Visfer: Camera-Based Visual Data Transfer for Cross-Device Visualization

Information Visualization
XX(X):1–19
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Sriram Karthik Badam and Niklas Elmqvist

Abstract

Going beyond the desktop to leverage novel devices—such as smartphones, tablets, or large displays—for visual sensemaking typically requires supporting extraneous operations for device discovery, interaction sharing, and view management. Such operations can be time-consuming and tedious, and distract the user from the actual analysis. Embodied interaction models in these multi-device environments can take advantage of the natural interaction and physicality afforded by multimodal devices and help effectively carry out these operations in visual sensemaking. In this paper, we present cross-device interaction models for visualization spaces, that are embodied in nature, by conducting a user study to elicit actions from participants that could trigger a portrayed effect of sharing visualizations (and therefore information) across devices. We then explore one common interaction style from this design elicitation called Visfer, a technique for effortlessly sharing visualizations across devices using the visual medium. More specifically, this technique involves taking pictures of visualizations, or rather the QR codes augmenting them, on a display using the built-in camera on a handheld device. Our contributions include a conceptual framework for cross-device interaction and the Visfer technique itself, as well as transformation guidelines to exploit the capabilities of each specific device and a web framework for encoding visualization components into animated QR codes, which capture multiple frames of QR codes to embed more information. Beyond this, we also present the results from a performance evaluation for the visual data transfer enabled by Visfer. We end the paper by presenting the application examples of our Visfer framework.

Keywords

Collaborative visualization, cross-device interaction, embodiment, sensemaking, software toolkits

Introduction

Visualization is increasingly spreading to multi-device settings, where separate devices—such as smartphones, tablets, laptops, wall displays, and tabletops—are used to show interactive visual representations.[1–3] This is known as *cross-device visualization* and is often used for collaborative sensemaking, where several analysts work together on a sensemaking task.[4] The motivation is simple: we are surrounding ourselves with an ensemble of digital devices capable of networking, computation, and high-performance graphics, and it makes sense to employ all of these devices for *ubiquitous analytics*:[5] sensemaking anytime and anywhere. However, even if frameworks for such ubiquitous analytics are beginning to appear in the literature,[6] building such environments is still challenging due to the need for fast and efficient methods for device discovery, view management, and interaction handling. Furthermore, existing frameworks mostly fail to capture—let alone leverage—the embodied nature[7] of a physical cross-device visualization space: the fact that the analysts are there, *in situ*, in the environment and are navigating physically in relation to the visualizations and devices.[8]

The fundamental operation for cross-device sensemaking activities including device discovery, view management, and interaction sharing, is the transfer of information[9;10] between devices such as between a wall-mounted display and a handheld mobile device. For example, view management can involve transferring a part of the large display to a small display (as described by Badam et al.[9]), while interaction sharing can be seen as capturing activities on one display and sending the corresponding data bindings to other displays.[6] To develop embodied interactions[7] to carry out this operation, we first elicit interaction designs through a user study by placing novice computer science students within fictional visual exploration scenarios. We present the observations from this design elicitation as a conceptual framework for cross-device interactions based on the proximity of devices and the sensemaking task.

A common interaction mechanism for cross-device visualization from our user study was based on actions that capture the visual focus of target devices (similar to taking a picture). Motivated by this, we propose VISFER (VISualization TransFER): an interaction technique for cross-device visualization environments through the use of QR codes decorating each of the component visualizations within the interface. The idea behind Visfer is simple: the user can capture a fully functional version of a visualization from a display to their handheld device simply by taking a picture of it—or, more specifically, of the QR code associated with the visualization. This simple physical

University of Maryland, College Park, MD, USA.

Corresponding author:
Sriram Karthik Badam, 2117 Hornbake Building—South Wing, University of Maryland, College Park, MD 20742, USA.
Email: sbadam@umd.edu

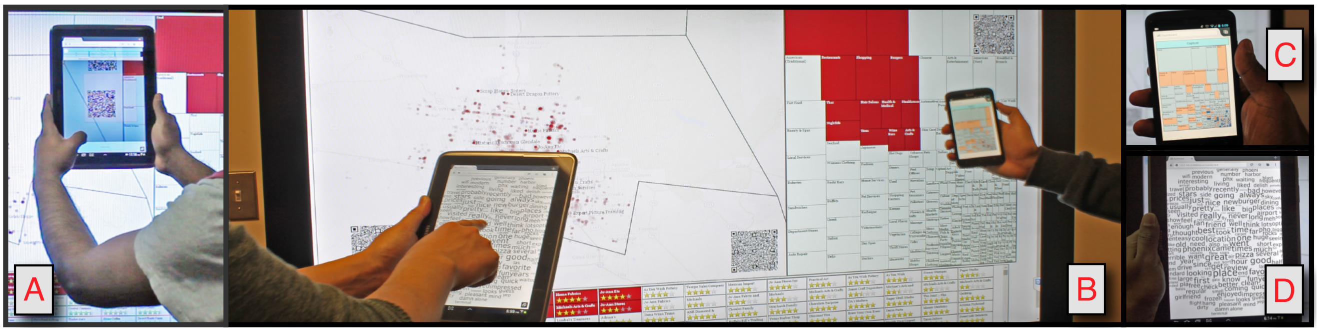


Figure 1. The Visfer framework enables transferring visualizations across devices to support cross-device visual sensemaking. (A) User captures a QR code; (B, C, D) visualizations loaded on personal devices from the large display. Transferred visualizations are adapted to the target device.

gesture draws upon current common practice for millions of mobile users—pulling out your phone to take a picture of something worth remembering, sharing with friends, or capturing for posterity—that has become a natural part of the vocabulary of smartphone usage. The gesture is also popular as a depiction of physical computing in movies such as *Minority Report* (2002) and *Iron Man* (2008). 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*[5] 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.

We have implemented our Visfer framework as a web application toolkit in JavaScript that is integrated with the standard web visualization toolkit D3.[11] To enable the transfer of large amounts of data using QR codes, we propose the use of *animated QR-codes*: a QR code with multiple frames of data that is animated over time,[12] increasing the information-carrying capacity of the code. We implemented a reader and generator of such animated QR codes as part of the Visfer framework using animated GIFs, making it trivially compatible with existing web browsers. As part of the validation of our toolkit, we tested the performance of this QR GIF reader, and found an average transfer rate of more than 500 Bytes/sec for an animated QR code with 10 fps. While this is not comparable to the typical network connection bandwidths, we found that this provides a sufficiently fast visual transfer of information for our application examples and scenarios.

Overall, the contributions of this paper include,

1. Embodied interactions for transfer-related tasks across devices during visual sensemaking in multi-device environments based on a design elicitation study.
2. Visfer framework for camera-based visual data transfer using QR codes for cross-device visualization and multiple application examples that take advantage of this framework.
3. Results from a performance evaluation revealing the performance of our framework and its effective bandwidth, as well as a comparison to previous techniques for visual data transfer.

Usage Scenarios

As Chung et al.[13] pointed out, the advantages of cross-device workspaces include (1) providing additional display and interaction space to enhance the visual perception and spatial interaction, (2) supporting collaboration among users by satisfying their individual analytical processes, and (3) allowing opportunistic use cases to take advantage of specific technologies for suitable tasks. There are many applications for the multi-display environments and cross-device interaction techniques (e.g., visual data transfer via QR codes) presented in this paper:

Collaborative Data Analysis in Office Settings

A well-established example for multiple devices in the data analysis space involves multiple users working together in the same location.[9;13;14] Consider a group of experts studying traffic flows within a city. Traffic data has multiple facets with real-time feeds, historical trends, and variable information such as signals, transit options, and weather. Modern visualization platforms* that support traffic data analysis store this data on a server and fetch multiple visualizations of the data when needed. It is however challenging to develop insights from this dataset by just showing these representations on a single interface for a single analyst. In this scenario, a real-time traffic feed is presented on a shared large display with contextual information shown on handheld devices. Analysts can connect this contextual information, such as weather, construction activity, and historical/seasonal trends, with patterns shown on the large display through cross-device interaction. Furthermore, multiple experts can quickly work together by focusing on specific geographical regions. They can take advantage of the visual data transfer mechanism by taking a picture to quickly extract information from the shared large display containing the real-time feed and combine it into the visualization space on their personal device. They can also use it as a way to share information

*RITIS: http://www.cattlab.umd.edu/?portfolio=ritis with other analysts surrounding them by just letting them take a picture with their handheld device. Physical interactions, involving an explicit action for transferring information, can help maintain awareness for the analysts within this co-located space and further help them coordinate the analysis. For instance, in air traffic control, MacKay[15] identified that physical operations on paper strips containing flight information aid the social processes in a control room, acting as a means for non-verbal communication, coordination, and cooperation between controllers.

Dedicated visualization environments also contain user and device tracking mechanisms that can further help utilize physical navigation and spatial awareness of the users, and enable collaborative visual exploration. In such visualization spaces, the content transferred during visual data transfer can contain (or link to) the visualizations and the user interactions during visual exploration.

Casual and Serendipitous Workspaces

These scenarios include opportunistic use of devices to support collaborative visual analytics.

