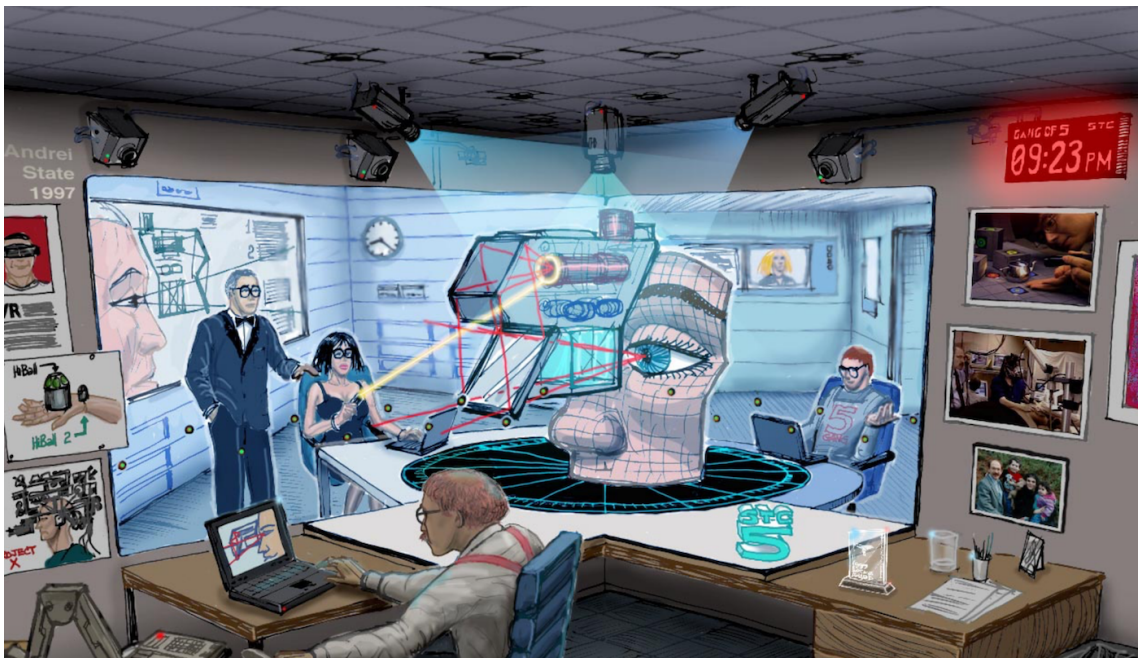


Figure 2.6: A Telepresence Office: A concept sketch of the office of the future from Raskar et al. [44] with projections and room-scale tracking of users.

[3]Portal: https://portal.facebook.com/

2.3.1 The CAVE

The CAVE system is one of the first commercial immersive virtual reality systems arising from the University of Illinois Chicago[4] that can simulate mixed-reality settings. CAVE platforms contain multiple projectors displaying on three to six of the walls within a room. Together with motion-capture mechanisms[5], they can help users interact within a room while having their "presence" captured. Doing so enables a remote collaboration space where distributed rooms are completely stitched together in a single virtual space.

2.3.2 The Office of the Future

Envisioned by Raskar et al. [44], the office of the future is an augmented reality space where projectors and image processing systems actively project remote users into the physical space. Raskar et al.'s vision uses real surfaces in an office as spatially immersive displays by projecting high-resolution graphics onto them. It also requires the transmission of dynamic models of the space over a network for viewing at a remote site. Initially prototyped for telepresence meetings [44], the complete tracking of motion and activity in space opens the door for a wide range of activities. The closest path for realizing this vision is through the use of AR headsets such as Microsoft Hololens. In fact, commercial agencies are already pushing towards such applications[6] (Figure 2.7).

[4]Visbox CAVE: http://www.visbox.com/products/cave/

[5]Vicon: https://www.vicon.com/

[6]Spatial: https://spatial.is/

Figure 2.7: Configuration of the Telepresence Office: Pictures from the demo for "shared infinite workplace" envisioned by Spatial.

2.4 The Intelligent Remote

In this setting, mobile users perform remote activities in the field aided by a group of analysts in an office. This combination of powerful remote intelligence and the reachable mobile user has a massive impact for first responders in disaster situations or even in dangerous and harsh working conditions such as an offshore oil rig, coal mine, or a power plant. For first responders, this can provide simple decision tools to work with to take timely actions. Data visualization techniques are especially powerful here due to their reliance on the high bandwidth sense of human vision. This setting has been an active area of interest for the Department of Homeland Security and other intelligence organizations in the USA [47]. Analytical activities are rich in this space as the analysts need to process data to come up with quick insights to guide to the mobile users (akin to Alfred guiding Batman in the DC universe). This setting was also the focus for the idea of ubiquitous analytics [48], where data analytics can happen anywhere and anytime.

2.4.1 The Human Agency

The situated agency can be a group of people offering support to mobile users. In transportation analytics, this is essential to identify blockades and traffic jams in real time to guide the authorities to the location. In law enforcement, the remote agency can enable predictive policing—usage of predictive analytics to identify potential criminal activity—and deploy the police force to the location. In this situation, the agency would be in a cockpit or a control room with rich computing capabilities, while the mobile users would have, say, smartphones or tablet devices. The closest commercial platform for this configuration is Tableau Mobile together with Tableau Desktop—analytical work done on the Desktop is configurable and presentable on the mobile with features adapted to mobile use.

2.4.2 The Virtual Agency

The situated agency does not have to be human. In modern big data scenarios, remote assistance can be provided through the use of artificial intelligence (e.g., JARVIS in the Iron Man movies). Following the Tableau Mobile example, having features such as Ask Data (Tableau's natural language query platform) [49] can help realize this configuration of an intelligent remote. Effective collaboration between human and machine is critical in this particular configuration.

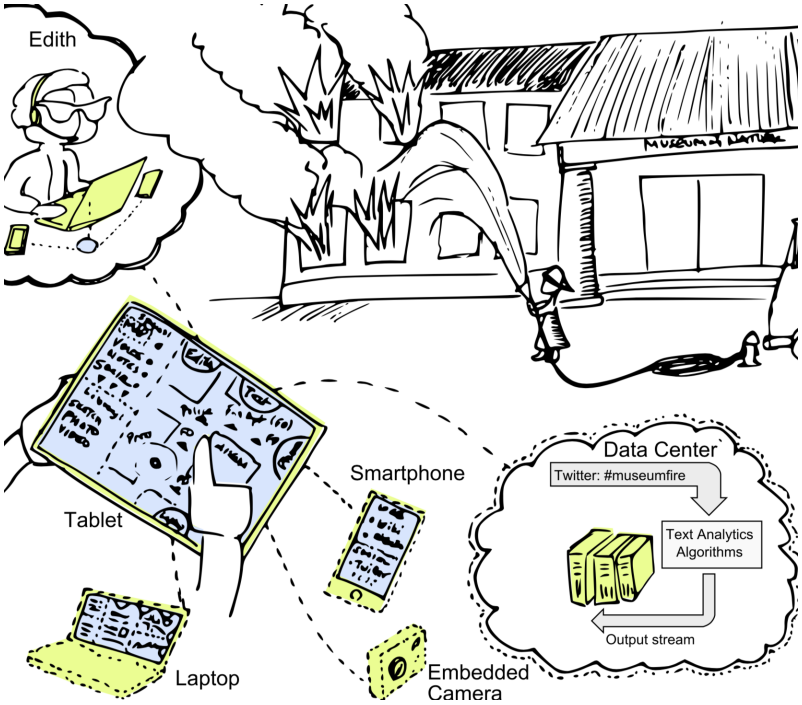


Figure 2.8: Intelligent Remote: A ubiquitous analytics [1] scenario from Prof. Niklas Elmqvist at the University of Maryland: a journalist covers a disaster event while being guided by her editor Edith who is in her office with visual analytics tools.

2.5 The Mobile

Modern information access and analytics can also be fully mobile. Devices in mobile situations can be serendipitously connected to develop insights and share them among groups of users; thus, creating a "virtual" workspace for the modern information worker. For example, a coffee shop owner can use his tablet to process payments and keep track of the daily sales and connect it to his laptop at the end of the day to analyze the best sellers. Similarly, an interactive installation on urban issues can be placed in a public city square from which passing citizens can download and view visualizations of their local community on their smartphones. It will allow

them to study and interact with the data on their personal devices and contribute their opinions back to the public display at a later time. These situations also capture an ad-hoc use of devices and collaboration between users; thus, show signs of a case of C2-VA. There are two interesting possibilities here:

Figure 2.9: The Mobile workspace: Picture from Langner et al. [10]'s VisTiles, a tool for visual analytics on ad hoc combinations of portable devices. Such tools can fully create Mobile workspaces—your office is where you meet your colleagues with your device. Researchers have also explored the analytical implications of such device combinations: for trip planning [50] and socio-economic data analysis [51].

2.5.1 The Water Cooler

This is a mobile situation where the analysts while away from their office might bump into each other at a landmark—a water cooler, a coffee machine, or even in a hallway (cf. Organizational Patterns identified by Coplien and Harrison [52]). This situation leads to a casual analytical activity. For example, when a group of business analysts meets around a water cooler in their office: one analyst shows a new projection that she has been working on her tablet, and the other analysts quickly transfer the financial visualizations to acquire the new proposal to their smartphones from the tablet and discuss it with their teams later.

2.5.2 The On-the-go

This is the fully mobile situation where multiple users communicate across devices from anywhere and anytime and contribute to a shared analytical task. For example, this setting has become the context for interpersonal informatics. Lin et al.'s Fish N' Steps [53] exemplifies a platform where groups can compete with each other for the highest scores in daily step count, and this shared information can encourage them to increase their physical activity. The analytical task performed here involves tracking the step count of groups to compete with each other.

2.6 Summary

The modern analytical settings showcase unique patterns of collaboration, device use, and analytical activity. While the cockpit and the control room target co-located activity, the rest move towards distributed settings. There are also stark differences in the type of devices involved in these configurations with large displays dominating the first three settings, and portable devices playing a significant role in the last two. However, they all represent device ecosystems as they contain interacting devices used for a shared goal. By covering these workspaces, this chapter provides an overview of when, where, and why collaborative, cross-device visual analytics (C2-VA) can be essential.

Chapter 3: A Visualization Pipeline and Activity Perspective

What C2-VA is composed of.

As we saw, modern analytical settings capture unique device ecosystems and user groups. They approach a wide range of analytical tasks; in fact, varying with the context of the application. At the same time, they should utilize these ecosystems to their advantages in supporting shared vs. personal exploration of data, natural interaction, and flexible workflows. To create user interfaces for these settings, we need to understand the aspects that need to be considered, beyond the traditional visual analysis on desktop, to develop solutions optimized for these environments.

3.1 Definitions

As defined earlier, a **device ecosystem** refers to an environment (physical or virtual) of interacting device technologies used by multiple users to achieve a shared goal. The sensemaking process performed in a device ecosystem is therefore across devices, often switching back and forth between different devices or using them together. This process is also across multiple users; the nature of many modern workspace configurations is such that multiple users work together. We call this process, **collaborative, cross-device visual analytics** (C2-VA).

A device ecosystem is also referred to as an environment. Interaction within this environment is called **cross-device interaction** when the user action spans multiple device technologies to carry out an intent (e.g., a smartphone and a smartwatch or a smartwatch and a large display). When the environment is co-located (all the devices and users in the same place), the movement within this environment is called **physical navigation.** It is in contrast to **virtual navigation** that is performed using a mouse on a computer. Physical navigation builds on the theory of proxemics—the study of the human use of space [8].

From an HCI standpoint, the heterogeneous devices in device ecosystems differ in their modalities. Each commercial device for consumer use contains some input and output modalities. A **modality** refers to "a particular mode in which something exists or is experienced or expressed."[1] For example, a smartphone has three forms of output—visual, auditory, and tactile—and many forms of input—touch, speech, vision, motion, orientation, etc. For the sake of simplicity, this dissertation refers to devices with only output modalities as **displays** and devices with only input modalities as **sensors** (nevertheless, they are all devices). With each modality comes an **affordance**—what it offers—for an analytical task. In C2-VA, the modalities of devices should be combined and complemented to support analytical activities, while allowing multiple users to work with them. Hence interaction design for C2-VA has many possibilities as the number of modalities in modern devices increases.

From a visual analytics standpoint, analytical settings support analytical activities among the users—analysts, domain experts, stakeholders, or even the general

[1] https://en.oxforddictionaries.com/definition/modality

public. Inspired by Activity Theory [54, 55], an **analytical activity** is defined as the purposeful actions of human analysts towards an idealized analytic product (e.g., some combination of insights, artifacts, and their intended effects) [56, p. 2]. This activity is mediated by the device ecosystem—the workspace of the users—and visual representations that are the **instruments** for achieving the target outcomes. Analytical activities are built upon **analytical tasks** that require users to perform actions to work with internal and external representations (i.e., visualizations) of data. As characterized by Brehmer and Munzner [57], these tasks have multiple levels with the lowest level covering specific user interactions. The research presented in this dissertation considers some **general activity contexts**—e.g., exploratory analysis of crime datasets by police departments and presentation of best restaurants in a city to a tourism company. The tools and platforms introduced in the research target better outcomes within these activities by supporting multiple users leverage the instruments—the devices in the ecosystem—for visual analytics.

3.2 Vision for Collaborative, Cross-Device Visual Analytics (C2-VA)

Modern analytical settings showcase configurations of multiple devices and users with different expertise. They cover a wide range of analytical activities from exploration of data within a cockpit to presentation and sharing of insights in front of a water cooler. The common thread in these settings is a notion of *transcendence*. Data analysis here goes beyond the traditional individual user model to encompass a wide variety of collaboration styles, types of devices, and analytical activities.

Transcendence across these dimensions represents the concept of collaborative, cross-device visual analytics (C2-VA). The next few sections will cover these aspects of C2-VA and also compare to the traditional visual analytics models.

3.3 C2-VA Pipeline and Reference Model

Traditional visual analysis is driven by a sequence of operations transforming data into interactive visual representations [16,58] presented on a user interface. Reacting to user's feedback, the data filters, representations, and presentation aspects can be modified to show visualizations that offer evidence for the user's analytical goal. Over time, by viewing these external representations of data—visualizations—and building internal representations of the observations, users can create an analytical outcome that drives decisions within the context. As Heer et al. [58] pointed out (top of Figure 3.1), multiple user aspects within the pipeline need to be considered for developing visual interfaces. This includes choices for, (1) cleaning, categorizing, and moderating data, (2) changing visual encodings, and view layouts, and (3) development of hypotheses, collection of evidence, summarization, and reporting.

The reference model needs to adapted to support collaborative, cross-device visual analytics as new aspects arise at different stages within the pipeline. Prominently, the presence of multiple devices needs to be handled at different stages of the pipeline. Besides processing the data at the earlier stages to fit the context of device use, adaptations to match the visual encoding to output modality of the device (i.e., the resolution, the size, and colors) are needed. Tracking findings across devices to

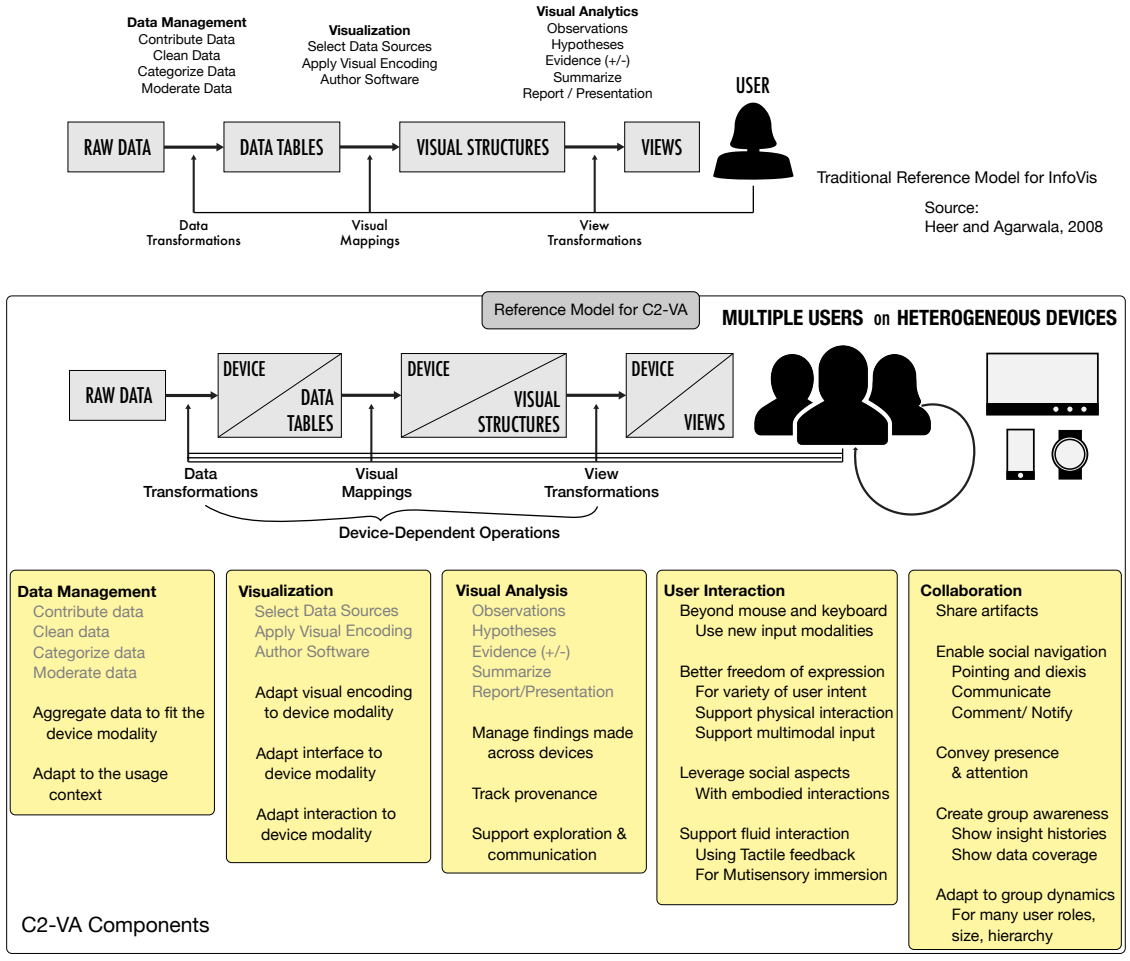


Figure 3.1: Figure presents the reference model for visual interfaces supporting C2-VA (bottom), alongside a traditional pipeline for visual analysis (top). It captures the research challenges and considerations for different aspects of these interfaces.

bring them together is another obvious implication in the presence of heterogeneous devices. Figure 3.1 captures these aspects within the new reference model for C2-VA. In comparison to traditional visual analysis, the data tables, visual structures, and views contain a device tag signifying responsiveness to the device modalities. These aspects are closely considered in later chapters to introduce the cross-device interaction techniques developed in our research.

The presence of multiple users itself adds more complications as methods to help them coordinate are needed. In reality, many of these aspects are often tackled together; however, Figure 3.1 acts as a checklist to ensure new systems are created by considering this model holistically. Vistrates [12] has been the culmination of the presented research and exemplifies how to tackle these aspects.

3.4 Analytical Activities

Another dimension to think about for C2-VA is the analytical activities performed by the users. By understanding these activities, we can create tools that can optimize the outcomes from these activities.[2] Chapter 2 discussed general analytical activities in modern analytical settings. The cockpit is suited for the situation awareness and **real-time monitoring** of data by a limited number of users (two or three). The target outcomes are immediate; for instance, requiring the users to monitor and guide the aircraft in real time. It is also ideal as a personal workspace to perform individual **exploratory analysis** or even **development of analytical**

[2]Following Don Norman's view that tools should refine the activities [56, 59] tackled in an activity-centered design process.

tools. such as visualizations. The control room expands the bandwidth in the participation of the users. The target outcomes from the control room are more long term or requiring many agencies; for instance, for charting the course of a ship, planning the patrol routes of police officers, or managing the traffic with a city. The users bring together diverse expertise into the control room; therefore that can also act as conference spaces to communicate ideas. The telepresence office and intelligent remote take the capabilities of the control room one step further by connecting distributed locations. Finally, the mobile configurations enable **explanatory activities** between users casually meeting at a water cooler or chatting on the go.

Visualization, for all its success within academia, industry, and practice, is still very much a fragmented area in terms of providing a standard, unified method that applies in all (or even most) the analytical settings. For any visual analytics setting, the choice of tool, technique, and approach depends heavily on the dataset, the goals of the analysis, and the expertise of the analyst and audience. Many additional factors come into play: At what stage is the visualization going to be used: during initial analysis or presentation of results? Is the analyst alone, or is there a team consisting of multiple people, each with their own roles and expertise? Are there special devices or equipment, such as smartphones, tablets, display walls, or tabletops, that should be integrated?

All of these concerns give rise to specific choices among the available tools and techniques in the visualization field today. For example, in terms of expertise, a novice may go for a template-based visualization tool such as Excel, a financial analyst may choose a shelf configuration tool such as Tableau [60], and a data scientist

may opt for Jupyter Notebooks [61]. Early on in exploratory analysis, a developer may choose Observable [62] to interactively code their analyses and see immediate results, whereas a more mature project may call for a custom-designed web visualization built in D3 [15] or Vega [63], and a final report for communication may require a narrative visualization tool such as Graph Comics [64], Data Clips [65], or even Microsoft PowerPoint. Graph data may influence a designer to pick NodeXL [66] or Gephi [67], whereas tabular data may require Spotfire [68], and event sequences may mean using EventFlow [69] or EventPad [70]. Obviously, there are currently few synergies between these five determining criteria that we identify—expertise, analytical activity, data, single/multi-user, and device—and committing to one typically means disregarding the others. This fragmentation takes its toll on as participants need to make a "collective compromise," negotiate a common software denominator [71], and expend additional effort to share information, import and export artifacts, and work across visualization systems. Therefore, this fragmentation—in terms of analytical activity, single/multi-user, and device support—remains to be a significant deterrent in creating general solutions for the C2-VA settings from Chapter 2.

3.5 Analytical Tasks

As described earlier, analytical activities are made up of individual tasks that require external representations of data (e.g., visual representations) to develop observations. Visualization community has been active in describing abstract tasks performed with visual interfaces [57, 72]. Yi et al. [72] provide a characterization of

tasks from a low-level interaction perspective—selecting elements in a visualization, encoding a new representation of data, etc. Amar et al. [73] describe the comprehension tasks connected to visualization—finding extrema, sorting, determining the range, etc. Developed from these early efforts, the most popular of the task taxonomies is from Brehmer and Munzner [57] and it considers a multi-level taxonomy for tasks based on why to perform a task, how to perform it, and what are its input/output. Some highlights from this taxonomy [57]:

WHY: Analytical tasks involving visualization are performed to consume, search, query, or produce information. Consumption is motivated by the need to present information, discover evidence, or just enjoy the representation.

HOW: The tasks are achieved by, (1) encoding information into visualizations, (2) by manipulating existing visualizations (e.g., selection), or (3) by introducing new elements into a visualization (e.g., annotation).

WHAT: The tasks are performed on the semantics within the data—the values, ranges, clusters, correlations, etc. This could be anything from the data that has an analytical value to the user.

As we move towards C2-VA, supporting these tasks in the analytical settings is essential. I will discuss this taxonomy again when introducing the interaction models in Chapter 7 to showcase how the tasks can be performed by utilizing multiple devices in an ecosystem.

3.6 Types of Devices

Another dimension to consider in C2-VA is the type of devices involved in the sensemaking process. Device technologies have evolved at an unprecedented rate over the past decade. Beyond personal computers (laptops and desktops), the research presented considers these technologies:

- **Wall-sized displays:** Often created using projectors, these displays offer a large canvas for displaying a visual interface. However, limited resolution, brightness, and color spectrum of projectors affect the user experience. CAVE systems[3] often contain wall-size displays.

- **Large devices:** Devices that contain a large display (> 50-inch) with a high resolution (1080p HD or 4K UHD). Beyond keyboard and mouse, these devices may support touch input. They also fit well with 3D input using depth cameras. Examples: Promethean ActivPanel[4] and Microsoft Surface Hub.[5]

- **Tiled-monitor displays:** Created by placing monitors in a grid, these setups are ideal for creating cockpits from Chapter 2. However, they require special stands—sometimes, custom built—to put the displays together. The presence of bezels or gaps and the lack of input capabilities that work across displays haunt these setups. Examples include configurations from TV companies. [6]

[3]CAVE: http://www.visbox.com/products/cave/

[4]ActivPanel: https://www.prometheanworld.com/products/interactive-displays

[5]Surface Hub: https://www.microsoft.com/en-us/surface/business/surface-hub

[6]https://displaysolutions.samsung.com/digital-signage/video-walls

- **Portable devices:** Tablets, smartphones, and smartwatches are modern personal devices owned by many people. While these devices are physically small, their displays are rich in high resolution and colors and support touch input. Many of these devices contain cameras (visual input), microphones (speech input), and sensors for movement, orientation, tilt, and gravity; these are ideal for advanced input. While smartphones are handheld, smartwatches are wrist-worn and keep the user's hands-free.

- **Tracking sensors:** Motion capture systems can work together with large displays to track user motion within a room. Examples include VICON devices[7] and depth cameras.[8]

Many other technologies are not considered in this research including the internet of things (IoT devices), augmented reality and virtual reality headsets, and brain-computer interfaces (BCI). All of these devices have a high potential to add to the heterogeneity in input and output modalities in C2-VA.

3.7 Components of the C2-VA stack

Compared to traditional visual analysis on a single desktop computer, there are many challenges in developing user interfaces for C2-VA (Figure 3.1). In general, tools for C2-VA require a full stack development for creating and using device ecosystems for collaborative visual analytics. Figure 3.2 captures the components

[7]VICON: https://www.vicon.com/products/vicon-devices

[8]Intel Realsense: https://realsense.intel.com/depth-camera/

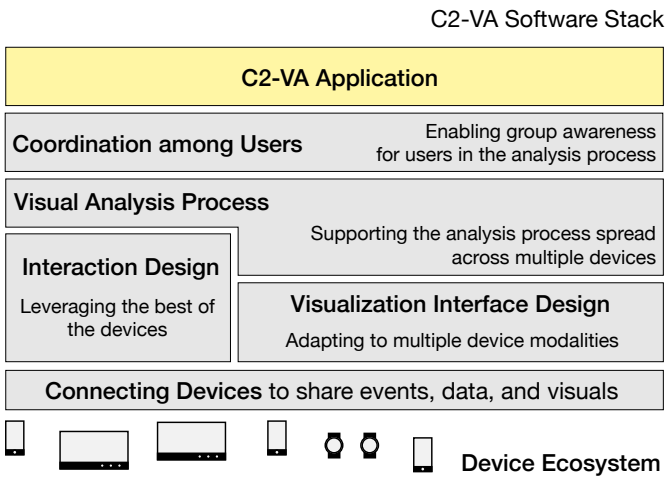


Figure 3.2: Full stack development for C2-VA requires holistic solutions that answer these components. This dissertation research tackles each one of these components and then introducing a platform called Vistrates [12] to connect them together.

within the C2-VA software stack. At a low-level, it contains infrastructure for **connecting devices** together, access their graphics engine, and interaction capabilities. Building on top is logical modules that define the **interactions** within the device ecosystem and visualization across devices. Building on interaction, an **analytical process** consists of many stages including collecting data, gathering evidence, and generating hypotheses (cf. sensemaking process 3.3). The roles of the devices in supporting this process should be considered. When engaged in this analytical process, **coordination mechanisms** among multiple users are further needed. Finally, **application interfaces** can be built on top of these components to support visual analytics of target data within the application domain.

Beyond the above system's perspective, another component of C2-VA is the sensemaking process itself. As illustrated by Pirolli and Card [2], the sensemak-

ing process characterizes that the users would go through multiple stages, iterating back and forth, to develop the final outcomes (Figure 3.3). It connects to the type of analytical activities supported in the application context. C2-VA requires interventions to support these stages during collaborations and across devices. In this dissertation, the focus is on the parts of the sensemaking loop (and the later stages of the foraging loop), where visualization and visual exploration play a significant role in collecting evidence towards developing and verifying hypotheses for insights.

A common goal across all the components of C2-VA is to support multiple users work across heterogeneous devices. As such the research presented in this dissertation (Chapters 6, 7, and 8) supports team-first visual analytics, where the visualization interface considers the needs of the team as a whole and seamlessly processes the group activity without deviating the users from their tasks. This work is combined into a platform for C2-VA application development called Vistrates [12] presented in Chapter 9.

3.8 Summary

Developing visualization interfaces for the modern analytical settings in Chapter 2 is not straightforward. The standard visualization pipeline that defines how data transforms into an interactive visualization needs to be adapted to consider multiple users and devices (Figure 3.1). Beyond supporting analytical activities and tasks, applications need to be built ground up to answer some of the core components of C2-VA. This includes communication channels between devices (Chapter 5),

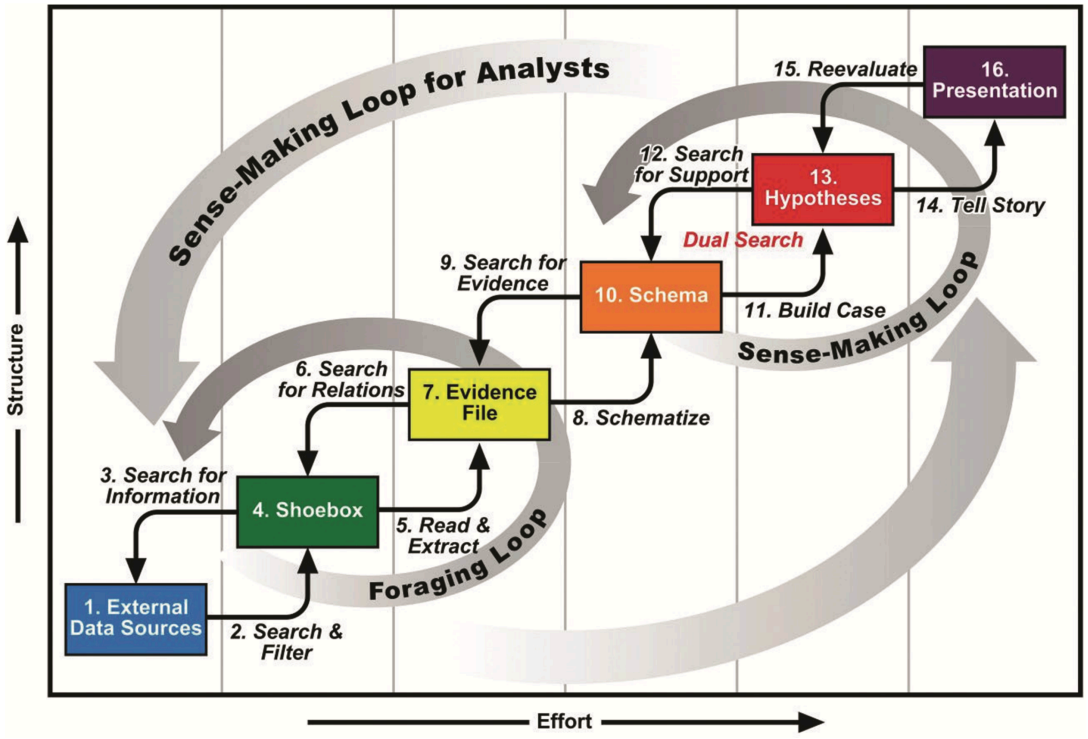


Figure 3.3: The sensemaking process by Pirolli and Card [2] capturing the stages of data analysis. While this is an abstraction of the process, analysts in reality would focus on particular steps based on the data and the target outcome.

visualization and interaction design for the device modalities (Chapters 6 and 7), support for the visual analysis process (Chapter 7), and coordination among multiple users (Chapter 8) involved in the analytical activities (Figure 3.2).

Chapter 4: Related Work

Who was targeted and how—users, devices, and contexts.

The concept of C2-VA tackles many aspects of human-computer interaction (HCI), sensemaking [2], visual analytics [6], and computer-support cooperative work (CSCW). As such, there is a considerable body of related work that provides important knowledge about these individual aspects relevant to accomplishing C2-VA. The following sections in this chapter will present the foundations of C2-VA. Following that, Table 4.1 highlights some new visual analytics platforms to understand their support for the C2-VA settings from Chapter 2.

4.1 Device Ecosystems

An ecosystem is defined as a "community of interacting organisms and their physical environment." A device ecosystem in the context of C2-VA refers to an environment of interacting devices used together to reach a shared goal. When moving from the traditional visualization model on a desktop, a significant effort went into using heterogeneous devices to create ecosystems for visual analytics [17, 74]. In the next paragraphs, we provide an overview of distributed user interfaces and interaction techniques for heterogeneous devices in visualization and HCI.

4.1.1 Distributed User Interfaces (DUI)

Distributed user interfaces (DUIs) distribute interface components across one or several of the dimensions input, output, platform, space, or time [48]. As we draw nearer to a vision of truly ubiquitous computing (ubicomp) [3], such interfaces are becoming increasingly important. Several models for DUI and ubicomp interfaces exist, such as the CAMELEON-RT middleware [75], distributed versions of the model-view-controller (MVC) paradigm, and the VIGO model [76] for building distributed ubiquitous instrumental interaction applications. In terms of toolkits and frameworks, examples include BEACH [77], MediaBroker [78], and the Proximity toolkit [79], and ZOIL [80]. jBricks [81] is a Java toolkit for rapid development of applications on cluster-driven wall displays.

Existing work by Bi and Balakrishnan [82] and Endert et al. [34] investigate the use of large displays and multi-screen environments in personal computing and standard office environments respectively. Bi and Balakrishanan [82] found that the usage patterns for large displays indicate that the users spend more time organizing the contents into focal and peripheral regions, thus establishing the need for automated/semi-automated layout management, legibility, and window management operations to save time. Endert et al. [34] discuss the various ways in which large displays enable extended memory. They also provide suggestions regarding the display configuration, keyboard placement, mouse placement, and user stance for large display use. Moreland [83] present various lessons learned in using large-format displays and try to extend their use as a visualization space.

More recently, Panelrama [84] supported the creation of cross-device web applications by splitting views and synchronizing interaction across devices. XD-Browser [85, 86] enables non-technical users to adapt existing single-device web applications for cross-device use and describes the design space in this context. Park et al. [87] proposed AdaM, a distribution technique for spreading interface components automatically across devices based on the semantics provided by developers (or users) and a constraint solving algorithm.

4.1.2 Interaction and Interface Design for Heterogeneous Devices

To achieve Mark Weiser's vision [3], public displays and handheld computers present in an ecosystem should be harnessed to create a context-aware system that captures knowledge about surrounding users and devices (actors).

Proxemics—the study of the human use of space [8]—has been utilized in this context as a way to create connections between users and devices in an environment [41, 88, 89]. Greenberg et al. [41] discuss the use of proxemic attributes such as distance, orientation, identity, movement, and location, for building a structured implicit interaction model. It includes (1) using distance-based interaction zones that define the reaction of the display [90], and (2) orientation-based understanding of the user attention [91]. Peck et al. [92] used distance-based interaction zones to control the scale of interaction and found evidence of naturalness in physical navigation in front of a large display as the users tend to associate visual scale (e.g., seeing overview/detail based on distance) with the interaction scale.

Gestural interaction has been possible since the advent of stereoscopy in HCI using depth cameras and sensors [93]. Gestures can be predefined or user-defined, and they can only be triggered by explicit motion that deviates from regular user interaction in an environment. From a gesture design perspective, Nancel et al. [93] studied the design of mid-air pan-and-zoom movements, including uni-bi-manual interaction, linear vs. circular movements, and guidance for mid-air gestures. They found that linear gestures, involving a linear movement and clutching, were faster than circular gestures, and two-handed gestures were faster than one-handed. Recently, Vogiatzidakis et al. [94] outlined the gesture elicitation studies in HCI contexts and classification of gestures thus obtained. Finally, interaction with arm-mounted devices through gestures was also investigated; e.g., Rekimoto [95] and Ashbrook et al. [96] explored their use as unobtrusive on-body input devices.

Cross-device interaction has been used to share information, chain tasks, and manage sessions across devices [97]. Pick and Drop [98] was one of the first cross-device techniques to exploit the physicality of large displays and mobile devices in an environment using a pen. Duet [99] enabled joint interaction across a watch and a phone using multi-device gestures (for instance, flip the watch and tap phone). More recently, WatchConnect [100] toolkit helps rapid prototyping of cross-device applications for smart devices through events created from on-surface, over-the-surface, and proxemics-based interaction. In workspaces with large displays, SleeD [101] used a sleeve display to interact with a large display wall. Finally, Brudy et al. [102] provide an exhaustive literature review and a taxonomy for cross-device systems, along with a unified terminology to connect the state-of-the-art.

Developing adaptive interfaces that can run on any device is a prominent challenge beyond interaction design. Thevenin and Coutaz formalized the idea of *plastic user interfaces* [103] that can adapt to a target device modality by modifying the rendering techniques and behavior of the system. Calvary et al. [104] extended the idea to create a reference framework to generate and classify user interfaces supporting multiple targets and multiple contexts. However, to develop interfaces that can adapt to any device, it is crucial to understand the tradeoffs of using the target devices. Tan et al. [105] studied a large projection display against a desktop monitor to quantify its benefits to an individual user.

4.1.3 Visualization on Large Displays

Utility. Large displays have been shown to improve productivity in office settings [106]. They have also long been of particular interest to the visualization and visual analytics community, presumably due to their large screen real estate and the potential for collaborative analysis [107]. The size of such displays allows for using *physical navigation* to support the classic visual information seeking mantra [108]: get an overview of the data from a distance, and move closer to the display to access more details [107, 109–111]. This general characteristic has motivated work explicitly focusing on physical navigation and spatial memory: Ball and North [112] as well as Ball et al. [113] showed that physical navigation is an efficient alternative to virtual navigation; however, the effects depend on the actual setup, interface, and tasks [114–116]. Liu et al. [116] compared physical navigation on an ultra-high-

resolution wall display against virtual navigation on a desktop display for a data classification task. They found that the wall display was more effective for such tasks with higher difficulty levels. In general, large displays can be beneficial for co-located collaborative scenarios [117], especially as they can promote different collaboration styles [118, 119] and benefit from physical navigation [88]. However, challenges regarding territoriality [30, 118], coordination costs [119], and privacy [120] must be considered. Finally, Jakobsen and Hornbæk [114] studied the relationship between display size and usability of map visualizations. However, they did not find any significant benefits in using the large display for their scenarios.

Interaction. Ball et al. applied embodied interaction [121]—interaction based on our familiarity and facility with the everyday world—to visualization on large displays [112, 122]. They found that devices such as 3D gyro mouse, touch screens, and head tracking equipment dramatically increase the user performance by improving their physical range of movement and performance time. Andrews and North [25, 123] discussed the importance of embodiment for sensemaking on large displays through a new analytical environment called Analyst's Workspace. This workspace aims at permitting the use of space as a cohesive whole where the position has a meaning to the analyst. In multi-user scenarios, proxemics [115, 124–126] can be used to provide personalized views or lenses. The comprehensive design space of such lenses was explored through BodyLenses by Kister et al. [127]. Yost et al. [128] explored physically adaptive visualizations for taking advantage of the human perceptual abilities by say using light colors that blend with the background and can only be seen when close to a display. Isenberg et al. [111] explore this form of re-

sponsiveness based on the spatial attributes of the user. For future visual interfaces that aim to support sensemaking in large display and multi-device environments, embodied interaction models can be beneficial to leverage our innate knowledge of naïve physics, body awareness, and social awareness [129].

Tabletop displays. These displays have been used for creating visualization systems for tree comparison [130], collaborative document analysis [131], and mixed-presence collaboration in general [45]. For tabletops, interaction techniques within the physical space around the display have been developed using tangibles that can be freely carried around (e.g., transparent lenses [132]). This was further extended to create *graspable* tangible views [133]. Isenberg et al. [134] compared the tabletop interfaces in office workspaces and public settings such as museums.

4.1.4 Visualization on Heterogeneous Devices and Technologies

Touch and speech. Novel input and output modalities of modern devices to interact with data are essential to the vision of C2-VA in a device ecosystem. In this direction, SketchStory [135] introduced one of the early attempts to use touch input and sketching as a means for storytelling with data visualizations. Touch interaction has also been explored for interaction and exploration of specific visual representations (cf. hierarchical stacked graphs [136] and network visualization [137]). Users of visualization systems can also express their questions and intents more freely using natural language and speech queries [49, 138, 139], allowing them to naturally perform the tasks in visual data exploration.

