Augmented reality-supported spatially aware visual links for multiple displays and devices

Jordan Gilbreath

Follow this and additional works at: https://louis.uah.edu/uah-theses

Recommended Citation
https://louis.uah.edu/uah-theses/381

This Thesis is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Theses by an authorized administrator of LOUIS.
AUGMENTED REALITY-SUPPORTED SPATIALLY
AWARE VISUAL LINKS FOR MULTIPLE DISPLAYS
AND DEVICES

by

JORDAN GILBREATH

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in
The Department of Computer Science
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2022
In presenting this thesis in partial fulfillment of the requirements for a master’s de-
gree from The University of Alabama in Huntsville, I agree that the Library of this
University shall make it freely available for inspection. I further agree that permis-
sion for extensive copying for scholarly purposes may be granted by my advisor or,
in his/her absence, by the Chair of the Department or the Dean of the School of
Graduate Studies. It is also understood that due recognition shall be given to me
and to The University of Alabama in Huntsville in any scholarly use which may be
made of any material in this thesis.

Jordan Gilbreath

06/08/2022
(date)
Submitted by Jordan Gilbreath in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Dr. Haeyong Chung  6/28/2022  (Date)  Committee Chair

Dr. Timothy Newman  6/28/22  (Date)  Dr. Jacob Hauenstein  6/29/22  (Date)

Dr. Letha Etzkorn  6/28/22  (Date)  Department Chair

Dr. Rainer Steinwandt  6/29/22  (Date)  College Dean

Dr. Jon Hakkila  6/29/22  (Date)  Graduate Dean
ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Master of Science
in Computer Science

College/Dept. Science/Computer Science

Name of Candidate Jordan Gilbreath

Title Augmented Reality-supported Spatially Aware Visual Links for Visual Data Analysis and Sensemaking for Multiple Displays and Devices

Visual data analysis and sensemaking can benefit from utilizing multiple displays to organize large amounts of information. However, analyzing information scattered across multiple displays can be challenging. Existing solutions to solving this problem often require mental effort to understand cross-display relationships. A hybrid approach, named AR-SAViL, is presented which addresses this problem by using an augmented reality headset to track displays and their contents in real time, drawing visual links directly between related information. The performance of the system is evaluated through exploration of an existing dataset and expert review, and findings suggest AR-SAViL has potential to assist visual analysis and sensemaking when using multiple displays.

Abstract Approval: Committee Chair

Dr. Haeyong Chung

Department Chair

Dr. Letha Etzkorn

Graduate Dean

Dr. Jon Hakkila
ACKNOWLEDGMENTS

This thesis would have been impossible without the guidance and materials provided to me by others. I would like to thank in particular Dr. Haeyong Chung, who suggested this thesis topic, provided his expertise in data visualization to guide its development, assisted in its writing, and provided necessary hardware for development and testing. Thank you to Dr. Jacob Hauenstein and Dr. Timothy Newman, who provided their guidance and time as committee members to make this thesis possible. Additionally, I would like to thank Nicholas Diliberti who assisted in research and editing and Dr. William MacKenzie who provided the HoloLens 2 device used for the final stage of development as well as valuable feedback on this work.
# TABLE OF CONTENTS

List of Figures .......................................................... ix  
List of Tables ........................................................... xi  
List of Symbols .......................................................... xii

Chapter

1 Introduction ............................................................ 1

2 Background and Related Work ......................................... 6  
   2.1 Cross-display Visual Representations ........................... 6  
      2.1.1 Highlighting ................................................. 7  
      2.1.2 Contextual Views ........................................... 8  
      2.1.3 Partially-Out-of-Frame Approaches ...................... 10  
   2.2 Display Ecologies ................................................ 10  
   2.3 Multi-Display Systems and Environments ..................... 12  
   2.4 AR-supported Visual Data Analysis ........................... 13  
   2.5 AR Applications using Multiple Displays .................... 15