- Consider a group of business analysts discussing a planned stock acquisition around the water cooler: one analyst shows a new projection that she has been working on her tablet, and the other analysts can quickly take pictures of the financial visualization to acquire the new proposal to their smartphones without having to bother about sharing URLs via email or instant messaging.
- Consider a casual traveler coming across a retirement savings visualization on an electronic billboard in the airport: the traveler can easily grab an interactive version of it by capturing the QR code without ever connecting to a remote and untrusted cloud server.
- Consider an interactive installation on urban issues in a public city square: passing citizens can download and view interactive visualizations of their local community merely by snapping pictures of the charts that interest them, allowing them to study and interact with the data on their personal devices.

In these public scenarios, there is a need to avoid logins and going through untrusted sources to *download* the content. At the same time, as identified by Isenberg et al.,[16] it is not unusual for large groups of people to perform these interactions in public settings. As such, the user experience should not be affected (due to delays) in the presence of multiple users performing the same interaction at the same time. In such cases, the content transferred during visual data transfer should be enough to recreate the visualization.

Public Presentations

During a lecture in a classroom or a presentation at a conference, there are often multiple devices owned by the audience. The learning experience of the audience can be enhanced by allowing them to test the content being covered. For example, a visualization lecture can be made more engaging by allowing the students to extract a visualization from the presentation and explore it on their personal device (e.g., to change the visual representation or add interaction components to it). With a visual data transfer mechanism for cross-device interaction, they could just take a picture of the current slide (by zooming with the camera if needed) and get the visualization content of the slide directly on their computer. However, this scenario can span settings where a dedicated server to fetch and serve visual representations or even a fast internet connection may not be present (e.g., at a conference with thousands of attendees). At the same time, the audience should still be engaged in the presentation and should not be going through logins or indirect URLs that could deviate them. In these presentation spaces, the content shared during the visual data transfer itself can contain the visualization pipeline (or even the code) and a sample small dataset to recreate the visualization, which could be manipulated by the audience on their computers.

Background

Our visual communication technique for cross-device visualization was inspired by existing cross-device interaction models for using multiple devices together, approaches for enabling visual sensemaking beyond a single desktop computer, and finally existing methods for visual data transfer through screen-camera communication. Here, we review research in these areas and highlight specific inspirations. Considering that a major focus of this paper is on developing cross-device interactions for visual exploration, we start with a review of existing cross-device interactions in general HCI.

Cross-Device Interaction in HCI

With the recent surge in smart devices—smartphones, tablets, smart eyewear—cross-device interaction to share information, chain tasks, and manage sessions across devices has become popular.[17] Pick and Drop[18] was one of the first cross-device techniques to exploit the physicality of large displays and mobile devices in an environment using a pen. Hinckley et al.[19] presented a cross-device interaction technique called stitching to interact with multiple mobile devices. Duet[20] enables joint interaction across watch and phone using multi-device gestures (for instance, flip watch and tap phone). More recently, WatchConnect[21] toolkit was created for rapid prototyping of cross-device applications for smart devices through a rich set of input and output events that are created from on-surface, over-the-surface, and proxemics-based interaction. In workspaces with large displays, SleeD[22] uses a sleeve display to interact with a large display wall. Compared to hand-held devices, a sleeve display allows free use of both hands, thus improving physical coupling between the displays.

In the past few years, there have also been many frameworks developed for cross-device interaction across smart portable devices and large displays. Panelrama[23] supported creation of cross-device web applications by splitting views and synchronizing interaction across devices. The Conductor[17] and WatchConnect[21] frameworks are built with specific low-level cross-device interactions in picture, however, they do not fully extend to complex device coupling scenarios that need interaction flow and output display management. For example, supporting private and public interaction spaces during an activity requires control over when the interaction is synchronized and how display information is transferred across devices.[2] PolyChrome[9] provides framework-level support for creating these hierarchies in different collaboration modes (synchronous vs. asynchronous, co-located vs. distributed), while managing concurrent use.

Visualization beyond the Desktop

Large displays have been shown to improve productivity in office settings.[24] For sensemaking through visualization, they provide a large space to think[25] and support better collaboration[4] between analysts, which in turn leads to better hypothesis generation. Bradel et al.[26] studied the spatial and territorial behaviors exhibited by users when working with a large display using document analysis tools. Apart from large displays, tabletop displays have been used for creating visualization systems for tree comparison,[27] collaborative document analysis,[28] and mixed-presence collaboration in general.[14] For tabletops, interaction techniques within the physical space around the display have been developed using tangibles that can be freely carried around such as physical transparent lenses[29] and paper lenses. This was further extended to create *graspable* tangible views.[30] Isenberg et al.[16] compared the tabletop interfaces in office workspaces and public settings such as museums.

In the large display environments, physical navigation (moving eyes, head, and body) is found to be more efficient and preferred than virtual navigation (zooming and panning).[8] To explore this pattern, the use of *proxemics*[31]—the social relationships between users and objects in an environment—has been proposed for interacting with visualizations on large displays.[1] This relates to their distance, orientation, position, movement, and identity.[32] Badam et al.[33] designed proxemic interactions for performing simple UI tasks in a visualization interface on a wall-sized display, and evaluated them against gestural interaction to create a balanced model. Kister et al.[34] presented the concept of BodyLenses, an egocentric interaction style through magic lenses controlled by body movement. Furthermore, the effects of this body movement and physical navigation in front of a large display on the human perception of visual encodings have been studied by Endert et al.[35]

An alternative is to develop interaction that is more direct in nature through gestures (for multi-touch or in 3D space). Andrews et al.[36] discussed the ability of gestures such as pinch and two-handed lateral selection to replace traditional control panels in InfoVis. The aforementioned work does not apply to visualization across devices, which brings about its own advantages. McGrath et al.[2] found that the ability to branch into a private interaction space on a local display and merge when needed empowers users to freely collaborate in a co-located space. However, their work was focused on the usage patterns of these spaces rather than how to build them.

Ball et al. applied embodied interaction (EI) models—interaction based on our familiarity and facility with the everyday world—to visualization on large displays.[37;38] They found that EI 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. These devices were also rated to be most preferred. Andrews and North[25;39] 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 position has a meaning to the analyst.

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.[40]

Camera-Based Discovery and Communication

The development of spatially immersive displays[41] through multiple projectors and device displays triggered the use of cameras for discovery and calibration of the displays in the 3D environment,[42] and furthermore, for stereoscopic vision and gestural interaction. While latter applications are very popular in modern HCI systems utilizing commercial depth cameras such as PrimeSense Carmine, Microsoft Kinect, and Leap Motion, we are more interested in the former for using cameras for device discovery and transfer of its content. For device discovery, popular methods include using visual tags/patterns[43] and fiducial markers for tracking objects.[44;45] Alternative methods include using vision-based algorithms to discover the device silhouttes.[46] Rohl and Zweifel[47] introduced a conceptual framework for interaction using camera phones and visual codes. They presented interaction models based on pointing, rotating, tilting, distance, and movement of the camera phone in front of a visual code, and studied the usability of the interactions from these primitives.

Our technique for cross-device visualization targets the use of built-in cameras for data transfer between devices during the visual sensemaking process. In the past, Hesselmann et al.[48] used the built-in cameras (and flashlight) on mobile phones to establish a communication channel with a tabletop when the phone is placed on it. They establish this connection through a color-based encoding directly underneath the phone on the tabletop. They further used an external camera on the tabletop and the flashlight on the phone to create bidirectional communication. Langlotz and Bimber[49] introduced 4D barcodes by encoding data in four dimensions: width, height, color, and time, for visual communication. The participants from their user study criticized the decoding time for this representation, but gave positive usability ratings overall. Animated QR codes have been utilized in the SENSeTREAM approach by Yonezawa et al.,[12] to augment videos with captured sensor data (e.g., the performer's movement within the video) embedded within the QR codes (in each animation frame). As they describe, this approach promises a high theoritical maximum of 88,590 bytes/sec for a 30fps animation and 60fps camera capture, but there are many factors in reality such as QR size, lighting conditions, and the camera parameters that significantly change the transfer rates. One of their application scenarios includes an augmented TV experience where smartphones and tablets are used as a second screen to visualize human-motion graphics by decoding the animated QR codes, to enhance TV sports and musical programs.