Tablets and smartphones. Visualization on mobile devices has been of interest with the prevalence of these devices [140, 141]. Techniques suitable for smartphones and tablet devices were also introduced for various datasets including categorical [142], spatiotemporal [143, 144], and hierarchical data [145]. Flexible analytical work can be performed by utilizing mobiles with other devices (cf. SRS system [146]), thus creating multi-device environments (MDEs). In this direction, VisTiles [10] presented the styles in coupling multiple small-screen devices to explore data. Thaddeus by Woźniak et al. [147] coupled spatially-aware mobile phones with tablets for interaction with visualizations. Their cross-device interactions were spatially sensitive, leveraging the configuration and movement of devices. A recent study by Plank et al. [51] showed that in the presence of multiple tablets per user, there was an under-utilization and hesitancy in using them for exploring data visualizations. This nature shows that the devices are not yet entirely "invisible" (cf. ubicomp [3]) but require better interaction methods and interface strategies to help users overcome the hesitancy in using them in MDEs. Beyond this simple combination, the combination with a large display is promising as it allows to separate shared and private information and enables users to switch between working in concert and working alone [148]. A fundamental operation in an MDE is the ability to transfer content from one device to another; Langner et al. [149, 150] investigated this for a spatially-aware smartphone and a large display and Chung et al. [151], through the VisPorter system, presented concepts for using a tablet as a document container for sensemaking tasks. Focusing on interaction with a wall display, Chapuis et al. [152] proposed to use a tablet as storage for multiple cursors and content items, while Liu

et al. [153] investigated collaborative touch gestures for content manipulation. For data exploration, Spindler et al. [133] used handheld displays above a tabletop as graspable views to show altered perspectives of visualizations. Recently, Kister et al. [154], through the GraSp system, investigated the use of spatially-aware mobiles in front of a display wall as personal views into a graph.

Smartwatches. Visualizations for smartwatches have also been studied in recent years. Chen [155] supported visualization of large time series on a smartwatch through aggregated statistics placed on the borders along with a detail visualization in the center. More recently, the design of glanceable visualizations for smartwatches and physical trackers [156,157] has been explored to find designs that are radial (cf. donut charts) to be suitable. In MDEs, von Zadow et al. [158] used an arm-mounted (mobile) device to allow users to have their hands free for interaction. However, the combination of smartwatches with large displays, especially for visual analysis, is underexplored. The CurationSpace of Brudy et al. [159] utilized a smartwatch for selecting, adjusting, and applying instruments—in essence, for content curation on large displays—as well as personalized feedback. Most research on smartwatches focused on how to overcome the limitations of these devices, i.e., the limited input and output possibilities. The input to a smartwatch can be expanded with physical controls (e.g., a rotatable bezel [160]), mid-air gestures [161], and spatial movements [162]. Furthermore, the watch's native inertial sensors can be used to enrich touch input on other devices such as smartphones [99] or tablets [163] with pressure and posture information. For output, haptic feedback [164], mid-air visuals [165], and on-body projections [166] have been proposed.

4.2 Collaboration

An obvious implication of a device ecosystem is the ability to support multiple users work together. Amid increasingly complex data, collaboration is becoming a necessity for effective data analysis [167]—to bring together the expertise of multiple analysts or even to communicate insights and knowledge among users.

4.2.1 Computer-Supported Cooperative Work (CSCW)

Collaboration is often classified by space (co-located or distributed) and time (synchronous or asynchronous) [168] (cf. Johansen's CSCW matrix [169]). The field of computer-supported cooperative work (CSCW) focuses primarily on the theory, design, and practice of software used concurrently by multiple users [168]. While the scope of CSCW spans decades and disciplines, here we focus on *coordination*: mechanisms that facilitate the collaborative process on a meta level without directly contributing to the collaborative task [170]. In general, such coordination is vital to ensure efficient collaboration, particularly as the number of collaborators grows. For example, source revision control systems provide operations to check out, commit, and resolve conflicts during collaborative work between multiple developers. Collaborative editors such as Google Docs provide coordination mechanisms such as chat, comments, shared highlighting, and revision histories.

Common ground, or "mutual knowledge, mutual beliefs, and mutual assumptions" [171] is one of the key aspects of efficient coordination. Achieving and maintaining such grounding in communication requires *group awareness*: an

up-to-date understanding of the interactions of other collaborators in the shared space [172]. It is vital in remote collaborative sessions since such settings lack familiar physical awareness cues. Several approaches in general HCI and CSCW focus on providing group awareness. Gutwin et al. [173, 174] propose several awareness widgets based on "radar" overviews of the shared space. Tang et al. [175] takes this a step further by drawing "ghost" arms of the remote participants on a tabletop display. Tuddenham and Robinson [176] derive design guidelines for providing group awareness on shared touch surfaces using territories, orientation, and implicit communication, and later study these effects empirically in different collaborative settings [177]. In general, Kiani et al. [178] conducted a survey of task awareness and presence awareness of 26 teams from 12 companies worldwide. They found that many factors influence group awareness including prior work experience, team size, and frequency of interaction.

Presence is a unique form of group awareness, where the idea is that the spatial proximity of a user to an object conveys an interest in that object. This effect is intrinsic to the physical world, but is more elusive in digital settings; for this reason, 3D virtual environments often use presence and proximity. Nevertheless, the concept can be used to good effect in standard desktop applications. For example, Chris Harrison's "Inhabited Web"[1] shows the current viewers on a webpage and their position on the page in the browser itself. Laufer et al. [179] created a synchronous collaboration extension to the Prezi presentation tool where avatars represent the current focus of each collaborator on the zoomable canvas.

[1] http://www.chrisharrison.net/index.php/Fun/InhabitedWeb

4.2.2 Collaborative Visual Analytics

Collaborative visual analytics can be succinctly defined as the shared use of visual analytics software by multiple users working towards a common goal. It has been named one of the grand challenges of the field [167]. The value proposition for this practice is simple: involving multiple analysts generally improves the analytical outcomes in terms of time, quality, or both. As a case in point, Mark et al. [180] discovered significant improvement for collaborative visualization compared to single-analyst usage, and Balakrishnan et al. [181] similarly point to substantial performance gains when analysts used a shared visual representation. However, while collaborative VA and visualization has many similarities with CSCW and groupware, it also has its own distinct set of challenges [58, 117], including its typically expert analyst audience, its focus on sensemaking rather than productivity, and its long-term, multi-stage, and multi-representation workflow.

Asynchronous and distributed collaboration is the most common setting of collaboration. Web technologies are useful for this collaboration style. Asynchronous social data analysis [4] was best captured in IBM's now-defunct ManyEyes [182] website. Sense.us [183] supported bookmarking of visualizations, graphical annotations, and social interaction—all essential components for distributed collaboration in visual analytics. Dashiki [184] enabled multiple users to build wiki-based visualization dashboards. Many of these ideas have propagated into modern visualization platforms—for example, Tableau [185] supports users to share their visualization dashboards and stories on the web through its public platform.

Co-located and synchronous settings are also standard. Here multiple analysts work together on an analytical task in the same room. Many of the visual analytics systems for co-located collaboration have been guided by work by Robinson [186] as well as Isenberg et al. [187], which both study the behavior of individuals as well as groups in the co-located paper-based analysis. The most straightforward approach is to connect multiple laptops and devices in the same—VisPorter [151] is an example system to enable this. Visual analytics in such environments was pioneered by a collaborative tree analysis tool for digital tabletops from Isenberg and Carpendale [130], but similar work includes Lark [188], which externalizes data pipelines on a shared touch surface, and Cambiera [131], which captures documents read and queried within text collections. A more detailed overview of this space is provided earlier through the discussion about multi-device environments (MDEs).

Other forms of collaboration are seen in special use cases. Provenance (i.e., for understanding a users reasoning process) and handoff (i.e., for transfer of knowledge) play a significant role in establishing asynchronous and co-located settings Zhao et al. [189] explored techniques—*annotation graphs*—for capturing, grouping, and analyzing annotations during visual exploration to support handoff. Capturing and visualizing analytic provenance has been an active interest area in recent times [190–192]. In this direction, techniques for grouping and capturing visualizations and their states through meta-representations can be essential to understand the user's process (cf. chart constellations [193]). Beyond this, group awareness and coverage are especially useful in synchronous and distributed settings (and all forms of collaboration in general).

4.2.3 Coordination and Group Awareness in Visual Analytics

Collaborative visual analytics requires particular attention to coordination mechanisms due to the complex nature of sensemaking. For example, the branch-explore-merge protocol [148] is a prime example of a sophisticated coordination mechanism that enables participants to branch from the shared state, explore the data independently, and merge back any new findings to the shared exploration. Heer and Agrawala cite awareness as one of the primary design considerations of collaborative visual analytics [58]. Similarly, Balakrishnan et al. [181] provide awareness to users using shared visualizations. Finally, the Hugin [45] visual analytics tool provides awareness based on radar widgets [173, 174] and remote interactions [175].

Heer and Agrawala also propose *social navigation* [194], where the presence and activities of multiple users in digital space are recorded and visualized, as a way to aggregate the actions of multiple analysts in collaborative visual analytics [58]. One concrete approach based on social navigation is Scented Widgets [195], which embed visual representations of prior use in-situ on the interface elements—such as range sliders, lists, and hierarchies—themselves. In a similar vein, the collaborative brushing proposed by Isenberg and Fisher [131] for text documents was extended to tabular data by Hajizadeh and Tory [196]. Mahyar and Tory [197] take this even further by connecting collaborators' findings using an approach they call "Linked Common Ground." Finally, Sarvghad and Tory found that dimension coverage increases the breadth of exploration without sacrificing depth for a single user [198] and reduces work duplication in asynchronous collaboration [199].

4.3 Toolkits, Grammars, and Platforms for Visual Analytics

Visualization tools for novice users, such as Excel, support basic charting and data transformations. Shelf-based visualization tools such as Tableau [185] support easier configuration of visualizations by drag-and-drop into "shelves" of visual variables. For visualization design, iVoLVER [200] uses visual programming to help non-programmers extract data from multiple sources and transform it into animated visualizations. iVisDesigner [201] supports the creation of visualizations by utilizing template-based mappings. More recently, Data Illustrator [202] and DataInk [203] explore the concept of lazy data bindings for better expressiveness in visualization design for non-programmers. Faceted visualization browsers such as Keshif [204] use predefined interactive facets representing data for novice users.

Visualization toolkits for developers such as Protovis [205] and D3 [15] support web visualization with a data model that maps data items to visual marks for SVG-based interactive graphics. D3 also binds the data to the Document Object Model (DOM) of web browsers and supports basic extract, transform, load (ETL) operations to create data objects for custom visualizations. Recently, high-level visualization specification grammars have been developed, such as Vega [206], Vega-lite [207], and Atom [208]. These grammars are oriented towards analysts with technical expertise in visualization development.

Interactive notebooks have also recently gained popularity in data science and visual analytics, as they promote collaboration by sharing. They adopt a literate computing-based approach to programming, where executable code is interweaved

with text and images to create interactive narratives. Jupyter Notebook [61] is a web-based interactive notebook that connects to a kernel capable of executing code in languages such as Python, R, Ruby, JavaScript and many more. Jupyter provides integrations with analytics and visualization frameworks such as SciPy [209] for analytics and Altair [210] for declarative visualizations. However, Jupyter does not support real-time collaboration out of the box. Google's Colaboratory [211] is a Jupyter implementation using the Google Drive backend for real-time collaboration. However, collaboration is on the level of editing and not interaction. Among new platforms, Observable [62] has been popular for JavaScript-based interactive notebooks, primarily used for creating interactive visualizations. It provides a reactive programming model where re-execution of a code-block will result in a re-execution of any code block that depends on it. Notebooks written in Observable are easy to fork and share and they also support real-time collaboration in editing and commenting when using the notebooks (since March 12, 2019[2]). However, they are not focused on analytical work across heterogeneous devices and activities in the sensemaking beyond visualization development.

4.4 Summary

Tables 4.1 and 4.2 outline the related work in terms of specific dimensions:

- Collaboration styles supported including **S**ynchronous-**C**olocated, **A**synchronous-**D**istributed, and **other** combinations. As you will notice from the tables, a

[2] https://observablehq.com/teams

lot of existing work with heterogeneous devices considered co-located spaces. There are a few unique systems that support multiple collaboration styles such as VisPorter [151], Hugin [45], and our own InsightsDrive [24].

- In terms of devices, support for novel devices such as large displays (**LD**), handheld devices (**HH**), and wrist-worn devices (**WW**) is highlighted. Notice that a lot of work has tackled large displays, with a few systems combining them with handheld devices (and even fewer with wearables). Early large display research has focused on tabletop systems [45, 130, 131, 188], but vertical/wall-mounted large displays are of focus in recent years [137, 150, 154].
- In terms of Analytical **Act**ivities, many of the systems are focused on a specific activity with a specific dataset. For example, the activity can be a visual exploration of a crime dataset. They have also been tested for that activity. Exceptions include (1) SketchStory [135]: supports storytelling along with visualization creation/interaction, (2) Dashiki [184]: supports charting along with dashboard creation for visual exploration (similar to Tableau [185]), and (3) our own Vistrates platform [12] presented in Chapter 9.
- For abstract tasks [57], the support from systems is subjective to how we define these tasks. We considered a basic definition where **consume** implies the ability to present/create new visualizations of data for consumption, **search** means the ability to navigate, locate, or explore, and **query** is the ability to compare or filter visualizations. It is interesting that systems that combine multiple devices tend to support more tasks. However, this observation needs further investigation.

- **Users** corresponds to the type of users considered for the tool or the evaluation of the tool. Many systems consider **general** user groups and perform user studies with graduate students or university staff. **Specific** users refers to programmers, domain experts, or a particular user group (e.g., police officers for the use of mobile VALET [144]).

In general, these systems form the building blocks for supporting the analytical configurations in Chapter 2.

Table 4.1: Collaboration categories: **S**ynchronous-**C**o-located, **A**synchronous-**D**istributed. **LD** is large display, **HH** is handheld, and **WW** is wristworn. **Act.** is activity.

Systems	**Collaboration**			**Devices**			**Act.**	**Tasks [57]**			**Users**	
	S-C	A-D	Other	LD	HH	WW	Many	Consume	Search	Query	General	Specific
Jupyter [61]			✓					✓	✓	✓		✓
Colaboratory [211]			✓					✓	✓	✓		✓
Observable [62]			✓					✓	✓	✓		✓
Tableau [185]					✓		✓	✓	✓	✓	✓	
Spotfire [68]								✓	✓	✓	✓	
iVisDesigner [201]								✓			✓	
Data Illustrator [202]								✓			✓	✓
Andrews et al. [25]	✓			✓					✓		✓	
Isenberg et al. [111]				✓					✓		✓	
SketchStory [135]				✓			✓	✓			✓	
BodyLenses [127]	✓			✓					✓		✓	
VisPorter [151]	✓		✓	✓	✓				✓	✓	✓	
Smarties [152]	✓			✓	✓				✓	✓	✓	
GraSp [154]	✓			✓	✓				✓	✓	✓	
MCV [150]	✓			✓	✓				✓	✓	✓	
ITC [130]	✓			✓						✓	✓	
LARK [188]	✓			✓				✓	✓	✓	✓	
Cambiera [131]	✓			✓					✓		✓	
Hugin [45]	✓		✓	✓	✓			✓	✓		✓	
Embodied Lens [132]	✓			✓						✓	✓	
BEM [148]	✓			✓	✓				✓	✓	✓	

Table 4.2: Continued from previous page ...

Systems	Collaboration			Devices			Act.	Tasks [57]			Users	
	S-C	A-D	Other	LD	HH	WW	Many	Consume	Search	Query	General	Specific
Spindler et al. [133]				✓	✓				✓	✓	✓	
VALET [144]					✓			✓	✓	✓		✓
Thaddeus [147]	✓				✓				✓	✓	✓	
Plank et al. [51]	✓				✓					✓	✓	
Sollich et al. [212]				✓	✓				✓	✓		✓
CurationSpace [159]				✓		✓				✓	✓	
SleeD [158]				✓		✓				✓	✓	
VisTiles [10]					✓				✓	✓	✓	
Sense.us [58, 213]		✓						✓		✓	✓	
ManyEyes [182]		✓						✓		✓	✓	
Dashiki [184]			✓				✓	✓		✓		✓
Balakrishnan et al. [181]			✓						✓	✓	✓	
Munin [20]	✓			✓	✓							✓
PolyChrome [19]	✓	✓		✓	✓							✓
Proxemic Lens [21]	✓			✓					✓	✓	✓	
Visfer [22]	✓			✓	✓			✓		✓	✓	
David/Goliath [23]	✓			✓		✓			✓	✓	✓	
InsightsDrive [24]	✓		✓						✓	✓	✓	
Codestrates [214]	✓	✓	✓		✓		✓					✓
Vistrates [12]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Chapter 5: Fundamentals: Device Roles, User Interaction, and Visualization Interfaces

Basics on how to create a device ecosystem.

The related work offers evidence of the advantages of heterogeneous device ecosystems. The purpose of this chapter is to set up the fundamentals of C2-VA. It includes our understanding of the roles of heterogeneous devices derived from existing work and middleware frameworks that we developed to establish connections between multiple users and devices. This chapter lays the foundation for the interaction techniques and platforms introduced in the rest of the dissertation.

5.1 Roles of Devices

Different displays have different capabilities during visual analytics. Here we talk about their tradeoffs and potential roles during an analytical activity.

5.1.1 Large Devices

By virtue of its size, a large device can serve as the primary display that provides multiple visualizations of a dataset. Their positive effects on the visual exploration process are well known—providing a large "space to think" [25,107,150]

and opportunities for multiple users to work together [30,127] Thanks to its size, the large display can be used by multiple analysts in parallel, thus serving as a **public and shared display** that is visible and accessible to everyone. The environment in front of the large display is also powerful, often offering contextual information about the user's activity [154]. For instance, when a single person is using the display to talk to an audience, there is a good chance that it is a presentation. With touch interaction [9,136,142], analysts can directly interact with the visualizations on the large displays: data elements can be selected by tapping them, the axes can be used to offer additional functionality (e.g., to sort the data), and layouts can be changed by dragging. Bimanual interaction is also possible to expand the interactivity, along with support for multi-user interaction.

5.1.2 Personal Workstations

Workstations are personal computers located in an office or at home. They could be tiled-monitor setups, in which they provide a **large space to collect and organize evidence** during visual analytics [25]. Personal workstations also come with input capabilities such as the keyboard and mouse—familiar to many users. In fact, many visualization tools support this setup. For instance, the Jigsaw tool [215] and the SRS system [146] provide holistic support for the sensemaking process on these devices. They are ideal for personal exploration of data or development of tools for visual exploration—e.g., creating a dashboard on Tableau [185] by dragging and dropping visualizations.

5.1.3 Portable Devices

In contrast, portable devices such as smartphones and smartwatches are personal—and smaller—devices only used by their owner. Consequently, they can act as a secondary displays, but can take on different roles. As a user-specific device, the secondary device can keep track of the user's interaction activities and corresponding data items (cf. VisPorter [151]). It can act as a **user-specific storage**—a container for points of interests or parameter settings—that can be easily accessed at any time. This role can further be extended by allowing the user to manage the stored content on the secondary device itself (e.g., combining, manipulating, or deleting content items). In this way, the secondary device itself acts a **primary device for personal exploration.** In the interest of managing the available display space while supporting multiple users, the secondary device enhances the interaction capabilities to support a wide range of exploration tasks. It can serve as a **mediator** (cf. Brudy et al. [159]), i.e., defining or altering system reactions when interacting with the large display. This mediation can happen in both an active and passive way: either the device is used to switch modes, or it offers additional functionality based on the interaction context and the user. Finally, to flexibly use the space in front of the large display, the secondary device can also take on the role of a **remote control** by allowing the user to interact from a distance.

Tablets and smartphones are quite capable for exploring individual lines of thought. Thus they can decouple the user from the large displays and let them come to a consensus later (cf. Branch-explore-merge [148]).

Smartwatches are special since they are very seamless in their capabilities. Beyond being lightweight and non-intrusive, their key advantage is that they are wearable. This not only frees the user's hands to interact with the other devices, they also provide anytime access without the need for persistent hand-held usage while leveraging proprioception for eyes-free, on-body interaction [95, 96].

5.2 Zones of Interaction in a Device Ecosystem

A device ecosystem is an environment. As any other work environment, there will be space to stand, walk-around, and even sit down. Imagining a device ecosystem is centered around a large display that acts as a shared display, interaction can happen in three zones: either **at the display** using say direct touch, in **close proximity** to the display but without touching it, or from intermediate and even **far distance** (Figure 5.1). This information can itself be used design interaction models with the shared display (as you will notice in Chapter 6).

As related work on physical navigation illustrates [113, 118, 127], working from an overview distance, close proximity, or directly at the large display is not an either-or decision. There is always an interplay between the three: analysts interact in front of the large display to focus on details, step back to orient themselves, and again move closer to continue exploration. Consequently, cross-device interaction should bridge these zones. For instance, an analyst may first work near the large display and perform interactions incorporating a smartphone or smartwatch. She then steps back to continue exploration from a more convenient position to analyze

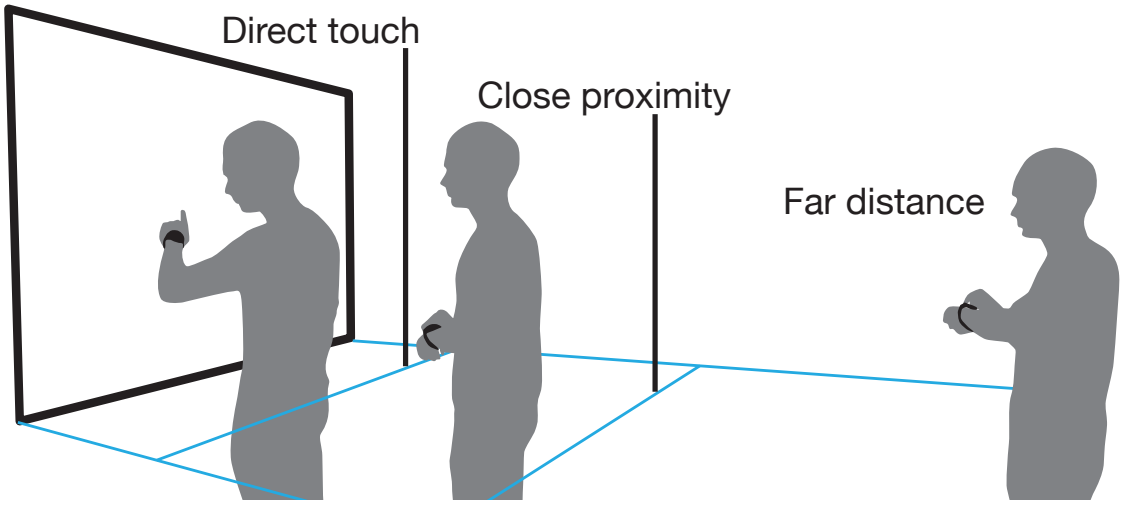


Figure 5.1: Interaction can happen with direct touch, in close proximity, or from intermediate or far distance.

other visualizations on the large display based on the stored data. The next chapters introduce interaction models targeting this flexibility within the device ecosystem.

5.3 Interaction Design in a Device Ecosystem

Physical navigation [113], spatial awareness [216], coupled and decoupled activities [148], and direct manipulation [48] facilitate collaboration and sensemaking in device ecosystems. This evidence makes a case for embodied interaction models. This is because, as defined by Paul Dourish, embodied interaction (EI) exploits our familiarity and facility with the everyday world, thus making it seamless to interact with other users and devices [121]. There are two distinct patterns of exploiting physicality and social familiarity of users in interaction design:

- **Implicit interaction:** Here the physical attributes of the devices and users such as their presence, position, and orientation in a physical space are used as

implicit triggers for interaction [100,217]. This style of interaction is defined in HCI as proxemic interaction; inspired by the study of *proxemics*—the spatial relationships between people and objects in a physical space—from Edward Hall [8]. This allows the system to react without interrupting the users.

- **Explicit interaction:** Explicit actions by the user through gestural interaction [93] such as touch, tap, and drag actions are used as input to the multi-device system [97, 99, 148, 218].

Supporting pervasive interaction can be beneficial for C2-VA [48]. There are many common ways in which we interact with the everyday world with our attention, movement, and communication. Several interaction techniques have been inspired from such physical and social activities. For example, Jakobsen et al. [88] utilized proxemics to interact with visualizations on wall-sized displays. These interaction models can be performed anywhere, and do not require much training. Furthermore, instead of indirect dialogs and control panels, supporting direct manipulation of visual representations promotes a flow in user activity (cf. fluid interaction [48]).

5.4 Software Primitives for Creating a Device Ecosystem

Many platforms utilize the idea of connecting the devices in the ecosystem with a client-server or a peer-to-peer architecture [45, 148]. Here I present an overview of two frameworks—Munin [20] and PolyChrome [19]—that we developed to provide primitives for connecting networked devices and users in C2-VA.

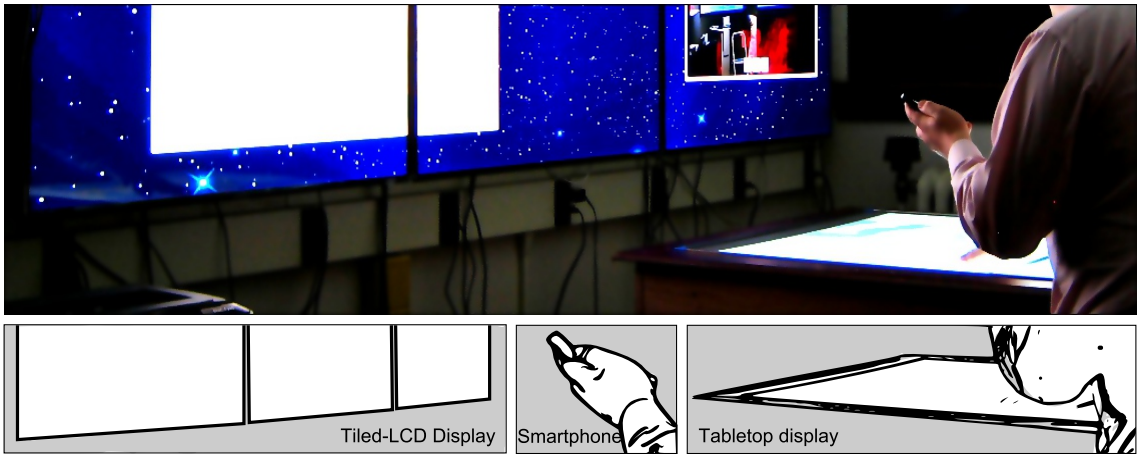


Figure 5.2: The Munin framework running on a display wall, a tabletop, and a smartphone.

5.4.1 The Munin Framework

The Munin framework explores network-centric abstractions for developing distributed user interfaces in the context of C2-VA.

5.4.1.1 Network Management

Munin is a software framework implemented in Java to create a peer-to-peer (P2P) connection between devices for replicating state across connected *peers* (application captured in Figure 5.2). It is a layered system with three primary layers: a Shared State layer for replicating data across peers, a Service layer for extending the framework's support for novel input and output, and a Visualization layer for managing a distributed scene graph and high-level constructs for visualization, including multiple views and visualizations.

A collection of Munin peers in a specific physical environment (either a mobile or static setting) is called a *Munin space* and is a pure peer-to-peer system with

no dedicated server. However, each Munin space has a global *space configuration* that maps out all peers, their capabilities, and their physical arrangements as well as the input and output surfaces they constitute. This configuration also maintains the peer-specific services necessary for a particular device to render, manage input, handle events, and perform computation. Since there is no central server for coordination, the space configuration is global and known by all peers so that each peer's own area of concern (in visual and data space) can be reliably determined.

5.4.1.2 Shared State Layer

The Shared State layer provides the basic network communication for replicating state and events.

Shared Associative State. Inspired by prior work on tuple spaces [219] and event heaps [220], Munin uses a shared *associative memory* that contains shared objects to which peers can subscribe. Subscribers will automatically be notified of changes to a shared object. This enables peers to create, modify, and update shared state that is replicated across all of the devices that constitute the Munin space.

Shared objects are the main form of communication and synchronization for services in other layers of the framework. The fact that our shared objects are dynamically typed is both a strength and a weakness. Dynamic typing makes for more flexible data modeling and means that the developer does not have to define interfaces for all data exchanges. However, the disadvantage is that all data exchanged between services must be manually checked.

Shared Event Channel. Munin incorporates a shared *event channel* where peers can distribute real-time events. Exposing the event system in shared state allows for easily decoupling input and output subsystems, which is common in ubiquitous computing environments—consider, for example, a tiled display wall consisting of multiple computers responsible for output, but only a single computer managing gestural input from a Vicon motion-capture system.

Persistence. Since Munin lacks a central server, the shared state and event channel is replicated on all peers. This means that the space itself, including all state, disappears when the the last peer in a Munin space is shut down. However, the fact that the shared state is fully replicated across all peers means that it is straightforward to persistently store a snapshot of the shared state. All shared objects are already required to support serialization, so this is simply a matter of iterating through all shared objects to serialize them.

5.4.1.3 Service Layer

The Service layer provides a mechanism for dynamically-loaded *services* that extend the Munin framework in several different ways. In fact, without core services from the Service layer, a Munin peer is simply an empty service platform with no functionality beyond the shared state and event channel.

Service Types. The Munin Service layer defines the following service types:

- `Display`: Display services are responsible for creating a graphics context on the peer so that visual output can be generated.

- `Renderer:` Renderer services transform an abstract node in the distributed scene graph into graphical output on the peer's display. Renderers are therefore heavily reliant on the display service.
- `Input:` Input services create the connection between input devices and the system. This involves transforming from the device's coordinate space into the global coordinate space.
- `App:` Application services tie together a collection of services using business logic for a particular task.
- `Simulation:` Whereas other services are event-driven, simulation services have an active thread of execution for online processes such as animation.
- `Computation:` A computation service is one that operates on shared data to produce new data. This can be used to control how expensive algorithms are distributed in the Munin space, e.g., to avoid overloading mobile devices.

5.4.1.4 Visualization Layer

Shared Data Table. The relational data model is a common data model for visualization [221], and many existing toolkits are based on this construct [222]. Munin defines a *shared relational data table* using the functionality of this layer.

Distributed Scene Graph. Each surface in the space configuration generates a scene graph that is distributed across the Munin space. Similar to most scene graphs, the Munin scene graph model is an instance of the Composite software design pattern, and is realized by Munin's shared objects.

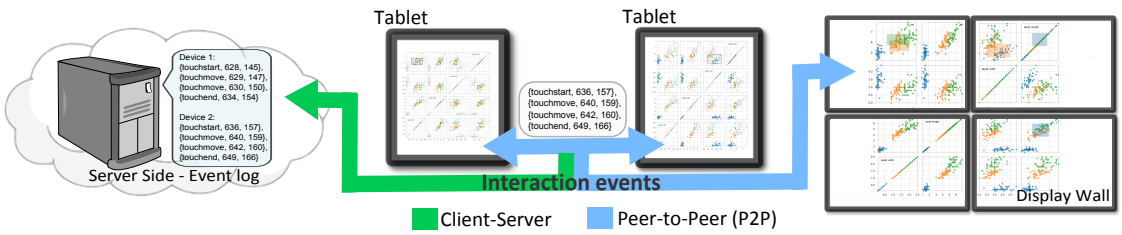


Figure 5.3: A web-based collaborative visualization of a scatterplot matrix of Anderson's *Iris* dataset with brush and link interaction enabled. Using PolyChrome, the brushes created on the tablets are represented on the display wall through operation (event) distribution, and also stored on a server.

Executing Assemblies. Finally, since Munin spaces generally consist of multiple networked devices, it can be painful to separately manage all of the services, configurations, and run scripts associated with each peer, one at a time. Instead, the Munin LaunchPad provides a peer-to-peer application frontend that runs on all connected peers and makes it simple to execute a Munin assembly.

These are the components of the Visualization layer in Munin.

5.4.2 The PolyChrome Framework

The PolyChrome framework extends the abstractions within Munin by adapting them to be visualization-centric. The motivation is simple: interaction has a meaning in a visualization space. A `mouseclick` is not just an event, but a selection of an item in a visualization. Similarly, there are certain configurations [223] in which a visualization scene graph is distributed across large and small displays.

In contrast to Munin, PolyChrome was built using the web technologies—HTML, JavaScript, and CSS—to keep up the current trends in the information

visualization field. Due to this strict reliance on standard web technology, it is entirely cross-platform and works on any device with a modern web browser.

5.4.2.1 Operation Definition

In PolyChrome, user operations can be defined to be interaction-level events or data-level selections such as filters and transformations applied by the user.

1. **Data-centric operation:** Here an operation can be defined on the data structures guiding the visualizations. Any interaction performed in this approach needs to be translated into a change in the data variables (operation) that is shared with other devices in the network.
2. **Interaction-centric operation:** Here an operation is a low-level interaction event handled by the device such as *mouseclick*, *mousemove*, and *mouseup*.

5.4.2.2 Operation Sharing

At a software level, user operations form the building blocks of any interaction with a visualization. For example, brush-and-link coordination is one of the common interaction techniques used in multiview visualizations, and it is performed through one or more selection operations. In C2-VA, these operations need to be captured and shared with the devices to propagate the effects of an interaction.

The sharing modules in PolyChrome provide the basic communication support to share and synchronize the user operations during visual analysis on different devices. PolyChrome achieves this by capturing browser events, communicating

these events in a serialized form through a shared peer-to-peer channel. PolyChrome allows two types of event capture mechanisms (Figure 5.4):

1. **Explicit Sharing:** An application developer using PolyChrome API can choose to handle event sharing explicitly. By verifying whether an event is generated PolyChrome or otherwise, the users can recycle native DOM events, which involves sharing the event with all devices in the P2P network. The recycled DOM event is then triggered by PolyChrome. The event is at the same time shared with the server by the PolyChrome client (on which the event was created). This is helpful in building private and public workspaces that can allow for branch and merge style collaboration [148].
2. **Implicit Sharing:** PolyChrome also allows the users to choose an implicit style in which all the events generated on a client are automatically captured at the document level irrespective of their targets. The events captured are analyzed to find their actual targets and are automatically shared with other clients without explicit application logic. This creates a fully-aware environment where each device knows the interaction happening on others. This method requires additional application logic for consistency management since interaction can happen at the same time on multiple devices. Typical usage scenarios include distributed user collaboration scenarios that convey a user interaction to all the other users.

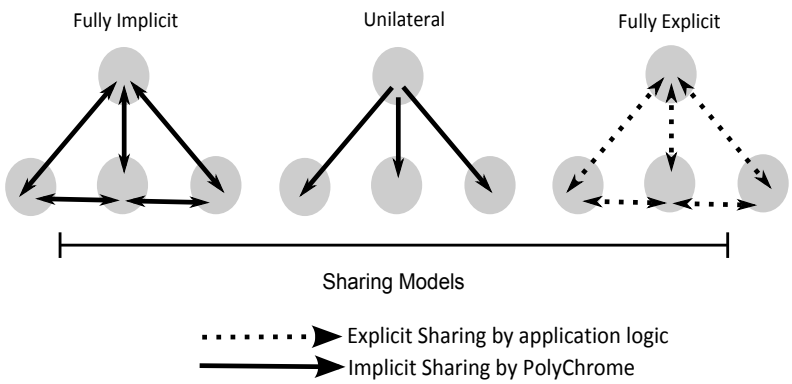


Figure 5.4: A hypothetical sharing scale ranging from fully implicit to fully explicit sharing. During implicit sharing, the operations are automatically shared with all connected devices. On the other end, explicit sharing allows the developer of the collaborative visualization to define explicit application logic for sharing interaction. Unilateral sharing model is a real-world example during presentations.

5.4.2.3 Managing Display Space: Distributing Visualizations

Device ecosystems consist of devices of different resolutions, aspect ratios, and screen sizes. This causes a distribution of the unified display space, i.e., the rendering of the visualization interface between multiple devices in the ecosystem. The distribution of the display space leads to different renderings on the devices that may cover whole or part of the global display space, or the unified display space covering all the devices (Figure 5.5).

1. **Stitching:** Here the entire display space is split between devices, with no two devices sharing any part of the display. The global display space is now formed by *juxtaposing* the individual displays.

   Usage scenarios: Multi-display visual analytics.

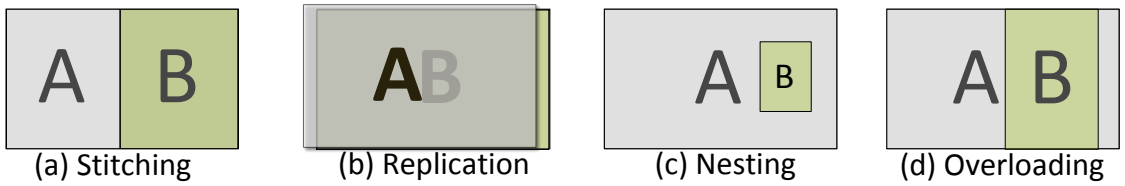


Figure 5.5: Display space configurations: (a) Splitting the display space between the two devices is useful for multi-screen displays, (b) replicating (mirroring) the display space among the two devices is useful during distributed collaboration, and (c, d) nesting and overloading are useful for small-screen mobile devices.

2. **Replicating:** This strategy shows the exact same view on all devices. The global display space of the ecosystem is, therefore, a *superimposition* of the individual screens.

   Usage scenarios: Public presentations where the view of the presenter can be mirrored on the displays held by the audience.

3. **Nesting:** Here one or more devices hold the entire display space, while others show bits and pieces of the display space relevant to the device or the user.

   Usage scenarios: Co-located settings with mobile devices and large displays.

PolyChrome achieves these configurations by maintaining a global space—i.e., the DOM rendering of the web application on a standard web browser—along with local space configurations for specific devices achieved through CSS transforms.

5.5 Designing Responsive Visual Interfaces for a Device Ecosystem

A visualization interface for a device ecosystem should adapts itself depending on the physical and computational capabilities of the device it is being rendered

on [22, 224]. This **responsiveness** is essential to support C2-VA and it can be enabled in many other ways, which is our motivation for discussing the following design space. Achieving responsiveness is traditionally viewed from a small display's perspective—how to create the interface primarily for the small screen (this is sometimes called "mobile first" in responsive web design) [225]. This is because an interface designed for a small display can easily (if not optimally) scale to a large display, but a large display interface when scaled to a small display faces issues with readability of the interface content. Therefore, we provide the design considerations for creating a visualization interface for a small display by transforming a classical multiple view interface (cf. CMV [226]).

5.5.1 Layout

A Coordinated Multiple View (CMV) layout in a visualization interface consists of an $m \times n$ grid of coordinated views. When working on a small display, the layout of the grid, including the width, height, and positions of the individual elements, can be modified to fit the display. The design choices in layout include:

1. **Stack:** Similar to responsive web design, the layout can be adapted by stacking the grid elements vertically or horizontally. However, this can lead to a loss of positional information and spatial relationships within the layout. For example, views that capture the same dimensions may be rearranged to no longer be in spatial proximity.
2. **Distort:** Techniques such as fisheyes [227] and hyperbolic distortion [228] are

popularly used to magnify areas of interest within a layout while compressing the rest. These techniques can be applied to a CMV layout to focus on specific views of interest while compressing the rest to save space.

3. **Proxy:** When there are specific views of interest (focus region) in the UI, the rest of the views can be replaced with a proxy widget(s) that can range from markers, icons, or even dynamic insets summarizing the visualizations [229].

5.5.2 Visual Encoding

The visual representation may also need to be adapted to the available display space. Based on the existing literature, we identify three choices:

1. **Fill:** Space-filling techniques can take full advantage of the available space to restructure the visualization. These techniques allow the graphical primitives to cover the entire space (e.g., treemaps [230]). Similarly, pixel matrices [231] can capture the trends in line charts and bar charts by capturing information at each pixel in the visualization view.
2. **Layer/Fold:** For charts with graphical primitives spanning either the X or Y dimension (e.g., paths), layering (or folding a dimension) saves space by splitting the dimension into parts which are overlaid and distinguished using other visual variables such as color or opacity. For example, horizon charts perform layering to significantly save space over traditional line charts.
3. **Merge:** Alternatively, merging visualizations within the interface into composite representations can save save space. As Javed and Elmqvist [223] dis-

cuss, there are four design choices for composite visualization: juxtaposition, superimposition, overloading, and nesting.

5.5.3 Data Content

Finally, the data content embedded within the visualizations can be changed to make it better perceivable on a small display. The design choice here corresponds to avoiding sharp and unreadable features of a visualization on a small display by intelligently adapting these features at a data level by:

1. **Aggregate or Sample:** These techniques are often used for managing the amount of information rendered through grouping [232], discretization, and sampling the data. They can be repurposed in responsive visualization to manage the content based on the physical display size.
2. **Identify Points of Interest:** Another approach for managing content is use only perceptually important points. These points capture important features in a visual representation in terms of visual perception [233].