3 Design Considerations ................................................ 17  
   3.1 Visualize Relationships between Information across Physically-separated Legacy Visualizations and Displays ......................... 17  
   3.2 Create Virtual Representations of Analysis Targets in the AR Scene ........................... 18
3.3 Support Foraging-oriented and Synthesis-oriented Tasks using Multiple Displays ........................................ 19
3.4 Identify the Physical Locations of Independent Displays .............. 20
3.5 Facilitating Display Discovery .................................................. 21
3.6 Reduce Visual Clutter when Drawing Numerous Visual Links ....... 22
3.7 Support the Use of Multiple Devices .......................................... 22

4 Cross-Display Visual Links ........................................................................ 25
4.1 Nodes .......................................................................................... 25
  4.1.1 Screen-space nodes .............................................................. 26
  4.1.2 AR-space nodes ................................................................... 28
4.2 Links ............................................................................................ 29

5 The AR-SAViL System ............................................................................. 32
5.1 System Description ........................................................................... 32
  5.1.1 Connect the two applications (Display and AR) ................. 34
  5.1.2 Register displays and devices .............................................. 36
  5.1.3 Inform AR application of screen contents ......................... 39
  5.1.4 Determine the AR position of screen content .................... 41
  5.1.5 Determine relationships between different nodes ............. 45
  5.1.6 Draw the AR visual links across registered displays ........ 46
5.2 Implementation ................................................................................ 49

6 Sensemaking Tool .................................................................................. 51
6.1 Dataset Views ................................................................................ 52
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Example of highlighting using color coding</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Example of contextual views</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Example of visual links drawn using a partially-out-of-frame approach</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Diagram of non-bundled vs. bundled cross-display links</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Screenshot showing screen-space and AR-space nodes</td>
<td>26</td>
</tr>
<tr>
<td>4.2 Diagram showing the categorization of screen nodes into displays, windows, and entities.</td>
<td>27</td>
</tr>
<tr>
<td>4.3 Example of bicluster and proxy nodes.</td>
<td>29</td>
</tr>
<tr>
<td>4.4 Diagram showing different link types.</td>
<td>31</td>
</tr>
<tr>
<td>5.1 Architecture of the AR-SAViL system</td>
<td>34</td>
</tr>
<tr>
<td>5.2 AR device detects a QR code and connects to the display application using encoded information</td>
<td>36</td>
</tr>
<tr>
<td>5.3 Example of a shared image target for a 16:9 aspect ratio display</td>
<td>38</td>
</tr>
<tr>
<td>5.4 Diagram illustrating display content reporting.</td>
<td>40</td>
</tr>
<tr>
<td>5.5 Example object metadata</td>
<td>41</td>
</tr>
<tr>
<td>5.6 Position a screen-space node according to its screen coordinates</td>
<td>44</td>
</tr>
<tr>
<td>5.7 Example of links drawn between nodes as Bezier curves</td>
<td>46</td>
</tr>
<tr>
<td>5.8 Connecting screen-space objects</td>
<td>48</td>
</tr>
<tr>
<td>5.9 Connecting bicluster nodes to related screen-space nodes</td>
<td>48</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Information encoded in the connection QR code</td>
<td>35</td>
</tr>
<tr>
<td>5.2 Locations for each Bezier control point</td>
<td>46</td>
</tr>
</tbody>
</table>
# List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>First endpoint of a visual link</td>
</tr>
<tr>
<td>$B$</td>
<td>Second endpoint of a visual link</td>
</tr>
<tr>
<td>$h$</td>
<td>Height of an on-screen object in pixels</td>
</tr>
<tr>
<td>$h_a$</td>
<td>Height of a screen-space node in the AR scene</td>
</tr>
<tr>
<td>$h_d$</td>
<td>Vertical resolution of a display in pixels</td>
</tr>
<tr>
<td>$h_{dn}$</td>
<td>Height of a display node in the AR scene</td>
</tr>
<tr>
<td>$h_n$</td>
<td>Normalized height of an on-screen object in the range $[0, 1]$</td>
</tr>
<tr>
<td>$M$</td>
<td>Midpoint between the two endpoints of a visual link</td>
</tr>
<tr>
<td>$P_i$</td>
<td>Control point $i$ of a Bézier curve, where $i \geq 0$</td>
</tr>
<tr>
<td>$w$</td>
<td>Width of an on-screen object in pixels</td>
</tr>
<tr>
<td>$w_a$</td>
<td>Width of a screen-space node in the AR scene</td>
</tr>
<tr>
<td>$w_d$</td>
<td>Horizontal resolution of a display in pixels</td>
</tr>
<tr>
<td>$w_{dn}$</td>
<td>Width of a display node in the AR scene</td>
</tr>
<tr>
<td>$w_n$</td>
<td>Normalized height of an on-screen object in the range $[0, 1]$</td>
</tr>
<tr>
<td>$x$</td>
<td>Horizontal position of an on-screen object relative to its parent in pixels</td>
</tr>
</tbody>
</table>
$x_a$ Horizontal position of a screen-space node relative to its parent in the AR scene