More recently, HiLight[50;51] bypassed using barcodes to directly transfer information by hiding it in the transparency channel of the displayed computer graphics (e.g., images and videos). The throughput of this approach was influenced by the screen-camera distance, environment factors, foreground image colors, and hand motion. The foreground images and videos were also affected when encoded with information. For cross-device visualization, we considered these obtrusive (with codes) and unobtrusive approaches (using transparency) for visual communication and decided to use animated QR codes as they promise a high theoretical transfer rate without being too obtrusive[12] or changing the foreground visualizations themselves.[51]

Embodied Interaction for Sensemaking

Sensemaking as a process involves developing insights from information for decision making. This can happen in a range of application scenarios in office spaces and public settings (as introduced in Usage Scenarios section). Multi-device environments containing a set of heterogeneous input and output devices including large wall-mounted displays, tabletop displays, portable tablets, and smartphones can (1) aid in the analytical process by providing more display and interaction space and (2) also facilitate collaboration among analysts, where goals, hypotheses, observations, and insights developed during sensemaking are coordinated within the group. However, such sensemaking support requires methods for leveraging the input and output modalities,[13;33] providing flexible means for coupling analysts' work in a group,[52] support coordination mechanisms among analysts to manage shared resources without conflicts,[53] and support development of territories common in group activity.[26]

Multi-device environments are advantageous due to their natural support for some of these aspects. Having heterogeneous devices in the environment can better support the visual exploration process (e.g., large wall-sized displays improved quality and breadth of insights[54]). Due to the presence of multiple devices, the conflicts are mitigated to a good extent when analysts work in groups as they can work on their individual devices, and share their findings when they want to. In co-located collaborations, spatial aspects such as position and orientation of analysts can be used to understand which analysts are working together,[32;33] and provide adapted UI features for coupling their interactions. Similarly, large displays naturally facilitate creation of territories,[26;34] while these territories span across devices in multi-device environments.[2] Finally, the users' body can itself be used to create contextual interactions.[55]

These particular aspects support specific user needs during the sensemaking process in multi-device environments. Physical navigation techniques take advantage of the fact that the analysts are situated within the environment along with the devices to provide different types of control over the information based on the context—a user can remotely interact with a large display or even directly manipulate the information when close to the display. Branch-Explore-Merge protocol[2] uses the personal and public displays to allow a flexible coupling style where analysts use personal devices as their territory to explore the data by themselves and then coordinate with others over the public displays. Furthermore, fluid interaction[56] in these contexts can help seamlessly share information—hypotheses, observations, or insights–from one device to another. Existing frameworks[9;10] for collaborative sensemaking support data *pushing* operations to send the interface content of a remote user's device to the shared display space during collaboration.

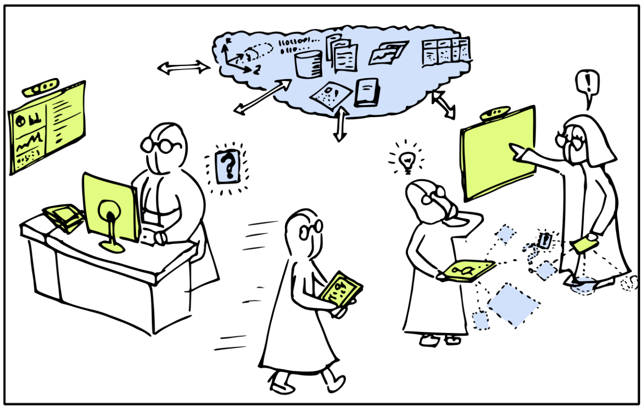


Figure 2. Sensemaking in multi-device environments. Analysts can take advantage of the unique device technologies, collaborate with others, and also flexibly work from different physical locations (on the field or in an office) at any time. This phenomenon was called ubiquitous analytics.[5] Figure reproduced with permission from Elmqvist and Irani.[5]

At a device level, the devices used for sensemaking can have different roles based on the context. Handheld devices act as private interaction spaces when *branched* from the public display,[2] aid many unit tasks to filter content and change visualization parameters on other displays[57], and create additional views into the data presented on a large shared display.[30] Beyond this, the large displays themselves have multiple roles aiding the analytical process of a single user or multiple users. To associate these roles to devices and still maintain a fluid interaction platform for visual sensemaking,[56] the intent of the user to perform a particular operation should be seamlessly conveyed to the system. By formulating cross-device interactions that ideally convey the user intent, we can developed systems for visual sensemaking directly based on the ubiquitous computing paradigm to enable analysis of data anywhere, anytime, and over any device[5] (Figure 2).

The above goal can be achieved by developing cross-device interaction models that use the physicality and spatial nature of the devices spread around the environment to convey user intent and expected outcomes of interactions. Embodied interaction enables this to an extent by exploiting the embodiment of the devices in the environment—the participative status in the physical and social world.[7] As defined by Paul Dourish, embodied interaction exploits our familiarity and facility with the everyday world. It relies on tangibility (physicality) of the interaction medium as well as the social aspects from how we experience such an interaction in the everyday world.

Over the past decade, the principles of embodied interaction have been applied to various domains in HCI focusing on learning,[58] gaming, and sensemaking.[37;38] The gaming industry has explored this interaction through devices such as Nintendo Wii, PlayStation Move, and Microsoft Kinect to leverage the player's innate knowledge and skills of the physical world.

There are two distinct patterns of exploiting physicality and social familiarity in interaction models between devices:

1. **Implicit interaction:** The physical attributes of the devices such as their presence, position, and orientation within a space are used as implicit triggers for interaction.[21;59]
2. **Explicit interaction:** Explicit actions by the user such as touch, tap, and drag actions using the devices, are used as input to the multi-device system.[2;17;19;20]

One of the goals for this paper is to develop embodied interactions for visual exploration across devices in multi-device environments. We are interested in knowing (1) what kind of interactions the users would perform for transferring information from one device to another, and (2) would these interactions differ for different physical contexts within the multi-device environment and visual exploration tasks.

Design Elicitation: Formative Evaluation

We conducted a formative evaluation to elicit interactions that enable the use of multiple devices for different tasks in visual sensemaking. This study was conducted with a protocol similar to Wobbrock et al.,[60] 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.

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 of this study 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.

Apparatus

Multi-device environments 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[†] (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 within our study.

Methods

We identified three effects when coupling a large display with a handheld device during sensemaking. Table 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). 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 (or the embodiment) 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 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 2). Sessions lasted for less than 20 minutes. This procedure is similar to the user study conducted by Wobbrock et al.[60] to develop a taxonomy of tabletop gestures.

Results: Cross-Device Interaction Patterns

The main cross-device interactions that were brought up in our study are highlighted in Figure 3.

Interaction Styles: Participants designed three cross-device 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 (similar a remote laser pointer). 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 filters it (with distance mapped to size of the region selected). Participants coupled these interactions with an explicit tap on the handheld device to actually trigger the effects after the physical 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

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

Table 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	Overview or aggregate a visual representation from large display on the smartphone.

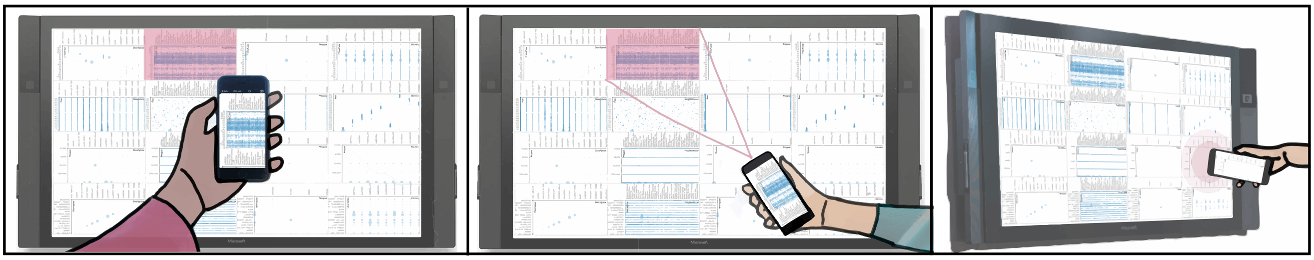


Figure 3. Three cross-device interactions suggested by the participants during our design elicitation study. Participants suggested holding the mobile device vertically to capture a region of interest (pink) in the field-of-view (left), using the handheld device to point to a region on the large display (middle), and using the handheld device to tap a visualization when close to the large display (right).

Table 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?

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 from the large display (middle), participants (all except P1, P5, P7) preferred holding the phone vertically to grab a region.

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 (this connects to **bi-directional communication** between the displays).

Cross-Device Interaction for Visualization

Based on our user study, we found three distinct 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 3). 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 multi-device environments.

Visfer: Visual Data Transfer

Cross-device visualizations are visual representations that are distributed across two or more displays and/or interaction surfaces. This form of visualizations has been previously used to combine a public tabletop display with individual private mobile displays for visual sensemaking.[2] These representations enable analysts not only work efficiently with each other, but also physically with the interaction surfaces including wall-mounted displays, tabletops, tablets, and smartphones. To develop these cross-device visualizations, we need methods to share representations across devices.

As identified in our 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 (Figure 4). 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.[11]

To support our motivating usage scenarios through the Visfer framework, we define design guidelines that guide the content and position of the QR codes on the visualization interface, and enable fluid interaction[56] and spatial interaction models[1;61] 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 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.[62] 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 to the user. When created, they should not occlude important 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 placed manually using a drag-and-drop user interaction. It should be possible to resize or remove them to save some display space. This guideline also makes the QR code based visual data transfer 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,[2] they should be perceivable and interactive on any device modality. For this purpose, we adapt Thevenin and Coutaz's notion of *plasticity*:[63] 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[56] and engagement of the analyst by expanding the notion of plasticity further based on standard visualization tasks,[64–66] such as creating an overview, considering details, and linking patterns across data. For example, it should be possible to use a smartphone as the medium for overviewing data visualizations on a large display. This type of adaptive visualizations is defined by Elmqvist and Irani[5] 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).[61] This information can be used to control visualization parameters such as zoom and detail levels. Similar interactions have been proposed using proxemic relationships[31] for updating visualizations based on the user's spatial attributes.[1] This can promote physical navigation among users in the multi-device environment.