5.5.4 Interaction

Interaction mechanisms in a visualization are connected to the input modality (e.g., mouse or touch) and the elements within the visual representation. As identified in existing literature [19, 224], differences in input modalities can be bridged by abstracting input events to work across device platforms (e.g., click becomes tap on touch displays). However, such adaptation may not be enough as the input is also

affected by the display size. For example, its hard to precisely touch a point on a smartphone due to the so-called "fat finger" problem. Furthermore, when the visual representation changes, adapting the interaction model becomes more complex and dependent on the representation itself. Adapting the interaction to a device is therefore inherently connected to the display size.

5.5.5 Sensemaking Task

For visual analysis, we can also specialize our notion of responsiveness further based on standard visualization tasks [108], such as creating an overview, looking into details, and linking patterns across data. This can maintain the flow and engagement in visual analysis by ensuring that the responsive representations of the visualization can reflect the task that the user wants to perform with the device. For example, using a smartphone as the medium to see an overview of the data visualizations present on a large display.

5.5.6 Design Guidelines

Given the above design dimensions, a visual interface should take advantage of one or more of these adaptations to achieve responsiveness. To explore the design space, we conducted a user study to evaluate two responsive interfaces to understand their affordances in visual sensemaking. This user study and results are accessible in Appendix A. Based on the results and our responsive designs, we determined the following guidelines for responsive visualization interfaces.

- **Target responsive encodings and combine them with other transformations.** Based on our user study (Appendix A), the responsive visual encodings should be combined with transformations in other design dimensions to achieve further responsiveness, since there may be a limit to how much a visual encoding can be compressed.
- **Highlight visual objects for tracking changes.** Users often placed their fingers on objects they wanted to track on the interface on both small and large displays. Explicit highlights can support this practice.
- **Provide change indicators on large display.** Users found it difficult to track changes during interaction on the large display due to its size. Visualizing change, such as through time-lapse representations, can help improve user performance for large displays and achieve better responsiveness.

5.6 Summary: Contributions and Next Steps

This chapter provides the fundamentals of how different devices in an ecosystem can be used together. The understanding of the device roles and interaction styles helped us target suitable tasks and complement the devices. It is used in Chapter 6 to design implicit and explicit interaction models based on body movement in the physical space of a large display. It also helped couple large displays with portable devices in Chapter 7 to support a visual analysis process in the device ecosystem. Notably, a portable device acting as storage, mediator, and remote control is found to be powerful for visual analytics in Chapter 7. Furthermore, the

design space for adapting visualization interfaces to the heterogeneous devices helped us investigate new techniques for responsive visualization in a device ecosystem (as seen in Appendix A).

The software primitives introduced in our early frameworks answered how the devices can be connected into an ecosystem. For instance, the PolyChrome framework defined the "user operation" as a shareable object representing an action on the data to synchronize multiple devices. However, being frameworks, both Munin and PolyChrome contain the essential classes and methods to connect devices. They only provide a skeleton for a C2-VA application, which still needs flesh and blood to analyze a dataset. To create a C2-VA application, they should be coupled with other frameworks to process data, build the visualizations (e.g., D3 [15]), and generate insights.

Having said this, putting together the skeleton, flesh, and blood into a living and breathing C2-VA application is an alchemical task. It is because the composition needs to work effectively (1) for multiple users and their expertise, (2) for the heterogeneous devices such large wall displays, tablets, smartphones, and smartwatches in the ecosystem, and (3) to support the target analytical activities in the application domain. We composed the primitives to develop solutions for specific application scenarios introduced in Chapters 7 and 8. To go beyond these monolithic solutions to generate general applications, we need more than a framework. Hence, we introduced a software platform in Chapter 9 for development of new C2-VA applications, with a component model to compose the fundamental components for data management, visualization, interaction, distribution, and collaboration.

Chapter 6: Designing Multi-User Interaction for Large Displays

How to interact in large display environments.

Figure 6.1: Two different types of operations performed using Proxemic Lens technique. Selecting a region of interest using a mid-air gesture (left), vs. merging two time-series plots when two users approach one another (right). A key feature of the lens is that it can be controlled by both implicit (proxemics) and explicit actions (gestures).

Large displays offer physical space for multiple users to gather and perform visual analysis together. There can be two types of interactions for large displays,

- **Explicit interaction:** User action where the purpose **is** primarily to interact with a computer system.
- **Implicit interaction:** User action where the purpose is **not** primarily to interact with a computer.

Explicit interaction is the traditional mode of interacting with computers, and includes actions such as mouse clicks and drags, keyboard presses, and touchscreen taps and swipes. Implicit interaction is a more novel approach, and it targets automatically using the body states and movements that are known or observed to exist when users interact in a physical space [234] to avoid learning explicit actions and reduce additional physical activity. Therefore, implicit interaction can be tightly coupled with utilizing proxemics attributes [8] of the users such as their position, posture, movement, orientation, and identity of users within a physical space to control a computer system [90, 127, 235]—also referred to as *proxemic interaction* [41]. The design space for proxemic interaction with visualizations—mapping proxemics dimensions to high-level tasks [236]—has been presented by Jakobsen et al. [88]. However, these designs do not fully extend to parallel individual and collaborative work due to inherent presentation conflicts. It is worth noting that implicit actions may go further than proxemics (e.g., using facial expressions).

6.1 Interaction Design

We propose a combined presentation and interaction technique for multi-user visual exploration in large display environments (see Figure 6.1). Drawing on design guidelines by Tang et al. [237] and Kister et al. [127], the **presentation** technique uses focus+context lenses owned by each collaborator. The lenses act as views for the users during parallel individual or loosely coupled work [127, 188], and can also be combined with other lenses for tightly coupled collaborative analytics. We refer to

Kister et al.'s design space exploration [127] for lens design including placement, size, shape, and rendering. We focus on the **interaction** techniques—both implicit and explicit design alternatives (Figure 6.2)—for exploring data visualizations through the lenses. Here, we list the abstract lens operations and elaborate on possible options for their implicit and explicit designs.

Initiate: Lenses visualize specific parts of a dataset, and the lens initiate operation involves selection of a region/part of an overview visualization. Selection operations in large display and mixed-modal environments are typically done through pointing and touch with hand [89] and other devices [238] in an explicit way. However, in contrast to selection of discrete objects (for instance, photos), selections in a visualization are more precise and granular. For selection, we define a gaze-controlled cursor highlighting the region-of-interest (cf. Peck et al. [92]).

- *Implicit Initiate:* When the user's gaze dwells on a region in the overview visualization, a lens is created at that location.
- *Explicit Initiate:* User creates a lens by navigating the cursor using hand and confirms the selection with a hand roll.

Scale: The ability to modify the size of a lens is needed in a large display environment since the distance from the screen affects the user's view. View scaling was previously performed by tracking the user's position and distance from the screen [92, 239], and by direct manipulation, popularly seen in movies such as Iron Man (2008) and Minority Report (2002).

- *Implicit Scale:* The size of the lens is directly mapped to the distance of the user from the display.

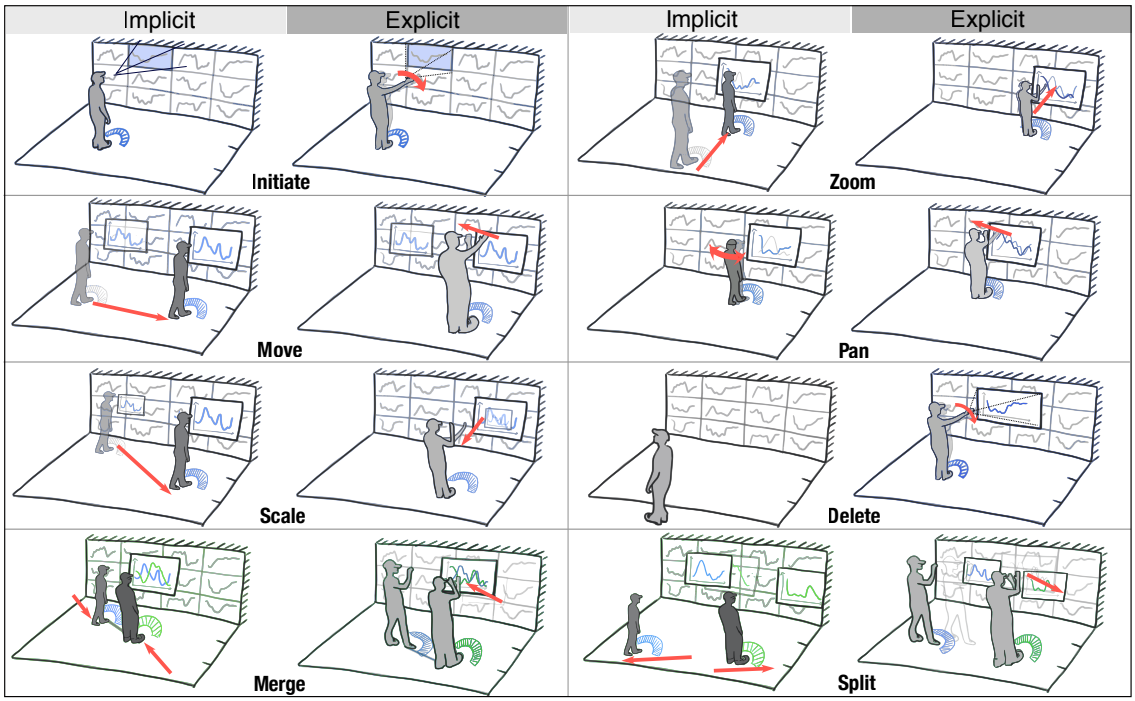


Figure 6.2: Visual summary of using proxemics and gestures to interact with focus+context lenses on a large multi-user display space. The direction of arrows in the illustration represent the direction of movement of head, body, or hand to trigger the corresponding actions.

- *Explicit Scale:* The lens size is changed by hand movement pulling/pushing away the lens.

Move: Positioning the lens helps organize the workspace. Move operations are done on large displays and across devices through explicit gestures such as flicking, drag-and-drop, grab-and-move [89, 240, 241].

- *Implicit Move:* The user's gaze or spatial movement controls the lens.
- *Explicit Move:* The user's hand directly moves the lens.

Zoom and Filter: Zooming and filtering the data within a view is a very common task in visualization [108]. These operations have been previously done

through implicit actions such as leaning [239] and distance-based semantic zoom [88], and explicit actions such as pinch-to-zoom, linear, and circular gestures [93].

- *Implicit Zoom:* The user's distance and orientation is used to zoom and filter the lens content.
- *Explicit Zoom:* The lens content is zoomed using a mid-air hand zoom gesture—linear hand push/pull [93]—while standing still in front of the wall display.

Pan: This operation shifts the content of a lens (i.e., not its position) based on the data attributes. For example, this can be based on time for time-series data or based on spatial location for spatial data. Pan operations in large display spaces have been explored through movement of handheld devices in a two-dimensional space [242] and through mid-air gestures [93], which are both explicit.

- *Implicit Pan:* The lens is panned based on the orientation of the user's head when dwelling on it.
- *Explicit Pan:* Hand movement (left and right)—hand swipe—is mapped to the pan operation on the lens content [93], when the user is standing still in front of the wall display.

Merge: Merging content is helpful for collaborative analysis, cross validation, and trend analysis across lenses. Merge also corresponds to the *relate* visualization task connecting different sets of visualizations [108]. Various styles of composite visualizations have been illustrated by Javed et al. [243], and these form the design choices for the merge operation. Related work on shape-shifting wall displays [244] explores controls for managing the arrangement of individual displays. They merge the individual displays implicitly by observing when users move close to them.

- *Implicit Merge:* Lenses are merged when two users are close and aligned towards each other [244].
- *Explicit Merge:* Lenses are merged when the users position them close to each other with their hands.

Split: Lens splitting can be mapped to a similar implicit action or explicit gesture as merge. Takashima et al. [244] supported this by identifying when users walk away or through a split gesture.

- *Implicit Split:* Lenses are split when the users move away while facing away from each other.
- *Explicit Split:* Lenses split when users separate them by pointing and dragging.

Delete: Being the inverse operation of initiation, it can be triggered based on similar metrics as initiation. Tracking attention is a popular way of understanding if a view is of interest [240].

- *Implicit Delete:* Lenses are deleted when the user moves out of her workspace while facing away.
- *Explicit Delete:* Lenses are deleted using a hand roll gesture.

6.2 Formative Evaluation: Implicit vs. Explicit

To gain a better understanding of the usefulness of both implicit and explicit interaction styles, we conducted a formative user study on the physical affordances of our design ideas. The user study focused on collaborative visual analysis of multivariate time-series data, and required users to interact with the data through

lenses to figure out patterns within and across variables. The user interaction was not controlled or constrained, thus allowing the participants to freely interact and explore various features provided by the lens technique. Our focus through this qualitative inquiry is not only to observe which interactions style suits each lens operation but also to gain interesting and unexpected design opportunities that can expand the conceptual model of our lens. Furthermore, through this evaluation we intend to combine ideal proxemics and gestures into a hybrid lens interaction model that can be applied to any visualization on large displays.

6.2.1 Dataset

The dataset was sensor data measured by different types of sensors (e.g., moisture, temperature, and humidity) in a building over time. It contained 13,500 records for 30 sensors spanning over two weeks.

6.2.2 Participants

We recruited 12 participants (5 female, 7 male) from our university's student population (6 groups of two). Participants signed up voluntarily and were rewarded $10 upon successful completion. All participants had experience with data analysis using visualization and more than 6 years of experience using computer systems. Only 7 participants had experience with mid-air gestural interactions.

6.2.3 Experimental Design and Procedure

The experiment followed a 2 × 6 within-participant design with Interaction Mode I (implicit, explicit) and Analytics Tasks A, resulting in 72 total trials (i.e., 12 per team). The participants were scheduled for one-hour sessions in groups of two. They were first introduced to the equipment used in the study, followed by a brief introduction of the study goal and tasks they were expected to perform. After signing their consent, the participants were asked to wear the props on their head, dominant hand and their feet. The investigator then described a list of gestures for the explicit mode and the proxemic attributes tracked by the system in the implicit mode. The participants were allowed to practice by testing each operation in the technique until they were comfortable. The participants were then quizzed to test their knowledge of the lens operations.

6.2.4 Tasks

Each team was given six tasks; three were low-level comprehension tasks involving finding specific values, trends, and extrema in the visualization, whereas three were high-level synthesis tasks involving identifying anomalies, comparing data, and correlating data. These categories were inspired by the work of Shneiderman [108] that summarizes the types of tasks performed on different data types. Two variants of these six tasks were prepared, giving two task lists (TS1 and TS2), one for each interaction mode. The tasks within each set were randomly shuffled for each group to counter sequence effects. Each task required multiple implicit or ex-

plicit actions that were mapped to corresponding lens operations. The presentation order of interaction modes was fully counterbalanced.

6.2.5 Data Collection

All participants were encouraged to "think aloud" and announce their decisions while interacting with the system to perform the tasks. During the session, the investigator took notes about important observed events and verbal comments made by the participants. The sessions were also video recorded (with consent).

6.2.6 Results

Inspired by grounded theory [245], we analyzed the participant data by open-coding the notes and transcribed interviews. Two researchers, who had observed and conducted sessions, developed two initial code-sets independently, and after reaching an agreement on a final code-set, one researcher proceeded to code the remaining session data.

6.2.6.1 Implicit actions

Lens initiation was not implicit enough. While participants liked the idea of a smart environment that could guess when a lens needs to be initiated, they unanimously agreed that head dwell was not much efficient, as it led to false negatives (low discoverability) and false positives. Two participants said, "*Being interested in one chart does not mean I will only look at that one chart*" (G4), and

in fact the chart of interest is usually decided by shifting focus between different charts in the overview. When participants knew the chart of interest, they had to consciously focus their line of sight to that chart region to create a lens and this was contrary to the implicit nature of the design. Five participants mentioned that the dwell time taken into consideration by the system maybe "too long". On the other hand, there were also multiple observations of unwanted lens initiations. This was frustrating to the participants, especially when they already have a lens of interest that they are analyzing. Two participants said that they have no clue as to why the extra lens was initiated (G2 and G5). We conclude that deciding on an accurate dwell time is a non-trivial task, and gaze dwell may not be an appropriate action for the lens initiation task.

Lens move and scale were liked. Perhaps the most interesting observation about these interactions was the description given by one participant as "unnoticed interaction" (G3). This participant (G3) further said, "*the interaction was so natural and intuitive that I almost did not notice!*". Intuitiveness was a common reason given by all participants who expressed positive opinions about these interactions. These results confirm the findings of Jakobsen et al. [88], and align with our motivation behind implicit actions.

Lens zoom and delete were perceived as "fun to do". Both zoom and delete required moving body across different proxemics regions. Three participants mentioned that they like the need for being active (G1, G2, G5) referring to the physical navigation that triggers the zoom. One participant described the zoom interaction as being "fun to do" (G2). Contrary to the observations of Jakobsen

et al. [88] about similar zoom interactions, we did not observe awkward, slow, or uncertain movements from our participants. We can argue that this could be due to (1) mapping the movement speed directly to the zoom rate, and (2) the visual cues on the floor in the form of lines.

Lens zoom and pan were not accurate. In many cases, participants were observed to undo (or redo) zoom and pan actions because they had zoomed or panned the lens content too much or too little on their first attempt. Several comments were made during the interview about the need for more "control" during zoom and pan, and using "gestures" (G6) was a common solution suggested by the participants. One participant suggested "*having a way of turning this feature on and off*" (G3). Implicit actions for these operations lacked precision and accuracy on the first attempt. When further attempts were made, the implicit nature is questioned.

Lens merge and split had mixed feedback. Lens merge was found to be the most interesting feature in implicit interaction mode. 75% of the participants described this feature as either "cool" (G3), "useful" (G2, G4), or "interesting" (G6). One participant remarked that "*this is similar to real world, where people have to get close to each other to share physical copies of documents*" (G6). Implicit lens split, on the other hand, received some conflicting results. While some participants still liked the feature, others expressed the desire for "keeping lenses merged" (G2, G3) while being able to "move around" (G2) referring to the need for both merged and separate work spaces during collaborative data analysis.

6.2.6.2 Explicit Interactions

Lens content manipulation gave users more control. A common theme observed across all explicit actions was that the participants liked having the direct control provided by these actions. One participant remarked, "*it is great to zoom in and out as much as you like*" (G6). Generally, participants learned the gestures related to lens content manipulation quickly. The participants who needed relatively more time to learn these gestures were observed to use the gestures more efficiently. In particular, it is worth highlighting a remark made by one participant about the initiate gesture. where the gesture referred to as "natural" (G1) and "like drilling into the data" (G1). We also observed minimal false negatives for these gestures. In these cases, the participants were able to identify the issue through the visual feedback (rather the lack of visual confirmation) for each gesture in the form of textual labels. They successfully corrected the gesture to achieve the desired results.

Lens move and scale gestures were demanding. Explicit lens move and scale did not receive good feedback from participants. Through observation, we noticed that participants usually did not put their hands down after completing a move or scale action. When asked, one participant mentioned that it was because they thought "*the lens would go back to where it was, if they let go of it*" (G4). Seven participants pointed out their hands were tired and experienced exhaustion when moving the lens. We rarely saw participants use lens scale. Three participants mentioned that this gesture "*required too much work for little return*" (G6).

6.2.6.3 Subjective Ratings

We also collected participant opinions about each action in both implicit and explicit modes through a post-session questionnaire. It included seven statements focusing on a different metric for each lens operation: Preference, Accuracy, Intuitiveness, Efficiency, Enjoyment, Collaboration, and Physical effort. Participants rated statements on a Likert-scale ranging from 1 (e.g., Not Preferred) to 5 (e.g., Preferred). For Physical Effort, the ratings ranged from 1 (Very hard to perform) to 5 (Very easy to perform). We performed Wilcoxon Signed Ranks tests at a significant level of $\alpha = .05$ to test for differences between 2 samples of participant ratings (implicit and explicit) for each statement and lens operation. For example, this meant comparing implicit and explicit lens zoom in terms of physical effort.

There was a significant difference between implicit and explicit lens initiation when it comes to perceived physical effort ($Z = -2.821, p = .005$). The majority of the metrics ranked significantly higher in the explicit mode for lens initiation.

Lens scale was ranked significantly higher in the implicit mode for both collaboration ($Z = -2.239, p = .025$) and physical effort ($Z = -2.209, p = .027$). The implicit lens move action was enjoyed more than explicit ($Z = -2.140, p = .032$).

Lens pan ranked significantly more intuitive ($Z = -2.360, p = .018$) when performed using explicit gestures. We did not, however, observe any other significant effects for zooming and panning.

Preference ($Z = -2.230, p = .026$) and accuracy ($Z = -2.877, p = .004$) ranked significantly higher for lens implicit merge operation. Lens split, on the

Figure 6.3: A hybrid merge that uses proxemics to stack lenses when the users are close (left) and changes the merge mode when they perform a collaborative gesture (right).

other hand, required significantly less physical effort ($Z = -2.539, p = .011$) in the implicit mode, however, other ratings were in favor of the explicit mode. In fact, the participants preferred the explicit lens split operation significantly more ($Z = -2.235, p = .025$).

6.3 The Proxemic Lens Technique

Based on the results, we designed a hybrid interaction technique combining both implicit and explicit interactions.In the hybrid version, we included "lens store," a new operation to allow storage of lenses on the floor display through a foot click gesture. User feedback indicated that seeing the area behind a lens is necessary, since users often go back and forth between the lens and the overview.

The hybrid version of our lens technique uses a mix of both implicit and explicit actions for lens initiate, scale, move, pan, zoom, merge, split, delete, and

store. Table 6.1 summarizes the proxemic and gestural actions that were used. This does not mean that for every action designed under the hybrid technique, we have to necessarily include a mix of both proxemic and gestural interactions. However, when designing a set of such lens actions, we can now choose from a mixed pool of interactions based on both proxemics and gestures. For instance, we kept the lens move action to be performed by the implicit interaction based on user's position relative to the wall display. One criticism from the previous study was that the implicit lens move action led to unwanted lens movements especially in collaborative scenarios when two users converse or when the users are very casual with their interactions, which lead to small movements of their heads. To avoid this, we introduced a region mapping for lens move, in which users can fixate the lens to avoid lens movement when they are close to the display. The lens merge and lens split operations in the hybrid version included both implicit and explicit aspects through a two-step process. During lens merge, the lenses would stack when users are close (resembling implicit merge), and switch to a different merge mode (e.g., content overlay) through a collaborative "hand raise" gesture (Figure 6.3) that captures group intent.

6.4 Summative Evaluation: Proxemic Lens Technique

To evaluate our hybrid lens designs, we conducted a separate summative study using the same tasks as the formative evaluation and following the same methodology when collecting and analyzing the results. The goal of this evaluation was to

Table 6.1: Overview of the Proxemic Lens interactions in hybrid mode.

Context	**Action**	**Interaction Details**
Lens	Initiate	Hand orientation, Hand-roll gesture
	Delete	Hand orientation, Hand-roll gesture
	Scale	Body distance to the wall display
	Move	Body position and distance to display
	Store	Body orientation, Foot click gesture
Lens Content	Pan	Body orientation + hand swipe gesture
	Zoom	Body position/orientation + hand gesture
Multi-User	Merge	User distance + body orient. + simultaneous gesture
	Split	User distance + body orient. + simultaneous gesture

learn about basic aspects of the usability of hybrid actions—would users find them intuitive or confusing? Physically easy or difficult to perform? Accurate or not?

6.4.1 Method

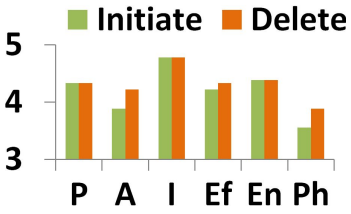
Similar to the formative study, each session lasted just under an hour and involved a group of two participants performing the tasks collaboratively. For the purpose of this study, we recruited four participants for two pilot sessions. After making minor adjustments based on the results of the pilot sessions, 18 additional participants (9 groups; G1-G9)—4 female and 14 males—were recruited for the actual trials. The participants were drawn from the student population at our uni-

versity. There were five parts to the study (identical to the formative evaluation): (1) training, (2) action-performance quiz, (3) six-task data analysis, (4) completing a 5-point Likert scale questionnaire, and finishing with a (5) semi-structured interview. In what follows, we report on the results of the hybrid action study and discuss interesting insights from our observations and user feedback. We did not include the "collaboration" item in the Likert scale questionnaire as we found that the participants had different perceptions of collaboration quality during the formative evaluation (since all the tasks were successfully completed). Instead, we chose to observe their collaboration style during the tasks, and inquire their individual opinions about their collaboration during the interview after the session. This left us with 6 dimensions (Preference (P), Accuracy (A), Intuitiveness (I), Efficiency (Ef), Enjoyment (En), and Physical Effort (Ph)).

6.4.2 Results and Discussion

Overall, participants found the hybrid actions easy to learn. Most subjects (16 out of 18) were able to recreate the actions during the quiz part at first try. However, during the actual experiment we observed that four participants could not recall the actions for pan and zoom. In one group (G6), the teammate helped the participant to remember the interaction, and in all four cases, participants were fully able to perform all actions correctly. Two participants (G6 and G9) noted that actions take some practice to master, but once learned were easy to recall and use.

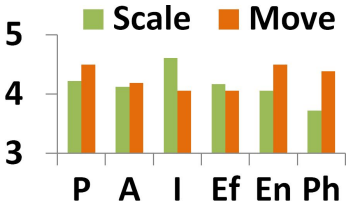
6.4.2.1 Lens Initiate and Delete



The lens initiate and delete actions both received high scores for all categories ($\mu > 4.2$), with the exception of physical effort ($\mu = 3.5$ and $\mu = 3.8$, respectively). Participants felt that these operations were hard to perform. This might be due to the fact that the interaction required pointing to the specific chart while keeping the cursor inside the borders of the chart and performing the hand roll gesture. This was slightly easier when performing deletion since usually users deleted their lenses when far from the display at which their lenses were at the maximum size and easier to point at.

There are a number of available solutions proposed in the visualization literature to tackle this issue. For example, we can use the fisheye technique to enlarge the chart the user is pointing at, making it easier to not cross boundaries. Participants found these actions very intuitive ($\mu = 4.7$); G1 described the action "like opening and closing doors."

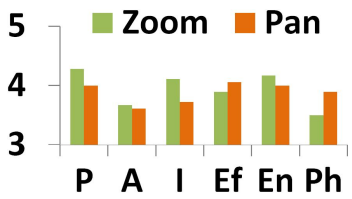
6.4.2.2 Lens Scale and Move



The lens scale and move actions, which mostly included proxemic interactions, received high ratings across all categories in the Likert scale ($\mu \geq 4$). Participants particularly enjoyed the implicit movement of the chart ($\mu = 4.5$), and stated it as "natural to want your lens close by at all times" (G8).

One participant brought up a point about lens movement, saying that "it let him focus on the chart more" and called it "efficient" (G1). Subjects also found the scaling of the lenses based on distance to the display to be intuitive ($\mu = 4.6$) and something one "almost does not notice" (G2). One criticism we received from the previous study in the implicit mode was that since the movement of the lens follows the head movements, in collaborative scenarios when two people are viewing the same lens, natural movements of the head happening during conversation would move the lens and it would be distracting for the collaborator. To avoid this issue, in our hybrid mode, we introduced the close region of the floor display in which the lenses are automatically fixated. In eight groups, we observed subjects taking advantage of this feature when discussing and exploring a chart with their partner.

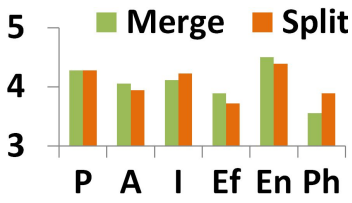
6.4.2.3 Lens Zoom and Pan



The lens zoom and pan actions received relatively low ratings for accuracy ($\mu = 3.6$ for both), efficiency ($\mu = 3.8$ and $\mu = 4$), and physical effort ($\mu = 3.5$ and $\mu = 3.8$). Both of these actions required explicit hand gestures, and we observed participants get confused and misuse the interactions at the beginning of the trial session even though they had successfully gone through the exercise and quiz phases. Similar to any other gestural and direct interactions, we can argue that the learning curves could be steep especially if the user has no prior experience with mid-air interactions and the particular interaction type.

Three participants mentioned that they did not find the pan interactions intuitive and expected them to be "backwards" (for instance, hand movement to the left leading to a pan right). One participant compared this to how users switching from Mac to Windows or vice versa would have a hard time adjusting to how the mouse pad scrolling works differently in the other system. Prior experiences and work habits do affect the learnability of new concepts, and gestural interactions are of no exception. One solution to improve the experience would be to introduce implicit components into these actions since they have a direct mapping. We tried to accomplish this by showing extra information on the charts automatically as annotations as the user enters the middle region (similar to Jakobsen et al. [88]), thus eliminating the need for some of the explicit content zooming. Participants' feedback confirms the improvements targeted by this design.

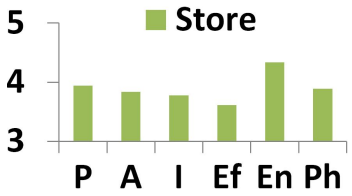
6.4.2.4 Lens Merge and Split



Lens merge and split both got mixed ratings and feedback from the participants. While participants found these actions physically demanding ($\mu = 3.5$ and $\mu = 3.8$) and rated both of this actions somewhat low for efficiency ($\mu = 3.8$ and $\mu = 3.7$), their ratings for the enjoyability dimension were high ($\mu = 4.5$ and $\mu = 4.3$) deeming both of these actions as fun to perform. Seven groups attributed this to the collaborative gesture, which was used for switching the chart type after an implicit merge, and in some cases even suggested alternative

interactions for this purpose: "high five" (G3) and "hand shake" (G6). While we can conclude that the new hybrid design of these actions in which the merge and split are a two-step process is liked by the participants, similar studies have shown that collaborative actions involving touch might not always be welcomed in cases where the two collaborators do not know each other [246]. Participants also commented in several cases on how these interactions promote involvement and collaboration: "*merge was fun because it required us to work together*" (G7). The implicit first step interaction sometimes happened accidentally (3 cases) but welcomed and further pursued to the second step of explicit superimposition. In two groups (G1 and G8), we observed that team members started collaborating and helping out their partners only after the first instance of the merge action.

6.4.2.5 Lens Store



Participants were very excited to try out the foot click gesture interaction method (tapping the floor with one's foot to temporarily store the lens). Two participants called this interaction "useful" (G3), while another (G1) mentioned he enjoyed using his foot to interact. In two cases (G6, G8), we observed that participants tried reading the lenses while on the floor, but soon brought back the lens to the wall display for better visibility. Several participants mentioned that it was "*easy to bring the lenses to the wall and back to the floor, so there was no need for using the lens on the floor*". One participant mentioned that "*it might*

be awkward to use foot for interaction in a formal meeting setup" (G1). One interesting usage of this interaction was made by a participant (G7) in order to quickly move away his lens and let his team member create a lens in the close proximity. We expect that better projection technologies in the near future will allow wider use of this unexplored surface as an additional interactive surface in the device ecosystem.

6.5 Summary: Contributions and Next Steps

Large displays appear in many of the analytical settings described in Chapter 2. Traditional input devices such as touch, mouse, and keyboard are often used with them [123, 247], but they are not ideal for the size of the large display. Inspired by the use of proxemics in HCI [41, 79], we designed multi-user interaction for visual exploration of time-series data using proxemics and gestures. We targeted simple exploration tasks to navigate time-series variables to identify values, trends, and correlations. Through a laboratory study, we identified the affordances of proxemics and gestures. Implicit interaction through proxemics is suited for seamless operations to enhance users' view of the data by say scaling, moving, or merging visualizations automatically. It takes advantage of the physical space on the display and in front of the display—using lenses to create personal territories and merging these lenses to perform group work. Explicit interaction through hand gestures is more suited for precise creation, configuration, and navigation of the visualizations.

Building on these affordances, the combination of proxemics and gestures—seen in our Proxemic Lens technique—unleashes the real power of large displays for

co-located collaboration. While Proxemic Lens was natural and useful for the simple tasks considered, it can fall short in supporting an entire visual analysis process. It is because there is a limitation to what can be expressed (or rather understood) from proxemics and how many gestures can be learned by the user. Furthermore, physical effort tends to be higher with these interaction styles. The qualitative feedback provided by the participants of our user studies reflects this nature.

To utilize these natural interactions while balancing their tradeoffs, we need more flexibility in terms of devices in the analytical process. We need more than what large displays and body-based interaction can afford for visual analysis. This flexibility can be achieved by introducing more modalities for input and output into the analytical environment. There are many choices. For instance, speech interaction can reduce some of the physical effort involved in body movement. In my research, I considered combination of devices—large and small devices—to support flexible analytical work across large displays and portable devices. Chapter 7 targets device combination and introduces interaction techniques elicited from a user study as well as a conceptual framework to support an analytical process across devices.

Chapter 7: Combining Large Displays and Portable Devices for Visual Analysis of Data

How to use tablets, phones, and smartwatches along with a large display.

Figure 7.1: A device ecosystem containing a large display, a personal laptop, and two smartwatches used together for visual analysis. This picture is from our large display + smartwatches project [23] introduced later in this chapter.

Following the shared exploration process enabled by Proxemic Lens technique, we wanted to explore how multiple devices can be connected together for visual

exploration (Figure 7.1). We target a scenario that extends the use of the large, shared display to integrating a secondary device for personal visual exploration of data. This situation is very common: Branch-explore-merge [148], GraSp [154] and VisPorter [151] consider similar cross-device workflows. However, it is firstly not clear how to design the interactions to connect multiple devices, seamlessly transfer information across them, and utilize them together or individually for visual analytics. Therefore, we sort out to first conduct a formative study to elicit cross-device interaction styles in device ecosystems.

7.1 Formative Evaluation: Eliciting Interactions in an Ecosystem

We conducted a formative evaluation to elicit interactions that enable the use of multiple devices in a device ecosystem for different tasks in visual analysis. This study was conducted with a protocol similar to Wobbrock et al., [248] where we explained to participants the expected outcome of an interaction (*effect*), and asked them to perform a physical action (*signal*) they thought appropriate for the effect. We focused on a specific device coupling between a fixed wall-mounted large display and a portable handheld device, but we believe that these observations can be extended to other combinations.

7.1.1 Participants

We recruited 9 unpaid participants (2 female, 7 male) from the student population in our university. Participants were between 23 to 32 years of age. All

participants had experience working with visualizations including creating charts for reporting and two participants developed visualization tools. All participants are avid users of touch devices (six of them also used large displays in the past). Participants were all right-handed.

We motivate the choice of using university students as a representative population as the focus is to extract cross-device interactions that make sense in particular sensemaking contexts (which were explained) and therefore no specific expertise except the experience of using handheld or portable devices was needed.

7.1.2 Apparatus

Device ecosystems can contain devices of different input and output modalities. In sensemaking, the type of visualization interface can also play a major role in guiding the cross-device interaction. We limited our study to cross-device interaction between a large wall-mounted display and a handheld smartphone, and simple visual exploration tasks including filtering, accessing details, and creating overviews for data of interest. The large display showed a grid layout with multiple visualizations (like a dashboard). The participants were also shown what will appear on the smartphone—the effect of their interaction. We used a Microsoft Perceptive Pixel[1] (55-inch display) as the large wall-mounted display and an Apple iPhone 7 (4.7-inch touch display) as the handheld device to elicit cross-device interactions.

[1]Perceptive Pixel: http://www.perceptivepixel.com/

Table 7.1: Types of effects used in our design elicitation study.

Type	Effects
Filter	Extract region from large display to smartphone (or vice versa).
Detail	Show details for region of interest from large display on the smartphone.
Overview	Aggregate a visual representation from large display on the smartphone.

7.1.3 Methods

We identified three effects when coupling a large display with a handheld device during sensemaking. Table 7.1 captures these scenarios which cover (1) filtering, (2) extracting details, and (3) developing overviews for regions of interest (visualizations) from a large display. For these three scenarios, participants were asked to invent the interactions (signals) at three distances from the large display ($d < .75m$: close; $d < 1.5m$: middle; $d > 1.5m$: far) (Figure 5.1). To focus on cross-device interactions, participants were asked to invent interactions that span/involve both the large display and the smartphone. A counter example was given to help them understand what would not constitute a cross-device interaction: showing the large display interface on the smartphone and using pan/zoom for selection. While such an interaction is useful in some scenarios, it is not our focus as it does not take advantage of the physicality of one of the displays in the environment.

During each session, the participants performed one trial for each effect in a fixed sequence of filtering, showing details, and then generating overviews. For each effect, participants were presented with a region within the large display and

Table 7.2: Questions asked during the post-session interview.

#	**Question**
Q1	Do you think coupling two devices—a large display and a handheld device—is useful?
Q2	What do you think is the purpose for combining two devices in the context of visualization and data analysis?
Q3	How did you come up with the cross-device interactions?
Q4	A common design alternative for multiple devices is using multiple windows (focii) on the same device. What are the strengths and weaknesses of each (is one better than the other in some scenarios)?
Q5	Can you think of any strengths and weaknesses of using multiple devices for collaboration in front of the wall-mounted display?

asked to invent a cross-device interaction (a signal) that would trigger the effect. Participants were suggested to think aloud their ideas and reasons. At the end, they were interviewed to gain their feedback about the importance/benefits of the interactions they designed and their basis for inventing them (Table 7.2). Sessions lasted for less than 20 minutes. This procedure is similar to the user study conducted by Wobbrock et al. [248] to develop a taxonomy of tabletop gestures.

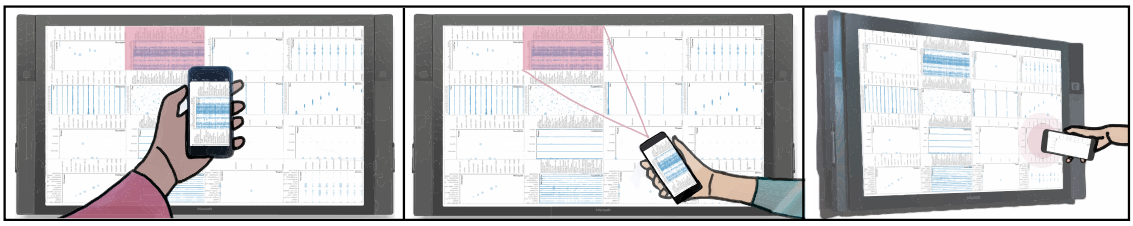


Figure 7.2: Three cross-device interactions suggested by the participants during our design elicitation study. Participants suggested holding the mobile device vertically to **capture a region of interest** (pink) in the field-of-view (left), using the handheld device to **point to a region** on the large display (middle), and using the handheld device to **tap a visualization** when close to the large display (right).

7.1.4 Results: Cross-Device Interaction Patterns

The main cross-device interactions that were brought up in our study are highlighted in Figure 7.2.

Interaction Styles: Participants designed three interactions when they are not close to the large display. Six participants (all except P1, P3, P7) suggested interactions around holding the smartphone vertically parallel to the large display (posture of taking a picture with a phone camera). Some participants (P1-P3, P5, P7) imagined holding the smartphone horizontally to point and select a region of interest. When far from the display, participants coupled these interactions with traditional zoom and pan to precisely select regions. One participant (P6) thought about spatial interactions where moving the smartphone along the perpendicular to the region of interest to filter it. Participants coupled these interactions with an explicit tap on the handheld device to actually trigger the effects after the action.

Effect of Exploration Task: Most participants (all except P1, P8) saw cross-device interactions for showing details and overviews to be a small variant of the ones designed for the filtering operation described above. This was expected since these operations are in fact related from a visual exploration standpoint. For example, showing details is essentially filtering together with more visual embedding. Participants saw showing details as combining two visualizations from the large display (e.g., get X, Y dimensions from one and Z from the another) and then switching between different modes of details on the smartphone (e.g., Z will be captured by color by default and can be changed). For camera and pointer-style interactions (from earlier), this is done by performing them twice or more to combine aspects within visualizations. Participant (P6) suggested to show details based on spatial locations in the 3D space around the user to find these details (guided by the organization of attributes on the large display). P1 and P8 suggested a physical drag (or brush) action with the smartphone where the movement of the phone decides the visualizations on the large display to combine. Participants imagined creating overviews by removing features from a visualization with gestures. For example, participants (P1-P6, P9) suggested to remove dimension X by shaking or swiping the phone in that direction in front of the large display.

Effect of Device Distance: All participants suggested different interactions based on the distance from the large display. Most participants (all except P4) preferred tapping with phone when close to the large display to convey a region of interest. P4 suggested using the front camera of the phone to reflect a region (like a mirror) and use it for the selection. When far from the display, participants

suggested camera-style and pointer-style interactions depending on the size of the region of interest. Pointing can be hard to precisely choose regions on the large display when far away and therefore using the camera to select and zoom into a region was seen as more tractable. At a moderate distance (middle), participants (all except P1, P5, P7) preferred holding the phone vertically to grab a region.