$x_c$ Centered horizontal position of an on-screen object in pixels

$x_n$ Normalized horizontal position of an on-screen object in the range $[0, 1]$

$x_p$ Horizontal position of an on-screen parent object in pixels

$y$ Vertical position of an on-screen object relative to its parent in pixels

$y_a$ Vertical position of a screen-space node relative to its parent in the AR scene

$y_c$ Centered vertical position of an on-screen object in pixels

$y_n$ Normalized vertical position of an on-screen object in the range $[0, 1]$

$y_p$ Vertical position of an on-screen parent object in pixels
CHAPTER 1

INTRODUCTION

Many workspaces utilize multiple displays to extend the total screen area available for on-screen information. Environments employing multiple displays for visual data analysis can provide an improved workspace compared to those using only a single display, for example, spreading information across physical space can aid visual data analysis compared to navigating through data completely virtually, such as with scrolling and panning [1, 2]. Certain types of displays also have unique advantages that can be leveraged to perform visual data analysis. Adding interactive displays to a workspace, such as touch screens and interactive tabletops, has been found to enhance analysis by allowing more natural interactions compared to mouse and keyboard input, and workspaces employing portable displays like tablets can benefit from the ability to physically arrange information at will [3–5].

We define sensemaking as the process of investigating information to make sense of a dataset’s underlying patterns and trends and make decisions [6]. For visual analysis and sensemaking with multiple displays, one common strategy is to take advantage of extended screen space from multiple displays by assigning and distributing visualizations and information across different displays. As an example,
we consider a drone operator who is responsible for monitoring a complex feed of information from a drone and other sources, such as maps, sensor readouts, radar, and live camera feeds. The operator must also be prepared to take quick action based on rapid analysis of this information, for example, changing the drone’s flight path, deploying flares, or taking defensive action against an incoming threat. When using only a single display, simultaneously viewing all of this information may be challenging for the operator, and she must rely on virtual navigation such as switching windows or scrolling. To overcome this challenge, the operator could employ multiple displays. For instance, she may use a large television and four desktop monitors, arranging live camera feeds onto the television, visualizations of sensor readouts like altitude and orientation across two monitors, a radar feed onto a third monitor, and an annotated map of the region onto the fourth. By extending available screen space and making all information organized and visually available, she can simply look at a different display rather than switching windows or scrolling to check different information sources.

When performing visual analysis, mentally relating and connecting distributed information is critical. However, this can be challenging when using multiple displays. If information is scattered across displays outside of the user’s immediate field of view, or when physical gaps exist between displays, supporting awareness of this information becomes difficult [4,5,7]. Prior techniques have been presented to mitigate these challenges with visual analysis using multiple displays. One such technique is highlighting [4,8], which visually guides users to related information with color coding, borders, animation, or other forms of emphasis. Other techniques include “partially-
out-of-frame” approaches [9, 10], which use partially-visible visual elements that guide users toward out-of-frame information.

In prior studies [4, 5, 8, 11], it was observed that while using multiple displays has notable benefits for data analysis, mentally connecting data scattered across displays remains a critical obstacle. Users found performing visual analysis with multiple displays challenging specifically due to this obstacle. The findings of these studies suggest that the exploration of data becomes increasingly problematic as the quantity of information, displays, and devices increases. Not all displays can necessarily be seen at once when multiple displays are used, meaning some information will likely be out of a user’s field of view at any given time. It was observed that the effectiveness of highlighting and “partially-out-of-frame” approaches degrades when information is spread across more than three displays. These approaches may require significant mental effort on the user’s part, repeatedly shifting focus between different displays to understand relationships between scattered information.