The Visfer Framework

The philosophy behind the Visfer technique is to support cross-device and collaborative visualization spaces by

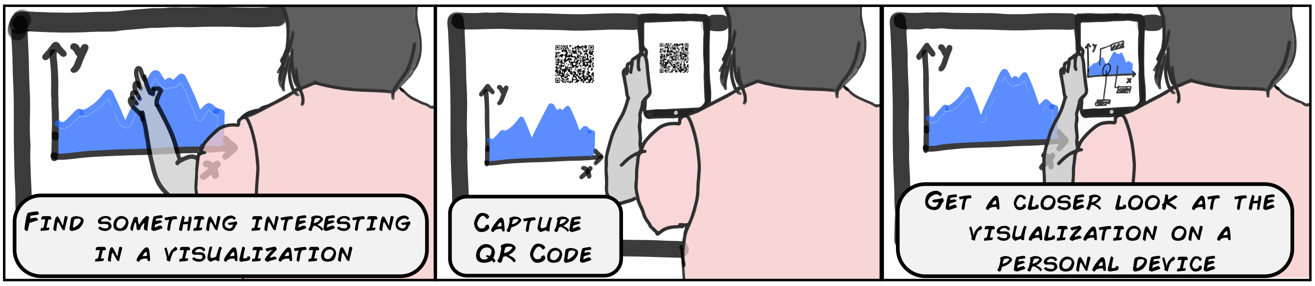


Figure 4. The simplest Visfer interaction scenario is when interacting with a large display, a user can simply take a picture of the QR code augmenting a visualization and load it on a personal tablet for further visual exploration.

visually transferring information between devices (large display to mobile, and mobile to mobile) using built-in cameras on smartphones and tablets. For example, as seen in Figure 4, a user interacting with a large display can simply take a picture of the QR code attached to a visualization and get a closer look at it on a personal device, while still keeping the large display visualization intact. To prototype this technique, we developed the Visfer framework to create cross-device visualization spaces over the web based on the design guidelines described in the previous section.

The Visfer framework is built upon the D3 toolkit[11] for web-based visualization. Visfer provides modules to augment web visualizations with QR codes through a QR generator. On a personal mobile device, the framework provides access to the device camera directly from the web application, along with a QR decoder (Figure 5). The framework also supports *plastic visual representations* and provides options to (1) transform the captured visualization by default to the target device modality (input and output capabilities), (2) transform some standard visualizations directly based on the InfoVis tasks being performed by the user, and (3) support explicit transformation logic from the application developer and the end-user (analyst).

The motivating usage scenarios for Visfer typically contain different types of infrastructures. Co-located collaborative sensemaking by analysts in an office setting can have the necessary server-side technologies to store the data, generate dynamic links to access all the visualizations, and keep track of the visual exploration of all the users. This scenario may require a coupling between devices that needs bidirectional communication to allow the analysts have a private visual exploration space on the smartphone and also coordinate with other collaborators through the large display. On the other hand, a more opportunistic use case at an airport or at a public square does not require bidirectional communication but rather just the flow of the information from the public display to the personal device. To support these different sensemaking scenarios, the framework supports three types of content within QR codes for cross-device visualization. Figure 5 shows these content types (levels 1-3) along with the features supported by each.

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[67] 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

[‡]Vega: https://vega.github.io/

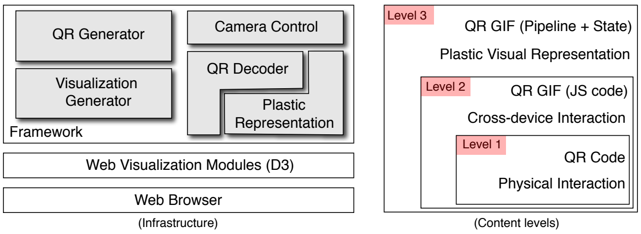


Figure 5. Visfer framework infrastructure and the three content levels along with framework-level support provided.

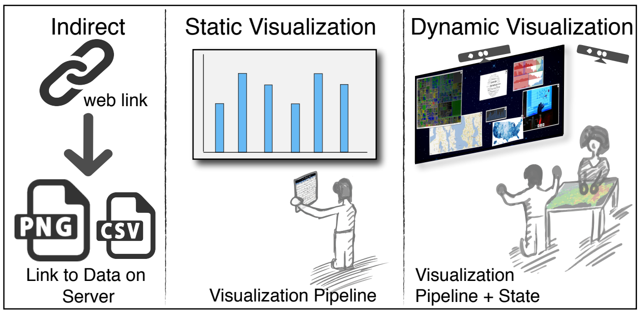


Figure 6. Types of content communicated in a cross-device visualization setting. This includes transfer of arbitrary data through a link, static visualizations, and interactive dynamically generated visualizations.

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 (based on user interaction), 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.

Note that these three levels in the framework capture three types of information that are needed to share visualizations across devices in different application settings. These levels have different infrastructure requirements as well—level 1 needs a server and level 2 and level 3 need a high-resolution camera on the phone quickly capture the animated QR codes. However, these levels have an inheritance structure (explaining the hierarchy in Figure 5); for example, the JSON representation of level 3 can be embedded into a URL by storing it on a server (which was introduced in level 1). Overall, by supporting these three content types and QR representations (Figure 6), we can create a common interaction style for a wide range of usage scenarios. The difference between these levels is also the additive design considerations supported by the framework. For Level 1, the application developer has the complete responsibility to handle the transferred information across devices by initiating the framework methods to encode/decode content maintained by a server and transforming the visualizations using the framework or by explicit application logic. Level 3 can utilize higher framework capabilities to automatically create plastic visualizations that are responsive just by reading the custom JSON representation within the animated QR codes. The common aspect among them is the embodied interaction of taking a picture by holding up a phone (as identified in our design elicitation study), made possible through the use of QR codes.

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.

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.[67]

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[67] 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

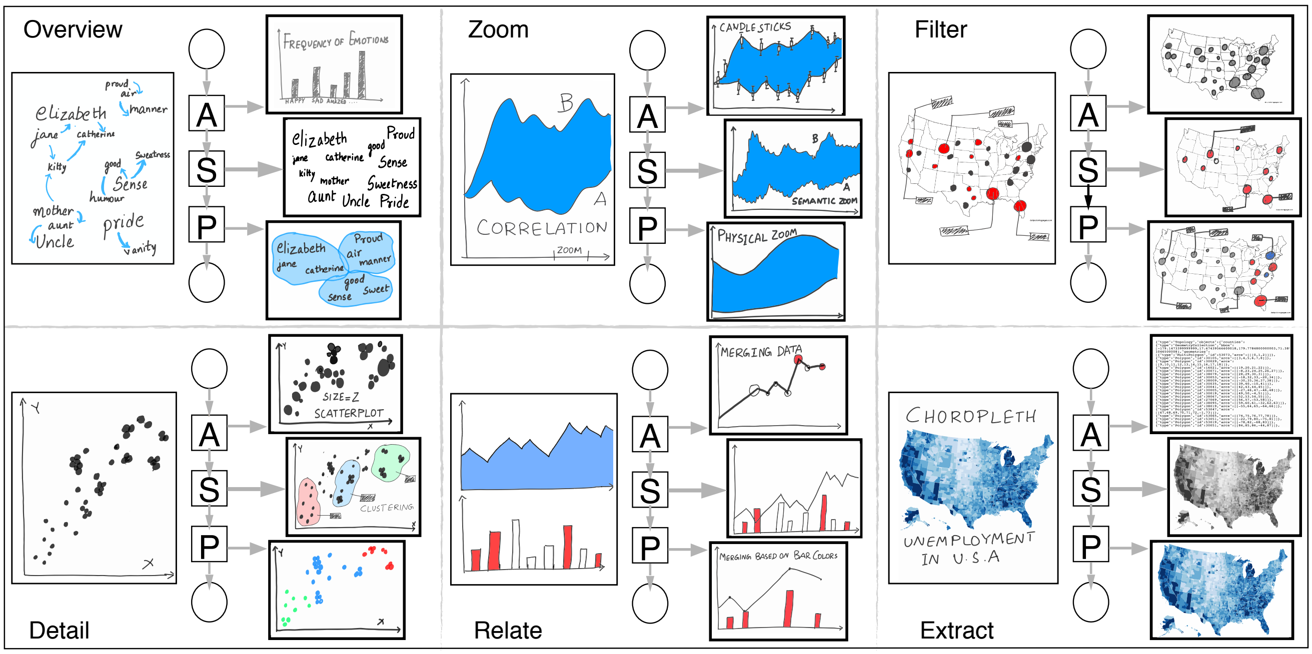


Figure 7. In this figure, A, S, and P stand for the analytical abstraction, spatial layout, and presentation layers in a visualization pipeline. Plastic visualizations are created by modifying the pipeline by branching out from any of these layers to create new and interesting visual representations 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 (based on Vega's visualization grammar) of the visualization pipeline and state.

more readable. The interaction definitions are translated to the input type on the target: converting mouse interaction handlers to touch and vice versa.

Adapting to Visualization Task. To further extend plastic representations, the attributes within the JSON representation should be transformed to fit the visualization tasks[65] being performed by the user of the cross-device visualization. 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,[68] 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 provides examples of these transformations for each visualization task. Here, we describe the conditions under and mechanisms through which these transformations are handled by the framework, or through explicit specification from the Visfer application developer.