7.1.5 Observations: Participant Feedback

All participants agreed with the utility of multiple devices (Q1, Q2) for reasons including, (1) the ability to add an additional layer of information through the handheld device on top of the large display, (2) the added interaction abilities through the smartphone to easily manipulate the content of the large display, and (3) support for multiple users to work together through their devices without affecting the large display. They built their interactions based on their social familiarity with other technologies—participants who came up with remote pointer interaction often cited the modern television as a source of inspiration (Q3) and some of them stated that their interactions just felt natural in that context.

Participants identified that the ability to work more flexibly with a handheld device (remote or in front of the public display) makes it more suitable for working with data compared to having multiple windows on the interface. On the other hand, some of them (P1, P2, P6, P7) also identified the potential drawback of dividing their attention between devices (Q4). Few participants (P2, P6) in fact used it as a motivation for using the camera-style interaction as it requires the user to hold

the handheld device vertically, which would keep the large display in the line of sight. Also, the added advantage of directly interacting with other users through a smartphone-smartphone connection was identified as a benefit. Finally, the ability to collaborate with others more naturally was apparent; all participants noticed that they could access interesting data and interact with it without affecting the views of others (Q5). They suggested a push gesture when far and a tap gesture when close to send the information back to the shared large display.

7.1.6 Guidelines for Connecting Heterogeneous Devices

Based on our study, we found three physical cross-device interaction styles, based on the social familiarity of our participants with such interactions, for sharing information across devices during visual sensemaking (Figure 7.2). This was because the users saw differences in the type of interactions based on the distance from a large display that is shared between users in the co-located multi-device environment. They felt that directly grabbing a region of interest by taking a picture was the easiest and a natural thing to do at a moderate distance. Depending on the size of the region of interest, they also suggested using the handheld device as a pointer. When close to the display, users preferred the interactions to be even more direct in nature based on the contact of one device with another (or proximity to a region on the large display) to perform the same operations. To extend this to different visualization tasks, adapting the above interactions by combining them with other actions (e.g., taking a picture and then performing a gesture to develop overview)

was preferred than developing completely new cross-device interactions. Together these cross-device interactions create a complete embodied interaction framework for visual exploration in a device ecosystem.

7.2 Visfer Technique: Smartphones/Tablets + Large Display

As identified in our formative study, one of the main cross-device interactions developed by our participants was based on the notion of holding up a handheld device vertically to directly capture what is in front of it (similar to taking a picture with a camera). We enable this cross-device interaction through a camera-based visual data transfer technique called Visfer (VISualization TransFER) (Figure 7.3). This technique encodes visualizations in QR codes, which can be captured by using the cameras that are now built into most mobile devices. We thus extend the common practice of "taking a picture" to capture visual information through the camera. Furthermore, we develop this technique as part of a web-based framework that couples with existing web visualization framework such as D3 [15].

7.2.1 Design Considerations

We defined design guidelines that guide the content and position of the QR codes on the visualization interface, and enable fluid interaction [48] and spatial interaction models [88, 249] within the environment.

Make the QR code context-aware. The content shared through the QR code should be based on the available software infrastructure and the application

Figure 7.3: The Visfer technique enables transferring visualizations across devices to support cross-device visual sensemaking. (A) User captures a QR code; (B, C, D) visualizations loaded on personal devices from the large display. Transferred visualizations are adapted to the target device.

scenario. Our usage scenarios introduced earlier in the paper demonstrate the differences among various multi-device application settings. For example, a casual capture of visualization and underlying data from a public display at an airport could use a different QR code content compared to, say, a cross-device visualization being used by a co-located collaboration of analysts in front of immersive displays connected to a high-performance server [38]. The QR codes should remove the need for using indirect dialogs and control panels for sharing visualizations and focus on providing a direct and minimalistic interaction model based on the scenario.

Augment, not replace. The QR codes should be shown on demand. They should not occlude information on the visualization—the visualization itself should give precedence to the user's eyes, not to the camera. The QR codes should be automatically placed in the free space or manually placed by drag-and-drop. It should be possible to resize or remove them. This guideline also makes the QR code-based technique more closer to the interaction proposed by our participants.

Adapt visualizations to the device. To actually use visualizations across private and public devices with a branch-explore-merge protocol, [148] they should be perceivable and interactive on any device modality. For this purpose, we adapt Thevenin and Coutaz's notion of *plasticity*: [103] transforming a user interface to a form that best uses the device's modality. Upon transferring a visualization from one device to another, it should be possible to interact with the visualization right away using the input capabilities of the target device. The transferred visualizations should also be responsively adapted to the display size of the target, either by scaling them, or by transforming them to compact representations for small displays.

Adapt cross-device representations to the task. The transferred visualizations should also maintain the flow [48] and engagement of the analyst by expanding the notion of plasticity further based on standard visualization tasks, [72,108,236] such as creating an overview, considering details, and linking patterns across data. For example, it should be possible to use a smartphone for overviewing the data present on a large display. This type of adaptive visualizations is defined by Elmqvist and Irani [1] as *plastic visualizations* or *plastic visual representations*.

Create spatially-aware representations. The QR code on a display can provide a low-fidelity tracking of distance and orientation between the display and the camera(s) [249]. This information can be used to control visualization parameters such as zoom and detail levels. Similar interactions have been proposed using proxemics for updating visualizations based on the user's spatial attributes [88].

7.2.2 Visfer Framework

The Visfer framework, developed around this technique, embeds not only a URL in the QR code, which is the most common use for QR codes today, but it also uses the QR code as a transport layer to transfer visualization source code, pipeline, and state changes as a result of interaction, over the visual channel to the target device. This allows the target device to receive the current state of the captured visualization, including any filters, selections, or annotations. Finally, the framework also supports *plastic visual representations* that can adapt to the capabilities of a device; for example, a node-link diagram of a social network on a large display can be automatically aggregated into communities when rendered on a smartphone display.

7.2.2.1 Visualization Transfer: Levels of Content

The three content levels primarily differ in the type of the content encoded into the QR codes including data, visualization pipeline, and dynamic state:

- **Level 1:** At the basic level, the framework supports creating static QR codes containing URLs or links to the data driving the visualization. This data, which is stored on a server, can range from open standard formats for communication to byte code and database indices. Due to the support for generic data types, the application developer using the Visfer framework has complete control over how to handle the content once the QR code is decoded by the framework. The developer can connect the data from the URL to the plastic

representation modules of the framework or use the data to carry out some other application logic. Due to the simplicity and flexibility of the content being encoded here (just URLs), there needs to be enough application support to generate the URLs on a server and network-level support to transfer the actual data once the URL is decoded by the end-user application.

- **Level 2:** The second type of content is the visual representation itself, or rather the pipeline to recreate the visual representation. Here, Visfer supports transfer of the static visualization pipeline in the form of JavaScript code through the QR code. This level supports simple application scenarios for cross-device visualization on non-interactive public displays such as ones at airports or restaurants, by offloading the visualizations to the personal devices of the users. The dataset for the visual representation can be either hard-coded in the JS code (avoiding indirection through a server), or provided through a link depending on the size and the available infrastructure. To support embedding the JS code without increasing the physical size of the QR code on the large display, the framework supports animated QR codes that contain multiple QR codes played one after the other and looped.

- **Level 3:** Here, the content takes the form of the visual representation *and* its dynamic state, which is represented by the interactions performed by the user. For this level, we developed a custom Visfer transfer protocol using the JSON communication format, based on Vega [63] grammar. This protocol helps encode the data, scales, marks (the granular representations such as

rectangles, lines, and circles), as well as interaction styles for the visualization pipeline and the visualization state through user selections. These attributes are automatically transformed by the plastic representation modules to fit the device modality by changing the width, height, and locations, and also converting between mouse interaction and touch interaction. Furthermore, in this level, the representation (marks and scales) are also be changed by the Visfer framework and the application developer, to fit to the InfoVis tasks (detailed later in the application examples). This level differs from the second level due to its support for the dynamic state of the visualizations, and targets a different application scenario. This level can use animated QR codes in the absence of a server to transfer information as the JSON content representation can go beyond the content handled by a single QR code.

The QR codes, both static and animated representations, can be repositioned by drag-and-drop operations, and resized through pinch-to-zoom operations. While the codes are initially placed in the corners of the visualizations to reduce occlusion, more topology-aware strategies are required to appropriately place them in free spaces on the interface. The resize operation spreads the QR content over more (or fewer) frames when the size is increased (or decreased), to maintain the readability of the individual QR code frames.

7.2.2.2 Visualization Adaptivity

The Visfer framework converts cross-device visualizations into *plastic representations* that adapt to device modality and visualization tasks being performed by the user. These plastic representations are an integral part of the cross-device interaction as they actually make this interaction more scalable by adapting any region of interest on the large display (however big it is) to the small screen space on the handheld device. This is carried out by transforming the visualization attributes within the JSON representation (defined in level 3) based on Vega [63].

Visfer JSON Content Representation. The JSON representation consists of (1) definitions of *width*, *height*, position, and *padding* of the visualization; (2) a *data* key with value as the raw data table or an array of links pointing to the raw data stored on the file system (or server); (3) *scales* defining the mapping between data attributes to visual boundaries and presentation attributes (e.g., color, opacity levels); (4) *axes* definitions pointing to the scales; (5) *marks* storing the graphical primitives assigned to each datum, corresponding properties based on the scales, and update definitions for handling interactions; and (6) *signals* driving the membership of data points in selections (*predicates*) from the user interaction (for example, brushing). The signals also drive the scales to change them based on the current interaction. These attributes are directly borrowed from Vega's JSON-based grammar [63] to provide a generic way to recreate visualizations. Beyond this representation, the Visfer framework also stores the current state in the JSON by saving the current selection of data points either in an intensional predicate form

(for e.g., $5 < data.variable < 10$) or an extensional predicate form (for e.g., select points $\#10, \#25, \#30, \ldots$) based on the interaction.

Adapting to device. Based on the JSON representation described above, adapting to a specific device is a process of changing the layout attributes such as width, height, and padding, and interaction events to the target device. The JSON representation stores the layout attributes along with the source device width and height (global attributes), which, when passed to the target device, are transformed to the new resolution. The Visfer framework also uses a 1D layout on small-resolution devices by stacking visualizations one below the other rather than a 2D arrangement to make them more readable. The interaction definitions are translated to the input type on the target: converting mouse interaction handlers to touch and vice versa.

Visualization responsiveness. To further extend plastic representations, the attributes within the JSON representation should be transformed to fit the visualization tasks [108] being performed by the user. There are multiple design choices in applying these transformations in terms of where to branch out from the original visualization pipeline. For example, as seen in LARK [188] this can happen at the levels of analytical data abstraction, spatial layout, and presentation. As identified in our design elicitation study, the Visfer interaction technique is augmented with simple options to select the appropriate transformations. Figure 7.4 provides examples of these transformations for each visualization task. Here, we describe the conditions under and mechanisms through which these transformations are handled, or through explicit specification from the application developer.

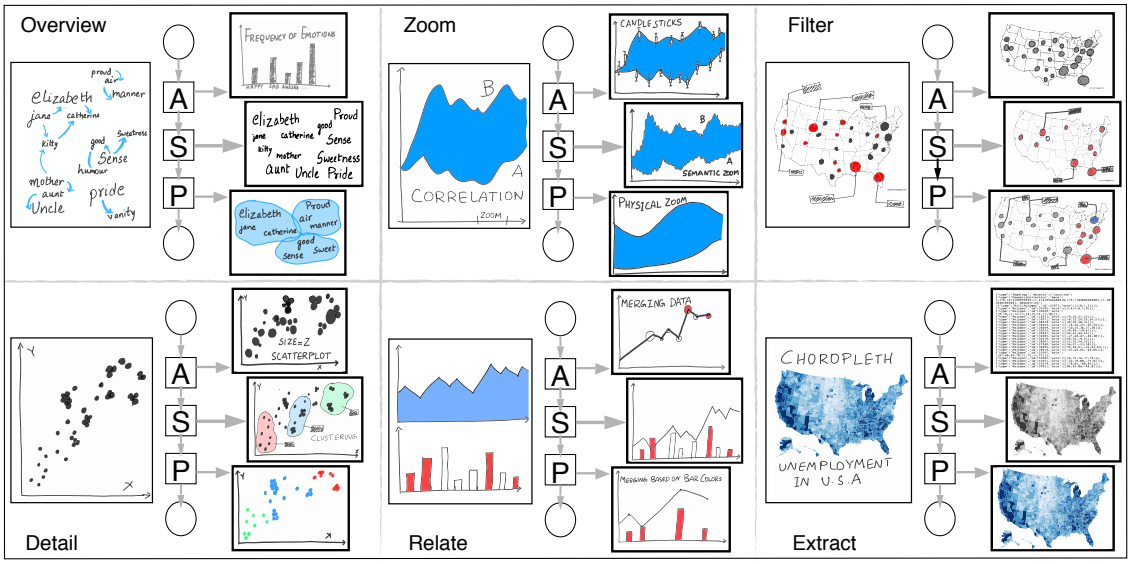


Figure 7.4: In this figure, A, S, and P stand for the analytical abstraction, spatial layout, and presentation layers in a visualization pipeline. Plastic visualizations are created by modifying the pipeline by branching out from any of these layers to create new and interesting visual representations on a target device. In Visfer, we combined this with visualization tasks to come up with a structured way to generate plastic visual representations. For example (top left), you can capture a phrase net and branch out from its analytical abstraction layer to create a sentiment histogram. The transformations happen by changing the scales, marks, and other attributes in Visfer JSON representation of the visualization pipeline and state.

- **Overview:** This transformation across devices take three different forms: (1) creating alternate representations to show aggregation at data abstraction layer—for instance, a word cloud visualization of product reviews can be transformed into a bar chart by abstracting the data as the review sentiment; (2) transforming visualization into alternate layouts—for instance, by sorting the words in a word cloud based on frequencies; and (3) changing the presentation attributes—for instance, coloring based on frequency ranges for words in the word cloud. While the latter two forms are automatically performed by changing the properties of the marks in the JSON representation, overview at abstraction requires explicit developer logic to define the new abstraction.
- **Zoom:** The framework allows semantic [250] and geometric zooming by manipulating the spatial layout and presentation layers. This is carried out by updating the dimensions and positions of the marks in the JSON representation based on a zoom position. At the data abstraction layer, a zoom transformation means looking at more attributes associated with each data point, which should be assigned by the developers based on their application. While the framework uses a default zoom position based on distance and orientation, it can be further controlled by the end user (e.g, the analyst).
- **Filter:** This transformation can also be seen as branching from the original visualization based on the selections on the source device. The framework handles branching the pipeline at spatial layout and presentation layers for this transformation by changing the visibility (e.g., through transparencies) of the selection. For example, a scatterplot matrix with brush-and-link selec-

tions transferred to a target device, is transformed to only show the current brushes by making the rest of the points completely transparent. Filtering at an abstraction level is similar to the overview transformation as it involving removing a data variable from the visual representation.

- **Details:** The inverse of the overview transformation is details-on-demand. The framework requires explicit definitions from the application developer to create details. The details can be of different kinds, ranging from more data attributes encoded in the visualization at the data abstraction level, to switching to more granular and categorized visual representations at the spatial layout and presentation layers. Due to the sheer amount of design opportunities here, the developer should define which visual attributes should be attached to the visualization to show details in terms of the graphical primitives (marks), layout, and presentation attributes.
- **Relate:** A relate transformation shows relationships between data. The Visfer framework supports combining two visual representations to create composite visualizations [111] by capturing their QR codes consecutively. The visualizations corresponding to the captured QR codes are automatically overlaid on the spatial layout. For transformation at the presentation level, the overlay can also be based on a particular presentation attribute from the application developer. The relate operation at an abstraction layer requires definition of the new abstractions by the application developer.
- **History and extract:** By maintaining the visualization states, the framework supports storage and extraction of historical states of the visualization.

Overall, by taking control over the pipeline, the framework handles transformation at the presentation and spatial layouts for most task types. In case of conflicting automatic transformation choices, the framework gives higher preference to layout. The application developer handles the remaining transformations, especially at the abstraction layer, based on their design. Beyond these features, the application user can switch between transformations on the target device.

For more implementation details, please refer to our original publication [22].

7.2.3 Application: BusinessVis

This cross-device application was created to visualize business data from the Yelp academic dataset, covering about 10,000 businesses in Phoenix, Arizona and 300,000 user reviews, across multiple devices. It was completely created with web technologies: HTML, JS, and CSS. The BusinessVis application supports collaboration among users and devices to analyze this big dataset through the Visfer framework. The interface has three default views: a geospatial map of the businesses, a category treemap, and a rating view showing the list of companies along with their user ratings (Figure 7.5). These views are connected to each other through brush-and-link interaction—any selection on one view is reflected on the rest. The goal of this application is to provide insights into the spatial locations, popular and top rated businesses, and feedback from the reviews. The users can explore the data without being restricted to a single screen, which is packed with information, and without interfering with each other's work.

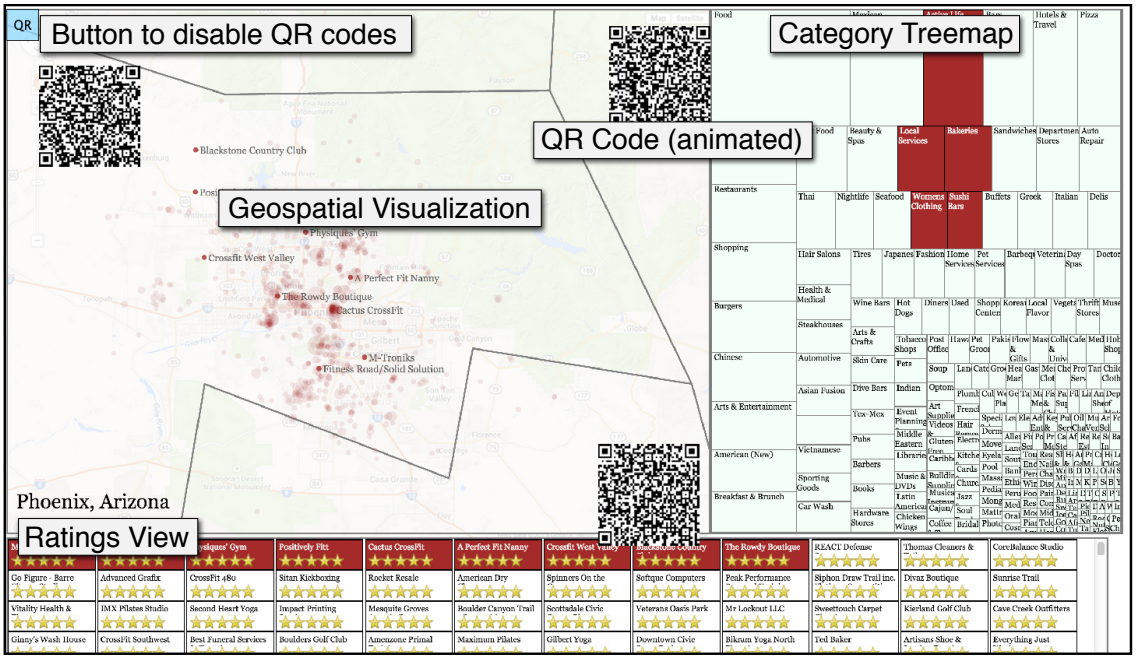


Figure 7.5: BusinessVis allows visual exploration of business reviews on Yelp through three visualizations. QR codes are shown/hidden in the interface when the button on top left of the interface is clicked. To further explore the data, the QR codes can be captured with a handheld device (tablet/smartphone as seen in Figure 7.3) by multiple users.

The BusinessVis application is showcases the possibilities brought about by cross-device visualizations augmented with QR codes for visual discovery and data transfer. BusinessVis uses all three types of QR contents presented in the framework description. It uses level 1 to share the business data among the devices. Level 2 is used to share visualization pipelines initially, when no user interaction is performed yet. The third level is most commonly used to share the pipeline and the dynamic state of the visualizations.

The interaction principle behind BusinessVis is to first support exploration of the visualizations on the large display, and then provide an additional "layer"

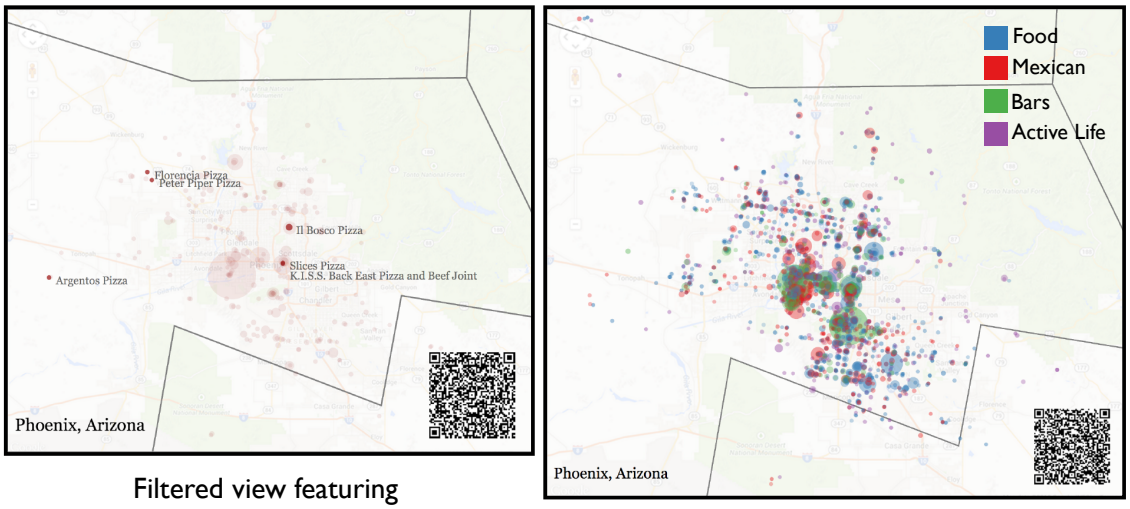


Figure 7.6: Two plastic representations of the map visualization (in Figure 7.5). (Left) A filtered view shown on the handheld device capturing five best-rated Pizza restaurants in Phoenix. (Right) A details view shown on the handheld device adding more visual encodings into the map visualization with circle colors capturing top 4 popular business categories, opacity encoding the average reviews, and size encoding popularity of businesses. Each view on a user's handheld device can also be augmented with a QR code to share it with other analysts.

to view other perspectives through handheld devices into the data underlying the large-display visualizations. This will help us create flexible analytical scenarios that happen through visual exploration on the large display, as well as opportunistic interaction on mobile devices.

Beyond the aforementioned views, the BusinessVis interface creates plastic visualizations to show overviews, filtered views, and more details by adapting these visualizations from different levels of their pipelines (Figure 7.3). The geospatial visualization transforms into, (1) a heatmap of the business categories upon overview

to show their spatial distribution, and (2) a filtered view based on the user selections in the connected visualizations (Figure 7.6). These transformations are automatically performed by the framework and can also be controlled by the user (through a button tap). This map visualization also transforms into a detail view to show the categories and business ratings using presentation attributes such as opacity, size, and color (Figure 7.6). The treemap visualization can transform to show more details such as aggregate user ratings for each category and their popularity, and filter to show current user selection from the brush-and-link operations (part C in Figure 7.3). Finally, the ratings view can transform into a overview word cloud of all the user reviews, and details with the sentiment data (part D in Figure 7.3). For relate tasks, the framework automatically merges the states of geospatial and rating visualizations to create a hybrid visualization.

7.3 When David meets Goliath: Smartwatches + Large Display

The Visfer technique combines smartphones and tablets with large displays for visual exploration by supporting a cross-device interaction. Recent works have shown the power of such techniques to create a flexible workflow in front of the large displays (cf. GraSp [154]). Developing on such ideas, we created a conceptual framework (Figure 7.7) to combine large displays with small devices—covering what, where, and how the combination happens—to explore specific roles for each device and interactions to achieve these roles. We also discuss how visualization tasks can be supported through this conceptual framework.

Figure 7.7: Visual data analysis using large displays and smartwatches together. Multiple analysts can extract data from views on a large display (A) to their smartwatches (B) and compare the data on other visualizations distributed over the large display by physical movements followed by direct touch (C) or remote interaction. This pull/preview/push metaphor can be extended to many visualization tasks. The watch enhances the large display by acting as a user-specific storage, a mediator, and a remote control, and aids multiple users work in concert or by themselves.

Beyond smartphones and tablets, we combined smartwatches with large displays to allow the watch to serve as a personalized analysis toolbox. In this function, the watch supports the multivariate data exploration on a large display interface containing multiple views (cf. coordinated and multiple views [226]). The devices represent two extremes—like David and Goliath—of interactive surfaces in many ways (e.g., small vs. large, private vs. public, mobile vs. stationary), which yields several fundamental design challenges for their combination. To tackle these, we first derive the basic roles of the two devices by drawing on the literature as well as an example data analysis scenario. Based on these considerations, we propose a conceptual framework defining the specific interplay between the smartwatch and the large display for a single-user. Within this framework, users can interact with the large display alone, and also benefit from the watch as a container to store and preview content of interest from the visualizations, and manipulate view configura-

tions (Figure 7.7). While collaboration is not explicitly considered yet (cf. group awareness [196], communication [197], and coordination [148]), the concepts allow for simultaneous (parallel) work from multiple users during visual exploration.

7.3.1 Elementary Interaction Principles on the Smartwatch

Generally, the smartwatch supports four types of input: simple touch, touch gestures, physical controls, and spatial movements. As the analysts mainly focus on the large display during visual exploration, the input on the watch should be limited to simple, clearly distinguishable interactions that can also be performed eyes-free to reduce attention switches (cf. Pasquero et al. [164], von Zadow et al. [158]). Therefore, we use three interactions on the watch (see Figure 7.8a-c): *swiping horizontally* (i.e., left or right), *swiping vertically* (i.e., upwards or downwards), and, if available, *rotating a physical control* of the smartwatch [160] as, e.g., the rotatable bezel of the Samsung Gear or the crown of the Apple Watch. For more advanced functionality, long taps as well as simple menus and widgets can be used. Finally, using the internal sensors of the watch, the users' *arm movements* or poses (Figure 7.8d) can be used to support pointing or detect different states [95, 162, 163].

When the smartwatch takes the role of user-specific storage, we assume that users have a mental model of two directions for transferring content; towards the smartwatch or towards the large display. Based on this, a specific axis of the smartwatch can be derived: The *proximodistal axis* (i.e., along the arm) is suitable for transferring content; swiping towards the shoulder (i.e., left or right depending on

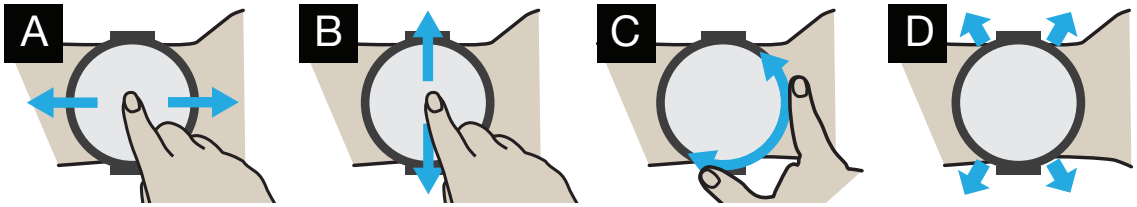


Figure 7.8: Smartwatch interactions: (a) swiping horizontally, i.e., along the arm axis for transferring content; (b) swiping vertically or (c) rotating a physical control for scrolling through stored content; and (d) moving the arm for pointing interaction.

the arm on which the user wears the watch; Figure 7.8a) can pull content from the large display onto the smartwatch. Vice versa, swiping from the wrist towards the hand, i.e., towards the large display, can allow to push content back to the visualizations. Additionally, the *axial axis* (i.e. orthogonal to the arm) can be defined as a second axis (cf. von Zadow et al. [158]). We suggest scrolling through the stored content by either swiping vertically (Figure 7.8b) or rotating the bezel or crown of the watch (Figure 7.8c).

7.3.2 Conceptual Framework for Multiple Devices in Visual Analysis

By incorporating the different roles of the smartwatch and the large display, our conceptual framework supports a multitude of tasks during visual exploration [57, 72]. In the role of user-specific storage, the smartwatch provides access to the data, i.e., points of interest. Both the shared large display and the smartwatch (as remote control) determine or define the context of an interaction. Regarding the task topology from Brehmer and Munzner [57], the combination of these two aspects—data and context—represents the *what* of an interaction, and enables the

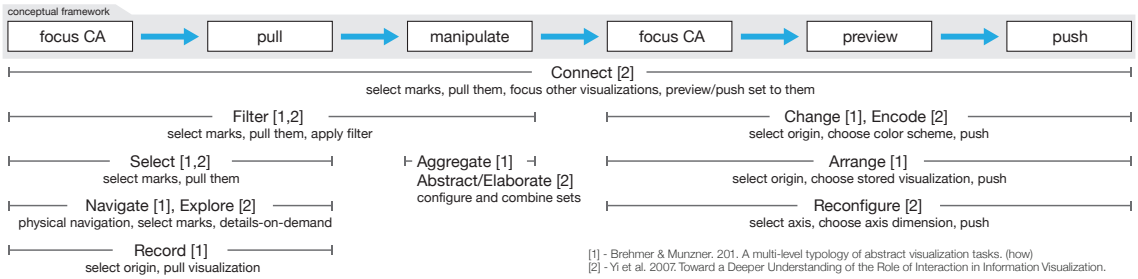


Figure 7.9: Our conceptual framework addresses a wide range of tasks, illustrated here by mapping two established task classifications [57, 72] onto interaction sequences that are enabled by our framework. For some tasks, certain aspects are also still supported by the large display itself, e.g., zooming and panning from *abstract/elaborate* and *explore* [72]. Regarding the typology by Brehmer and Munzner [57], we focus on their *how* part. From this part, a few tasks (*encode*, *annotate*, *import*, *derive*) are not considered as they are going beyond the scope of this paper. **CA: Connective Area**.

smartwatch to act as mediator defining the *how*. This mediation enables the analyst to solve a given task coming from questions raised in the scenario (*why*). Our framework provides components that blend together into specific interaction sequences and address the various task classes (Figure 7.9). In the following, we will introduce these components and describe their interplay. We will also reference the matching tasks from Figure 7.9 in small caps (Example).

7.3.2.1 Item Sets & Connective Areas

The primary role of the smartwatch is to act as a personalized storage of *sets*. We define *sets* as a generalized term for a collection of multiple entities of a certain type. In our framework, we currently consider two different set types: data items and configuration properties (e.g., axis dimension, chart type). These sets can also

be predefined; for instance, for each existing axis dimension, a corresponding set is generated. On the smartwatch, the stored sets are provided as a list. As shown in Figure 7.10, each set is represented by a description, a miniature view or icon, and further details (e.g., value range). Consistent with the set notion, sets of the same type can be combined using set operations (i.e., union, intersection, complement). Finally, to allow managing sets over time, they are grouped per session. Former sessions can be accessed using the watch.

During the data exploration, the region that a user interacts with can provide a valuable indication of the user's intent. We therefore define four zones for each visualization—called *connective areas* (CA)—that will provide the context (*what*) of an interaction: the marks, canvas, axes, as well as a special element close to the origin. Connective areas define the set type (Figure 7.11) and control the functionalities accessed on the two devices. To **focus on a CA**, the interaction comprises of, in the simplest case, tapping or circling marks (i.e., data points) for selection. For other CAs, user can set the focus in two ways: by performing a touch-and-hold

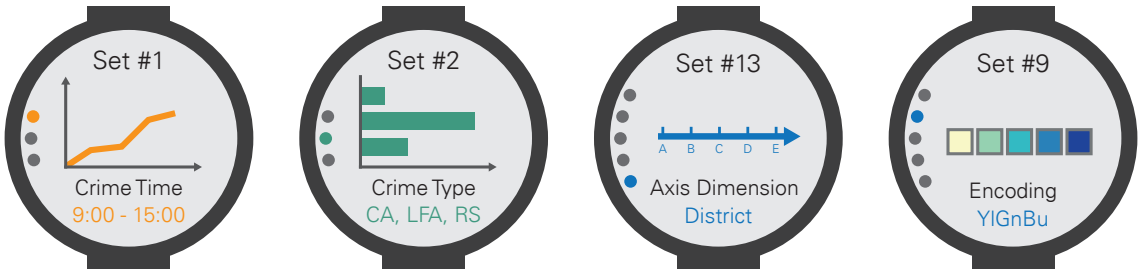


Figure 7.10: *Sets* are represented by labels and a miniature: for sets with data items, the miniature is based on the view where it was created (left); for sets containing configuration items an iconic figure is shown (right).

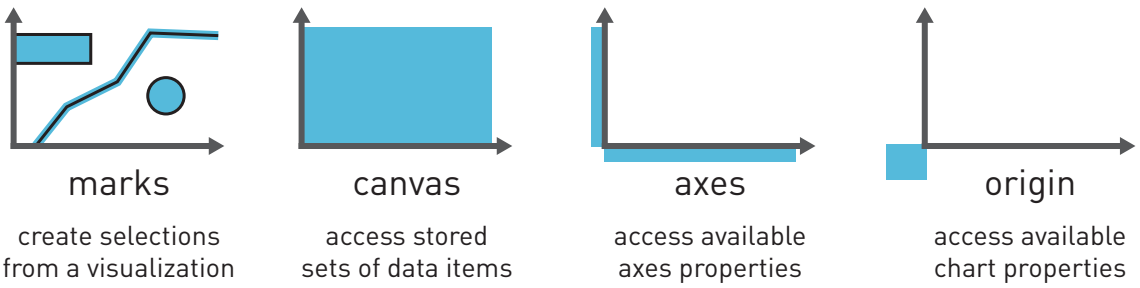


Figure 7.11: *Connective Areas* (CA) represent semantic components of a visualization that have a specific interaction context with respect to a secondary device.

(long touch), the focus is set onto the respective area underneath the touch point but stays only active for the duration of the hold; by performing a double tap, the focus is kept as long as not actively changed. Setting the focus activates suitable functionalities for the specific connective area on the watch. On focus, stored set content can also be previewed on the large display.

While we consider working in close proximity to the large display as the primary mode of interaction, certain situations exist where this is not appropriate or preferred. For instance, a common behavior when working with large displays is to physically step back to gain a better overview of the provided content. To remotely switch the focus onto a different view or connective area, the user can perform a double tap on the smartwatch to enable *distant interaction* and enter a coarse pointing mode. Similar to Katsuragawa et al. [162], the pointing can be realized by detecting the movements of the watch using its built-in accelerometer. Alternatively, it is also possible to scroll through the visualizations instead of moving the arm. In both cases, the current focus is represented as a colored border around the corresponding view on the large display. After confirming the focus, the analyst can

Figure 7.12: David and Goliath Technique: Interactions for (a) pulling, (b) previewing, and (c) pushing of sets of data from visualizations.

select the desired connective area within the focused visualization in a second step and then access and preview stored sets. This remote interaction provides the same functionality as the direct touch interaction. Users can explicitly switch between interaction based on direct touch or on remote access from both close proximity and far distance. This transition could also be extended by using proxemics (cf. proxemic interaction [124, 126]).

7.3.2.2 Creating & Managing Sets for Visual Exploration

To develop insights through visual exploration, the interactions in our framework are (Figure 7.12) focused on selecting, manipulating, and previewing data points of interest, as well as applying the previews permanently to a visualization. These interactions are mediated by the smartwatch based on context of the user. The concepts enabling these four functionality also define the *how* of the analyst's task. To **pull** (i.e., create) a set, the analyst first selects marks in the visualization on the large display by tapping or lasso selection, and then swipes towards herself on the watch (SELECT). The resulting set is stored on the smartwatch. Now, by

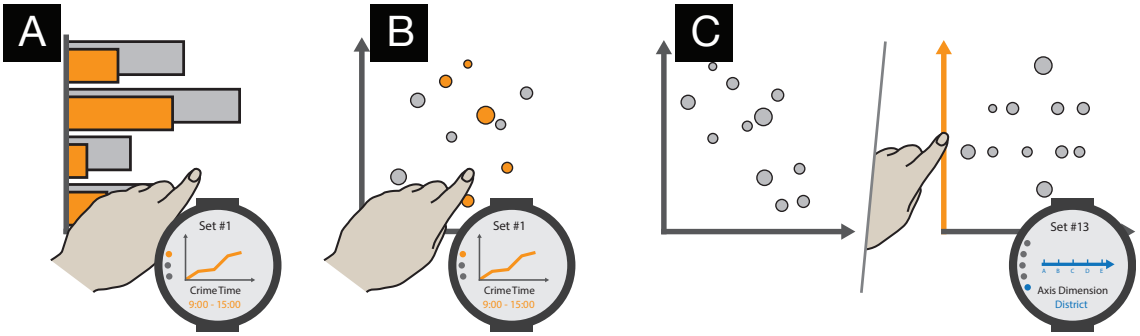


Figure 7.13: Previewing stored sets results in (a+b) inserting or highlighting the containing data points in the visualization, or (c) adapting the visualization to the respective configuration item (here: axis dimension).

again switching the focus to another view on the large display (i.e., by holding, double tapping, or pointing), the set currently in focus on the watch gets instantly **previewed** on the target visualization. The preview is only shown for a few seconds, or, in the case of holding, for the duration of the hold. Depending on the visualization type and the encoding strategy (aggregated vs. individual points), the items are inserted as separate elements or highlighted (Figure 7.13a,b). As the focus is set on a connective area, the smartwatch can still be used for further exploration. For instance, by swiping vertically on the watch or rotating its bezel, the user can switch through the list of stored sets and preview others for comparison. Again, the preview is shown only for a few seconds. To permanently **push** the changes to the view on the large display, a horizontal swipe towards the large display, i.e., the visualization, can be performed on the watch (Connect). As push is considered a concluding interaction, the system then switches back to a neutral state by defocusing the view.

Besides data items, visualization properties can also be accessed and adapted. Based on the connective areas, we distinguish between axis properties (e.g., dimension, scale) and chart properties (e.g., chart type, encoding). These configuration sets are mostly predefined, as only a limited number of possible values/configurations exist. For instance, when tapping on an axis, all dimensions as well as scales are offered as individual configuration sets on the watch. As with data items, scrolling through this list of sets results in instantly previewing the sets, e.g., the marks would automatically rearrange accordingly to the changed dimension or scale (Figure 7.13c). By performing a push gesture, this adaptation is permanently applied to the visualization on the large display (Change, Encode, Reconfigure). Naturally, more possibilities for visualization configuration may exist; however, covering all of them is beyond the scope of this work. In addition to single configuration properties, the origin can also provide access to the visualization in its entirety, i.e., a set containing all active properties at once. This allows storing a visualization for later use, or moving it to another spot in the interface (Arrange, Record).

As an extension to storing sets, the smartwatch also offers the possibility to **manipulate** and combine sets on the watch. By performing a long tap on a set, these operations are shown in a menu. For all set types, this involves the possibility to combine sets based on a chosen set operation (e.g., union or intersection), which results in a new set (Aggregate). For sets containing data items, sets can also be bundled; previewing or pushing such a bundle shows all the contained sets as separated overlays at once; thus, merging them on the view itself. Furthermore, it is possible to create new filters and change the set representation on the watch.

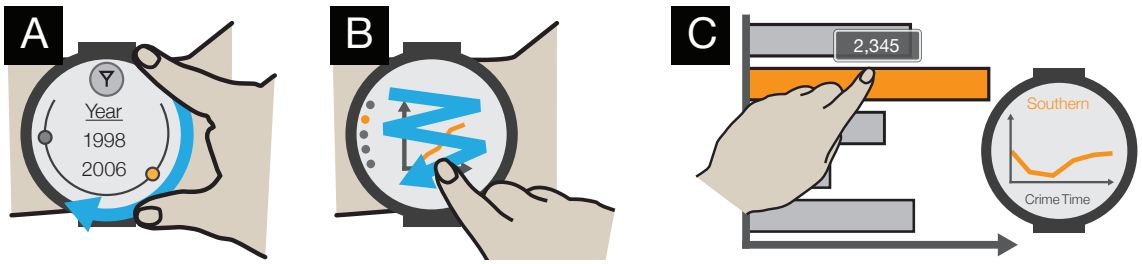


Figure 7.14: The smartwatch allows (a) applying filters to data item sets; (b) deleting sets by wiping; and (c) displaying additional details-on-demand.

The filter option allows the analyst to select a property first and then to define the filter condition (e.g., crime date in July 2015). For numeric filter options, sliders are provided (Figure 7.14a). To delete a set on the watch, a wipe gesture can be performed (Figure 7.14b).