To assist visual data analysis and sensemaking when using multiple displays, we present AR-SA ViL (AR-supported Spatially Aware Visual Links), a novel visual analysis system that visually connects information scattered across independent displays using persistent overlays in augmented reality, or AR. The main goal of AR-SA ViL is to use AR overlays and visualizations to integrate multiple displays and devices into a single visual analysis space. We connect information using visual links, which are lines drawn between two information items (e.g., text, visualizations, elements within visualizations, etc.) to visually represent a relationship between those
items. Specifically, we draw these visual links as AR overlays so that items can be visually connected, even if they reside across multiple displays.

We evaluate AR-SA ViL with respect to several key research questions. First, how can visual links be effectively drawn between physically separated displays to support visual analysis in augmented reality? Second, how can physically separate displays and their contents be precisely tracked to support visual analysis in augmented reality? Third, how does AR-SA ViL support navigational and sensemaking tasks with multiple displays? Fourth, how can AR-SA ViL assist in collecting and relating information from separate devices? We present three primary contributions of AR-SA ViL which are described next.

First, we contribute novel visual link representations which connect information items across physical space using AR overlays. Using a wearable AR headset, these links are overlaid onto the user’s vision to visually connect displays and their contents. Drawing numerous visual links can result in confusion due to visual clutter which hinders the user’s ability to perceive individual links. To reduce this visual clutter, we use a bundling technique that groups links based on shared relationships, decreasing the total number of links drawn.

Second, we contribute a novel method of integrating multiple displays into a single visual analysis workspace using augmented reality tracking technology. AR-SA ViL must identify the position and orientation of a display within the user’s environment before that display’s contained information can be connected with visual links. We present a method of locating and tracking displays using the cameras and sensors of an AR headset worn by the user rather than using external motion tracking
systems. AR-SA ViL identifies and locates displays in the user’s field of view based on their visual contents (i.e., what is shown on the display). Additionally, the user’s head movements are used to extrapolate display locations even when displays are no longer in the user’s field of view.

Third and finally, we contribute a method of collecting information through a local network connection to integrate multiple devices into a single data analysis workspace in augmented reality. Before AR-SA ViL can draw visual links between any information items, those items and their relationships must be known. Information about each display, such as its resolution and physical size, must also be known to support the tracking of displays and their contents. Each device in the user’s workspace, including the AR headset, are presumed to be independent devices on a common local network where information can be shared. Leveraging this infrastructure, AR-SA ViL coordinates information sharing between devices so that this necessary information can be collected by the AR headset for drawing visual links and tracking displays.

We also present a prototype which applies AR-SA ViL to two real-world visual analysis problems: visual analysis with multiple displays and sensemaking of text data. We extend prior investigations [5, 12] of sensemaking to an environment employing multiple displays, wherein the user utilizes AR-SA ViL to understand cross-display data relationships and perform visual analysis. We investigate how an expert reviews AR-SA ViL’s effectiveness in assisting visual analysis and sensemaking with multiple displays, and find that AR-SA ViL has potential to aid in visual data analysis.
CHAPTER 2

BACKGROUND AND RELATED WORK

AR-SAViL draws on prior research on cross-display visual representations, display ecologies, multi-display environments, AR-supported visual data analysis, and AR applications using multiple displays. In this chapter, we will discuss related works on these topics.

2.1 Cross-display Visual Representations

In this section, we explore existing approaches to connecting information items across multiple displays. In multi-display environments (MDEs), visual representations of relationships can be used to support users’ awareness and analysis of scattered information [4, 5]. These cross-display visual representations are used to direct analysts to relevant information in the MDE. We discuss three approaches to connecting information items across displays: highlighting, contextual views, and partially-out-of-frame approaches.
2.1.1 Highlighting

Highlighting refers to applying matching visual indicators to related on-screen items (e.g., text, images, elements within visualizations). Specifically, related items across one or more displays can be highlighted using visual features such as color, outlines, and blinking effects [13]. Some systems utilizing highlighting will also draw the user’s focus to a specific item by reducing the visibility of the surrounding screen region through changes in saturation [14], brightness [15], or sharpness [16]. By applying these visual features, highlighting provides users with a means of locating related information items. Highlighting is often used in brushing-and-linking techniques which connect multiple views for interactive visual analysis, for example, by highlighting information related to a user’s selection across multiple views and synchronizing changes in one view to other views [17, 18]. Highlighting can apply matching visual indicators to provide guidance toward related information across displays, acting as a cross-display visual representation of those relationships. An example of highlighting using color coding across two displays is shown in Figure 2.1, where two relationships are visually represented using yellow and red.