- **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 application developer logic to define the new abstraction.
- **Zoom:** The framework allows semantic[69] 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, analyst) using the Visfer applications.
- **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 selections 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/hybrid visualizations[70] 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, which requires specification from the application developer. The relate operation at an abstraction layer requires definition of the new abstractions and is handled by the application developer.
- **History and extract:** By maintaining the visualization states (from the QR codes), the framework supports storage and extraction of historical states of the visualization collected during collaboration.

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.

Spatial Awareness

The use of visual markers (QR codes) allows for a low-fidelity tracking of the spatial attributes—including screen-camera proximity and orientation—that could be used for creating spatial interactions. The parameters during overview, zoom, filter, and showing details (visualization tasks) can be associated by the Visfer's application developer to the device position and orientation deduced from these spatial attributes. For example, when showing more details, the proximity to the large display can define the level of detail provided—being close can add an additional layer to the visualization with colors or annotations, while being far from the display can create a new visual representation.

As Jakobsen et al.[1] identified, some interesting uses of these proxemics data include adjusting level of detail on visualizations, controlling aggregation, and selecting attribute values within visualizations. Examples of using the spatial attributes from our particular cross-device setting include (1) for the overview transformation, the QR code sizes can be used as a way to determine the binning parameters; (2) the orientation of the QR codes can determine the zoom parameters and the position to zoom in the 2D space; and (3) spatial aspects of the QR code can determine where the user is physically located, which can be used to control the type of visual representation shown. However, as mentioned earlier, this is a low-fidelity measure and may not always reflect the actual proxemics of the users and the devices since the users can freely move around in front of the large display when taking pictures.

Developer and End-User Controls

The Visfer framework prototypes the camera-based visual data transfer via QR codes during visual exploration across devices. While doing so, it further supports visualization tasks[65] by adapting Visualizations to the target device and the task itself. Furthermore, the spatial awareness creates a way to extend the camera-capture action to a more spatial interaction style by utilizing the low-fidelity measures of the user proxemics. Overall, these three aspects of our framework need to be controlled by the developers based on their applications, and further utilized by the end users to ease their visual exploration process. Here, we reiterate the specific controls left within the framework to the developer and the end users of Visfer applications.

Firstly, the three levels of QR content open up the space for more flexible usage scenarios based on the available infrastructure and the application setting. In most scenarios, the choice of the levels is made by the application developer. For instance, in casual and serendipitous scenarios, the developer can choose level 2 or level 3 QR codes (QR GIFs) to isolate the data transfer mechanism from maintaining the content on a server. However, there can also be application scenarios where multiple levels of content might be used. This is common in hybrid application scenarios where visualizations are transferred from one setting to another; from instance, when the insights from a co-located sensemaking session are opened up to a general audience who are outside the sensemaking environment. In this particular case, the end users can control the QR content by switching from embedding URLs, to creating animated QR codes with specific data of interest that could used to transfer the visualizations outside the sensemaking environment. From a user interface perspective, this end-user control can be a control option through a button or a menu that changes the mode of QR encoding. Furthermore, the control over which visualizations are augmented with QR codes for cross-device interaction, as well as the size and other properties of the QR codes, can be provided to the end user through toggle buttons, menu options, and direct manipulation.

For visualization adaptation, there are two types of controls. From a developer perspective, the layer in the visualization pipeline used to guide the adaptation of the visualization (Figure 7) needs to be configured. Once this information is encoded in the application, the end user can control the appropriate transformations based on the task being performed.[65] With this control, the user can, say, capture a QR code corresponding to a visualization and select the filter task on their personal tablet to add specific filters to the viewed content. At the current stage, the framework does not automatically figure out these adaptations; however, this is a potential direction for improvement to better support cross-device visual exploration.

Spatial awareness helps develop spatial interactions and support physical navigation in front of the large display. However, the mapping from the spatial attributes to the data shown in visualizations is not straightforward. While the framework currently captures the spatial attributes, this mapping needs to be set by the application developer using the Visfer framework. For example, when close to the large display, more details for the data items in visualizations can be shown on the personal device through additional visual encodings. Finally, entering the spatial interaction mode can by itself be explicit. The end user can control whether or not to utilize the spatial attributes for visual exploration, with a control option on their personal device.

Implementation

We implemented the Visfer framework using standard web technologies. It is written in JavaScript and currently couples with the D3[11] and Vega[67] frameworks. This means that the users could just access it by opening their web browser to a URL (hosting the web application created with Visfer) without the need for any installations. We developed the three types of QR content as discussed in the previous section. The framework is available on GitHub[§] for public use and we are developing more examples to make cross-device visualization design as convenient as using D3.[11]

In terms of the QR code content levels, the first level that can encode a link or a URL into a QR code currently requires the application developers to maintain a server component. For the second level of content design, which involves sharing the JavaScript code, the framework currently assumes that the application developers solves the dependencies in terms of the JS objects and application context required to execute the web application code on the browsers (i.e., the applications running on all devices use the same dependencies). For the third level, which promises plastic visual representations by capturing the state along with the pipeline, some transformations require explicit application-level logic or end-user control as described in the previous section. At its current stage, the framework performs transformations at the spatial layout and presentation layers, and it was used to develop the examples described in the next section. However, these transformations are not generic and we are currently expanding them to other visual representations. By studying different usage scenarios with Visfer, we also plan to further develop more implicit/automatic logics to adapt visualizations.

QR code generation. The animated (multi-frame) QR code is created by the QR generator by simply splitting the content into a predefined number of individual frames (based on the discussion given later in performance evaluation). The content in each frame is also attached with metadata about the frame number and total frame count. The QR codes can be made invisible and loaded whenever needed with a toggle button to reduce the distraction caused by the animation. Proximity sensing can be an alternative for showing/hiding the QR codes, however, this remains to be part of the future work. During the QR generation process, each QR code has error correction features implemented by the Reed-Solomon Code[71] added to the original content. This includes four levels of error correction: level L (7% content restored), level M (15% content restored), level Q (25% content restored), and level H (30% content restored). The Visfer framework uses level H correction by default, however, this can be changed by the application developer. To keep the content size of the QR codes to as minimal as possible, we used a JSON compression library[¶] that can compress up to 55% of the original content size.

QR code decoding. The animated QR codes are decoded frame-by-frame following the standard procedure.[12;72] While the correction mechanism provides a good amount of leverage in capturing it from a range of distances and orientations, the decoding process is still affected by the lighting, and camera parameters leading to frame dropping and processing delays. Furthermore, the frame rate of the animation (f_a) and the frame rate of the camera capture (f_c) should be matched for fast decoding ($\frac{f_a}{f_c} \leq 1$; $\frac{f_a}{f_c} = 0.5$ by default). Finally, auto-focus options on the camera also delay the process further.

Examples

We developed three application examples with the Visfer framework focusing on environments with one large display and a few portable multi-touch devices. These examples are available with the Visfer source code. Our most advanced example is a Yelp data visualization called BusinessVis.

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 8). 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.

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. Setting up these sharing mechanisms is quite simple, requiring instantiating the appropriate classes and methods in less than 15 lines of code for each visualization.

The interaction principle behind BusinessVis is to first support exploration of the visualizations on the large display, and then provide an additional "layer" to view other perspectives through handheld devices into the

[§]Visfer framework: URL hidden for anonymity.
[¶]jsonpack: https://github.com/sapienlab/jsonpack

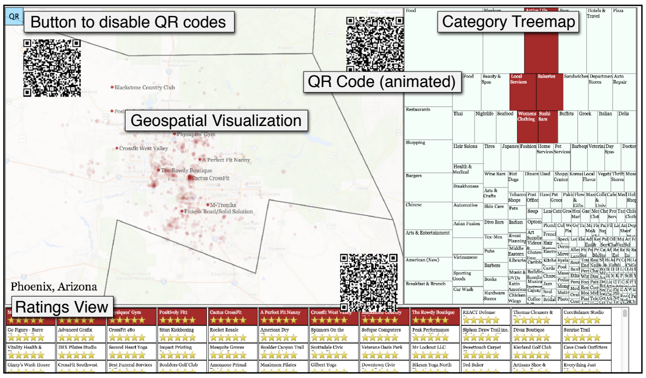
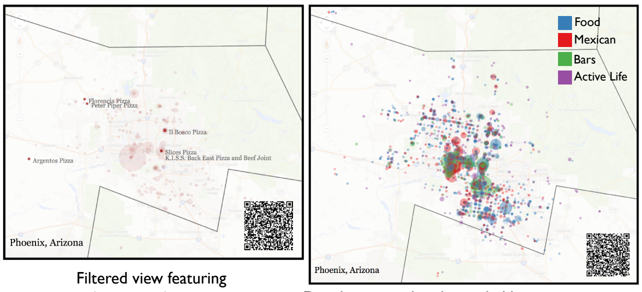


Figure 8. BusinessVis allows visual exploration of business reviews on Yelp through three visualizations. QR codes are shown/hidden in the interface when the button on top left of the interface is clicked. To further explore the data, the QR codes can be captured with a handheld device (tablet/smartphone as seen in Figure 1) by multiple users.