All in all, the *set* metaphor is ideal for visually comparing regions of interest on the large display because data items can be extracted from the views, manipulated or combined on the watch, and then previewed on target visualizations (Connect). The ephemeral nature of our proposed preview techniques enables analysts to explore aspects without worrying about reverting to the original state of a visualization. In addition, the set storage further acts as a history of user interactions, to undo, replay, or progressively refine the interactions [108] (Record). During the exploration, the watch can also be used for tasks not involving sets. For instance, existing details-on-demand mechanisms on the large display (e.g., displaying a specific value for a mark) can be extended by displaying further details on the watch, e.g., an alternative representation or related data items (Figure 7.14c; Navigate).

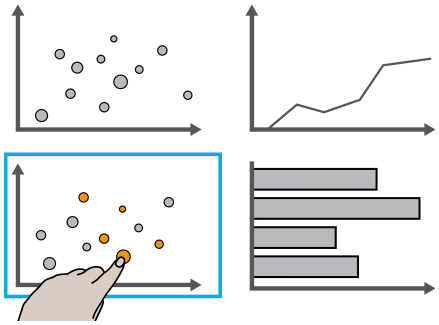


Figure 7.15: The scope of user interactions is limited to the views in focus.

7.3.2.3 Scope of Interactions in Multi-User Setups

In common coordinated multiple view (CMV) applications [226], changes in one visualization (e.g., selection, filter, encoding) have global impact, i.e., they are applied to all displayed views. This behavior may lead to interference between analysts working in parallel [148]. To avoid this issue, the effects of an interaction should by default only be applied to the visualization(s) currently in focus of the analyst (Figure 7.15). Further, we constrain the scope of an interaction mediated by the smartwatch to a short time period. More specifically, on touching a visualization to apply a selected adaptation from the smartwatch, the resulting change is only visible for a few seconds or as long as the touch interaction lasts. At the same time, there also exist situations where changes should be applied permanently, i.e., merged back into the shared visualization [148]. Therefore, it must be possible to push these adaptations to the large display and keep the altered data visualization.

7.3.2.4 Feedback Mechanisms

For cross-device setups, it is important to consider feedback mechanisms in the context of the interplay between devices, especially to avoid forced attention switches. In our setup, we are able to use three different feedback channels: visual feedback on the large display and on the smartwatch, as well as haptic feedback via the watch. On the large display, the feedback equals the system reaction on user interactions, e.g., previewing content. To further ease the exploration of different sets, a small overlay on the large display indicates the set currently in focus when scrolling through the list on the watch, thus reducing gaze switches between the two devices. The colored border around a view indicates if a connective area is focused and thus the watch can act as a mediator.

We use haptic feedback, i.e., vibrations of the smartwatch, for confirmation. When successfully performing an interaction, e.g., pulling a set onto the watch or pushing it to a visualization, the watch confirms this by vibrating. Alongside with the small overlays described above, this behavior also supports eyes-free interaction with the smartwatch. Further, the watch also vibrates to indicate that additional information or tools are available on the watch: While moving the finger over a visualization, the watch briefly vibrates when a new element is hit to indicate that details-on-demand or more functionality are available. To some degree, this also enables to "feel" the visualization, e.g., through multiple vibrations when moving across a cluster of data points in a scatterplot.

7.3.3 User Study: Interaction Patterns

The David and Goliath Technique has the potential to ease visual exploration, however, the way the techniques are utilized during sensemaking—and affect the developed observations from data—is not clear. Therefore, we conducted a user study with our large display and smartwatch combination (**LD+SW**), against an equivalent large display only interface (**LD**) for visual analysis tasks. This allows us to investigate the interaction patterns during visual exploration, and especially how the context-aware smartwatch and the different roles it takes alter these patterns.

Experiment Conditions. The study comprised two conditions: LD+SW and LD. The LD+SW interface allows participants to: (1) pull data from the large display to create sets (each set gets a unique color), (2) show a preview of sets on target visualizations, (3) push sets to the large display, (4) use the smartwatch as remote control to focus views on the large display, and (5) combine sets on the smartwatch. Except for the last two, equivalent capabilities were created for LD using a freely movable overlay menu with a scrollable set list that appears on long touch. All participants worked with both conditions; the order was counterbalanced.

Participants. We recruited 10 participants (age 22-40; 5 female; 5 male) from our universities (U1: P1-P4, U2: P5-P10). Participants were visualization literate with experience in using visualizations with tools such as Excel and Tableau; 4 of them used visualizations for data analysis (for their course or research work).

Apparatus and Dataset. The study was conducted in two setups as described in the Implementation section. They only differed in the size of the large

display (U1: 84-inch, U2: 55-inch); the smartwatch (Samsung Gear S2), the prototype version, as well as dataset (Baltimore crime) were the same.

Tasks. We used this dataset to develop user tasks that can be controlled for the study purposes. Tasks contained three question types: (**QT1**) finding specific values, (**QT2**) identifying extrema, and (**QT3**) comparison of visualization states [21]. In general, the complexity of a task results from the number of sets and the target visualizations to be considered to answer it. After pilot testing with two participants, we settled on a list of questions with different complexities: for QT1 and QT2 the number of targets was increased to create complex tasks, while for QT3 both the number of sets and the target visualizations were increased. Here are few sample questions used:

1. How many auto thefts happened in Southern district? (**QT1**)
2. What are two most frequent crime types in Central? (**QT2**)
3. What are the differences between crimes in the Northern and the Southern districts in terms of weapons used? (**QT3**)
4. For the two crime types that use firearms the most, what are the differences in crime time, district, and months? (**QT3**)

The task list contained 9 questions overall. Two comparable lists were developed for the two conditions to enable a within-subject study design. These tasks can promote engagement in LD+SW and effective use of LD.

Procedure. The experimenter first trained participants in the assigned interface by demonstrating the visualizations and interactions. The participants were then allowed to train on their own on a set of training tasks. Following this, they

worked on the nine tasks, answering each question verbally. They then moved on to the other condition and repeated the procedure. Afterwards, they completed a survey on the perceived usability of the two interface conditions, as well as on general interaction design aspects. Sessions lasted one hour.

Data Collected. Participants were asked to think out aloud to collect qualitative feedback. Their accuracy for the tasks was noted along with the participant's interactions and movement patterns as well as hand postures by the experimenter in both conditions. All sessions were video recorded and used to review the verbal feedback as well as noted observations.

7.3.3.1 Results: Summary

After analyzing the data collected, we found three main results:

- LD+SW interface allows flexible visual analysis patterns.
- Set management tends to be easier in LD+SW due to fewer attention switches; thus, simplifying comparison tasks.
- Participants rated the interactions within our LD+SW prototype as seamless, intuitive, and more suited for the tasks.

7.3.3.2 Interaction Patterns and Observed Workflow

As we expected, the interaction abilities of both devices in LD+SW and the ability to work from any distance lead to flexible workflows for visual analysis. Therefore, we focused on observing when and how these workflows manifest in our

tasks. In simple QT1 and QT2 tasks, participants used the basic touch interaction (long touch, double tap) to preview a set on the target visualization (workflow **F1**). Eight participants used physical navigation to move from one part of the display to the other to perform such tasks, while others did this remotely with their watch. For most of them (7/10), the long touch action was seen to be sufficient to quickly answer these tasks when only a value or extrema must be determined. For comparisons between two sets (QT3) on a target, eight participants preferred to disconnect from the large display by double tapping it and taking two or three steps back to gain a full view of the target visualization (**F2**), while only two remained close and used long touch. On LD, it was not possible to step back since participants had to stay close to the display to switch between sets to compare them.

In more complex tasks where two or more targets were considered, participants in LD+SW further showed this need to step back to get a better view of the large display. While eight participants mostly performed these tasks by moving back-and-forth in front of the display to collect sets and pick target visualizations to make comparisons (**F2**), three participants (P7 did both) used remote controls to access target views to avoid this movement to an extent (**F3**). To track the sets on their smartwatch, four participants held their hand up to view both displays at the same time, while the majority (seven) differentiated sets based on their assigned color. This set awareness was weaker in LD condition; the participants often shifted their focus between the sets menu and the visualizations repetitively to achieve the same. Finally, five participants used the combine option when related sets were already created for previous tasks, avoiding large display interaction (**F4**).

Overall, we observed that participants followed the pattern of *interact, step back, and examine* (F1, F2), as well as *interact remotely* from a distance (F3, F4). Further, they often interacted eyes-free with the watch, although the prototype could be further improved in that regard (e.g., by displaying set labels on the large display as more sets are being previewed). The rotatable bezel of the watch was exclusively used for switching sets, thus played an important role acting as a tangible control.

7.3.3.3 Differences in Developed Insights

Workflows F1-F4 were observed for different tasks on the LD+SW condition. Given these observations, we were interested in the differences in task answers from these workflows compared to their LD counterpart. In QT1 and QT2 tasks, participants answered accurately on both conditions. However, the LD condition was less preferred, e.g., participant P1 stated, "*the interaction in LD was a little complicated and felt slower than with the watch.*" More nuanced patterns existed in participant answers to visual comparison of two or more sets in target visualizations: they made observations about specific values, trend differences in the target, and relative differences in specific data items. To begin with, all participants mentioned specific value-based differences between the sets in the target visualization. To observe trend and relative differences more effectively in LD+SW, participants (following workflows F2 and F3) made use of the possibility to step back from the large display and to switch back-and-forth between sets with the help of the rotatable bezel on the watch. In the LD condition, participants tried to switch back-and-forth by tapping

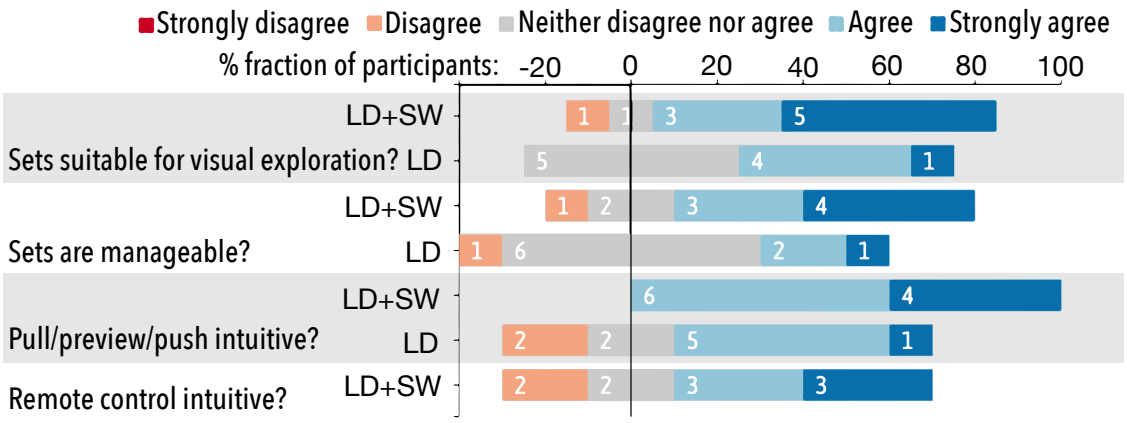


Figure 7.16: In LD+SW, sets were more suitable for exploration, and more manageable.

on the sets in the menu; however, this was more error-prone due to the missing physical guidance. As a result, this forced attention switches between set navigation and visual comparison and required some participants to repeat the interaction multiple times to develop their answers. For instance, one participant (P10; worked with LD+SW first) answered a comparison task (QT3, three sets on two targets) by rotating the bezel between the sets twice for each target, while he switched between the sets five times for each target to make a similar comparison on LD.

Finally, in the two large display setups (84-inch vs. 55-inch), the workflows differed slightly regarding the extent of physical navigation (stepping back) and distant interaction (F2, F3), while the answers given by the participants were similar.

7.3.3.4 Qualitative Feedback

After each session, participants rated the two conditions on a Likert scale from 1 to 5 for two groups of metrics: (1) the overall efficiency, ease of use, and utility, as well as (2) suitability of the devices for set-based tasks and the intuitiveness of the

specific interaction designs. Participants rated both conditions to be similar in efficiency, ease of use, and utility for visual exploration. This was expected as the LD condition supported equivalent operations to the LD+SW. The one negative rating of LD+SW was due to the perceived increase in interaction cost with an additional device. For remaining questions, participants found the LD+SW condition to be more suited for set creation and management, and the interactions on LD+SW to be more intuitive. In Figure 7.16, this pattern is visible with more participants strongly agreeing to these questions in case of LD+SW. As P6 says, "*The interactions correspond to the [cognitive] actions: pull reads data in, and preview/push by activating a focus visualization gives data back.*"

7.4 Summary: Contributions and Next Steps

In a device ecosystem, analysts should take advantage of the shared large displays as well as their portable and personal devices. We found three popular cross-device interaction styles to transfer information from one to the other during analytical activities in a device ecosystem—through contact, by taking a picture, and by pointing/gesturing. Our Visfer technique connected multiple devices for visual analysis by designing a cross-device interaction utilizing the built-in cameras of the devices. Extending this idea further, our David and Goliath technique considered a broader set of cross-device interactions based on the roles of large and small devices. This contributed a conceptual framework for combining large and small devices for a wide range of visual analysis tasks [57, 72]. In a laboratory study, we found that this

conceptual framework is effective for combining a large display with smartwatches. When the roles of devices complement each other, they provide a flexible workflow for the users in the ecosystem and enhance the insights from the data.

In this chapter, we presented specific device combinations with large displays—tablets, smartphones, and smartwatches. Through them, we elicited interaction techniques and designed a conceptual framework for device combinations in visual analytics. This particular framework needs to be evaluated further to understand the full potential of each device modality. For instance, we are missing an understanding of how smartphones and tablets improve visual analysis compared to a smartwatch. Roles of modern input and output modalities—speech, gaze, AR/VR, etc.—should also be reasoned in the future to improve the power of device ecosystems (cf. combining modalities for immersive analytics [18]). We need to also consider diverse analytical activities to enhance the conceptual framework. Beyond exploration, device combinations can be beneficial for activites of storytelling and communication of insights following the visual analysis process.

Finally, the interaction techniques and roles of devices considered here support multiple users to work simultaneously in a device ecosystem. This support is similar to the focus in Chapter 6 to create multi-user interaction in front of large displays. While these concepts are sufficient to target visualization tasks, as research in CSCW shows, multi-user work needs to be coordinated to ensure that the group is working collectively. Beyond multiple devices, Chapter 8, therefore, targets the coordination of multiple users through the notion of group awareness during an analytical activity.

Chapter 8: Coordinating the Activity of Multiple Users in Visual Analysis of Data

How to coordinate multiple users.

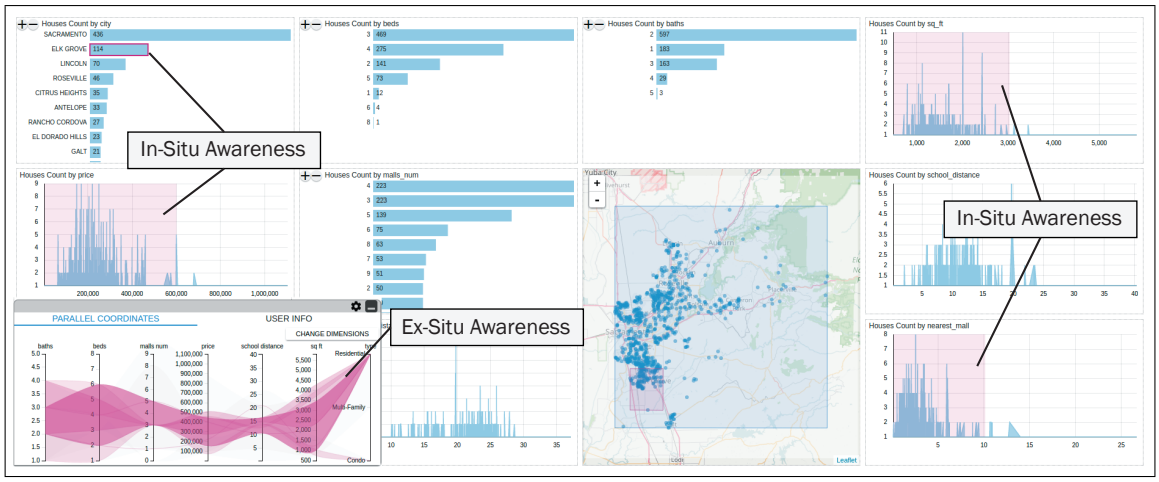


Figure 8.1: Our InsightsDrive tool provides a dashboard of interactive visualizations for real estate data. It supports collaborative visual analytics by combining seamless in-situ highlights (selection shadows in pink) with an ex-situ widget (parallel coordinates) providing group awareness for the team.

Interacting with multiple devices in a device ecosystem is useful to utilize the full potential of each device by the users. To bring forward the potential of every user, we need mechanisms in C2-VA settings to help the users coordinate their insights with the team.

8.1 Design Considerations

Being aware of the group's work in visual analytics allows for subdividing tasks, avoiding conflicts, and improving communication. However, providing complete awareness to each user can be a double-edged sword as the users can be significantly deviated from the actual analytical activity when overloaded with this information. In the presence of multi-device support, our main goal for achieving coordination is to embed and blend the awareness information within the visual analytics interface such that the users can perceive the group activity without requiring significant cognitive effort. In this section, we present the design space for group awareness, and then provide guidelines for effective awareness integration.

8.1.1 Presence and Attention

The digital *presence* of collaborators represents their interest solely based on their proximity. This can reduce conflicts during group activity. Knowing where the collaborators' attention is focused allows team members to understand their tasks and their interactions with the data. For example, Laufer et al. in Prezi Meeting [179] use avatars to represent the position and attention of collaborators within a presentation. In visual analytics, presence and attention have been explored using multiple techniques. Previous approaches allowed users to explicitly switch to see others' views, or show data items that are common to other collaborators' analyses [181, 197] to understand their focus and attention.

8.1.2 Analysis Coverage and History

The concept of *analysis coverage* [199] captures which parts of the data that a team is actively viewing or has viewed in the past (history). While it is not necessary that a team views the entire dataset, and further, it is not given that viewing data automatically yields all insights from it, it is still a useful metric on the completeness of a collaborative analysis. We distinguish three types of coverage:

- **Attribute Coverage:** The attributes that are currently being considered by the analyst. For example, if an analyst is viewing a bar chart capturing number of sports cars, sedans, coupes, and wagons in a cars dataset, he might select the bar containing sedans to filter the other views in the interface. The attribute coverage of the dataset would then be "car type."
- **Range Coverage:** The range of attribute values being examined. For example, suppose an analyst selects five cars of interest by filtering a specific range of values for gas and mileage attributes on the interface. These ranges would be considered as the range coverage. Range coverage is connected directly to the user interactions—e.g., by brushing the axes or the visualization substrate.
- **Feature Coverage:** The connections between different dimensions of data being examined. For example, say an analyst is exploring sports cars with a high top speed. Feature coverage relates to providing information about the interesting connections between other attributes including cylinders, year of release, gas mileage, and horse power. One such connection can be that a lot of sports cars have a poor gas mileage.

Depending the context of the collaboration (synchronous or asynchronous), the coverage information can reflect real-time exploration or history of states explored by the users. Providing a detailed history complements real time coverage by providing additional support for distributed asynchronous collaborations.

8.1.3 Self and Group Awareness

Users of an awareness visualization should be able to see their own coverage across the dataset as well as that of collaborators. Such group and self awareness allows users to adjust their analyses by identifying gaps in the data that has been explored or by identifying particularly interesting subsets of the data that are being examined by many collaborators. Another important aspect vital to supporting presence and attention for self and group awareness is notification. Once coverage has been ascertained, users need to be alerted to important changes in the analysis. In many cases, awareness can be effectively provided on demand, while other analysis scenarios may require more active forms of notification. For example, in the case of an expert on European cars, the user may want to receive notifications if other analysts' coverage extends to or focuses on European cars, eliminating the need for passive observation.

8.1.4 Communication and Deixis

Communication is a key part of effective collaborations, allowing team members to coordinate tasks and share insights. In an awareness visualization, communi-

cation can be facilitated directly using mechanisms such as textual, audio, or video chat. Another important aspect is supporting *deixis* [58]—essentially, the ability to point at elements of reference—to promote effective communication. For example, *collaborative brushing* [131] highlights selections made by a user on all of the remote displays for the entire team.

8.1.5 Presenting Group Activity

Presenting the group awareness in a VA interface—containing visualizations of a dataset of interest—based on the above categories quickly becomes a binary choice: should the awareness representation be separate (*ex-situ*) from the primary visualizations, or should it be integrated into (*in-situ*) said visualizations?

8.1.5.1 Ex-Situ Representation

Ex-situ group awareness visualizations provide a separate view that captures presence, coverage, and communication aspects. For example, the group awareness representations introduced by Sarvghad and Tory [199]—circular dimension co-mapping and treemap designs—are ex-situ as they are presented in a separate view from the actual data visualizations. Ex-situ representations minimize clutter, because the view is separated from the primary visualization interface and the visual encoding can thus be designed freely. However, adding a new view requires splitting the user's attention and introduces a risk of change blindness.

8.1.5.2 In-Situ Representation

In-situ representations for group awareness are blended into the primary visualization interface that may contain multiple visual representations of a dataset. In this case, the group activity information can be either directly overlaid on the content of a data visualization within the VA interface resembling a shadow, or directly attached to a target visualization within the interface resembling a scented widget [195]. This is meant to capture the collaborators' selections and interactions.

For both techniques, users can be distinguished through colors and labelling. These in-situ representations can make analysts be aware of what other team members are doing without having to divert their attention away from the main visualization window. However, information conveyed by these representations is limited compared to an ex-situ representation, which has its own dedicated space.

8.1.6 Design Guidelines

A group awareness approach should provide presence, attention, and coverage within the VA interface without deviating the users from their activity.

- G1 Adapt the group awareness representation to the sensemaking scenario—target dataset and collaboration style (async/sync. and distributed/co-located).
- G2 Target glanceable visual representations that convey group activity without heavyweight interactions and context switching.
- G3 Avoid visual clutter within in-situ and ex-situ awareness representations to aid in a quick understanding of group activity.

G4 Support customization of awareness representation, since users may be interested in different aspects of the group activity and have different perceptual capabilities (some can be faster at interpreting visualizations than others).

G5 Target extensible representations that can be applied to different visualization designs—line charts, bar charts, and graphs—to maintain consistency in awareness representation.

8.2 InsightsDrive: A Visualization Dashboard for Coordination

Based on our design space and guidelines, we developed a prototype visual analytics tool called InsightsDrive (Figure 8.1). This tool was developed for multidimensional data with synchronous collaboration in mind; therefore, the group activity representations capture the current focus and selections of users (G1). It is currently most suited for distributed teams of analysts with a flat hierarchy since all the team members have access to the same type of features within the interface.

8.2.1 Interface

The actual visualization interface within our tool (Figure 8.2) contains multiple views, with each view showing a summary for a particular dimension within the dataset as a bar chart, line chart, or map visualization. Each view is interactive and allows selections, and uses brushing and linking to coordinate the other views. To provide a quick understanding of the group activity without cluttering the visual interface, our InsightsDrive combines ex-situ and in-situ representations

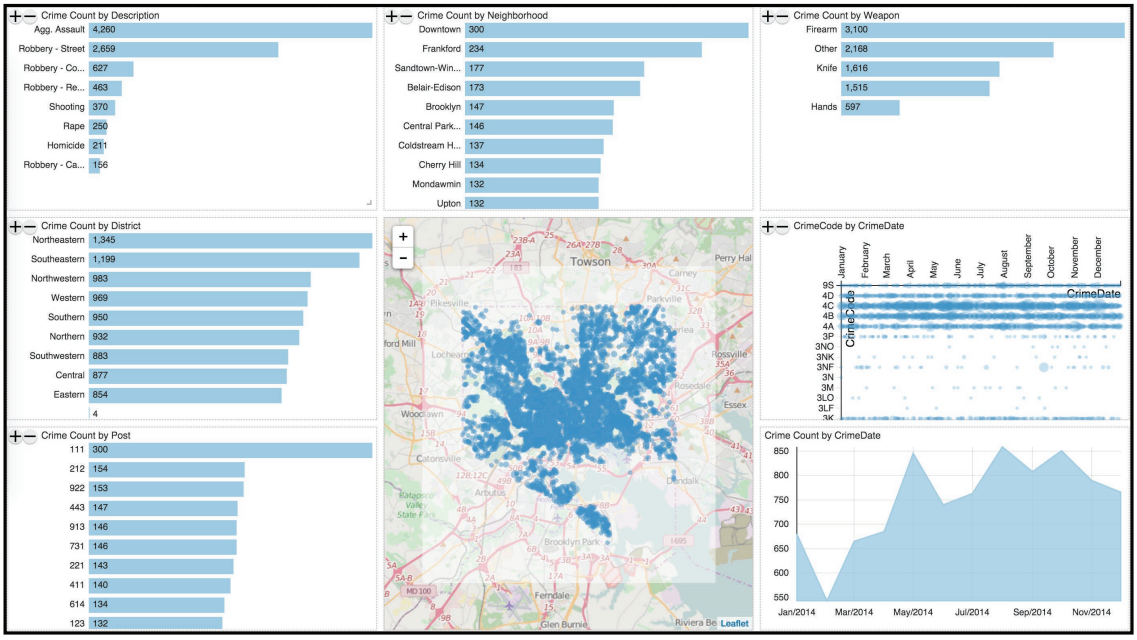


Figure 8.2: The InsightsDrive tool presenting a Baltimore crime dataset. By default, it shows bar charts for all the categorical variables, a line chart for visualizing the temporal component, and a map to capture the geographic data. Clicking the '+' button on top of each view allows for adding an extra dimension to the view to perform 2D analysis.

to automatically capture presence, attention, and coverage of the collaborators (in a glanceable way while minimizing context switching from the actual activity) and support further exploration and customization (G2, G4).

8.2.2 Ex-Situ Representation

We use a separate interface widget to provide ex-situ awareness that unobtrusively docks to the main visualization window. This widget is collapsible (Figure 8.1) and uses limited interface space (G3). Since we target general multidimensional data, we have two visualization designs—parallel coordinates plot and scatterplot—

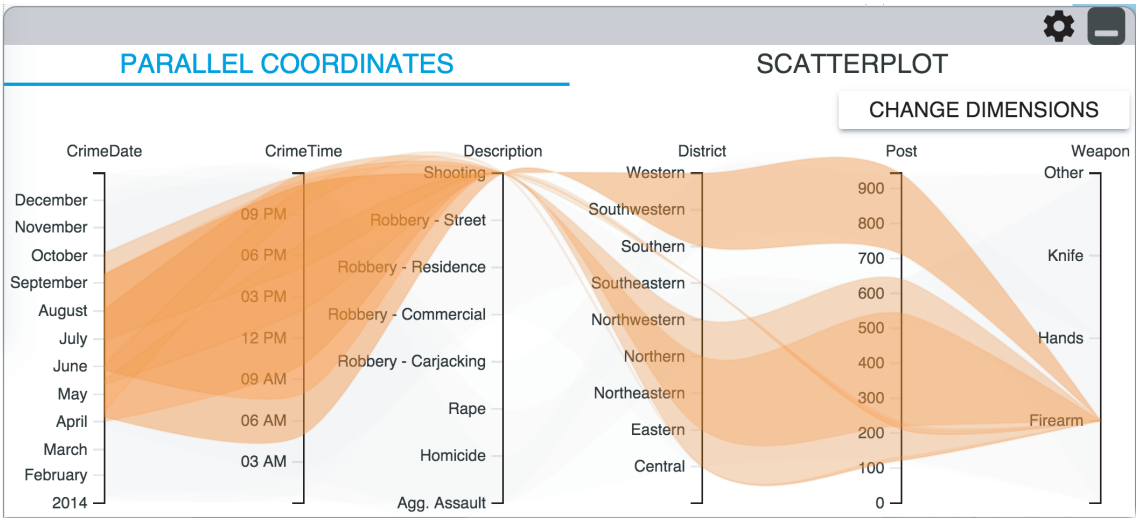


Figure 8.3: Parallel coordinates capturing coverage in a ex-situ widget in InsightDrive. To avoid clutter, the covered data points of each collaborator are clustered using hierarchical clustering into bands.

to show the team's presence and analysis coverage within this ex-situ widget (further studied in Section 8.3). These particular representations can apply to any multi-dimensional dataset and are also extensible (G5). This widget can also provide methods for communication.

8.2.2.1 Ex-Situ: Parallel Coordinates

A parallel coordinates view (Figure 8.3) can represent all of the data points in the dataset and the respective coverage of each team member. We use agglomerative clustering to create bands [251] to quickly understand the covered data points (G2), while avoiding clutter (G3). Transparency of each band encodes the fraction of the total number of data points it contains. Hovering over a band highlights the encoded points on collaborators' interface [131]. Axes can be added, removed, and

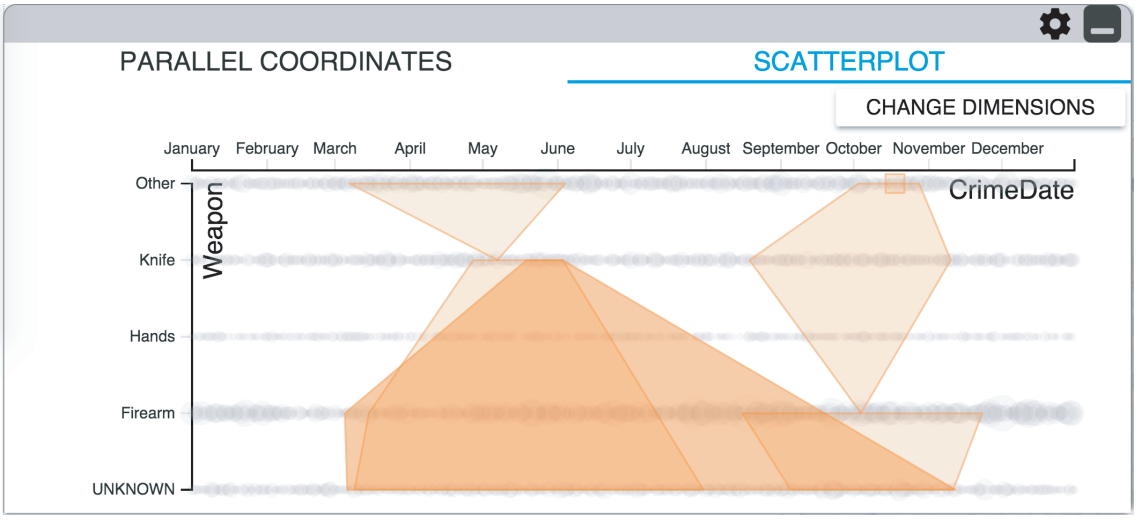


Figure 8.4: Scatterplot awareness on ex-situ widget in InsightsDrive. Two dimensions are chosen to create a scatterplot. The points viewed by a collaborator are clustered and shown as the regions.

reordered for customization (G4). This parallel coordinates view makes it easy to see sequential selections based on the band transitions (e.g., user first selects a range on dimension X and then dimension Y). However, showing coverage by aggregation comes at a cost as individual point-level information is lost.

8.2.2.2 Ex-Situ: Scatterplot

While a scatterplot matrix can provide an overview of the data on all dimensions, SPLOMs yield high clutter and require significant display space. As an alternative, we use a single scatterplot with editable axes to make the awareness widget compact (Figure 8.4). Again we use hierarchical clustering to visualize clusters of covered points in the scatterplot as two-dimensional regions.

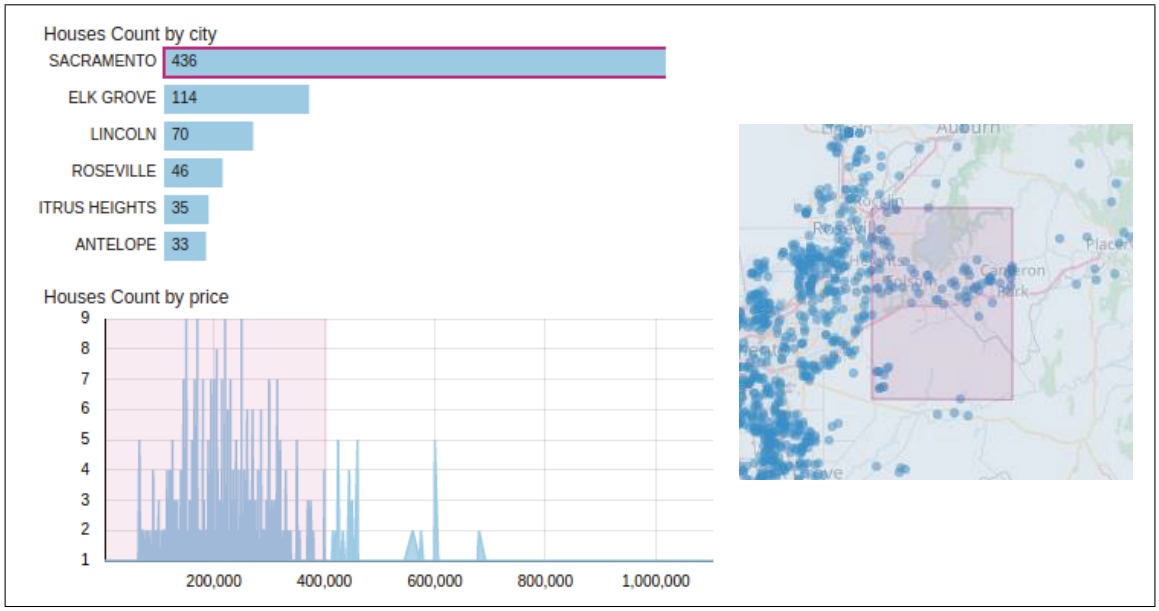


Figure 8.5: Selection shadows for in-situ awareness. Based on collaborative brushing [131], shadows show selections that collaborators have made as color-coded shapes in the background of each view.

8.2.3 In-Situ Representation: Selection Shadows

We visualize the selections made by all other team members as "shadows" (Figure 8.5) in the background of each individual visualization (G2, G3). These selection shadows are coded with a unique color and label assigned to each collaborator. Shadows are adapted to the underlying visualization—appearing as borders to bars in bar charts and as colored regions in line charts and maps (G5).

8.3 Formative Evaluation: Utility of Ex-Situ Awareness

We conducted an exploratory user study to compare the utility of our ex-situ representations—parallel coordinates and scatterplots—and their affordances.

8.3.1 Study Design

Participants. We recruited 6 participants (1 female, 5 male) between the ages of 18 and 45 from the student population within our university campus. They were paid $10 for participation. All participants self-reported as proficient computer users and as experienced with using visualizations for data analysis.

Dataset. We used a Baltimore crime dataset[1] that contains 11 attributes including date, time, location, description, and weapons used. We picked this dataset to enable investigative sensemaking by using questions related to trends and anomalies. Sessions were held in a lab setting using the InsightsDrive tool on a Google Chrome browser of a Macbook Pro (15-inch display; 1440×900 resolution).

Tasks and Protocol. Each task consisted of the participant following the awareness visualization (either parallel coordinates plot or scatterplot) while a VA expert (the study investigator) answered a question about the dataset. The participants were asked the speak out the observations (think-aloud protocol) they make from the awareness representation as the expert interacts with the interface to figure out the answer. The participants worked on eight tasks in the experiment: four with parallel coordinates and four with scatterplot (order counterbalanced across participants in the study). The motivation behind this methodology was to verify to what extent the participants can follow the activity of their collaborator in terms of presence and analysis coverage (attribute, range, and feature) just by viewing the ex-situ awareness representations. For this reason, the participants did not interact

[1] https://data.baltimorecity.gov/

with the interface or the investigator directly during the experiment session. The candidate question list used for the tasks was generated by two VA experts using InsightsDrive. It consisted of questions related to four high-level visual analytics tasks: specific value identification, trend identification, extrema detection, and comparison of two data items. The list includes questions such as,

- What is the most common weapon used in April?
- In what neighborhoods do most shootings occur?
- During what time of the day did assaults with a firearm most happen in the central district?
- What do crimes happening in Downtown in the Spring and Fall seasons have in common?

Procedure. Participants first went through a training procedure where the assigned awareness representation (parallel coordinates or scatterplot) was demonstrated. They then proceeded with the tasks, and later repeated the process with the other awareness representation. Finally, they named their preferred awareness visualization. Each session lasted for less than 40 minutes.

The participants were asked to think-out-aloud. Screen and audio recordings were captured for participants' answers as well as comments during the session.

8.3.2 Results and Observations

Here, we report the observations made based on how participants used the visualizations on our ex-situ widget in InsightsDrive.

8.3.2.1 Parallel Coordinates

All participants were able to easily identify which dimensions have been selected by just looking at the intersections of the bands with the axis of the parallel coordinates. The **attribute coverage** was the primary visual feature the participants observed after every expert interaction due to the change in shape of the bands (P3 described it as, "*when a dimension is selected, it appears like the free-flowing bands [that cover the entire space] are tied to a specific range [on a dimension]*"). All participants followed their observation of the attribute coverage with an observation of a **range coverage** aspect almost immediately. Typically, this was about the coverage over the date, time, district, crime description, and weapon dimensions.

Participants P3, P4, and P5 made complex observations related to **feature coverage** (Figure 8.6). For example, when the expert was viewing the street robberies, P4 remarked that there are a lot of crimes in the Southeastern and Central districts that happen after 9am in the morning. This specific feature is apparent due to our clustering approach. Beyond this, the participants could also sense the presence and attention of the collaborator based on the changes. However, a potential drawback (P2 and P6) was that the dimensional ordering in the parallel coordinates affected the perception of coverage. Overall, participants made more observations from parallel coordinates (2-4 per task) than scatterplot (1-2 per task).

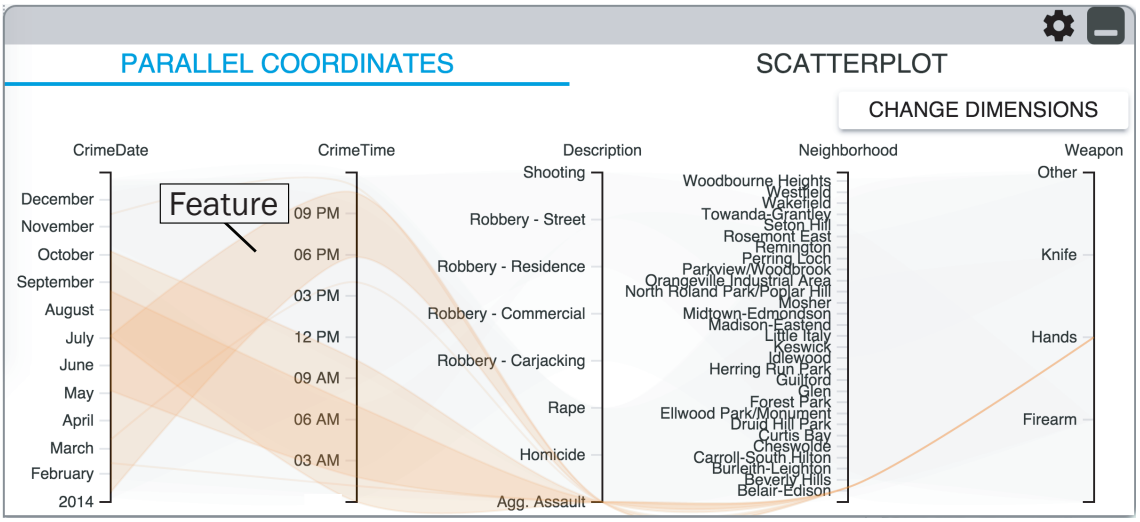


Figure 8.6: Example feature from the dataset covering Assaults with Hands in the Bel-air Edison neighborhood. It appears that crimes happening after 6pm mostly occurred in the first half of the year.

8.3.2.2 Scatterplot

Participants typically took longer to interpret the scatterplot visualization due to the inherent need to switch dimensions to get a complete perspective of the coverage. This was expected from the use of a scatterplot as it can only capture coverage on two dimensions at once. Participants in this scenario focused on the range coverage (all) and feature coverage (P3, P5, P6). For example, when the expert was viewing crimes happening in the Fall months, P5 remarked that "*crimes are [evenly] distributed on the weapons dimension, but knife is more commonly used during September to December, while firearms for August to October.*" We observed that the process of understanding the awareness on scatterplots can be viewed as the opposite of parallel coordinates. In parallel coordinates, the participants interpret

the coverage top-down (e.g., by first examining attribute coverage, then examining more specific details about the data space if possible). In contrast, they try to comprehend scatterplots bottom-up (e.g., by looking at individual data points first). This is because the participants had to look at 2D distributions and also explicitly switch between scatterplot dimensions, and therefore they made (range and feature) observations about the data on the current two dimensions first.