These highlighting approaches rely primarily on the user’s memory to mentally connect relevant information in the MDE [4, 5]. For example, we consider a user working on a police investigation. The user has a text document on one display containing information about a suspect and a visualized network of phone numbers on another display. If the user selects the name of a person in the text document, a highlight could be applied to that selection, and the user in turn creates a mental mapping
Figure 2.1: Example of highlighting using color coding

from the selected name to the visual features of the highlight. A matching highlight is subsequently applied to all phone numbers contacted by that person. When the user switches focus from the display containing names to the display containing phone numbers, she remembers the mapping and understands that those phone numbers are associated with the selected name since the items are highlighted with the same visual features.

Due to this reliance on memory when shifting focus between displays, highlighting can become difficult for users when information is spread across numerous displays [5]. Accordingly, more explicit representations of cross-display relationships are desirable in the context of MDEs.

2.1.2 Contextual Views

Contextual views convey approximate locations of unseen information items with on-screen overlays. These overlays are drawn on top of a view and guide users toward information which is not visually present within that view. For example, Figure 2.2 shows contextual views being used to provide visual cues to the user about
the approximate locations of related information residing outside of each window and display. Bars are drawn along window and display borders to show the direction where related information can be found. Some examples of systems using contextual views include City Lights [19] and EdgeRadar [20]. City Lights uses contextual views by drawing bars along the borders of windows to indicate the direction of unseen visual elements. EdgeRadar uses contextual views by drawing a radar-like visualization along the borders of windows, indicating the direction and approximate distance of unseen visual elements. Contextual views have also been used in video games to visually represent the direction of off-screen elements, such as arrows which indicate the direction of game objectives or opponents. Contextual views can be deployed in multi-display environments to guide the user’s focus from information on one display to information on another display, acting as a cross-display representation of relationships between information items.
2.1.3 Partially-Out-of-Frame Approaches

In partially-out-of-frame approaches, visual representations of unseen information items are drawn partially out of a view’s frame. For example, the Halo [9] visualization system uses partially-out-of-frame circles to guide users toward off-screen information on small form-factor handheld displays. Wedge [10] extends Halo by instead using partially-out-of-frame wedge shapes which point toward off-screen objects. SAViL [5] uses partially-out-of-frame visual links which connect scattered information items using straight lines. When drawn across multiple displays, SAViL’s visual links are interrupted by physical gaps which separate each display, and the user must mentally connect links across these gaps to understand cross-display relationships. Figure 2.3 shows an example of a partially-out-of-frame approach similar to SAViL applied to multi-display environments, where visual links drawn using straight lines connect related information items. When related items reside on different displays, links between those items are interrupted by gaps between the displays.

2.2 Display Ecologies

Chung [7] defines a display ecology as a set of displays which each contribute to an analyst’s workflow in support of their analysis goals. Displays within an ecology can be diverse, having differing sizes, form factors, and interactive capabilities, and relate to one another in varied ways to create a single combined visual analysis environment. Certain types of displays have been found to have unique advantages when employed for visual data analysis. For instance, prior investigations [1, 21] indicate
that large displays can open opportunities to create physical navigation methods that exploit our natural spatial senses, such as bodily movements like crouching and head movements, which would be comparatively less effective on smaller displays. Thus, it is desirable to consider how to combine varied displays into a single visual analysis workspace.

Four design considerations for enabling visual analysis in display ecologies have been proposed [7]: display composition, information coordination/transfer, information connection, and display membership. Display composition refers to how relationships are formed between multiple displays to form a single visual analysis space. Information coordination/transfer refers to how information and tasks can be distributed across multiple displays. Information connection refers to the process of connecting information that is scattered across displays. Display membership refers to how displays join or leave the display ecology during operation.
2.3 Multi-Display Systems and Environments

In multi-display environments, such as workroom or laboratory settings, displays can be deployed in parallel to create coordinated views. There are several multi-display systems that allow users to distribute information across multiple displays for sharing information and collaborating on different tasks. For example, Geyser et al. [22] used multiple displays to support collaborative sketching, and Forlines et al. [23] employed multiple displays and devices to create a multi-user molecular visualization system.