Filtered view featuring top 5 (best rated) Pizza places

Details view with color-coded business categories

Figure 9. Two plastic representations of the map visualization (in Figure 8). (Left) A filtered view shown on the handheld device capturing five best-rated Pizza restaurants in Pheonix. (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.

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 1). 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 9). 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 9). 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 1). 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 1). For relate tasks, the framework automatically merges the states of geospatial and rating visualizations to create a hybrid visualization. While the current implementation does not fully utilize the spatial awareness aspects, we discuss some example spatial interactions relevant to this application within the workflow to illustrate the full capabilities of the Visfer technique.

Workflow. Let's consider Eva, a business analyst, interested in understanding the public opinions about the businesses within Phoenix, Arizona. Eva can visualize this data using BusinessVis on a large display in her office. She uses her personal tablet along with the large display for cross-device visual exploration. Using the large display, she can observe the distributions of businesses on the geographical map and query it by selecting categories on the treemap or top companies on the ratings view. This helps her understand where the top businesses are located and which business categories are most common in different areas in Phoenix. However, the large display interface just gives a simple picture of the businesses based on their ratings. Eva understands that the real value of any company is often reflected in the actual reviews. To explore the reviews, she first scans the QR code of the map visualization and requests details to see the business category and popularity captured with color and size of the points on the map (Figure 9). She notices that the most popular businesses are restaurants in the Downtown area. She then selects the Downtown area of Phoenix on the large display to see the business ratings on the ratings view. She scans the QR code attached to the ratings view to get more details about the actual reviews on her personal tablet. This creates a word cloud of the popular words used to describe the businesses on her tablet without changing the visualization on the large display. To understand the reviews attached to specific business categories related to restaurants (Figure 9), she points the camera again to the QR code attached to the ratings view and then enters a spatial filtering mode on the tablet (accessed with a button click). In this mode, the business categories are grouped in alphabetical order and filtered based on the distance from the large display, which is estimated using the size of the QR code in the view of the tablet's camera. This allows for physical navigation in front of the large display, where Eva can move forward and backward in front of the large display to observe the common phrases in word clouds for different business categories. While Eva is exploring the restaurants in the Downtown area, another business analyst, Lana, picks the Pizza restaurants in a university campus in the northern part of Phoenix. Lana can extract the data items to her own tablet using the QR code attached the geographical map without interrupting Eva due to the visual data transfer process facilitated by Visfer. She can then open the filter view on her tablet for the best Pizza restaurants (Figure 9). Once she has some insights about the ratings and common phrases used in the northern region, Lana can capture the QR code on Eva's tablet to compare their visualizations and consolidate their observations. Finally, when writing a report to share their insights within their company, Eva and Lana can embed the visualizations (e.g., word clouds) that were created for specific businesses in

Phoenix through level 3 QR codes by clicking a "share and embed button on their tablets. This will create a shareable version of the visualizations along with the corresponding data filters. Now, a new analyst who wants to expand this analysis can scan the QR codes to replicate the visualizations along with the interactive filters applied. Note that due to the co-located nature of the scenario within the analytics company and the size of the dataset, the QR codes contain links to the dataset hosted on the company's public server.

We implemented many of the design choices in content and plastic representations available through Visfer in this example. Other examples are based on casual web visualizations created with the D3 [11] and Vega [67] frameworks.

HaloCloud

HaloCloud is a web application to augment legacy webpages with cross-device interaction abilities. HaloCloud can add a QR GIF to a webpage that can be captured by a portable/personal device with a camera running the web application. For example, while browsing a Wikipedia page, HaloCloud can generate a QR GIF of the text, which transforms into a tag cloud on a target device when captured. This example showcases how QR codes can be used to create a visual connection for transferring data across platforms.

The HaloCloud web application has two components: (1) a Chrome extension for capturing webpage content and creating a QR GIF with the Visfer, and (2) visualization components that can create a predefined set of visualizations—a word cloud, line chart, and bar chart—from the passed data.

Workflow: HaloCloud is useful when reading long textual articles on the web. In this application, QR codes containing data and the visualization pipeline are attached to the webpages. When a user captures the QR codes with a personal device, HaloCloud creates a cross-device experience to quickly understand the webpage content with the help of visual aids (a tag cloud of the text). The QR codes in this example are level 2 by default, as they contain the textual data in the webpage along with the pipeline for the visualization. When the user scrolls through the webpage, the QR codes become level 3, by capturing the text in the active viewing area of the user. Since this application targets a casual and opportunistic scenario where the developer cannot control the hosting of the webpage content, the textual data is embedded within the QR code itself using the animated QR representation (QR GIF).

QR-Vega

Vega is a declarative grammar language for creating and sharing visualization designs. It uses a JSON specification containing declarations of the data, scales, marks (graphical primitives), and the interaction definitions. As an application example, we were interested in connecting the Visfer framework with the Vega toolkit to transfer visualizations across devices. Therefore, we augmented the examples in the Vega toolkit to create a animated QR codes containing the underlying specifications. The decoded QR code on a target device, say a tablet, is fed directly into the Vega toolkit to create an interactive visualization on the tablet (Figure 11). The examples from the Vega toolkit include

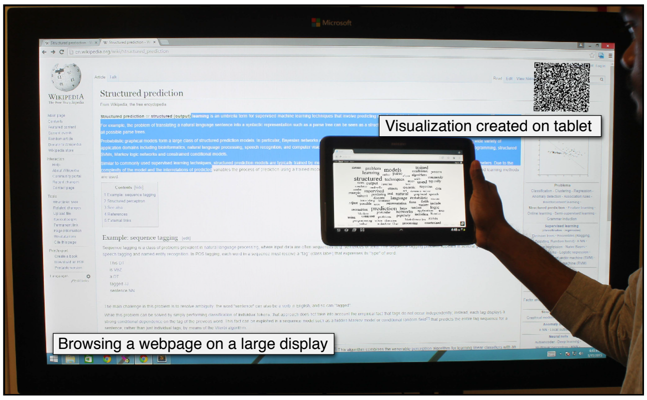


Figure 10. HaloCloud augmenting a Wikipedia page along with the QR GIF. The personal device reads the QR codes and creates a word cloud.

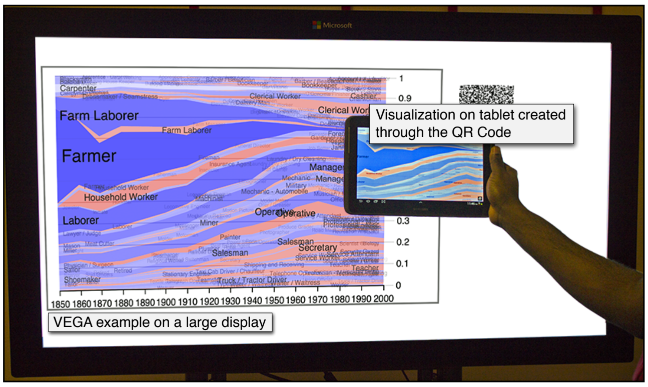


Figure 11. QR-Vega showcases QR code enabled cross-device versions of Vega toolkit examples (figure shows one of them). This image was processed to be more legible.

interactive visualizations of line charts, bar charts, area plots, scatterplots, and some abstract representations.

Performance Evaluation

As a performance evaluation of our Visfer framework, we tested the animated QR codes (QR GIFs) since they are the major component of our cross-device interaction that will affect the user experience. We recorded how long it takes for our QR decoder to read QR GIFs containing fictional data (alphanumerical) of different sizes over different frame counts (Table 3), to find a balance between the number of frames and the embedded content. Note that the user aspects of having time-multiplexed barcodes and animated QR codes have also been studied to an extent, [12;49] and they were found to be not too disruptive in terms of the viewing experience.

After testing some popular QR code readers for the Android platform, we realized that even these applications cannot read normal QR codes (single frame) that encode more than 500 characters, unless the physical size of the code itself is increased drastically. This is because QR codes with large content have closely packed patterns that are error-prone during the decoding process. In essence, there is a tradeoff between the content size and QR code dimensions for accurate visual transfer. With QR GIFs, we can circumvent this tradeoff and go beyond the regular limits of a single-frame QR code. However, the only drawback of the QR GIF is the drastic effects of missing frames during the decoding process, as there is now a waiting time to catch the frame at the next loop. While storing all the frames as they are captured before processing might help, it does not reduce the delay in the case when the camera-captured QR frame has too much error for efficient decoding.

Table 3. Read time (sec) for different content sizes and frame counts in a QR GIF, avoiding trivially good and bad combinations.

#Characters \ #Frames	1	3	5	7	9	11	13	15
300	0.38	0.90						
500	0.55	0.90	2.08					
1000		1.31	2.28	2.70	3.75			
2000		2.66	3.92	3.83	3.62	6.02		
4000				6.17	7.90	6.49	11.11	19.16
7000						10.45	14.35	21.91
10000								16.13

In the performance evaluation, we increased the content size and proportionally increased the range of the number of frames to find the ideal content per frame to get the maximum bandwidth through visual transfer. We used a HTC One X smartphone (product released in May 2012) to read the QR GIFs using a web application, created with Visfer framework, running on Firefox browser and recorded the time taken to capture and decode them. We placed the smartphone at a distance of appx. 1.5m from the large display. The dimensions of the QR codes being read are 200×200 pixels on the smartphone with 10 fps rate. For these specifications, we found that in order to get transfer rates of at least 500 characters per second, an average of 432 characters should be placed in each frame. Note that each character is of one byte size. This throughput is much lower than the theoretical maximum (88,590 bytes/sec) discussed by Yonezawa et al.[12] as the smartphone hardware and the web browser restrictions limit the decoder performance.