8.3.2.3 Subjective Feedback

All participants preferred parallel coordinates for the ex-situ group activity widget because, (1) it was harder to interpret clusters in scatterplots than bands on parallel coordinates, and (2) scatterplots require switching between dimensions.

8.4 Summative Evaluation: Ex-Situ vs. Combination

InsightsDrive provides both in-situ shadows and ex-situ coverage widget as a balanced way to provide awareness. We were interested in observing the tradeoffs of the combination of in-situ and ex-situ over just ex-situ awareness on time and accuracy measures, when a team of analysts (participants) try to solve a practical visual analytics task involving decision making. Note that just having in-situ awareness by itself is not ideal for capturing presence and providing complete coverage on the dimensions since this can inundate each view with shadows and highlights based on the group activity, making it hard to follow. Hence this condition is not considered in the user study. Also, based on the previous study, we decided to use only the

parallel coordinates plot for the ex-situ widget as it was the preferred visualization and led to more observations about the attribute, range, and feature coverage.

8.4.1 Study Design

Participants. We recruited 20 participants (6 female, 14 male) between the ages of 18 and 45 from the student population within our university campus. They were paid $10 for participation. All participants self-reported as proficient computer users and 18 of them had previously used visualization for data analysis. Participants were grouped into 10 teams based on their availability for the study. Participants in 9 teams knew each other, but only participants in one team worked with each other in a professional situation before.

Experimental Factors. The awareness technique (T) and the task type (Q) are the factors influencing the group performance. For the awareness technique, we tested two conditions:

- EX+IN: This involved using the InsightsDrive multi-dimensional dashboard for the real estate dataset with both awareness techniques: in-situ shadows and ex-situ widget (Figure 8.1).
- EX: Only the ex-situ widget with parallel coordinates was used to gain a complete awareness of the group activity.

The order of tasks and conditions was counterbalanced.

Dataset and Apparatus. We used a simulated real estate dataset with 10 attributes including address, bedrooms, bathrooms, size, and price, as well as dis-

tances from closest school, shopping mall, university, and golf course. This dataset helped us develop simple relatable tasks that can be controlled for the study purposes. Participants worked in a lab setting similar to the previous study. During the user study session, participants sat opposite to each other without being able to see each other's displays. Beyond following the awareness representations, communication through speech was the only means for them to consolidate their work during the tasks. This choice replicates a distributed collaboration scenario.

Tasks and Protocol. We used decision making tasks (four types) about real estate (house) search for the participants in our study. Each of these tasks involved giving a specific set of constraints (e.g., within 2 miles from a school) to each participant in a group and asking them to interact based on the constraints and coordinate with their collaborators to find the best choice.

- **Task 1 (T1):** Here, only one house in the dataset satisfies the constraints given to the participants. The participants would have to make appropriate selections based on their constraints and use the awareness visualizations to understand their collaborators' constraints. They then find the candidate houses on their interface based on their awareness of the group activity, discuss them with their collaborator, and pick a house.
- **Task 2 (T2):** There are multiple houses satisfying the constraints. The participants follow a similar procedure as Task 1, but now they need to consolidate and pick one final house among the satisfying ones. We were interested in seeing their performance in coming to consensus.
- **Task 3 (T3):** There is no house satisfying the constraints in this task. There-

fore, the participants need to negotiate to reach a compromise on some constraints to make a decision.

- **Task 4 (T4):** This task is similar to Task 3, but the participants are now aware of all the constraints (from their group).

  Example constraints include,

- Find a house within $200,000 price.
- Find a house within 5 miles from the closest school.
- Find a house with more than 2 bedrooms.

Participants worked on a total of eight tasks during the study: four (one per task type) with in-situ and ex-situ combination, and four with just the ex-situ awareness. For each task, groups of two participants worked as a team, along with a VA expert (the study administrator). The expert user added one more constraint to the task while encouraging the other participants to talk to each other. The expert user did not participate in the discussion between the two participants. This is a variant of the pair analytics protocol [252], modified for collaborative studies, giving the study administrator unfettered insight into the collaborative work. The time taken during each task from introducing the constraints to reaching a final consensus was measured. This represents the speed at which the participants become aware of the group activity and consolidate with the other, and thus captures the collaboration dynamics to an extent within this controlled setting for teams of two participants. The answers were also analyzed to evaluate their accuracy.

Procedure. Participants first trained with the assigned visual analytics interface by demonstrating the visualizations, interactions, and awareness representa-

tions. They were given a set of training questions to answer and could return to the training again if needed. Following this, they worked on the four tasks with their group. They then moved on to the second awareness condition and repeated the same procedure. At the end of the session, they individually filled a questionnaire providing feedback about the perceived usability of the awareness representations for solving the tasks. The participants' comments and answers were audio recorded. Each session lasted for less than one hour.

8.4.2 Hypotheses

H1: Participants will be faster when both in-situ and ex-situ awareness is provided, since it can balance the participant attention between the actual interface (in-situ) and ex-situ components.

H2: Participants will be more accurate when both forms of awareness are provided as this may give a high-fidelity awareness. The in-situ representation in EX+IN captures the user interaction on the VA dashboard and can ensure that the collaborator does not miss any group activity due to split attention.

8.4.3 Results

Here we report the results from the statistical analysis of the time and accuracy measures collected during the sessions.

8.4.3.1 Time

We first analyzed the time taken by the participants to solve the tasks for the two techniques and the four tasks using repeated-measures analysis of variance (Table 8.1). The technique had a significant effect, but an interaction between task and technique was also found to be significant. The combination of in-situ and ex-situ awareness (EX+IN) (M = 139 sec, SD = 79 sec) was faster than the ex-situ only condition (EX) (M = 207 sec, SD = 75 sec).

Table 8.1: Effects of technique (T) and task (Q) on time (repeated-measures ANOVA—all assumptions satisfied).

Factors	**df, den**	**F**	**p**
Awareness technique (T)	1, 80	23.83	<.001
Task type (Q)	3, 80	3.09	.033
T * Q	3, 80	9.71	<.001

We then analyzed the individual differences between the techniques for each task using paired T-tests. We found that the technique factor led to a significant difference in time for tasks T1 ($t(9) = 3.79$, $p = .004$), T2 ($t(9) = 3.00$, $p = .015$), and T3 ($t(9) = 4.49$, $p = .002$). For these tasks, the in-situ and ex-situ combination led to better performance (Figure 8.7). This **confirms** hypothesis **H1** for tasks T1, T2, and T3.

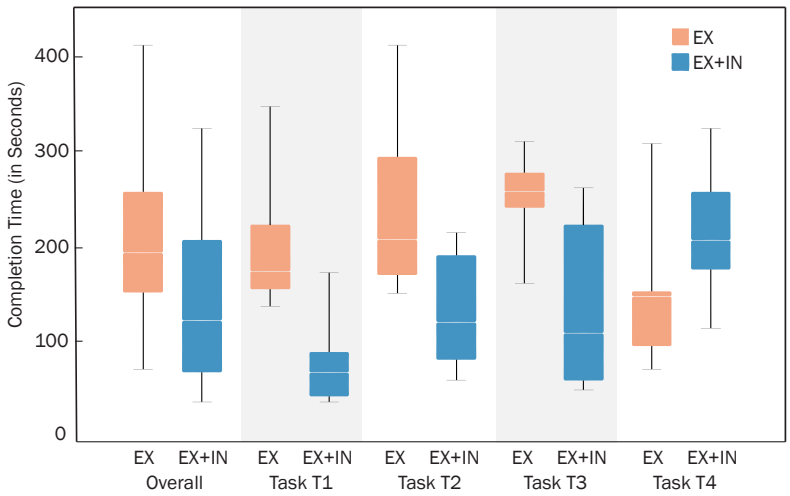


Figure 8.7: Differences between the task completion times. Statistical analyses revealed that having both forms of awareness (EX+IN) was faster than just ex-situ (EX) for tasks T1, T2, and T3.

8.4.3.2 Accuracy (Distance)

Accuracy meant different things across the four tasks. For tasks T1 and T2, accuracy was the correctness of the decision made (whether the final house selected satisfied the constraints). All groups responded to these tasks correctly by picking the house that satisfies the constraints. Therefore, there was no difference across conditions for these tasks.

For T3 and T4, which do not have a correct answer, accuracy is based on the concession distance that defines how closely the selected house matched the constraints (similar to the one used by McGrath et al. [148]). This concession distance is defined as the normalized euclidean distance between the selected house and the boundaries of the collective constraints given to the group. For instance, for price

range $\leq$ \$200,000 constraint, the boundary on the price attribute is \$200,000. During computation of this normalized distance, the attribute distances between the selected house and the constraints are scaled down by the overall range of the particular attribute in the entire dataset. For this reason, attributes in the dataset with higher values in general (e.g., price compared to distance) still have the same influence over the distance measure as others. Repeated-measures analysis of variance applied to this measure revealed significant differences across two techniques based on the effects shown in Table 8.2.

Table 8.2: Effects of technique (T) and task (Q) on distance (accuracy) (repeated-measures ANOVA—all assumptions satisfied).

Factors	**df, den**	**F**	**p**
Awareness technique (T)	1, 40	6.14	.020
Task type (Q)	1, 40	5.87	.022
T * Q	1, 40	16.07	$< .001$

Paired T-tests applied to the distance for the individual tasks revealed significant differences only for T4 ($t(9) = -7.73$, $p < .001$). In T4, the normalized distance was higher for the ex-situ + in-situ condition (M = .40, SD = .07) than just ex-situ (M = .22, SD = .06). The differences were not significant for T3. Hypothesis **H2** is therefore **not confirmed**.

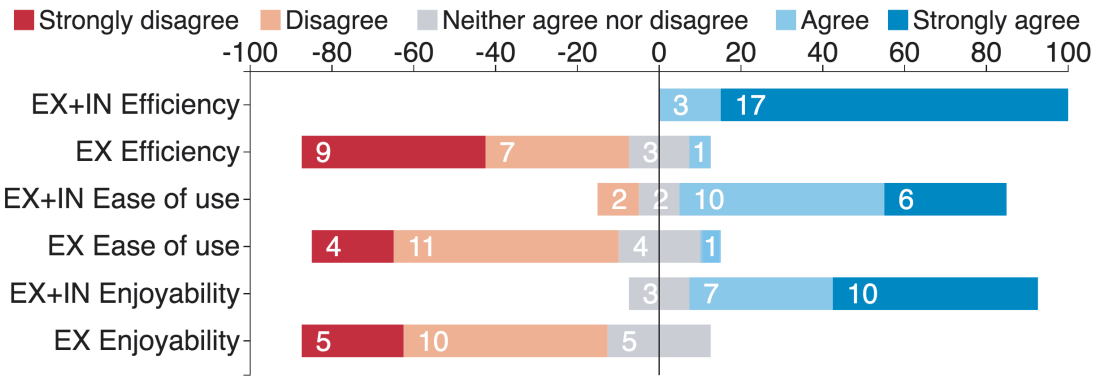


Figure 8.8: Differences between the condition with both forms of awareness (EX+IN) and ex-situ awareness only (EX) in terms of the Likert-scale ratings. Each bar in this chart captures the number of participants who gave the corresponding rating.

8.4.3.3 Subjective Ratings

The participants rated the awareness techniques on separate 5-point Likert scales for efficiency, ease of use, and enjoyability. This was analyzed using non-parametric Friedman tests and significant differences were found for all three scales (significance level: $p < .001$). As evidenced in Figure 8.8, having both forms of awareness (EX+IN) was perceived to be more efficient, easy to use, and enjoyable than just ex-situ (EX). Almost all participants agreed to these questions for the condition with both forms of awareness, while disagreeing in case of ex-situ (EX) condition (Figure 8.8). Note that this questionnaire was given after the tasks on both awareness conditions were completed, so the responses are comparing the ex-situ technique to the combination of the ex-situ and in-situ techniques.

8.5 Summary: Contributions and Next Steps

In a device ecosystem, multiple users will need help coordinating their work when immersed in an analytical activity across their devices. Beyond helping users work with the devices, mechanisms need to be designed to communicate and understand each collaborator's work in a team. In synchronous collaboration, this can be done through **in-situ** highlights to the user interface that show the data selected by each user, and **ex-situ** visualizations that represent each user's focus in the context of the entire dataset (cf. data coverage [199]). These awareness techniques were used in our visualization dashboard for real estate property search. In a laboratory study done with groups of two users, we found the effectiveness of the combination of in-situ and ex-situ awareness for a collaborative task of searching for a house that is ideal for both members of the group. Users in each group were better aware of their collaborator's search activity and were able to come to a faster consensus about the ideal house.

This chapter does not just provide the findings from the specific activity of property search. It contributes general design considerations for coordination including concepts from the field of CSCW such as presence and attention, communication, deixis, and group awareness, along with visualization-specific ideas (cf. analytical coverage [199]). These considerations also apply to other collaboration settings—asynchronous or co-located/distributed collaboration. However, we need optimal choices for specific considerations in those settings. For instance, to support asynchronous collaboration, history mechanisms need to be provided to see

the complete historical coverage of the team. Having said this, the major limitation of this research is the reliance on symmetric collaboration style—where all the users contribute at the same level to a common goal. Future research can target asymmetry—for instance, where one user focuses on finding the best house while the other user audits the analytical activity to ensure that the data and the insights are truthful. Generalizing these considerations to large groups is also essential.

Chapters 5-8 have been focused on individual components of the C2-VA stack. This isolation was ideal for focusing on specific challenges and providing detailed guidelines for designers of future C2-VA settings. For instance, from this chapter, designers could gain insights into the considerations towards team coordination in C2-VA. To drive the science of C2-VA, we need more than these individual and isolated applications. There is a need for platforms that help create new C2-VA applications while learning and applying the guidelines from previous efforts. Chapter 9 introduces our contribution to building such a platform for C2-VA applications.

Chapter 9: A Component Model for Multiple Analytical Activities

How to develop new C2-VA applications.

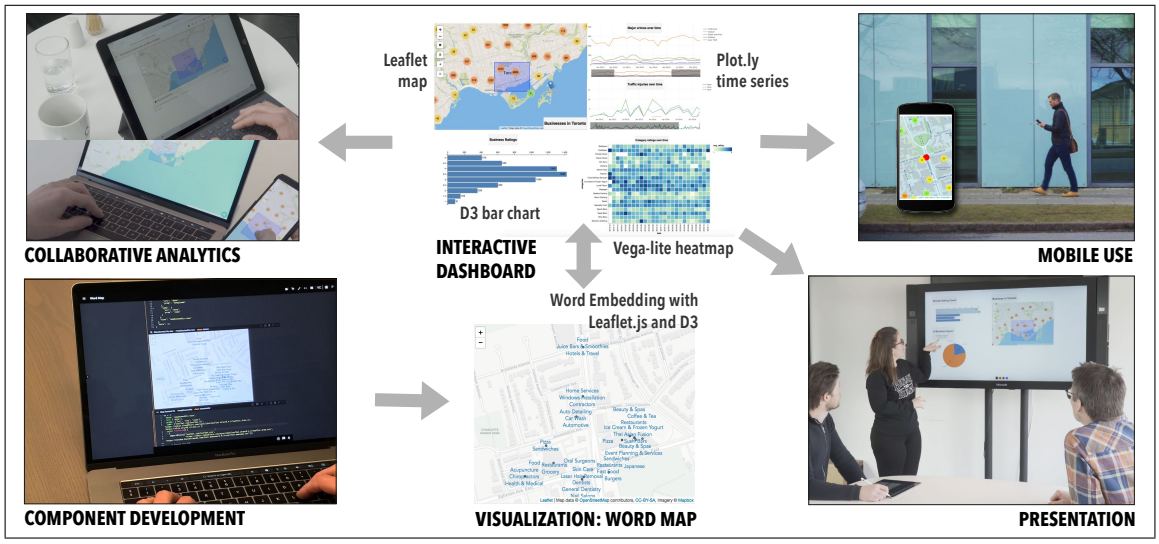


Figure 9.1: Vistrates transcend the traditional tool boundaries of analysis activities, analyst expertise, input and output devices, and modes of collaboration to enable a C2-VA workflow. Each vistrate (center) is a shareable dynamic media [253] where components encapsulating existing web technologies—such as D3 [15], Vega-Lite [207], Leaflet, and Plot.ly—can be made to interoperate seamlessly. The document can be accessed from multiple settings, using heterogeneous devices, and by multiple concurrent users in activities ranging from data wrangling and exploration to development and presentation.

The solutions considered so far for designing interactions and coordinating users in device ecosystems are targeting specific activities, tasks, datasets, and col-

laboration styles. As we saw in Chapter 2, this is not the case in practice. Many modern analytical settings support a wide range of analytical activities from exploration to communication of insights. To enable the development of collaborative, cross-device analytical tools for multiple analytical activities, we created Vistrates, a web-based collaborative platform that allows development of applications for (1) both single-user and collaborative work, (2) the full spectrum of data analysis activities, (3) all levels of user expertise, and (4) a menagerie of devices (Figure 9.1). As such, Vistrates is the culmination of the research efforts presented so far and targets holistic support for C2-VA.

To achieve this vision, Vistrates applies the concept of *shareable dynamic media* [253] as well as recent advances in conceptualizing and implementing software as *information substrates* [254] to the field of data analysis. Information substrates blur the traditional distinction between *application* and *document*, they embody content, computation and interaction, and can evolve and be repurposed over time. Vistrates uses a **component model** for assembling visualization and analytics pipelines into such information substrates. In this model, a component is a unit module with internal state, inputs, and outputs. Components provide visual analytics functionality and are reusable, replaceable, and extendable. This allows them to become building blocks for data analysis systems. Following the philosophy of information substrates, these systems can be integrated into media such as slideshows, interactive whiteboard canvases, reports, or interactive applications; thus supporting a wide range of analytical tasks. Figure 9.1 captures an example application built in the Vistrates platform for a business dataset from Yelp.

9.1 Motivating Scenario

Vergil is an experienced freelance travel writer. He has been commissioned by a new internet-based travel guide company called "TraLuver" that is trying to "disrupt" the travel guide industry by providing customized travel plans for their clients. Their business idea is to use data science to find an optimal match. TraLuver is rolling out their service to a select few North American cities, and Vergil has been tasked with curating and preparing the dataset for Toronto, Canada.

Prior to starting his field work, Vergil uses his laptop to familiarize himself with the TraLuver platform, which is built on top of a Vistrates installation. Vergil is not a data scientist, so he connects with Daria, an analyst in the data science team at TraLuver's headquarters. Using videoconferencing and a single vistrate document, Daria takes Vergil on a tour of the basic datasets available including Yelp! businesses and reviews and open data provided by the City of Toronto. She constructs a simple visualization interface, where a map of businesses in Toronto can be filtered to see their ratings in a bar chart, by putting together available components in the vistrate's graphical interface without any programming. Since Vergil knows he will be restricted to mobile devices when he is out in the field, he installs a mobile view following Daria's example, which shows one of the available views at a time.

After learning the system, Vergil heads out to the 553-meter CN Tower, a significant landmark of the city. He installs a GPS component to center the map in the vistrate to his location. This helps him visit surrounding restaurants and access

their reviews on the TraLuver platform. He realizes that the single map view would be more useful if it also incorporated relevant review keywords. He pulls out his tablet, sketches this idea, and discusses it with Daria, who provides a temporary solution by adding a simple word cloud. Daria gets in touch with Sam, a developer in TraLuver, who starts building the new component in a copy of Vergil's vistrate. After this, Sam calls Vergil and Daria and adds the new Word Map component to their vistrate.

A month later, after hard work by Vergil and his TraLuver team, Daria can finally present the finished Toronto project to the company's board. Basing her presentation on the same vistrate that she and Vergil created weeks back, she has created a slideshow of multiple canvases that show each feature, dataset, and visualization of the final product.

9.2 Designing Vistrates for Multiple Analytical Activities

Vistrates is a realization of a set of design choices that together form the vision of a holistic and sustainable environment for C2-VA.

Webstrates and Codestrates. Vistrates is built on top of Webstrates and Codestrates. Webstrates [253] is a web framework where webpages are made collaboratively editable in real-time. Changes made to the DOM of a webpage (called a *webstrate*) are persistent and synchronized to all clients of the webstrate. Codestrates [214] is an authoring environment built on top of Webstrates. A codestrate is a webstrate that includes tools for editing its own content, including writing and

executing code, following a literate computing approach similar to interactive notebooks. Individual codestrates contain their implementation, which means they can be reprogrammed from within. A codestrate is structured in sections consisting of paragraphs of code, data, styles, and web content. Sections can be turned into packages of functionality that can be shared between codestrates [255].

The design of Vistrates is rooted in the principles from information substrates (cf. Webstrates [253]) that software should be malleable, shareable, and distributable. Beyond this, it was based on the following considerations:

9.2.1 Component-based Pipeline Architecture

The typical architecture to go from data to visualization is through a visualization pipeline [256]. We propose a component-based architecture, where components (Figure 9.2(a)) are connected together in reconfigurable pipelines (Figure 9.2(b)). A component can be a data source (e.g., serving a file, connecting to a database or API, or providing coordinates from a phone's GPS module), a computation on data (e.g., filtering, aggregating, or analyzing), or a visualization (e.g., a bar chart, scatterplot, heatmap, etc.). Visualizations do not have to be endpoints in the pipeline, but can be interactive and hereby serve as data sources as well. Components should be executable blocks of code with an optional input, output, state, and view (Figure 9.2(d)). The pipeline should be reactive, so when the output of a source component changes, it will trigger updates of components that have the output of the source component as input. Components should adhere to a minimalistic interface

for connecting them together, and what a component does and what third-party software libraries it uses should be up to the developer.

9.2.2 Collaborative Pipeline

It should be possible to modify, run, and interact with components in the pipeline *collaboratively* from different clients. However, for most computations, it is faster and simpler to execute them locally than to distribute data across a potentially high-latency network. We therefore propose a design where each client executes its own instance of a pipeline, but synchronizes state locally to components between them. State includes the configuration of a component and any application state that should be synchronized or persisted, e.g., interactions. An example of the latter could be a rectangular selection on a map-based visualization, or a URL to a data file that a data source component should load into memory. It should be up to the developer to specify what application state is synchronized, allowing, e.g., the developer of the aforementioned map component to specify that selections should be synchronized but not, e.g., zoom levels. Components should synchronize execution between clients, i.e., rerunning a component on one client should trigger reruns on all other clients as well.

In other words, the collaborative pipeline principle consists of (1) a reactive data flow, (2) a shared execution flow, and (3) shared component state between clients of the same vistrate.

9.2.3 Prototype-based Components

Components should be prototype-based, and components should be instantiated by copying from a prototype and configuring the instance into the pipeline. An instance of a component should contain its own implementation and its state. This is a deliberate violation of the software architectural principle to avoid code duplication. However, by having components contain their own code they become directly reprogrammable, allowing a user to reprogram a single component and potentially turn it into a prototype for new components.

9.2.4 Multiple Levels of Abstraction

Users should be able to work on multiple levels of abstraction: from programming components, to configuring components in a pipeline, to creating presentations of the visualizations and to interact with said visualizations. At the lowest level of abstraction, all aspects of components should be manipulable as code. At a higher level of abstraction, components and their pipeline should be reconfigurable in an interactive fashion, allowing for even non-programmers to reconfigure without programming (Figure 9.2(b)). At an even higher level of abstraction, visualizations should be treated as content that can be composed, e.g., in the form of a slide deck, a document, or a dashboard (Figure 9.2(c)).

Collaboration should be possible on each level of abstraction—from writing the code to interacting with the visualizations—but it should also be possible to collaborate on different levels of abstraction at the same time. That is, while one

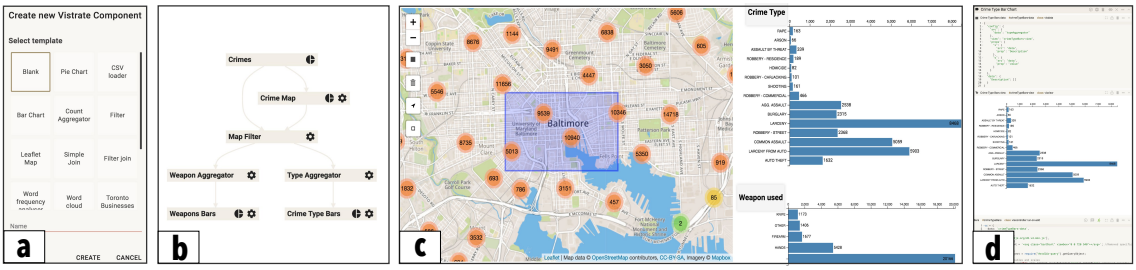


Figure 9.2: (From left to right) **Component instantiation**: Existing component prototypes are available to be edited, thus, promoting reusability and extensibility. **Pipeline view:** This view supports configuration through interaction—drop-down selection—to create visualizations and interactively explore data without programming. In this example, a crime dataset from Baltimore, MD is visualized through a map and bar charts for crime type and weapons by aggregation. A filter component is added to filter the bar charts based on the selection on the map. **Dashboard view**: This vistrate view creates a grid layout for visual exploration of the data and annotation using rich text. **Development view**: The lowest level of abstraction for a vistrate in which a programmer can edit the code and create new visual analytic components.

user is interacting with a component, it should be possible for another user to reprogram it and re-execute it without requiring the first user to restart their client.

9.2.5 Component Repository

It should be possible for a "programming literate" user to develop their own components or redevelop other people's components. However, we also wish for a non-programming-savvy user to be able to construct a visualization pipeline using components made by others. Components should, therefore, be shareable through a common component repository where users can publish their components, as well as retrieve components made by others.

9.2.6 Transcending Application Boundaries

It should be possible to integrate visualizations directly into other media types (e.g., presentations or reports). The components and pipeline should co-exist in an open-ended software ecology with tools not only designed for visualization work.

9.3 Implementation

Vistrates[1] is implemented using standard modern web technologies as well as Codestrates [214] and Webstrates [253]. It uses Codestrates' literate computing and package management [255]. Vistrates consists of a core framework package and individual components implemented as packages (Figure 9.3).

[1]Vistrates: http://vistrates.org

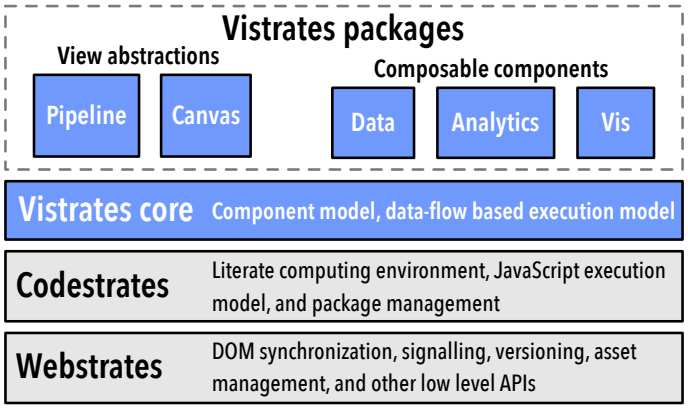


Figure 9.3: Vistrates architecture and relation to Codestrates/Webstrates.

In Codestrates, software is implemented in paragraphs grouped into sections. There are four basic types of paragraphs: *code paragraphs* containing JavaScript that can manually be executed by the user, toggled to run on page load, or be imported into other code paragraphs; *style paragraphs* containing CSS rules; *body paragraphs* that can contain any web content expressible as a DOM subtree; and *data paragraphs* containing JSON formatted data that can manually be edited by users, or programmatically through JavaScript. Every paragraph can be given a human readable name, a unique identifier, and a list of class names. Every paragraph can be toggled to be shown in full-screen either only local to a particular client, or for all clients. Vistrates utilizes these abstractions—paragraphs and sections—and defines an update logic and data flow between them, turning them into building blocks for visualization components and analytical activities.

```
vc = {
  data: "id-of-vis-data",
  src: ["mySourceName_1", ..., "mySourceName_n"],
  props: ["myProp_1", ..., "myProp_m"],
  libs: ["myLibraryStoredAsAsset.js", "https://somecdn.com/anotherLibrary.js"],
  init: function() { /* code goes here */ },
  destroy: function() { /* code goes here */ },
  update: function(source) { /* code goes here */ }
}
```

Listing 9.1: The code paragraph template.

9.3.1 The Core Framework

At its core, the Vistrates framework governs the control flow through component pipelines using the principles of inversion of control and dependency injection of components. The backbone of Vistrates is a singleton that registers all components in a vistrate, a component class that implements an observer pattern for connecting the input and output of components together, and an execution model for executing user-provided component code.

On load, the Vistrates singleton registers all existing components and registers observers between them. When components are updated or new components are created, the singleton also updates observers accordingly. All components have a controller that implements an observer pattern, such that the appropriate components in the pipeline are notified when the output of a component changes. A component consist of three paragraphs grouped in a section: a code paragraph, a data paragraph, and an optional view paragraph (web content paragraph).

The **code paragraph** of a component includes the definition of specific methods and properties following the format shown in Listing 9.1, which include the fields

`data`, `src`, `props`, and `libs` as well as the methods `init`, `destroy`, and `update`. All fields and methods in a component are optional, but defines how a certain component can function within the pipeline. Vistrates uses regular DOM IDs as references, and the `data` property contains the ID of the component's data paragraph containing configuration and stored state. The `src` property defines named sources that can be referred in its methods, and `props` refers to named properties of the sources that can be remapped dynamically through its configuration data. If a component does not have source references, it can only function as an entry node in the pipeline, which is typical for components that load data. `libs` is a list of references to JavaScript libraries for the component. The references can either be URLs to external files, or file names of files uploaded as assets to the webstrate.

The first time the code paragraph is executed, a controller object is instantiated from the controller class, and the properties and methods defined in the code paragraph are evaluated and copied to the controller object. If the controller references anything in `libs`, these are downloaded and evaluated before the `init` method is run. After `init` the `update` method is called, and it is later called any time any of the component's sources update their output. Code paragraphs can be rerun by pressing the play button, and whenever this happens, the previous `destroy` method is executed (to, e.g., remove event listeners) and the newly defined properties and methods are hotswapped on the controller object. Updating the `output` property of a component will trigger the `update` method on any observing components.

The **data paragraph** contains the configuration of the component and the shared state, encoded as JSON. The data paragraph template has the format shown

```
{
config: {
    src: {"mySourceName_1": "source_1_id", ..., "mySourceName_n": "source_n_id"},
    props: {
      "myProp_1": { "src": "mySourceName_1", "prop": "somePropOnSource"},
      ...,
      "myProp_m": { "src": "mySourceName_n", "prop": "someOtherPropOnSource"}
    },
    view: "id-of-vis-view"
  },
  data: { /* The data field of the component for storing state */ }
}
```

Listing 9.2: The data paragraph template.

in Listing 9.2. Besides observing the sources of a component, a Vistrates controller also observes its data paragraph and changes to the configuration will trigger changing dependencies to be hotswapped and changes to the state will trigger the `update` function, and thereby immediately be reflected in the component views on all clients. The chosen configuration of a component includes the mapping between source/property reference names and the actual ids in the vistrate, which allows users to change the mapping on the fly. This format was chosen to be able to reference specific data items in the output of a source component without forcing developers to follow a rigid output convention. As an example, the source with reference name `mySourceName_1` that is currently mapped to source id `source_1_id` in Listing 9.2 can be changed to refer to another source id simply by changing this mapping. Similarly, the property `myProp_1` is mapped to a specific data item in `mySourceName_1`. The configuration also includes a reference to the view paragraph. The shared state of a component can be encoded in the `data` field of the data paragraph, which for instance can be the interaction state of a visualization. Web-

strates will by default synchronize the DOM elements outside `transient` HTML tags, hence the state encoding will also be synchronized across clients. We have chosen an open format for the declarative state specification, which means that it is left to the developer to define this encoding and how to behave accordingly in the `update` method. The data paragraph can be edited by the user, or updated programmatically, e.g., by the pipeline view described below.

The **view paragraph** is a body paragraph containing the visual output of a component. The content can be any standard web content, which is then wrapped in a `transient` element. Transient elements are Webstrates-specific DOM elements that do not have their state synchronized nor persisted. This means that the content of views are not shared across clients. Clients share code and data paragraphs, but clients are executing their own pipeline and thereby creating the content of their own views. This makes it possible to have views where not all interactions are shared between clients. As an example, a map component can share area selections by writing those selections to the data paragraph, while at the same time allowing each user to define their own viewbox and zoom level. In the controller code, the view can be referred through the `view` property and its content can be replaced by setting the `view.content` property either to an HTML string or a DOM node reference, or by referring to the root DOM element of the view using the `view.element` property. Finally, style paragraphs can be added to define the appearance of a vistrate view.

Component updates in Vistrates are triggered in two ways: (1) when the output of a source is updated, or (2) when the configuration or the state in the data paragraph is updated. The cause of an update is encoded in the first argument in the

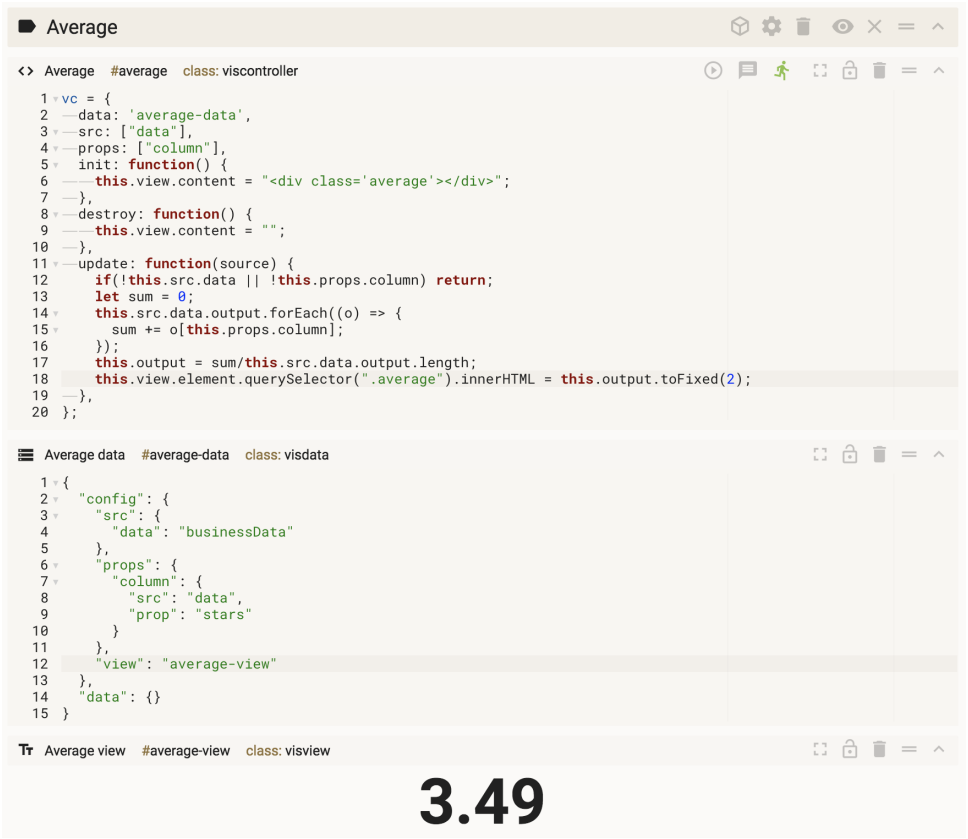


Figure 9.4: A simple Vistrates component that calculates the average of a single numeric data column. The component is easily reconfigurable by changing the mappings in the data paragraph. Other components can observe the output and act accordingly.

`update` method, which can be a specific source from the `src` list, the configuration, or the state. We chose this design as it allows the developer to update a visualization differently based on the type of update; if the data input changes the visualization needs to be redrawn, but if only the interaction state changes the visualization can be updated in a different manner: say, by highlighting specific visual marks. It is possible to create update cycles between components, but it is up to the developer to ensure that these cycles are finite. For instance, such update cycles are currently used to develop coordinated multiple views with brushing-and-linking [226].

In essence, Vistrates adapts these paragraphs to visualization and analytics. In contrast to Codestrates, paragraph definitions in Vistrates have an analytical value—the code captures the underlying logic for processing and visualizing data, the data paragraph captures the declarative specification to map properties to visual variables, and the view contains the analytical outcome of the component. Furthermore, Vistrates explicitly defines the update logic and control flow across components made from these paragraphs. Figure 9.4 shows an example component including controller (code), data, and view paragraphs that calculates the average of a data column and views the result.

9.3.2 The Pipeline

Vistrates components are composable through the configuration specification in the data paragraph. The pipeline view (Figure 9.2) is an abstraction layer on top of the textual specification, which provides graphical access to the configuration and composition of the components in a vistrate. In the pipeline view, components can be reconfigured and recomposed at any time, and changes are immediately reflected in their output, which also triggers updates of connected components. The components' views can be inspected within the pipeline view to immediately observe the effects of a reconfiguration or recomposition. The pipeline view is itself a component that observes the state of the pipeline through an observer installed on the Vistrate singleton. This means that the core of the pipeline view also follows the standard component template with a code paragraph, a data paragraph, and a

view paragraph. The pipeline is easily reprogrammable or replaceable with another abstraction layer. Our current pipeline view is a basic graph implemented in D3 with unfoldable nodes that can either contain the view or the configuration of a component. The pipeline can be accessed in a vistrate through a keyboard shortcut or by pressing the pipeline button in the global toolbar.

9.3.3 The Component Repository

The component repository is implemented using Codestrates' package management features [255]. Prototype components can be pushed to or installed from a repository. New instances of an installed component can be created through the "Create new Vistrate Component" dialog accessible through the global toolbar (Figure 9.2(a)). Instantiating a component will copy the selected prototype, insert the given name and ids, and add it to the vistrate document.

Any component can be turned into a reusable prototype by making it a package and pushing it to the repository. This adds metadata to the component including a short description, a list of assets (e.g., images, JavaScript libraries, or CSS files), dependencies to other packages, and a changelog. This approach allows for reappropriation and customization of existing components. Components in the repository are also ready to use, meaning that an instance can immediately be configured using the pipeline view and, therefore, allows users to create visualization pipelines without programming. The current Vistrates component repository contains components for standard visualizations such as the bar chart, pie chart, line chart, geographi-

cal map, scatterplot, parallel coordinates, etc., components to analyze, transform, combine, and filter data, as well as utility components to load data, spawn headless browsers, and offload heavy parts of the pipeline to strong peers.

9.3.4 Component Canvas

Space is an important cognitive resource; we think and work in physical space [25]. Our implementation of the "Vistrates Component Canvas" package, therefore, allows for such spatial arrangements of component views on a 2D canvas. This can facilitate the sensemaking process to externalize thoughts and for distributed cognition during collaborative work, or it can become a dashboard for interacting with the visualizations (Figure 9.2(c)). In addition, users can add rich text and other media supported by HTML5 and annotate the canvas with a digital pen. Any content on the canvas—including component views—can be moved, scaled, and rotated. When installed, the Vistrates interface has a button in the global toolbar to add a component canvas paragraph. A canvas paragraph is styled to look like a whiteboard.

9.3.5 Mobile List View

In contrast to the component canvas, the "Vistrates Component List View" displays a single (selected) component view at a time. It provides a responsive component container that scales component views according to a device's available screen real-estate, e.g., to fully show a component view on a smartphone (Figure 9.6).

Any available component view can be picked from an action menu. Together with logic for device responsiveness (Appendix A), this view can be very promising.

9.4 Application Examples

We accomplished the motivating scenario (Section 9.1) through Vistrates.[2] Beyond that, we present additional applications that Vistrates makes possible to develop. These are quite unique as they support various aspects of C2-VA and provide evidence of how Vistrates can be used in the future.

9.4.1 Computation Offloading

When all clients of the same vistrate execute their own code, it is possible for weaker peers to offload heavy computations to stronger peers. This can even be an entire subpart of the pipeline, as the example in Figure 9.5 shows, where the highlighted part of the pipeline is offloaded to stronger peers. Two components called *Heavy Start* and *Heavy End* handle the offloading. The start node will signal *help* to other clients if it is executed on a mobile device and pick one of the stronger clients that offers help. The communication between clients is realized using the Webstrates signaling API [257]. The chosen client will then execute their pipeline using the input of the weak client. When the heavy end node is reached, the strong client will provide the weak client with the result, and the weak client can then continue its own execution. This principle also works for multiple heavy start and

[2]Video demo: https://www.youtube.com/watch?v=nMmiWBJoJUc

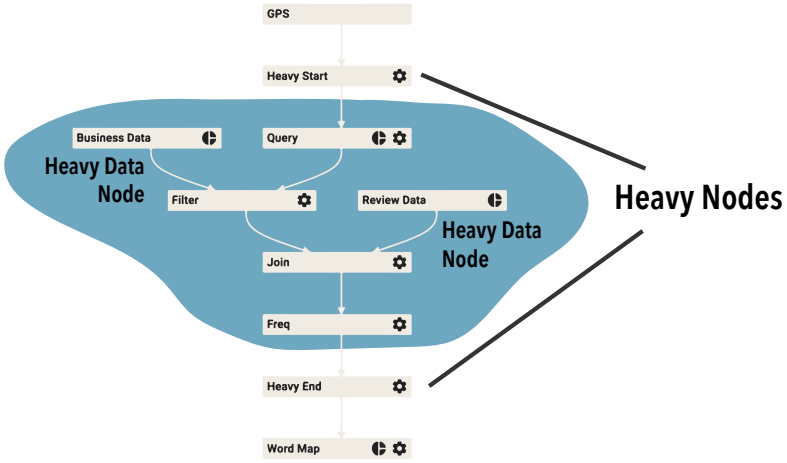


Figure 9.5: A pipeline where *heavy* nodes are used to offload part of the pipeline when the client of a vistrate is a weak peer, e.g., a phone. The large datasets in the marked region—business and reviews data from the scenario—are also not loaded.

end nodes. If no strong peer is available, a service implemented on the Webstrates server can be called through an HTTP request to spawn a headless browser instance pointing to the given vistrate. Beyond this, a *Heavy Data* component is also available to avoid attempting to load large datasets on weaker clients. This way, a client can present interactive visualizations without having to load the dataset.

This approach to computation offloading is implemented purely as new components without any changes to the core of Vistrates. This means that components for different approaches to distributed computing could be created, e.g., to support the kind of peer-to-peer distributed computation provided by VisHive [258].

9.4.2 Cross-Device Visualization

A vistrate can be opened on any device with a web browser. This provides an opportunity to create physical dashboards across multiple devices and for mobile

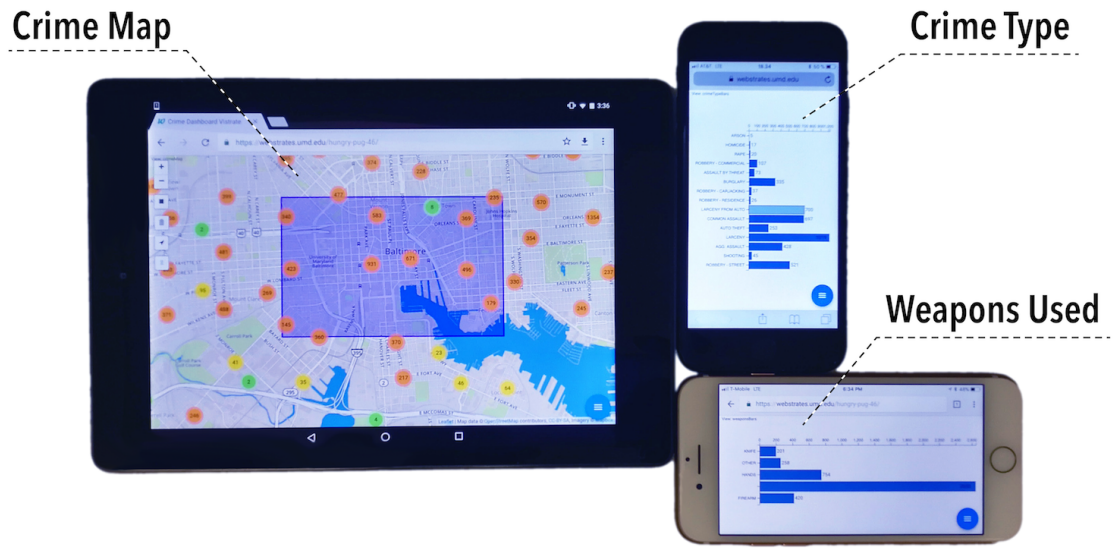


Figure 9.6: A physical dashboard using two phones and a tablet for the crime analysis vistrate in Figure 9.2. The vistrate is opened on the web browser with mobile list view component installed to fit to the screen.

use of vistrates. In Figure 9.6, we showcase a physical dashboard with two mobile phones and a tablet (inspired by VisTiles [10]). This dashboard is created by installing a Mobile List View on the vistrate from Figure 9.2 and selecting a single component view on each device. The phones show visualizations of aggregated crime data—crime type and weapon used—and the tablet show a geographical map of all the crimes in Baltimore. Filtering a view by interaction will trigger updates to synchronize views on other devices owing to the collaborative pipeline of Vistrates. By introducing heavy nodes, the phones never execute any aggregation, filtering, or analysis, but they can show the visualizations and support interaction.

9.4.3 Integrating Visualizations in a Slideshow

As Vistrates is built on top of Codestrates, it is out-of-the-box compatible with, e.g., a slideshow package from Codestrates. This package wraps a view paragraph into a slide as part of a slideshow—including cross-device presenter notes and remote pointing (Figure 9.1). Each slideshow can be styled by creating a theme—a set of stylesheets—that can specify how the visualizations and text appear in the presentation. Visualizations in a vistrate, when wrapped in a slide, are still interactive as their event handlers are retained, and this enables interactive and dynamic presentations when utilizing Vistrates.

9.5 Summary: Contributions and Next Steps

C2-VA work can target a wide range of analytical activities ranging from development and exploration [259] to sensemaking [2] and communication of data. To develop new C2-VA applications and also to utilize device ecosystems as seen earlier, we introduced the Vistrates platform. A vistrate is simply a web document containing executable code, text, and media. Like Google Docs, vistrate documents are inherently collaborative—you can share a URL with your teammate. The user interface of Vistrates is flexible and changes based on analytical activity. During visual exploration, for example, the interface turns into a dashboard (an abstraction) to filter and analyze data across multiple attributes. Similarly, during storytelling, the interface converts into a slideshow along with speaker notes to enable presentations. On a mobile device, the visualizations in the interface adapt to fit the device

characteristics. A vistrate is made up of **components** to achieve this flexibility. A component consists of a controller (written in JavaScript), a state specification (resembles JSON), and a view (in HTML). These components feed each other and together support a complex activity. Therefore, Vistrates acts as the culmination of this dissertation by unifying the components of C2-VA.

Since Vistrates is a platform, it can be used by application developers to create new software tools using existing visualization and data science libraries. To help in adoption, we prototyped simple application examples for device ecosystems in C2-VA using Vistrates, along with a collection of data processing, visualization, and interaction components. We have also maintained these examples online. Vistrates is beginning to be used for new research in C2-VA. It was used to develop an application to research ways to distribute visualizations across devices in an ecosystem [13] automatically. Furthermore, it was also used to create methods to convert an analytical activity into a shareable interactive article [14]. These efforts hint at the versatility of Vistrates to explore new ideas for C2-VA. It is also exciting that using the component model these efforts contributed to the knowledge base within Vistrates. Now, when developers use Vistrates, they can right away instantiate these components created from scientific research projects.

Having said this, Vistrates is only a platform. A successful platform should simplify the job of application developers. Hence, future research should evaluate and, thereby, improve the user interface abstractions, components, and the development environment of Vistrates. Furthermore, the component repository should be enhanced to support the ever-evolving device technologies.

Chapter 10: Conclusion and Future Directions

My research targeted the effective use of heterogeneous devices for collaboration in device ecosystems. In this direction, I introduced the concept of collaborative, cross-device visual analytics (C2-VA). I broke down this concept by discussing the individual components within it in Chapter 3 of this dissertation, along with the related work and fundamentals (Chapters 4 and 5). The goal of this dissertation has been to support C2-VA by considering specific **focus points** (in Chapters 6-9):

- **Analytical activities:** This dissertation has focused on exploratory analyses [259] by targeting visualization tasks to develop insights through a visual exploration of data. In Chapter 6, the tasks were low-level—related to comprehension of values, trends, and correlations. In Chapter 7, a more comprehensive task set was considered to cover the taxonomy from Brehmer and Munzner [57]. Chapter 8 considered collaborative tasks where the users need to come to a consensus in a visual exploration activity. In the future, more activities—e.g., in visual communication—can be considered from specialized application scenarios (cf. business meetings [260]) to enhance this research.

- **Datasets and domains:** This research was built on application scenarios with multivariate datasets. We targeted situations in energy manage-

ment (sensor data), criminal justice (crime data), urban development (housing data), and tourism (business data). Other dataset types can be considered such as geospatial, graphs, trees, and events in the future to understand C2-VA support for tasks in those targets.

- **Collaboration types:** We considered small to medium groups of users (e.g., 2 to 4 users) as the target of our applications. Furthermore, we focused on symmetric collaborations where all the users contribute to a shared goal in the device ecosystem. Analytical contexts are rich with a broader variety of group structures. Analytical enterprises can contain analysts, engineers, designers, service officers, managers, and other stakeholders working together with different goals.

- **Devices:** This dissertation focuses on large displays and small devices such as tablets, smartphones, and smartwatches. In terms of input, touch, body-based, and gestural interaction was the focus. Utilizing other input and output modalities is an immediate and exciting direction to extend this work.

- **Contributions:** My focus, for this dissertation, was to balance scientific contributions with engineering outcomes. As such, the early research in this dissertation provided the software frameworks that were used later to investigate and contribute interaction guidelines and considerations for designers. By putting this knowledge together, the final contribution was a software platform to develop new applications. This platform opens up the space for new research in both directions—new applications can be developed using the

Vistrates platform to investigate new research ideas and generate guidelines, which further feed into Vistrates as components.

Through these focus points, this dissertation provides the following takeaways.

10.1 Takeaways and Limitations

> **Takeaway 1**: **Fundamentals for creating visualization interfaces** across devices in a device ecosystem. It includes roles of devices, interaction styles, software primitives, and user interface design to fit the device form factor.

Chapters 3 and 5 presented the basic considerations for creating visual interfaces for multiple devices. Some highlights from these contributions include:

- The visualization pipeline from Chapter 3 (Figure 3.1) provides a reference model of the challenges in supporting C2-VA. The C2-VA stack (Figure 3.2) overviews the components needed in a C2-VA application. These components have been the focus for Chapters 6-8).

- The roles of the devices from Chapter 5 based on related work (Chapter 4) not only guide the interaction design but also the visual analysis process itself.

- Software primitives needed for C2-VA based on our initial work—Munin [20] and PolyChrome [19]. It includes input, shared state and display space management across devices. For instance, PolyChrome turns user interaction into serialized operations within a visualization interface shared across the device

ecosystem. It also supports the distribution of the user interface containing multiple visualizations across multiple devices.

Takeaway 2: **Interaction models and a conceptual framework for visual analysis** using a large display and multiple small devices in a device ecosystem. It includes the utilization of the physical space through proxemics and gestures and designing interactions that complement the device affordances in an ecosystem (by considering their roles).

Large displays such as Microsoft Surface Hub and Promethean ActivPanel are becoming common in office and educational spaces. Inspired by the general HCI work on full-body interaction, we developed the Proxemic Lens technique based on proxemics—the study of the human use of space—and mid-air gestures to respond to the users in front of a wall-sized display. By investigating two interaction models based on proxemics and gestures for visual analysis, a mixed Proxemic Lens technique [21] was introduced to leverage the best of both interaction styles. This interaction technique enables fluid data exploration for multiple users in front of the wall-sized display. However, there are **limitations** to these natural body-based interactions as they require extra physical effort from the users compared to standard interaction through touch or mouse. Furthermore, there is a limit to their freedom of expression for complex analytical tasks.

To overcome these limitations, we combined multiple devices into the ecosystem to enhance the tasks. Since there is a plethora of novel input and display

technologies out there, it is not straightforward to utilize them in visual data exploration. We first elicited multiple cross-device interaction styles through formative evaluation. Based on this, we developed two interaction techniques—Visfer [22] for combining smartphones/tablets with large displays and David and Goliath technique [23] for combining smartwatches with large displays. We also created a conceptual framework for combining a large display and smartwatches for visual analysis tasks based on their roles to complement each other. The visual analytics system based on this technique showed that the users were able to work flexibly in the device ecosystem when the roles of the devices were complementary and showcased the differences in insights compared to a baseline.

Limitations. Having said this, there are some limitations to our findings. Our user studies provide evidence of the utility of the device combination for specific tasks. An in-depth study of open-ended visual exploration (cf. Reda et al. [247]) would broaden this to a larger group of tasks covered in our framework. More questions remain to be answered: (1) which visual analysis tasks can be enhanced by handhelds vs. wearables, and (2) which visualizations and application scenarios most benefit from such device combinations. Finally, our conceptual framework for visual analysis should be extended by mechanisms to explicitly promote coordination during collaborative visual analysis by conveying presence, attention, and coverage and supporting group awareness and communication.

Takeaway 3: **Considerations for coordination among multiple users** engaged in visual analysis in a device ecosystem. For group awareness, it is essential to represent the focus of each user and their context in the dataset using in-situ and ex-situ representations in the visualization interface.

To aid coordination among analysts engaged in collaborative analysis, we designed awareness techniques on top of PolyChrome to study their affordances in supporting coordinated decisions from data. Our InsightsDrive [24] platform captured group activity in terms of the data coverage through ex-situ widgets and in-situ highlights within the visualization interface. Applied to house search, InsightsDrive summarized the housing properties being viewed by different users through multi-dimensional representations such as parallel coordinates within an external widget and highlighted the user interaction on the internal visualizations within the data dashboard. Our user study revealed that users were significantly faster and effective at making collective decisions when using our group awareness technique.

Limitations. Our user studies in this research endeavor focused on specific awareness designs for multidimensional datasets in synchronous collaboration with small user groups. As such, it is hard to generalize the findings to other settings. To scale to larger groups, (1) aggregation techniques [251] need to be taken into consideration to convey the activity of multiple users and (2) the representations in in-situ and ex-situ awareness should be designed to be more seamless. Finally, the tasks chosen—in housing property search—are simple and not representative of all possible visual analytic tasks. This aspect can be a target for future work.

Takeaway 4: **The Vistrates platform introducing a component model** for future research in collaborative, cross-device visual analytics (C2-VA).

A culmination of my dissertation research was a web platform called Vistrates [12], which introduces a component model for creating C2-VA systems. A core concept within the Vistrates platform is the notion of a visual analytic component: a building block of any analytical system. Each component—say, a data mining model, a visual representation, or an interaction technique—connects to other components within a vistrate to form a cohesive analytical system. This system can contain different interface abstractions such as a programming view as in Jupyter notebooks, a dashboard view as in Tableau or Power BI, an annotation view as in Sense.us [4], or even a presentation view as in Microsoft Powerpoint. This versatility in the user interface is crucial when supporting many analytical activities, users of diverse expertise, and devices of different capabilities. The application examples developed using Vistrates showcase this versatility.

Limitations. All code in Vistrates is currently executed within a web browser, which has at least two limitations: (1) reliance on JavaScript and (2) the available computational power. Data scientists often use languages such as Python and R that provide multiple efficient libraries for data transformation and machine learning, which JavaScript does not offer to a similar extent. As a next step, we plan to make Vistrates a mixed environment, where, e.g., data analysis components developed in Python can be executed in the cloud and interleaved with visualization components developed in JavaScript. Beyond this, it can be challenging to deal

with large datasets and execute complex computations driving the data analysis in a web browser. Another limitation of the current implementation is its usability. Vistrates supports multiple levels of abstraction through development, pipeline, and dashboard views. However, the current proof-of-concept is centered on a linear development view based on literate computing. More abstraction levels should be available to support specific activities and users: (1) shelf-based configuration such as in Tableau and Polestar to assemble components, (2) provenance tracking using interaction and insight histories [183] for visual exploration, and (3) better mobile interfaces driven by responsive visualization [22, 261].

10.2 Future Directions

In the future, I envision using Vistrates as a platform for a multitude of visualization projects. The fact of the matter is that the core Vistrate features are too convenient to give up, and they come at minimal cost; building a Vistrate component instead of freestanding D3 [15], or Vega code [207] will make the result collaborative, cross-device, and shareable.

10.2.1 Collaborative Visual Analytics

Collaborative visual analytics is in a nascent stage with a few collaborative systems used in practice. Platforms such as Vistrates can help us create and deploy collaborative systems for visual analytics. New methods for decision making, coordination, group awareness, and social interaction can be investigated on top to

utilize the power of a collective mind. There are many research opportunities here:

1. Capturing analysis outcomes and knowledge with provenance and annotation.
2. Transforming the analytical discovery process into presentable summaries.
3. Understanding the collaborative workflows suited for visual analysis tasks.

Vistrates can be quite useful in these directions. Firstly, the historical states of visualizations are already stored in Vistrates. This information together with a new component for annotation of visualizations can provide the primitives to support the first research direction. By doing so, we can bridge the physical and cognitive gap between multiple users in C2-VA. Secondly, using the historical states of visualizations, a new user interface abstraction can be added to sort through these visualizations to create a presentation. This abstraction can be similar to our dashboard but instead showing the entire history of visualizations accessed by the users. Such an endeavor can promote pedagogy and knowledge transfer between users—data scientists, analysts, engineers, or even students—in analytics. Finally, creating a formal understanding of collaborative workflows requires an extension of the Vistrates component repository. Vistrates is currently suited for an open analytical activity, where every user contributes freely. Fixing a workflow between users—e.g., two analysts first start exploring the data, while another analyst reviews and combines their work—requires different user interface abstractions for each user. Due to the current limited number of UI abstractions provided in Vistrates, this calls for further development. To this end, the current Vistrate components and examples offer a reference point to tackle these new research ideas.

10.2.2 From Data Exploration to Visual Communication

My research has mainly focused on the visual exploration of data to develop insights. The sensemaking process is more than exploration. It involves many stages including data collection, evidence filing, hypothesis verification, and communication [2]. Supporting the analytical activities in these different stages is essential. In particular, the following research directions are interesting:

1. Supporting data mining and automated analyses for evidence collection.
2. Enabling a variety of communication media including data-driven presentations, comics, videos, and articles.

Vistrates opens the door for these directions as well. While we focused on essential visualization and data processing components in Vistrates, our component model can also be used to implement advanced algorithms. For instance, a machine learning (ML) model can be added to Vistrates using libraries such as Tensorflow.js.[1] These Tensorflow components can also connect to visualization components already in Vistrates to support visual debugging of the ML models. The limitation here is the processing power of the web browser. Furthermore, Vistrates focuses on simple communication activities through slideshow presentations. Similar to the slideshow, user interface abstractions can be created within Vistrates to support other communication media. For instance, to support data-driven comics, a user interface abstraction is needed to clip and compose visualization snapshots along with annotations. It can be an extension of the current dashboard interface.

[1] Tensorflow: https://www.tensorflow.org/js

10.2.3 New Device Technologies

Another research direction is to explore new device technologies. This direction can be used to solidify the understanding of the affordances of these technologies in analytical activities. Our vision for visualization beyond a desktop discusses these new technologies [17], while our recent work on multimodal interaction for immersive analytics begins to unravel the affordances of various technologies [18]. There are many interesting ideas to explore:

1. Using emerging display modalities to represent and interact with information.
2. Combining modalities for multimodal input and output in analytics.

Within this space, speech, IoT, and augmented/virtual reality technologies are exciting as they are of increasing interest in recent years. Vistrates currently focuses on visualization applications using web technologies. Supporting augmented and virtual reality technologies requires connection to the native programming interfaces (API) to render directly on the AR/VR headsets. Vistrates can serve as the graphics engine in this case to create 3D graphics and visualizations for analytical activities in these settings. Using primitive libraries for 3D graphics such as WebGL,[2] new components can be created in Vistrates following the standard component model to create 3D scenes. These components along with the visualization components currently in Vistrates can create AR and VR interfaces. Finally, to combine multiple input modalities—speech, IoT sensors, depth cameras, etc.—additional effort is required from the developers. Specifically, components need to

[2]WebGL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

be developed for communicating with the Vistrates server to connect to the native APIs of these input devices. This effort will help us design and evaluate natural interaction techniques within the device ecosystems.

Overall, I am excited to explore these research directions to contribute to scientific research in C2-VA. I am also looking forward to creating open source contributions through Vistrates that are shareable, replicable, and extensible.

Appendix A: User Study: Responsive Data Visualization Interfaces

Figure A.1: A visual analytics dashboard on different devices. (A and B) User interface with coordinated multiple view layout shown on a classical large display along with screen-responsive visual encoding instances on a tablet and a smartphone. (C and D) Close-up of the two responsive instances—an overview+detail and a boundary visualization interface.

This elaborate design space in Section 5.5 can be explored to design techniques for responsiveness by choosing alternative layouts, encodings, and data content in visual interface. In fact, previous work on time-series visualization by Chen [155] showcase techniques grounded in this design space. Chen leverages two techniques to fit a traditional multi-view interface on a smartwatch: (1) using border visualizations that utilize encodings that fit the visualizations into the border of the small display (adapting visual mapping), and (2) using overview+detail layout that shows the entire interface in a small bird's-eye view with content of interest as a detail view.

We were interested in evaluating such responsive interfaces to understand their affordances in sensemaking (Figure A.1). The focus of this user study was to observe quantitative differences in time and accuracy between a classical visualization interface and two small-screen versions—boundary and overview interfaces [155].

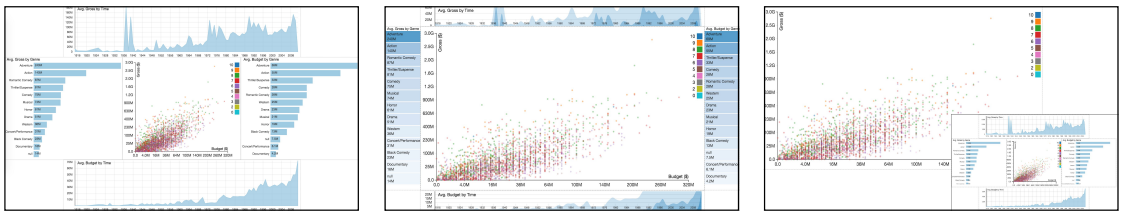


Figure A.2: (Left to right) Classical, boundary, and overview visualization interfaces used in the controlled study with the movies dataset. Boundary and overview interfaces are developed for a small screen (a tablet).

A.1 Dataset and Interfaces

We chose to study three interfaces: (1) a classical visualization interface (**CV**) showing a multidimensional dataset, (2) a boundary visualization interface (**BV**) developed for a small screen where the boundary views of the classical interface are compressed, and (3) an overview interface (**OV**) for a small screen that uses an overview+detail layout. Figure A.2 shows these three interfaces applied to a motion pictures (movies) dataset for this study. The movies dataset contains 3,201 movie records with 15 variables including information such as gross earnings, budget, genre, IMDB rating, Rotten Tomato rating, and date of release. We picked this dataset as it may be accessible and familiar to a general audience due to its real-world interest.

CV *Classical Interface:* The classical visualization interface consists of a CMV layout with five visualizations of a movies dataset. The center visualization is a scatterplot of movies organized by their budget (x-axis) and gross (y-axis), and colored based on their ratings. The visualization surrounding the center (focus) contain statistical information about these variables (average budget and gross) over the years and for different genres visualized with line charts and bar charts respectively. All visualizations take equal screen space. This interface represents a typical multi-view visual dashboard.

BV *Boundary Interface (Small Display):* The boundary visualization interface is created for a small display by compressing the peripheral views in the CMV layout using space-efficient visual encodings. The center scatterplot visualization takes most of the space (80%). The rest of the screen is equally divided among the four surrounding visualizations of average budget and gross. The line chart turns into a horizon chart with three layers and the bar chart adapts into a space-filling version with color intensity to capture value instead of bar size (cf. border views [155]).

OV *Overview Interface (Small Display):* The overview visualization interface is developed using an overview+detail layout on the classical interface. An overview+detail layout helps show multiple scales of information on a given limited display space. In this case, it shows the entire classical interface as an overview by making the viewport smaller, with the detail view showing a particular visualization.

A.2 Participants

We recruited 13 paid participants (5 female, 8 male) from the general student population of our university. The participants were between 18 and 45 years of age. All participants self-reported as proficient computer users. Furthermore, 8 participants had previously used visualization as a means of data analysis; however, it was mostly limited to charting in Excel, MATLAB, Mathematica, R, and SPSS. Only two participants had experience working with interactive visualization tools such as NodeXL, Gephi, and Tableau.

A.3 Apparatus (Devices)

The participants used a 55-inch display—Microsoft Perceptive Pixel—as the large display, and a 8.9-inch Google Nexus Tablet as the small display. The large display has a resolution of 1920×1080 pixels, while the tablet is 2048×1536 pixels (but with an effective CSS resolution of 1024×768 pixels). The interfaces were developed using web technologies—HTML, CSS, and JS—and the D3 framework [15].

A.4 Tasks

To measure the low-level costs of responsiveness through boundary and overview transformations in visual sensemaking, we chose to study typical visual analysis tasks such as value and trend identification, and comparison. The major feature of a CMV interface is the **coordination** across views: user interaction on one visu-

alization leading to visual changes in others. Therefore, it is an important aspect whose efficiency should be preserved across interfaces as they are made responsive to smaller screens. For this purpose, we chose to create tasks for the controlled study in which the participants had to verify statements about the data values and trends in the visualizations in each interface. We chose to simulate data selections on a focus view to see changes on the surrounding context of the CMV layout. The animations used to convey the changes in the CMV interface had a one-second duration and were staged with axes changing first and then the data. The statements provided to the participants are of two types: (1) perceiving changes in values in a chart in the interface (whether they increased or decreased over the animation created through simulated selections), and (2) perceiving retainment or reversal of a trend in a chart (whether the trend stayed the same or reversed at some location).

To come up with the statements, we created a list of observations with a goal of answering high-level questions about the data such as "how are the top rated movies different than others?", "are directors good at only specific genres?", and "are there differences in the typical gross from genres over time?" After this, we extracted the selection interactions that led to these observations, to simulate them in the actual experiment. The observations from answering the high-level questions were modified to create the statements for the study that can be either true or false. These statements were tested in two pilot studies with (1) a visualization expert to verify the correctness and complexity of the observations behind the statements, and (2) a novice student to verify if the statements are comprehensible. The statements were revised based on their feedback.

For controlling the study, statements had similar complexity—requiring the user to look at animated visual changes for two or more items within a visualization in the interface. Examples statements used in the participant tasks include,

1. The average budget values for thriller/suspense and musical genres are higher than default.

2. The average gross for 1972 and 2009 are lower than default.

3. The trend in average budget values between 2000 and 2008 is similar to default.

4. The trend in average gross values for Romantic Comedy and Horror genres is opposite to default.

Default is the state before animation, and trend refers to the profile of changes. These definitions were explained to the participants and verified during training.

A.5 Experiment Design and Procedure

We used a within-subjects design with the participants using all visual interfaces to verify statements about the data. Each interface was assigned a random set of statements, and the interface order was counterbalanced. This ensured that there was no effect of statement and interface order. There were eight statements (task repetitions) for each interface: four about time (line charts) and four about genres (bar charts). The statement type (S)—value comparison or trend comparison—and the data type (D)—about time (line chart) or genre (bar)—are also factors.

Each study session started with the participant reading and signing a consent form, as well as completing a demographic survey. Following this, they went through a training procedure on an assigned interface, including how to interpret the line charts, bar charts, and the statements. Their accuracy was tested during the training to make sure they understand the statements and the interface. Each task required participants to, (1) understand a statement and identify which visualization to look at, (2) click a *play* button to simulate an interaction leading to animated changes in the visualizations (after a 3-sec countdown), and (3) determine if the statement is true or not and submit the answer. They were also asked to verbally explain their reasoning for the answers before moving on to the next task. The participants were allowed to replay the simulated interactions for each statement any number of times. Following this, they click a *next* button to move to the next statement. Following the tasks on one interface, they completed a Likert-scale survey rating the efficiency, ease of use, and enjoyability of the interface. They then moved on to other conditions to follow the same process. Each session lasted up to 50 minutes.

A.6 Measures

We recorded the accuracy of each participant's assessments (true/false) and number of interaction replays performed to reach the assessment. We also recorded the time taken for each task, but this also includes the time taken by the participant to interpret the statements, which may differ across participants. We therefore treat the number of replays as a surrogate measure for time as the interaction replay itself

takes a fixed amount of time irrespective of the interface and this can implicitly isolate the subjective time for interpreting statements from the evaluation.

A.7 Hypothesis

Based on our design, we formulate the following hypothesis:

H1 The classical visualization interface will be more accurate than boundary and overview visualization interfaces as the large display contains more visual space, which can help track the visual changes in charts.

H2 Classical interface will be faster than boundary and overview interfaces for the same reason as above.

H3 Boundary will be faster and more accurate than overview as it uses the display space more efficiently with space-efficient encodings.

A.8 Analysis and Results

Here, we discuss accuracy/correctness and completion time (in terms of replays) for the tasks by reporting on the results from statistical analyses. Figure A.3 visualizes these results by calculating point estimates and 95% confidence intervals (CI) based on 1000 percentile bootstrap replicates. Considering recent concerns with null-hypotheses testing [262] and APA recommendations [263] regarding p-value statistics, our analysis combines the best of both worlds by reporting p-values as well as confidence intervals from Bootstrapping (see Figure A.3).

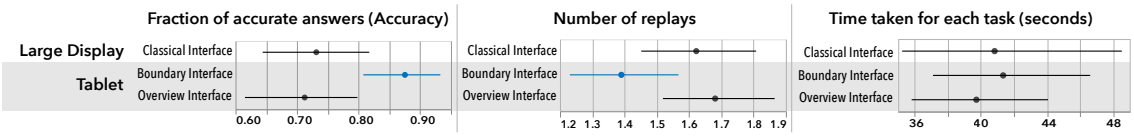


Figure A.3: (Left to right) Point estimates and 95% confidence intervals based on 1000 percentile bootstrap replicates for the measures from the user study.

A.8.1 Accuracy

Table A.1 summarizes the main effects and interactions on accuracy using logistic regression (all assumptions valid).

Table A.1: Effects of factors on accuracy (logistic regression).

Tasks	**Factors**	**df, den**	**F**	**p**
All	Display Interface (I)	2, 293.6	4.71	.009**
	Data Type (D)	1, 288.5	.09	.76
	Statement Type (S)	1, 299.3	1.67	.19
	I * D	2, 290.6	.51	.59
	I * S	2, 295.4	1.05	.35
	D * S	1, 289.1	1.39	.23
	I * D * S	2, 297.8	.53	.58

* $= p \leq 0.05$, ** $= p \leq 0.01$.

Post-hoc analysis with Tukey HSD revealed significant differences between boundary visualization and overview visualization interfaces ($p = .009$), and the classical interface and boundary visualization on tablet ($p = .017$). Boundary visu-

alization interface (total correct answers = 91/104) on the tablet was more accurate than the overview interface (total correct answers = 74/104) and the classical interface (total correct answers = 76/104). Figure A.3-left showcases these effects with the Boundary interface outperforming the classical and overview interfaces—with the effect being stronger between Overview and Boundary. There were no significant differences between the classical and overview interfaces. This rejects our first hypothesis (**H1 rejected**) since BV was significantly more accurate.

A.8.2 Number of Replays

As described above, time measurement is influenced by the time taken to interpret the statement in each task rather than just comprehension of the visualization and animation. In fact, the participants took similar amount of time to answer the statements in each condition as seen Figure A.3-right. We therefore focus on the number of interaction replays (an integer) and analyzed it using a generalized linear regression model (reported in detail in Table A.2).

Post-hoc analysis with Tukey HSD revealed significant differences between boundary and overview visualization interfaces ($p = .038$). There were no significant differences for the other two combinations. Overall, boundary visualization interface (mean = 1.39, s.d. = 0.86) had significantly less interaction replays than overview visualization interface (mean = 1.68, s.d. = 0.95). This can also be confirmed from Figure A.3-middle where the size of non-overlapping region between Boundary and Overview interfaces is more than 80% of the bands (showcasing a large effect) [264]).

Table A.2: Effects of factors on replays (gen. linear regression).

Tasks	**Factors**	**df, den**	**F**	**p**
All	Display Interface (I)	2, 24	3.50	.046*
	Data Type (D)	1, 12	.04	.84
	Statement Type (S)	1, 12	.00	.97
	I * D	2, 24	.14	.86
	I * S	2, 24	.35	.70
	D * S	1, 12	.26	.61
	I * D * S	2, 24	2.27	.12

* = $p \leq 0.05$, ** = $p \leq 0.01$.

Based on this, our second hypothesis is rejected (**H2 rejected**). However, the third hypothesis is confirmed since BV was faster based on interaction replays and more accurate than overview visualization on tablet (**H3 confirmed**).

A.9 Subjective Preferences

After each session, the participants rated the techniques on three interfaces: efficiency, ease of use, and enjoyability, on a Likert scale ranging from 1 (e.g., strongly disagree) to 5 (e.g., strongly agree). Figure A.4 showcases the differences between the three conditions as perceived by the participants. Across all the scales, both classical and boundary interface are perceived to be better than the overview interface. This especially reflects in the number of participants strongly disagreeing on all three

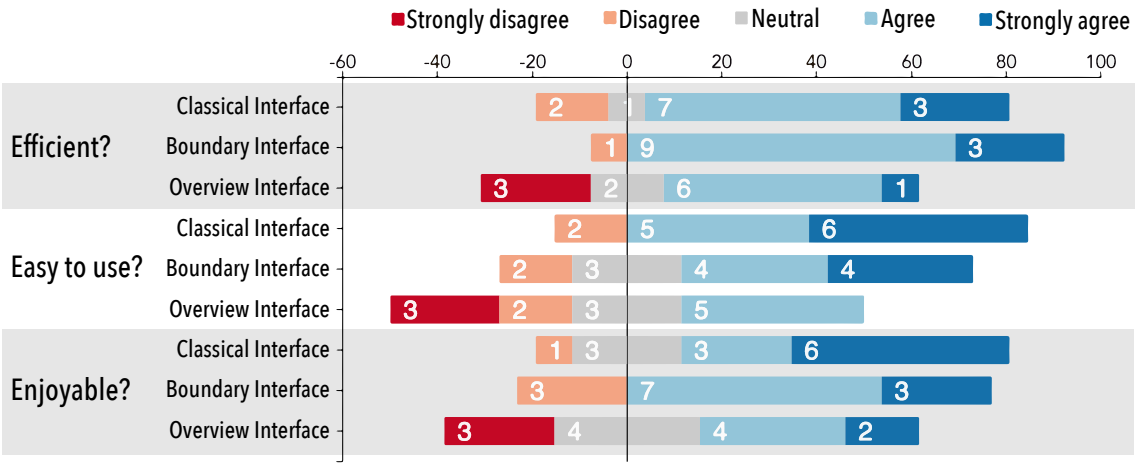


Figure A.4: Likert scale ratings provided by the participants for efficiency, ease of use, and enjoyability of the three conditions.

scales for the Overview condition on the tablet device. On the other hand, the differences between classical and boundary interfaces are little. Therefore, to better explain our results, we report the participant feedback in the following section.

A.10 Participant Feedback

The differences in performance can be explained through participant comments during the tasks as well as their preferences.

Boundary interface was better than classical (for some). Participants (8/13) expressed that the boundary interface suited the form factor of the small display used in this user study. Participants (4/13) commented that the classical interface on the large display required them to track a larger physical space on the screen to verify the statements in the tasks. This could have also contributed lower accuracy on the classical interface compared to the boundary (BV), which is a surprising

result. For instance, P3 commented, "*it is hard to look for the information that I want [on the classical interface] on large display. On boundary interface it is much easier to locate.*" Boundary interface was also seen to be more suited to the small screen than the overview interface. P5 said "the visualizations on the boundary are much easier to follow than the overview." In the post questionnaire, majority of the participants (12/13) therefore preferred boundary transformation—i.e., adapting the visual encodings to be more space efficient to achieve responsiveness—compared to the Overview. This is irrespective of the visualization (line or bar chart), and this aspect also reflects in the lack of an effect of the visualization type in our results. Four participants preferred the boundary interface on the tablet over the classical interface on the large display, citing the efficient nature of the boundary interface on a familiar personal device that they use in their everyday life.

Overview interface was familiar, but not effective. Participants felt that the overview interface provided a more familiar set of visualizations, since some of them (7/13) were not familiar with horizon charts. However, the added cost in tracking changes closely in a small overview overcame the convenience of reading familiar visual representations (line and bar charts). Five participants explicitly cited the small size of the overview visualization for giving it a poor rating on all scales. Therefore, overview transformation—a transformation of the layout of the interface to achieve responsiveness without changing the visual encodings—is not ideal when it comes to low-level sensemaking tasks.

Bibliography

[1] Niklas Elmqvist and Pourang Irani. Ubiquitous Analytics: Interacting with big data anywhere, anytime. *IEEE Computer*, 46(4):86–89, 2013.

[2] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of the International Conference on Intelligence Analysis*, volume 5, pages 2–4, 2005.

[3] Mark Weiser. The computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.

[4] Jeffrey Heer, Fernanda B. Viégas, and Martin Wattenberg. Voyagers and voyeurs: supporting asynchronous collaborative information visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1029–1038, New York, NY, USA, 2007. ACM.

[5] Andrew McAfee, Erik Brynjolfsson, Thomas H Davenport, DJ Patil, and Dominic Barton. Big data: The management revolution. *Harvard business review*, 90(10):60–68, 2012.

[6] Kristin A Cook and James J Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Lab (PNNL), Richland, WA (United States), 2005.

[7] Cole Nussbaumer Knaflic. Exploratory vs. Explanatory Analysis. https://web.archive.org/web/20190321033209/http://www.storytellingwithdata.com/blog/2014/04/exploratory-vs-explanatory-analysis, accessed March 2019.

[8] Edward T. Hall. *The hidden dimension*. Doubleday & Co, 1966.

[9] Ramik Sadana and John Stasko. Expanding selection for information visualization systems on tablet devices. In *Proceedings of the ACM Conference on Interactive Surfaces and Spaces*, pages 149–158. ACM, 2016.

[10] Ricardo Langner, Tom Horak, and Raimund Dachselt. VisTiles: Coordinating and combining co-located mobile devices for visual data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), 2017.

[11] Sriram Karthik Badam, Roman Rädle, Clemens Nylandsted Klokmose, and Niklas Elmqvist. Towards a unified visualization platform for ubiquitous analytics. In *Proceedings of the ACM CHI Workshop on Mobile Visualization*, 2018.

[12] Sriram Karthik Badam, Andreas Mathisen, Roman Rädle, Clemens N. Klokmose, and Niklas Elmqvist. Vistrates: A component model for ubiquitous analytics. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):586–596, 2019.

[13] Tom Horak, Andreas Mathisen, Clemens N. Klokmose, Raimund Dachselt, and Niklas Elmqvist. Vistribute: Distributing interactive visualizations in dynamic multi-device setups. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2019. To appear.

[14] Andreas Mathisen, Tom Horak, Clemens N. Klokmose, Kaj Grφnbæk, and Niklas Elmqvist. InsideInsights: Integrating data-driven reporting in collaborative visual analytics. *Computer Graphics Forum*, 2019. To appear.

[15] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[16] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[17] Jonathan C Roberts, Panagiotis D Ritsos, Sriram Karthik Badam, Dominique Brodbeck, Jessie Kennedy, and Niklas Elmqvist. Visualization beyond the desktop—the next big thing. *IEEE Computer Graphics and Applications*, 34(6):26–34, Nov 2014.

[18] Sriram Karthik Badam, Arjun Srinivasan, Niklas Elmqvist, and John Stasko. Affordances of input modalities for visual data exploration in immersive environments. In *Workshop on Immersive Analytics: Exploring Future Interaction and Visualization Technologies for Data Analytics at IEEE VIS*, 2017.

[19] Sriram Karthik Badam and Niklas Elmqvist. PolyChrome: A cross-device framework for collaborative web visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 109–118, 2014.

[20] Sriram Karthik Badam, Eli Fisher, and Niklas Elmqvist. Munin: A peer-to-peer middleware for ubiquitous analytics and visualization spaces. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):215–228, 2015.

[21] Sriram Karthik Badam, Fereshteh Amini, Niklas Elmqvist, and Pourang Irani. Supporting visual exploration for multiple users in large display environments. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 1–10. IEEE, 2016.

[22] Sriram Karthik Badam and Niklas Elmqvist. Visfer: Camera-based visual data transfer for cross-device visualization. *Information Visualization*, 18(1):68–93, 2019.

[23] Tom Horak, Sriram Karthik Badam, Niklas Elmqvist, and Raimund Dachselt. When David Meets Goliath: Combining smartwatches with a large vertical display for visual data exploration. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 19:1–19:13. ACM, 2018.

[24] Sriram Karthik Badam, Zehua Zeng, Emily Wall, Alex Endert, and Niklas Elmqvist. Supporting team-first visual analytics through group activity representations. In *Proceedings of the Graphics Interface Conference*, pages 208–213. Canadian Human-Computer Communications Society, 2017.

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

[26] Nicolai Marquardt, Frederik Brudy, Can Liu, Ben Bengler, and Christian Holz. SurfaceConstellations: A modular hardware platform for ad-hoc reconfigurable cross-device workspaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 354:1–354:14, 2018.