For this performance evaluation, we focused on a specific distance between middle and far regions in front of the display hosting the visualizations. By doing so, we ensured that the QR codes have a fixed size (200×200 pixels) on the smartphone when captured. While the effect of distance on reading performance is not evaluated, it depends on this pixel resolution of the captured QR code. With higher resolutions for the captured codes, the speed of the decoding process will increase since the chance of decoding errors that could lead to longer waiting times is less.

Discussion

Embodied interaction in a multi-device environment can allow better use of the physical space, thus, supporting better collaboration. However, little research exists on how we can fully leverage this physicality within an environment for visualization and visual analytics. Our design elicitation revealed three main cross-device interaction styles between a large display and a handheld device, (1) capturing the visual focus (field-of-view) of the handheld device covering the large display by holding it vertically (similar taking a picture), (2) pointing and drawing a region within the large display by holding the handheld device horizontally (similar to a TV remote or a laser pointer), and (3) tapping the large display with the handheld device when close to the display. Participants of our study felt that these interactions felt natural and often motivated by interactions they perform in their everyday life (like using a camera or a TV remote).

To support these interactions, we need technological support for tracking individual device positions, orientations, and field-of-view, as well as traditional interaction mechanism on each device (e.g., direct touch). This can be achieved through NFC, depth sensing, infrared tracking, and even native sensors within modern devices (e.g., accelerometer, gyroscope, camera, and pressure sensors). We focused on most common interaction suggested in our study—cross-device interaction based on visual data transfer (similar taking a picture). Instead of relying on additional hardware components to enable this, we relied on a more universal interaction of using the built-in device camera to capture QR codes in the context of visual exploration.

In Visfer, we created a distinction between the three types of content encoded in the QR codes based on the different usage scenarios. The basic content representation—a link or a URL to the content—is a generic way to create cross-device visualizations. The content connected to the link can be varied depending on the scenario, ranging from data in CSV format to state variables for synchronizing visualizations. However, it leaves development of the client-server platform, to capture each state of a visualization and generate links, to the Visfer application developer using the framework. For the other types of content, some of the usage scenarios may involve casual analytics settings where the necessary infrastructure to create client-server platforms is absent. As discussed in our usage scenarios, these can be casual sensemaking at a water cooler, at a public square, or in an airport. For this, we need a QR representation of the data that is more scalable than a simple QR code. Our animated QRs expand the application space of the Visfer framework to such analytics settings. The JSON content representation used in Visfer further supports adapting visual representations to the target device (plastic visual representation). The spatial interaction made possible by analyzing the captured dimensions of the QR codes also expands the interaction opportunities possible through our visual data transfer mechanism.

Visfer introduces the idea of plasticity for visual representations and extends its notion to support visualization tasks in cross-device visualization settings. This goes further beyond the philosophy of responsive web design. The framework also provides a structured approach for transforming visual representations based on the pipeline and task. The design space for these transformations is still complex and depends on the visual representation itself. However, there are few choices that could be handled by the framework, as showcased in the application examples. Concrete guidelines for plasticity require a deep analysis and evaluation of common visualizations, types of aggregations and sampling approaches for handling the corresponding data, and visual variations in presentation and layout attributes. This will be a significant part of our future work, along with creating the rest of the interaction techniques elicited in our user study for visual exploration across devices.

Comparison of Visfer's Performance

Our performance evaluation provides evidence that the Visfer framework can scale to even complex visualization pipelines. The bandwidth for our visual transfer is found to be 500 characters per second, which is equivalent to 4kbps. While this bandwidth is small compared to modern network connections, it was sufficient for our examples. Compared to our animated QR code implementation, past embodied visual data transfer approaches have only achieved similar or lower performance, which makes Visfer's method promising:

- FlashLight[48] enabled a tabletop-phone optical communication through a color-based encoding, leading to 33bps (with no error) and possibly up to 150bps.
- Langlotz and Bimber's 4D barcodes[49] could encode 70 characters per 2D barcode, leading to a maximum of 1400 characters per minute (23 characters/sec).
- Li et al.'s screen-camera communication[51] led to a throughput of 1.1Kbps for static foreground images when decoded with an iPhone 5s, and 6.6Kbps with a Canon 60D SLR camera.

More work is needed to improve the performance of our animated QR codes for complex pipelines, by parallelizing the decoding process (e.g., handling multiple frames at once), and developing better content representations.

Limitations

A limitation of our Visfer technique and framework is the lack of direct support for bidirectional communication unless both devices involved in the cross-device interaction have a camera. Considering that some commercial large displays may not have a built-in camera, Visfer requires an additional equipment to send information from a handheld device to the large display. Using an external camera for bi-directional communication is not uncommon.[48] With the external camera mounted on the large display, the user can hold their phone up to the large display so that the external camera can capture the QR codes. The act of showing animated QR codes on the smartphones to the external camera is also embodied (it involves physical movement and is based on the social act of showing information to another person). Another plausible solution for bi-directional communication is to maintain a web URL (if possible) to the large display visualization, and merge through the URL with a "push" gesture rather than the visual channel (as suggested by the participants of our study).

The act of taking pictures also does not fully suite continuous interactions—for example, for dynamically synchronizing views between two displays at all times, a user cannot be expected to keep taking pictures. In such settings, intervention through a server is required to react to the action of taking a picture for the first time and create a permanent connection between the two devices for synchronization, which could be stopped by the user if needed. Note that such an expectation of dynamic synchronization is unusual in casual and serendipitous scenarios discussed in this paper, and occurs more often in collaborative sensemaking scenarios in dedicated visualization environments where sufficient infrastructure exists to continuously synchronize views after the initial cross-device interaction (a handshake).

Conclusion and Future Work

In this paper, we introduced the concept of cross-device visualization for environments with multiple devices, where visual representations are inherently developed for sharing and working on multiple devices. We conducted a user study to elicit embodied interactions for cross-device visualization that take advantage of the physicality of the devices within the environment. We have presented the Visfer framework for visual data transfer, based on a popular interaction style that emerged from our study. Visfer utilizes QR code based visual communication through the built-in camera on devices. The framework also supports multiple levels of QR code content, plastic visual representations that adapt to the device, and low-fidelity spatial interaction. We have provided a detailed account of the Visfer framework, including implementation details and three application examples. Finally, through a performance evaluation, we found an ideal content per frame to reach high bandwidth of data transfer using the animated QR codes.

Our future plans include exploring plastic visualizations by studying what fits in different collaboration scenarios. We intend to understand the requirements for automatically transforming visualization based on the user activities when transferred across devices. Apart from creating more examples of Visfer, we intend to explore other communication techniques (for instance, through NFC) and develop more cross-device interactions for sensemaking in multi-device environments.

Acknowledgements

The authors thank the reviewers for their feedback during the review cycle. They also thank Dr. Senthil Chandrasegaran for his feedback that further helped improve this manuscript. This work was partially supported by US National Science Foundation award IIS-1539534. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the funding agency.

References

1. Jakobsen MR, Sahlemariam Haile Y, Knudsen S et al. Information visualization and proxemics: design opportunities and empirical findings. *IEEE Transactions on Visualization and Computer Graphics* 2013; 19(12): 2386–2395.
2. McGrath W, Bowman B, McCallum D et al. Branch-explore-merge: Facilitating real-time revision control in collaborative visual exploration. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. pp. 235–244, 2012.
3. Roberts J, Ritsos P, Badam SK et al. Visualization beyond the desktop—the next big thing. *IEEE Computer Graphics and Applications* 2014; 34(6): 26–34.
4. Isenberg P, Elmqvist N, Scholtz J et al. Collaborative visualization: definition, challenges, and research agenda. *Information Visualization* 2011; 10(4): 310–326.
5. Elmqvist N and Irani P. Ubiquitous analytics: Interacting with big data anywhere, anytime. *IEEE Computer* 2013; 46(4): 86–89.
6. Badam SK, Fisher E and Elmqvist N. Munin: A peer-to-peer middleware for ubiquitous analytics and visualization spaces. *IEEE Transactions on Visualization and Computer Graphics* 2015; 21(2): 215–228.