[27] Electronic flight instrument system. https://en.wikipedia.org/wiki/Electronic_flight_instrument_system, accessed March 2019.

[28] Flight cockpit design. http://aviationknowledge.wikidot.com/aviation:cockpit-design-and-human-factors, accessed March 2019.

[29] Zhicheng Liu, Nancy J. Nersessian, and John T. Stasko. Distributed cognition as a theoretical framework for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1173–1180, 2008.

[30] Lauren Bradel, Alex Endert, Kristen Koch, Christopher Andrews, and Chris North. Large high resolution displays for co-located collaborative sensemaking: Display usage and territoriality. *International Journal of Human-Computer Studies*, 71(11):1078–1088, 2013.

[31] Xiaojun Bi, Tovi Grossman, Justin Matejka, and George Fitzmaurice. Magic Desk: Bringing multi-touch surfaces into desktop work. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 2511–2520, 2011.

[32] Daniel Wigdor, Gerald Penn, Kathy Ryall, Alan Esenther, and Chia Shen. Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In *Proceedings of IEEE Tabletop*, pages 60–67, 2007.

[33] Roman Rädle, Hans-Christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. HuddleLamp: Spatially-aware mobile displays for ad-hoc around-the-table collaboration. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 45–54. ACM, 2014.

[34] Alex Endert, Lauren Bradel, Jessica Zeitz, Christopher Andrews, and Chris North. Designing large high-resolution display workspaces. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 58–65, 2012.

[35] Control Room. https://en.wikipedia.org/wiki/Control_room, accessed March 2019.

[36] Bridge of the enterprise. https://www.theverge.com/2013/4/19/4243082/star-trek-2009-set-photos-enterprise-bridge-bad-robot, accessed March 2019.

[37] Miami Gardens Police Department: Control room. https://www.cinemassive.com/case-studies-list/miami-gardens-real-time-crime-center/, accessed March 2019.

[38] Xavier Amatriain, JoAnn Kuchera-Morin, Tobias Hollerer, and Stephen Travis Pope. The Allosphere: Immersive multimedia for scientific discovery and artistic exploration. *IEEE Multimedia*, 16(2):64–75, 2009.

[39] Arnaud Prouzeau, Anastasia Bezerianos, and Olivier Chapuis. Towards road traffic management with forecasting on wall displays. In *Proceedings of the ACM Conference on Interactive Surfaces and Spaces*, pages 119–128, 2016.

[40] MIT Smartroom. http://vismod.media.mit.edu/vismod/demos/smartroom/, accessed March 2019.

[41] Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. Proxemic Interactions: The new ubicomp? *Interactions*, 18(1):42–50, 2011.

[42] Flavia Sparacino, Alex Pentland, Glorianna Davenport, Michal Hlavac, and Mary Obelnicki. City of news. In *ACM SIGGRAPH Abstracts and Applications*, page 189, 1999.

[43] Tobias Schwarz, Flavius Kehr, K Herme, and H Reitere. Holistic workspace: Future control room design. *Proceedings of the IADIS International Conference*, 2012.

[44] Ramesh Raskar, Michael S Brown, Ruigang Yang, Wei-Chao Chen, Greg Welch, Herman Towles, Brent Scales, and Henry Fuchs. Multi-projector displays using camera-based registration. In *Proceedings of the IEEE Conference on Visualization*, pages 161–522, 1999.

[45] KyungTae Kim, Waqas Javed, Cary Williams, Niklas Elmqvist, and Pourang Irani. Hugin: A framework for awareness and coordination in mixed-presence collaborative information visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 231–240. ACM, 2010.

[46] Remote surgery. https://en.wikipedia.org/wiki/Remote_surgery, accessed March 2019.

[47] ITACG: Enhancing first responder awareness. https://www.dni.gov/index.php/who-we-are/organizations/ise/ise-archive/ise-blog/2204-itacg-enhancing-first-responder-awareness, accessed March 2019.

[48] Niklas Elmqvist. Distributed user interfaces: State of the art. In José A. Gallud, Ricardo Tesoriero, and Victor M. R. Penichet, editors, *Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem*. Springer, 2011.

[49] Vidya Setlur, Sarah E Battersby, Melanie Tory, Rich Gossweiler, and Angel X Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 365–377, 2016.

[50] Frederik Brudy, Joshua Kevin Budiman, Steven Houben, and Nicolai Marquardt. Investigating practices when using an overview device in collaborative multi-surface trip-planning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2018.

[51] Thomas Plank, Hans-Christian Jetter, Roman Rädle, Clemens N. Klokmose, Thomas Luger, and Harald Reiterer. Is Two Enough?! Studying benefits, barriers, and biases of multi-tablet use for collaborative visualization. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, pages 4548–4560. ACM, 2017.

[52] James O Coplien and Neil B Harrison. *Organizational patterns of Agile software development*. Prentice-Hall, Inc., 2004.

[53] James J Lin, Lena Mamykina, Silvia Lindtner, Gregory Delajoux, and Henry B Strub. Fish'n'steps: Encouraging physical activity with an interactive computer game. In *International Conference on Ubiquitous Computing*, pages 261–278. Springer, 2006.

[54] Yrjö Engeström et al. Activity theory and individual and social transformation. *Perspectives on Activity Theory*, 19(38), 1999.

[55] Kari Kuutti. Activity theory as a potential framework for human-computer interaction research. In *Context and consciousness*, pages 17–44. Massachusetts Institute of Technology, 1995.

[56] Darren Edge, Nathalie Henry Riche, Jonathan Larson, and Christopher White. Beyond tasks: An activity typology for visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):267–277, 2018.

[57] Matthew Brehmer and Tamara Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, 2013.

[58] Jeffrey Heer and Maneesh Agrawala. Design considerations for collaborative visual analytics. *Information Visualization*, 7(1):49–62, 2008.

[59] Donald A. Norman. Human-centered design considered harmful. *Interactions*, 12(4):14–19, 2005.

[60] Chris Stolte and Pat Hanrahan. Polaris: a system for query, analysis and visualization of multi-dimensional relational databases. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 5–14, 2000.

[61] Fernando Pérez and Brian E Granger. IPython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29, 2007.

[62] Observable, 2018. https://beta.observablehq.com/, accessed Feb 2018.

[63] Arvind Satyanarayan, Kanit Wongsuphasawat, and Jeffrey Heer. Declarative interaction design for data visualization. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 669–678. ACM, 2014.

[64] Benjamin Bach, Natalie Kerracher, Kyle Wm. Hall, Sheelagh Carpendale, Jessie Kennedy, and Nathalie Henry Riche. Telling stories about dynamic networks with graph comics. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 3670–3682. ACM, 2016.

[65] Fereshteh Amini, Nathalie Henry Riche, Bongshin Lee, Andres Monroy-Hernandez, and Pourang Irani. Authoring data-driven videos with DataClips. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):501–510, 2017.

[66] Derek Hansen, Ben Shneiderman, and Marc A. Smith. *Analyzing Social Media Networks with NodeXL: Insights from a Connected World*. Morgan Kaufmann Publishers, September 2010.

[67] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the International Conference on Weblogs and Social Media*. The AAAI Press, 2009.

[68] Christopher Ahlberg. Spotfire: An information exploration environment. *SIGMOD Record*, 25(4):25–29, 1996.

[69] Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2227–2236, 2013.

[70] Bram C. M. Cappers and Jarke J. van Wijk. Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):532–541, 2018.

[71] Midas Nouwens and Clemens N. Klokmose. The application and its consequences for non-standard knowledge work. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 399:1–399:12, New York, NY, USA, 2018. ACM.

[72] Ji Soo Yi, Youn ah Kang, and John Stasko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[73] Robert A. Amar, James Eagan, and John T. Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 111–117, 2005.

[74] Bongshin Lee, Petra Isenberg, Nathalie Henry Riche, and Sheelagh Carpendale. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *IEEE Transactions of Visualization and Computer Graphics*, 18(12):2689–2698, 2012.

[75] Joëlle Coutaz, Lionel Balme, Christophe Lachenal, and Nicolas Barralon. Software infrastructure for distributed migratable user interfaces. In *Proceedings of UbiHCISys Workshop on UbiComp*, 2003.

[76] Clemens Nylandsted Klokmose and Michel Beaudouin-Lafon. VIGO: Instrumental interaction in multi-surface environments. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 869–878, 2009.

[77] Peter Tandler. Software infrastructure for ubiquitous computing environments: Supporting synchronous collaboration with heterogeneous devices. *LNCS*, 2201:96–115, 2001.

[78] Martin Modahl, Ilya Bagrak, Matthew Wolenetz, Phillip W. Hutto, and Umakishore Ramachandran. MediaBroker: An architecture for pervasive computing. In *Proceedings of the IEEE Conference on Pervasive Computing*, pages 253–262, 2004.

[79] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. The proximity toolkit: Prototyping proxemic interactions in ubiquitous

computing ecologies. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 315–326, 2011.

[80] Hans-Christian Jetter, Michael Zöllner, Jens Gerken, and Harald Reiterer. Design and implementation of post-WIMP distributed user interfaces with ZOIL. *International Journal of Human-Computer Interaction*, 28(11):737–747, 2012.

[81] Emmanuel Pietriga, Stéphane Huot, Mathieu Nancel, and Romain Primet. Rapid development of user interfaces on cluster-driven wall displays with jBricks. In *Proceedings of the ACM Symposium on Engineering Interactive Computing System*, pages 185–190, 2011.

[82] Xiaojun Bi and Ravin Balakrishnan. Comparing usage of a large high-resolution display to single or dual desktop displays for daily work. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1005–1014, 2009.

[83] Kenneth Moreland. Redirecting research in large-format displays for visualization. In *Proceedings of IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 91–95, 2012.

[84] Jishuo Yang and Daniel Wigdor. Panelrama: Enabling easy specification of cross-device web applications. In *Proceedings of the ACM conference on Human factors in computing systems*, pages 2783–2792, 2014.

[85] Michael Nebeling and Anind K. Dey. XDBrowser: User-defined cross-device web page designs. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 5494–5505. ACM, 2016.

[86] Michael Nebeling. XDBrowser 2.0: Semi-automatic generation of cross-device interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 4574–4584. ACM, 2017.

[87] Seonwook Park, Christoph Gebhardt, Roman Rädle, Anna Maria Feit, Hana Vrzakova, Niraj Ramesh Dayama, Hui-Shyong Yeo, Clemens Nylandsted Klokmose, Aaron Quigley, Antti Oulasvirta, and Otmar Hilliges. AdaM: Adapting multi-user interfaces for collaborative environments in real-time. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 184:1–184:14. ACM, 2018.

[88] Mikkel R. Jakobsen, Yonas Sahlemariam Haile, Soren Knudsen, and Kasper Hornbæk. Information visualization and proxemics: Design opportunities and empirical findings. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2386–2395, 2013.

[89] Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. Gradual engagement: Facilitating information exchange between digital devices as a function of proximity. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 31–40, 2012.

[90] Daniel Vogel and Ravin Balakrishnan. Interactive public ambient displays: Transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 137–146, 2004.

[91] Jeffrey S Shell, Ted Selker, and Roel Vertegaal. Interacting with groups of computers. *Communications of the ACM*, 46(3):40–46, 2003.

[92] Sarah M Peck, Chris North, and Doug Bowman. A multiscale interaction technique for large, high-resolution displays. In *IEEE Symposium on 3D User Interfaces*, pages 31–38, 2009.

[93] Mathieu Nancel, Julie Wagner, Emmanuel Pietriga, Olivier Chapuis, and Wendy Mackay. Mid-air pan-and-zoom on wall-sized displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 177–186, 2011.

[94] Panagiotis Vogiatzidakis and Panayiotis Koutsabasis. Gesture elicitation studies for mid-air interaction: A review. *Multimodal Technologies and Interaction*, 2(4):65, 2018.

[95] Jun Rekimoto. GestureWrist and GesturePad: Unobtrusive wearable interaction devices. In *Proceedings of the IEEE International Symposium on Wearable Computers*, pages 21–27. IEEE, 2001.

[96] Daniel L. Ashbrook, James R. Clawson, Kent Lyons, Thad E. Starner, and Nirmal Patel. Quickdraw: The impact of mobility and on-body placement on device access time. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 219–222. ACM, 2008.

[97] Peter Hamilton and Daniel J. Wigdor. Conductor: Enabling and understanding cross-device interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 2773–2782. ACM, 2014.

[98] Jun Rekimoto. Pick-and-drop: A direct manipulation technique for multiple computer environments. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 31–39. ACM, 1997.

[99] Xiang Chen, Tovi Grossman, Daniel J. Wigdor, and George Fitzmaurice. Duet: Exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 159–168. ACM, 2014.

[100] Steven Houben and Nicolai Marquardt. WATCHCONNECT: A toolkit for prototyping smartwatch-centric cross-device applications. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1247–1256. ACM, 2015.

[101] Ulrich von Zadow, Wolfgang Büschel, Ricardo Langner, et al. SleeD: Using a sleeve display to interact with touch-sensitive display walls. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 129–138, 2014.

[102] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Klokmose, and Nicolai Marquardt. Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, 2019.

[103] David Thevenin and Joëlle Coutaz. Plasticity of user interfaces: Framework and research agenda. In *Proceedings of IFIP INTERACT*, pages 110–117, 1999.

[104] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.

[105] Desney S Tan, Darren Gergle, Peter Scupelli, and Randy Pausch. With similar visual angles, larger displays improve spatial performance. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 217–224, 2003.

[106] Mary Czerwinski, Greg Smith, Tim Regan, Brian Meyers, George Robertson, and Gary Starkweather. Toward characterizing the productivity benefits of very large displays. In *Proceedings of INTERACT*, pages 9–16, 2003.

[107] Christopher Andrews, Alex Endert, Beth Yost, and Chris North. Information visualization on large, high-resolution displays: Issues, challenges, and opportunities. *Information Visualization*, 10(4):341–355, 2011.

[108] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996.

[109] Anastasia Bezerianos and Petra Isenberg. Perception of visual variables on tiled wall-sized displays for information visualization applications. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2516–2525, 2012.

[110] Alex Endert, Christopher Andrews, Yueh Hua Lee, and Chris North. Visual encodings that support physical navigation on large displays. In *Proceedings*

of Graphics Interface, pages 103–110. Canadian Human-Computer Communications Society, 2011.

[111] Petra Isenberg, Pierre Dragicevic, Wesley Willett, Anastasia Bezerianos, and Jean-Daniel Fekete. Hybrid-image visualization for large viewing environments. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2346–2355, 2013.

[112] Robert Ball and Chris North. Realizing embodied interaction for visual analytics through large displays. *Computers & Graphics*, 31(3):380–400, 2007.

[113] Robert Ball, Chris North, and Doug A. Bowman. Move to improve: Promoting physical navigation to increase user performance with large displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 191–200. ACM, 2007.

[114] Mikkel R. Jakobsen and Kasper Hornbæk. Interactive visualizations on large and small displays: The interrelation of display size, information space, and scale. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2336–2345, 2013.

[115] Mikkel R. Jakobsen and Kasper Hornbæk. Is moving improving?: Some effects of locomotion in wall-display interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 4169–4178. ACM, 2015.

[116] Can Liu, Olivier Chapuis, Michel Beaudouin-Lafon, Eric Lecolinet, and Wendy E. Mackay. Effects of display size and navigation type on a classification task. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 4147–4156. ACM, 2014.

[117] Petra Isenberg, Niklas Elmqvist, Jean Scholtz, Daniel Cernea, Kwan-Liu Ma, and Hans Hagen. Collaborative visualization: Definition, challenges, and research agenda. *Information Visualization*, 10(4):310–326, 2011.

[118] Mikkel R. Jakobsen and Kasper Hornbæk. Up close and personal. *ACM Transactions on Computer-Human Interaction*, 21(2):1–34, 2014.

[119] Arnaud Prouzeau, Anastasia Bezerianos, and Olivier Chapuis. Evaluating multi-user selection for exploring graph topology on wall-displays. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):1936–1951, 2017.

[120] Frederik Brudy, David Ledo, Saul Greenberg, and Andreas Butz. Is anyone looking? Mitigating shoulder surfing on public displays through awareness and protection. In *Proceedings of the International Symposium on Pervasive Displays*, pages 1:1–1:6. ACM, 2014.

[121] Paul Dourish. *Where the action is: The foundations of embodied interaction*. MIT press, 2004.

[122] Robert Ball, Michael DellaNoce, Tao Ni, Francis Quek, and Chris North. Applying embodied interaction and usability engineering to visualization on large displays. In *Proceedings of the ACM British HCI-Workshop on Visualization & Interaction*, pages 57–65, 2006.

[123] Christopher Andrews and Chris North. Analyst's workspace: An embodied sensemaking environment for large, high-resolution displays. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 123–131, 2012.

[124] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. Proxemic Interaction: Designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 121–130. ACM, 2010.

[125] Jakub Dostal, Uta Hinrichs, Per Ola Kristensson, and Aaron Quigley. SpiderEyes: Designing attention-and proximity-aware collaborative interfaces for wall-sized displays. In *Proceedings of the ACM Conference on Intelligent User Interfaces*, pages 143–152, 2014.

[126] Nicolai Marquardt and Saul Greenberg. Informing the design of proxemic interactions. *IEEE Pervasive Computing*, 11(2):14–23, 2012.

[127] Ulrike Kister, Patrick Reipschläger, Fabrice Matulic, and Raimund Dachselt. BodyLenses: Embodied magic lenses and personal territories for wall displays. In *Proceedings of the 2015 ACM International Conference on Interactive Tabletops & Surfaces*, pages 117–126. ACM, 2015.

[128] Beth Yost, Yonca Haciahmetoglu, and Chris North. Beyond visual acuity: The perceptual scalability of information visualizations for large displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 101–110, 2007.

[129] Robert J. K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: A framework for post-WIMP interfaces. In *Proceedings ACM Conference on Human Factors in Computing Systems*, pages 201–210, 2008.

[130] Petra Isenberg and Sheelagh Carpendale. Interactive tree comparison for co-located collaborative information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1232–1239, 2007.

[131] Petra Isenberg and Danyel Fisher. Collaborative brushing and linking for co-located visual analytics of document collections. *Computer Graphics Forum*, 28(3):1031–1038, 2009.

[132] KyungTae Kim and Niklas Elmqvist. Embodied lenses for collaborative visual queries on tabletop displays. *Information Visualization*, 11(4):319–338, 2012.

[133] Martin Spindler, Christian Tominski, Heidrun Schumann, and Raimund Dachselt. Tangible views for information visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 157–166, 2010.

[134] Petra Isenberg, Uta Hinrichs, Mark Hancock, Matthew Tobiasz, and Sheelagh Carpendale. Information visualization on interactive tabletops in work vs. public settings. *Collaborative Visualization on Interactive Surfaces*, 2010.

[135] Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. SketchStory: Telling more engaging stories with data through freeform sketching. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2416–2425, 2013.

[136] Dominikus Baur, Bongshin Lee, and Sheelagh Carpendale. Touchwave: Kinetic multi-touch manipulation for hierarchical stacked graphs. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 255–264. ACM, 2012.

[137] John Thompson, Arjun Srinivasan, and John T Stasko. Tangraphe: Interactive exploration of network visualizations using single hand, multi-touch gestures. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 43:1–43:5, 2018.

[138] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. DataTone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the ACM Symposium on User Interface Software & Technology*, pages 489–500, 2015.

[139] Arjun Srinivasan and John Stasko. Orko: Facilitating multimodal interaction for visual network exploration and analysis. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):511–521, 2018.

[140] Bongshin Lee, Matthew Brehmer, Petra Isenberg, Eun Kyoung Choe, Ricardo Langner, and Raimund Dachselt. Data visualization on mobile devices. In *Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems*, 2018.

[141] Benjamin Watson and Vidya Setlur. Emerging research in mobile visualization. In *Adjunct Proceedings of the ACM Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*, pages 883–887. ACM, 2015.

[142] Steven M. Drucker, Danyel Fisher, Ramik Sadana, Jessica Herron, and m.c. schraefel. TouchViz: A case study comparing two interfaces for data analytics on tablets. In *Proc. CHI '13*, pages 2301–2310. ACM, 2013.

[143] Luca Chittaro. Visualizing information on mobile devices. *IEEE Computer*, 39(3):40–45, 2006.

[144] Ahmad M. M. Razip, Abish Malik, Shehzad Afzal, Matthew Potrawski, Ross Maciejewski, Yun Jang, Niklas Elmqvist, and David S. Ebert. A mobile visual analytics approach for law enforcement situation awareness. In *Proceedings of the IEEE Pacific Symposium on Visualization*, pages 1235–1244. IEEE, 2014.

[145] Ragaad AlTarawneh, Shah Rukh Humayoun, and Abdel-karim Al-Jaafreh. Towards optimizing the sunburst visualization for smart mobile devices. In *Adjunct Proceedings of the International Conference on Human-Computer Interaction (INTERACT)*, volume 22, page 2015323, 2015.

[146] W. A. Pike, J. Bruce, B. Baddeley, D. Best, L. Franklin, R. May, D. M. Rice, R. Riensche, and K. Younkin. The scalable reasoning system: Lightweight visualization for distributed analytics. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 131–138, 2008.

[147] Paweł Woźniak, Lars Lischke, Benjamin Schmidt, Shengdong Zhao, and Morten Fjeld. Thaddeus: A dual device interaction space for exploring information visualisation. In *Proceedings of the ACM Nordic Conference on Human-Computer Interaction*, pages 41–50, 2014.

[148] Will McGrath, Brian Bowman, David McCallum, Juan David Hincapié-Ramos, Niklas Elmqvist, and Pourang Irani. Branch-explore-merge: Facilitating real-time revision control in collaborative visual exploration. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 235–244. ACM, 2012.

[149] Ricardo Langner, Ulrich von Zadow, Tom Horak, Annett Mitschick, and Raimund Dachselt. *Content Sharing Between Spatially-Aware Mobile Phones and Large Vertical Displays Supporting Collaborative Work*, pages 75–96. Springer International Publishing, 12 2016.

[150] Ricardo Langner, Ulrike Kister, and Raimund Dachselt. Multiple coordinated views at large displays for multiple users: Empirical findings on user behavior, movements, and distances. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):608–618, 2019.

[151] Haeyong Chung, Chris North, Jessica Zeitz Self, Sharon Lynn Chu, and Francis K. H. Quek. VisPorter: Facilitating information sharing for collaborative sensemaking on multiple displays. *Personal and Ubiquitous Computing*, 18(5):1169–1186, 2014.

[152] Olivier Chapuis, Anastasia Bezerianos, and Stelios Frantzeskakis. Smarties: An input system for wall display development. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 2763–2772. ACM, 2014.

[153] Can Liu, Olivier Chapuis, Michel Beaudouin-Lafon, and Eric Lecolinet. CoReach: Cooperative gestures for data manipulation on wall-sized displays. In

Proceedings of the ACM Conference on Human Factors in Computing Systems, pages 6730–6741. ACM, 2017.

[154] Ulrike Kister, Konstantin Klamka, Christian Tominski, and Raimund Dachselt. GraSp: Combining spatially-aware mobile devices and a display wall for graph visualization and interaction. *Computer Graphics Forum*, 36(3), 2017.

[155] Yang Chen. Visualizing large time-series data on very small screens. In *Short Paper Proceedings of the IEEE VGTC/Eurographics Conference on Visualization*, pages 37–41. The Eurographics Association, 2017.

[156] Tanja Blascheck, Lonni Besançon, Anastasia Bezerianos, Bongshin Lee, and Petra Isenberg. Glanceable visualization: Studies of data comparison performance on smartwatches. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):630–640, 2019.

[157] Rúben Gouveia, Fábio Pereira, Evangelos Karapanos, Sean A Munson, and Marc Hassenzahl. Exploring the design space of glanceable feedback for physical activity trackers. In *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing*, pages 144–155, 2016.

[158] Ulrich von Zadow, Wolfgang Büschel, Ricardo Langner, and Raimund Dachselt. SleeD: Using a sleeve display to interact with touch-sensitive display walls. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*, pages 129–138. ACM, 2014.

[159] Frederik Brudy, Steven Houben, Nicolai Marquardt, and Yvonne Rogers. CurationSpace: Cross-device content curation using instrumental interaction. In *Proceedings of the ACM Conference on Interactive Surfaces and Spaces*, pages 159–168. ACM, 2016.

[160] Xin Yi, Chun Yu, Weijie Xu, Xiaojun Bi, and Yuanchun Shi. Compass: Rotational keyboard on non-touch smartwatches. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 705–715. ACM, 2017.

[161] Jungsoo Kim, Jiasheng He, Kent Lyons, and Thad Starner. The gesture watch: A wireless contact-free gesture based wrist interface. In *Proceedings of the IEEE Symposium on Wearable Computers*, pages 15–22. IEEE, 2007.

[162] Keiko Katsuragawa, Krzysztof Pietroszek, James R. Wallace, and Edward Lank. Watchpoint: Freehand pointing with a smartwatch in a ubiquitous display environment. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 128–135. ACM, 2016.

[163] Gerard Wilkinson, Ahmed Kharrufa, Jonathan Hook, Bradley Pursglove, Gavin Wood, Hendrik Haeuser, Nils Y. Hammerla, Steve Hodges, and Patrick

Olivier. Expressy: Using a wrist-worn inertial measurement unit to add expressiveness to touch-based interactions. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 2832–2844. ACM, 2016.

[164] Jerome Pasquero, Scott J. Stobbe, and Noel Stonehouse. A haptic wristwatch for eyes-free interactions. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 3257–3266. ACM, 2011.

[165] Dirk Wenig, Johannes Schöning, Alex Olwal, Mathias Oben, and Rainer Malaka. WatchThru: Expanding smartwatch displays with mid-air visuals and wrist-worn augmented reality. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 716–721. ACM, 2017.

[166] Gierad Laput, Robert Xiao, Xiang 'Anthony' Chen, Scott E. Hudson, and Chris Harrison. Skin Buttons: Cheap, small, low-powered and clickable fixed-icon laser projectors. In *Proc. of the ACM Symposium on User Interface Software and Technology*, pages 389–394, 2014.

[167] James J. Thomas and Kristin A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.

[168] Ronald M Baecker. *Readings in Groupware and Computer-Supported Cooperative Work: Assisting Human-Human Collaboration*. Elsevier, 1993.

[169] Robert Johansen. *Groupware: Computer support for business teams*. The Free Press, 1988.

[170] John C. Tang. Findings from observational studies of collaborative work. *International Journal of Man-Machine Studies*, 34(2):143–160, 1991.

[171] Herbert H. Clark and Susan E. Brennan. Grounding in communication. In L. B. Resnick, J. M. Levine, and S. D. Teasley, editors, *Perspectives on Socially Shared Cognition*, pages 127–149. Amer. Psych. Assoc., 1991.

[172] Carl Gutwin and Saul Greenberg. Design for individuals, design for groups: Tradeoffs between power and workspace awareness. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, pages 207–216, 1998.

[173] Carl Gutwin and Saul Greenberg. Effects of awareness support on groupware usability. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 511–518, 1998.

[174] Carl Gutwin, Mark Roseman, and Saul Greenberg. A usability study of awareness widgets in a shared workspace groupware system. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, pages 258–267, 1996.

[175] Anthony Tang, Carmen Neustaedter, and Saul Greenberg. VideoArms: Embodiments for mixed presence groupware. In *Proceedings of the British Human-Computer Interaction Conference*, volume 20, pages 85–102, 2006.

[176] Philip Tuddenham and Peter Robinson. Distributed tabletops: Supporting remote and mixed-presence tabletop collaboration. In *Proceedings of IEEE Tabletop*, pages 19–26, 2007.

[177] Philip Tuddenham and Peter Robinson. Territorial coordination and workspace awareness in remote tabletop collaboration. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 2139–2148, 2009.

[178] Zia Ur Rehman Kiani, Darja Šmite, and Aamer Riaz. Measuring awareness in cross-team collaborations - distance matters. In *Proceedings of the IEEE Conference on Global Software Engineering*, pages 71–79, 2013.

[179] Laszlo Laufer, Peter Halacsy, and Adam Somlai-Fischer. Prezi meeting: Collaboration in a zoomable canvas based environment. In *Proceedings of the ACM Conference on Human factors in Computing Systems*, pages 749–752, 2011.

[180] Gloria Mark and Alfred Kobsa. The effects of collaboration and system transparency on CIVE usage: an empirical study and model. *Presence: Teleoperators & Virtual Environments*, 14(1):60–80, 2005.

[181] Aruna D Balakrishnan, Susan R Fussell, and Sara Kiesler. Do visualizations improve synchronous remote collaboration? In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1227–1236, 2008.

[182] Fernanda B Viégas, Martin Wattenberg, Frank Van Ham, Jesse Kriss, and Matt McKeon. ManyEyes: A site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007.

[183] Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1189–1196, 2008.

[184] Matt McKeon. Harnessing the information ecosystem with wiki-based visualization dashboards. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1081–1088, 2009.

[185] Tableau. https://www.tableau.com/, accessed June 2018.

[186] Anthony C. Robinson. Collaborative synthesis of visual analytic results. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 67–74, 2008.

[187] Petra Isenberg, Anthony Tang, and M. Sheelagh T. Carpendale. An exploratory study of visual information analysis. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1217–1226, 2008.

[188] Matthew Tobiasz, Petra Isenberg, and Sheelagh Carpendale. Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1065–1072, 2009.

[189] Jian Zhao, Michael Glueck, Simon Breslav, Fanny Chevalier, and Azam Khan. Annotation graphs: A graph-based visualization for meta-analysis of data based on user-authored annotations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):261–270, 2017.

[190] Phong H Nguyen, Kai Xu, Ashley Wheat, BL William Wong, Simon Attfield, and Bob Fields. SensePath: Understanding the sensemaking process through analytic provenance. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):41–50, 2016.

[191] Eric D Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):31–40, 2016.

[192] Kai Xu, Simon Attfield, TJ Jankun-Kelly, Ashley Wheat, Phong H Nguyen, and Nallini Selvaraj. Analytic provenance for sensemaking: A research agenda. *IEEE Computer Graphics and Applications*, 35(3):56–64, 2015.

[193] Shenyu Xu, Chris Bryan, Jianping Kelvin Li, Jian Zhao, and Kwan-Liu Ma. Chart constellations: Effective chart summarization for collaborative and multi-user analyses. *Computer Graphics Forum*, 37(3):75–86, 2018.

[194] Paul Dourish and Matthew Chalmers. Running out of space: Models of information navigation. In *Short paper presented at HCI*, volume 94, pages 23–26, 1994.

[195] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007.

[196] Amir Hossein Hajizadeh, Melanie Tory, and Rock Leung. Supporting awareness through collaborative brushing and linking of tabular data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2189–2197, 2013.

[197] Narges Mahyar and Melanie Tory. Supporting communication and coordination in collaborative sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1633–1642, 2014.

[198] Ali Sarvghad, Melanie Tory, and Narges Mahyar. Visualizing dimension coverage to support exploratory analysis. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2016.

[199] Ali Sarvghad and Melanie Tory. Exploiting analysis history to support collaborative data analysis. In *Proceedings of the Graphics Interface Conference*, pages 123–130, 2015.

[200] Gonzalo Gabriel Méndez, Miguel A Nacenta, and Sebastien Vandenheste. iVoLVER: Interactive visual language for visualization extraction and reconstruction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 4073–4085, 2016.

[201] Donghao Ren, Tobias Höllerer, and Xiaoru Yuan. iVisDesigner: Expressive interactive design of information visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2092–2101, 2014.

[202] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, John Stasko, and UT Lehi. Data Illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 123:1–123:13, 2018.

[203] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. DataInk: Direct and creative data-oriented drawing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 223:1–223:13, 2018.

[204] Mehmet Adil Yalcin, Niklas Elmqvist, and Benjamin B Bederson. Keshif: Rapid and expressive tabular data exploration for novices. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[205] Michael Bostock and Jeffrey Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1121–1128, 2009.

[206] Arvind Satyanarayan and Jeffrey Heer. Authoring narrative visualizations with Ellipsis. *Computer Graphics Forum*, 33(3):361–370, 2014.

[207] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017.

[208] Deok Gun Park, Steven M. Drucker, Roland Fernandez, and Niklas Elmqvist. ATOM: A grammar for unit visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3032–3043, 2018.

[209] SciPy. http://scipy.org, accessed March 2018.

[210] Altair. http://altair-viz.github.io, accessed March 2018.

[211] Google Colaboratory. https://colab.research.google.com/, accessed Feb 2018.

[212] Hendrik Sollich, Ulrich von Zadow, Tobias Pietzsch, Pavel Tomancak, and Raimund Dachselt. Exploring time-dependent scientific data using spatially aware mobiles and large displays. In *Proceedings of the ACM Conference on Interactive Surfaces and Spaces*, pages 349–354, 2016.

[213] Jeffrey Heer. The design of Sense.us. In Toby Segaran and Jeff Hammerbacher, editors, *Beautiful Data: The Stories Behind Elegant Data Solutions*, pages 183–204. O'Reilly Media, Inc., 2009.

[214] Roman Rädle, Midas Nouwens, Kristian Antonsen, James R. Eagan, and Clemens N. Klokmose. Codestrates: Literate computing with webstrates. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 715–725. ACM, 2017.

[215] John Stasko, Carsten Görg, and Zhicheng Liu. Jigsaw: Supporting investigative analysis through interactive visualization. *Information visualization*, 7(2):118–132, 2008.

[216] George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. The large-display user experience. *IEEE Computer Graphics and Applications*, 25(4):44–51, 2005.

[217] Nicolai Marquardt and Saul Greenberg. *Proxemic Interactions: From Theory to Practice*. Morgan & Claypool Publishers, 2015.

[218] Ken Hinckley, Gonzalo Ramos, Francois Guimbretiere, Patrick Baudisch, and Marc Smith. Stitching: Pen gestures that span multiple displays. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 23–31, 2004.

[219] David Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, January 1985.

[220] Brad Johanson and Armando Fox. Extending tuplespaces for coordination in interactive workspaces. *Journal Systems and Software*, 69(3):243–266, 2004.

[221] Jeffrey Heer and Maneesh Agrawala. Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):853–860, 2006.

[222] Jean-Daniel Fekete. The Infovis toolkit. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 167–174, 2004.

[223] Waqas Javed and Niklas Elmqvist. Exploring the design space of composite visualization. In *Proceedings of the IEEE Pacific Symposium on Visualization*, pages 1–8, 2012.

[224] Juliana Leclaire and Aurélien Tabard. R3s.js—Towards responsive visualizations. In *Proceedings of the Workshop on Data Exploration for Interactive Surfaces*, pages 16–19, 2015.

[225] Luke Wroblewski. Mobile first. http://www.lukew.com/ff/entry.asp?933, November 2009.

[226] Jonathan C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71. IEEE, 2007.

[227] George W. Furnas. Generalized fisheye views. In *Proceedings of the ACM Conference on Human Factors in Computer Systems*, pages 16–23, 1986.

[228] John Lamping, Ramana Rao, and Peter Pirolli. A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 401–408, 1995.

[229] Sohaib Ghani, N Henry Riche, and Niklas Elmqvist. Dynamic insets for context-aware graph navigation. *Computer Graphics Forum*, 30(3):861–870, 2011.

[230] Brian Johnson and Ben Shneiderman. Tree maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Conference on Visualization*, pages 284–291, 1991.

[231] Ming C Hao, Umeshwar Dayal, Daniel Keim, and Tobias Schreck. A visual analysis of multi-attribute data using pixel matrix displays. In *Proceedings of SPIE Visualization and Data Analysis*, 2007.

[232] Matej Novotny and Helwig Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.

[233] Fu-Lai Chung, Tak-Chung Fu, R. Luk, and V. Ng. Flexible time series pattern matching based on perceptually important points. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Learning from Temporal and Spatial Data*, pages 1–7, 2001.

[234] Adam Kendon. Spacing and orientation in co-present interaction. In *Development of Multimodal Interfaces: Active Listening and Synchrony*, pages 1–15. Springer, 2010.

[235] Wendy Ju, Brian A Lee, and Scott R Klemmer. Range: Exploring implicit interaction through electronic whiteboard design. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 17–26, 2008.

[236] Jeffrey Heer and Ben Shneiderman. A taxonomy of tools that support the fluent and flexible use of visualizations. *ACM Queue*, 10(2):1–26, 2012.

[237] Anthony Tang, Melanie Tory, Barry Po, Petra Neumann, and Sheelagh Carpendale. Collaborative coupling over tabletop displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1181–1190, 2006.

[238] Dominik Schmidt, Julian Seifert, Enrico Rukzio, and Hans Gellersen. A cross-device interaction style for mobiles and surfaces. In *Proceedings of the Designing Interactive Systems Conference*, pages 318–327. ACM, 2012.

[239] Chris Harrison and Anind K Dey. Lean and zoom: Proximity-aware user interface and content magnification. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 507–510, 2008.

[240] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. The Proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 315–326, 2011.

[241] Jun Rekimoto and Masanori Saitoh. Augmented Surfaces: A spatially continuous work space for hybrid computing environments. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 378–385, 1999.

[242] Alex Olwal and Steven Feiner. Spatially aware handhelds for high-precision tangible interaction with large displays. In *Proc. of the ACM Conference on Tangible and Embedded Interaction*, pages 181–188, 2009.

[243] Waqas Javed, Sohaib Ghani, and Niklas Elmqvist. PolyZoom: Multiscale and multifocus exploration in 2D visual spaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 287–296, 2012.

[244] Kazuki Takashima, Saul Greenberg, Ehud Sharlin, and Yoshifumi Kitamura. A shape-shifting wall display that supports individual and group activities. Technical report, Department of Computer Science, University of Calgary, 2015.

[245] Barney G Glaser and Anselm L Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, 1999.

[246] Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. Cooperative gestures: Multi-user gestural interactions for co-located groupware. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 1201–1210, 2006.

[247] Khairi Reda, Andrew E. Johnson, Michael E. Papka, and Jason Leigh. Effects of display size and resolution on user behavior and insight acquisition in visual exploration. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 2759–2768. ACM, 2015.

[248] Jacob O Wobbrock, Meredith Ringel Morris, and Andrew D Wilson. User-defined gestures for surface computing. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1083–1092, 2009.

[249] Michael Rohs. Marker-based embodied interaction for handheld augmented reality games. *Journal of Virtual Reality and Broadcasting*, 4(5):1860–2037, 2007.

[250] Ken Perlin and David Fox. Pad: An alternative approach to the computer interface. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, pages 57–64, 1993.

[251] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the IEEE Conference on Visualization*, pages 43–50, 1999.

[252] Richard Arias-Hernandez, L. T. Kaastra, Tera Marie Green, and Brian Fisher. Pair analytics: Capturing reasoning processes in collaborative visual analytics. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 1–10, 2011.

[253] Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. Webstrates: Shareable dynamic media. In *Proceedings of the ACM Symposium on User Interface Software & Technology*, pages 280–290. ACM, 2015.

[254] Michel Beaudouin-Lafon. Towards unified principles of interaction. In *Proceedings of the Biannual Conference on Italian SIGCHI Chapter*, pages 1:1–1:2, New York, NY, USA, 2017. ACM.

[255] Marcel Borowski, Roman Rädle, and Clemens Nylandsted Klokmose. Codestrates packages: An alternative to "one-size-fits-all" software. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, pages LBW103:1–6, New York, NY, USA, 2018. ACM.

[256] Ed Huai-hsin Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 69–75. IEEE, 2000.

[257] Webstrates Signaling API. https://webstrates.github.io/userguide/api/signaling.html, accessed June 2018.

[258] Zhe Cui, Shivalik Sen, Sriram Karthik Badam, and Niklas Elmqvist. VisHive: Supporting web-based visualization through ad hoc computational clusters of mobile devices. *Information Visualization*, 2018.

[259] John Wilder Tukey. *Exploratory data analysis*. Addison-Wesley series in behavioral science: quantitative methods. Addison-Wesley, Reading (Mass.), 1977.

[260] Tom Horak, Ulrike Kister, and Raimund Dachselt. Presenting business data: Challenges during board meetings in multi-display environments. In *Proceedings of the ACM International Conference on Interactive Surfaces and Spaces*, pages 319–324. ACM, 2016.

[261] Keith Andrews and Ale Smrdel. Responsive Data Visualisation. In *EuroVis 2017 - Posters*. The Eurographics Association, 2017.

[262] Geoff Cumming. The new statistics: Why and how. *Psychological science*, 25(1):7–29, 2014.

[263] Publication manual of the American Psychological Association, 2016. https://www.apastyle.org/manual/, accessed May 2019.

[264] Gail M Sullivan and Richard Feinn. Using effect size—or why the P value is not enough. *Journal of Graduate Medical Education*, 4(3):279–282, 2012.