7. Dourish P. *Where the Action Is: The Foundations of Embodied Interaction*. MIT press, 2004.
8. Ball R, North C and Bowman DA. 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*. pp. 191–200, 2007.
9. Badam SK and Elmqvist N. PolyChrome: A cross-device framework for collaborative web visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. pp. 109–118, 2014.
10. Marrinan T, Aurisano J, Nishimoto A et al. Sage2: A new approach for data intensive collaboration using scalable resolution shared displays. In *IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. pp. 177–186, 2014.
11. Bostock M, Ogievetsky V and Heer J. D^3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 2011; 17(12): 2301–2309.
12. Yonezawa T, Ogawa M, Kyono Y et al. Sensetream: Enhancing online live experience with sensor-federated video stream using animated two-dimensional code. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*. pp. 301–305, 2014.
13. Chung H, North C, Self JZ et al. VisPorter: facilitating information sharing for collaborative sensemaking on multiple displays. *Personal and Ubiquitous Computing* 2014; 18(5): 1169–1186.
14. Kim K, Javed W, Williams C et al. Hugin: A framework for awareness and coordination in mixed-presence collaborative information visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. pp. 231–240, 2010.
15. MacKay WE. Is paper safer? the role of paper flight strips in air traffic control. *ACM Transactions on Computer-Human Interaction (TOCHI)* 1999; 6(4): 311–340.
16. Isenberg P, Hinrichs U, Hancock M et al. Information visualization on interactive tabletops in work vs. public settings. *Collaborative Visualization on Interactive Surfaces-CoVIS'09* 2010; .
17. Hamilton P and Wigdor DJ. Conductor: enabling and understanding cross-device interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 2773–2782, 2014.
18. Rekimoto J. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. pp. 31–39, 1997.
19. Hinckley K, Ramos G, Guimbretiere F et al. Stitching: pen gestures that span multiple displays. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*. pp. 23–31, 2004.
20. Chen X, Grossman T, Wigdor DJ et al. Duet: exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 159–168, 2014.
21. Houben S and Marquardt N. WATCHCONNECT: A toolkit for prototyping smartwatch-centric cross-device applications. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 1247–1256, 2015.
22. von Zadow U, Büschel W, Langner R 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*. pp. 129–138, 2014.
23. Yang J and Wigdor D. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 2783–2792, 2014.
24. Czerwinski M, Smith G, Regan T et al. Toward characterizing the productivity benefits of very large displays. In *Proceedings of INTERACT*. pp. 9–16, 2003.
25. Andrews C, Endert A and North C. Space to think: large high-resolution displays for sensemaking. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 55–64, 2010.
26. Bradel L, Endert A, Koch K et al. Large high resolution displays for co-located collaborative sensemaking: Display usage and territoriality. *International Journal of Human-Computer Studies* 2013; 71(11): 1078–1088.
27. Isenberg P and Carpendale S. Interactive tree comparison for co-located collaborative information visualization. *IEEE Transactions on Visualization and Computer Graphics* 2007; 13(6): 1232–1239.
28. Isenberg P and Fisher D. Collaborative brushing and linking for co-located visual analytics of document collections. In *Computer Graphics Forum*, volume 28. Wiley Online Library, pp. 1031–1038, 2009.
29. Kim K and Elmqvist N. Embodied lenses for collaborative visual queries on tabletop displays. *Information Visualization* 2012; : 1473871612441874.
30. Spindler M, Tominski C, Schumann H et al. Tangible views for information visualization. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ACM, pp. 157–166, 2010.
31. Hall ET. *The Hidden Dimension*. Garden City, NY: Anchor Books, 1966.
32. Ballendat T, Marquardt N and Greenberg S. Proxemic interaction: designing for a proximity and orientation-aware environment. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. pp. 121–130, 2010.
33. Badam SK, Amini F, Elmqvist N et al. Supporting visual exploration for multiple users in large display environments. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*. 2016.
34. Kister U, Reipschläger P, Matulic F et al. Bodylenses: Embodied magic lenses and personal territories for wall displays. In *Proceedings of the ACM International Conference on Interactive Tabletops & Surfaces*. pp. 117–126, 2015.
35. Endert A, Andrews C, Lee YH et al. Visual encodings that support physical navigation on large displays. In *Proceedings of Graphics Interface*. Canadian Human-Computer Communications Society, pp. 103–110, 2011.
36. Andrews C, Endert A, Yost B et al. Information visualization on large, high-resolution displays: Issues, challenges, and opportunities. *Information Visualization* 2011; 10(4): 341–355.
37. Ball R, DellaNoce M, Ni T et al. Applying embodied interaction and usability engineering to visualization on large displays. In *Proceedings of the ACM British HCI-Workshop on Visualization & Interaction*. pp. 57–65, 2006.
38. Ball R and North C. Realizing embodied interaction for visual analytics through large displays. *Computers & Graphics* 2007; 31(3): 380–400.

39. Andrews C and North C. Analyst's workspace: An embodied sensemaking environment for large, high-resolution displays. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*. pp. 123–131, 2012.
40. Jacob RJK, Girouard A, Hirshfield LM et al. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 201–210, 2008.
41. Raskar R, Welch G, Cutts M et al. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*. pp. 179–188, 1998.
42. Raskar R, Brown MS, Yang R et al. Multi-projector displays using camera-based registration. In *Proceedings of the IEEE Conference on Visualization*. pp. 161–522, 1999.
43. Scott D, Sharp R, Madhavapeddy A et al. Using visual tags to bypass bluetooth device discovery. *ACM SIGMOBILE Mobile Computing and Communications Review* 2005; 9(1): 41–53.
44. Kaltenbrunner M and Bencina R. reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the ACM International Conference on Tangible and Embedded Interaction*. pp. 69–74, 2007.
45. Klokmose CN, Kristensen JB, Bagge R et al. Bullseye: High-precision fiducial tracking for table-based tangible interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. pp. 269–278, 2014.
46. Rädle R, Jetter HC, Marquardt N et al. HuddleLamp: Spatially-aware mobile displays for ad-hoc around-the-table collaboration. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. pp. 45–54, 2014.
47. Rohs M and Zweifel P. A conceptual framework for camera phone-based interaction techniques. In *International Conference on Pervasive Computing*. Springer, pp. 171–189, 2005.
48. Hesselmann T, Henze N and Boll S. Flashlight: optical communication between mobile phones and interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. pp. 135–138, 2010.
49. Langlotz T and Bimber O. Unsynchronized 4d barcodes. In *Advances in Visual Computing*. Springer, 2007. pp. 363–374.
50. Li T, An C, Campbell AT et al. Hilight: Hiding bits in pixel translucency changes. *ACM SIGMOBILE Mobile Computing and Communications Review* 2015; 18(3): 62–70.
51. Li T, An C, Xiao X et al. Real-time screen-camera communication behind any scene. In *Proceedings of the ACM Conference on Mobile Systems, Applications, and Services*. pp. 197–211, 2015.
52. Tang A, Tory M, Po B et al. Collaborative coupling over tabletop displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. pp. 1181–1190, 2006.
53. Morris MR, Ryall K, Shen C et al. Beyond 'social protocols': multi-user coordination policies for co-located groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. pp. 262–265, 2004.
54. Reda K, Johnson AE, Papka ME et al. 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*. pp. 2759–2768, 2015.
55. Shoemaker G, Tsukitani T, Kitamura Y et al. Body-centric interaction techniques for very large wall displays. In *Proceedings of the ACM Nordic Conference on Human-Computer Interaction: Extending Boundaries*. pp. 463–472, 2010.
56. Elmqvist N, Moere AV, Jetter HC et al. Fluid interaction for information visualization. *Information Visualization* 2011; 10(4): 327–340.
57. Langner R, Horak T and Dachselt R. Towards combining mobile devices for visual data exploration. In *Poster Program of the 2016 IEEE Conference on Information Visualization (InfoVis)*. 2016.
58. Abrahamson D and Trninic D. Toward an embodied-interaction design framework for mathematical concepts. In *Proc. ACM Conference on Interaction Design and Children*. pp. 1–10, 2011.
59. Marquardt N and Greenberg S. *Proxemic Interactions: From Theory to Practice*. Morgan & Claypool Publishers, 2015.
60. Wobbrock JO, Morris MR and Wilson AD. User-defined gestures for surface computing. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. pp. 1083–1092, 2009.
61. Rohs M. Marker-based embodied interaction for handheld augmented reality games. *Journal of Virtual Reality and Broadcasting* 2007; 4(5): 1860–2037.
62. Amatriain X, Kuchera-Morin J, Hollerer T et al. The Allosphere: Immersive multimedia for scientific discovery and artistic exploration. *IEEE Multimedia* 2009; 16(2): 64–75.
63. Thevenin D and Coutaz J. Plasticity of user interfaces: Framework and research agenda. In *Proceedings of INTERACT*, volume 99. pp. 110–117, 1999.
64. Heer J and Shneiderman B. A taxonomy of tools that support the fluent and flexible use of visualizations. *ACM Queue* 2012; 10(2): 1–26.
65. Shneiderman B. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*. pp. 336–343, 1996.
66. Yi JS, ah Kang Y, Stasko JT et al. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics* 2007; 13(6): 1224–1231.
67. Satyanarayan A, Wongsuphasawat K and Heer J. Declarative interaction design for data visualization. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. pp. 669–678, 2014.
68. Tobiasz M, Isenberg P and Carpendale S. Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization and Computer Graphics* 2009; 15(6): 1065–1072.
69. Perlin K and Fox D. Pad: An alternative approach to the computer interface. In *Computer Graphics*. pp. 57–64, 1993.
70. Isenberg P, Dragicevic P, Willett W et al. Hybrid-image visualization for large viewing environments. *IEEE Transactions on Visualization and Computer Graphics* 2013; 19(12): 2346–2355.
71. Wicker SB and Bhargava VK. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.
72. Chen W, Yang G and Zhang G. A simple and efficient image pre-processing for QR decoder. In *Proceedings of the International Conference on Electronic & Mechanical Engineering and Information Technology*. 2012.