We identified four previous sensemaking tools which focus on using multiple displays and devices. These tools’ authors investigate how using multiple displays can aid sensemaking, or the discovery of a dataset’s underlying patterns [6]. Plaue et al. [24] investigated how using two desktop monitors can aid sensemaking with a detailed dataset, finding that users strongly preferred using two side-by-side monitors compared to using only a single monitor. Wallace et al. [6] experimented with collaborative sensemaking using different display configurations including tablet computers and a shared digital tabletop, and found that the introduction of the digital tabletop improved sensemaking compared to using only tablet computers. The Conductor system [25] facilitated the use of numerous tablet computers in parallel, and the system’s authors found that users benefited from the use of multiple displays. SAViL [5] was also designed to support sensemaking with multiple displays by connecting information with partially-out-of-frame visual links drawn as straight lines. SAViL’s authors
concluded that its visual links assisted users with sensemaking within multi-display environments.

Some multi-display systems utilize multiple independent devices to promote visual data analysis. Munin [26] and PolyChrome [18] both support distributed architectures where devices communicate over a network to share information and coordinate separated views. Some systems have leveraged spatially-aware displays, which are aware of their locations in physical space, for visual analysis. For example, Schreiner et al. [27] and Langner et al. [8] presented a design concept for using spatially-aware displays to create “tangible views,” which are interacted with via physical movement, and later combined this concept with augmented reality [28,29]. Pixel-oriented Treemap for Multiple Displays (PTM) [11] divides a visualization across two displays to analyze the status of one million online computers. PTM visualizes detailed information, such as machine class, function, unit, facility, etc., on a user’s personal display, while providing an overview of the system on a wall display.

2.4 AR-supported Visual Data Analysis

Many efforts have been made to create, refine, and evaluate data-analysis tools that implement augmented reality technologies. These approaches not only span a large set of dissimilar problem spaces, but also employ a variety of technological setups, from headsets to desktops to smartphones.

While AR representations of visualizations like graphs [30] have been created, there are also efforts to use AR as a means of organizing information in physical
space. This has led to AR applications such as content organization by a single user in a personalized space [31, 32] and sensemaking by a group in a shared space [33].

Many AR-based visualizations vary in how they project data and data representations into real space instead of residing on the user’s screen. Some visualizations create virtual displays in the user’s environment [34], some overlay data atop existing objects in the user’s environment [35], and some project data relative to a real-world anchoring object [36]. The large diversity of these visualizations leave many opportunities open for the creation of new techniques and the refinement of existing ones.

Some developments with augmented reality focus on improving the effectiveness of existing user interaction patterns instead of the development of new tools or techniques. Some work in this area includes extending existing visualizations on physical displays into AR [36, 37], providing additional information in AR based on a user’s interactions on the physical display. Reipschläger et al. [29] used AR head-mounted displays to provide users with personalized views while using a large interactive display shared with other users.

Other work has been done in order to evaluate augmented reality in comparison to more traditional workspaces used by data analysts [38, 39]. The capabilities of AR displays are both powerful and unfamiliar, necessitating evaluation of when AR displays or non-AR displays are more effective [40], as well as evaluation of when different kinds of AR visualizations are most appropriate [41].
2.5 AR Applications using Multiple Displays

There exist many works which combine augmented reality displays with traditional non-AR displays. AR head-mounted displays (HMDs) have been used in many of these works, including surrounding users with multiple interactive displays and AR visualizations [29, 36], using AR to overlay widgets containing information next to mobile devices [42], and combining an HMD’s display with a real-world display [43]. Since elements shown on HMDs are specific to the user wearing the HMD, collaborative applications with user-specific data visualizations have also been presented [43, 44]. Alongside these, works exist using HMDs to augment existing experiences with mobile devices [42] and desktop applications such as CAD software [45].

Mobile phones, tablets, and desktops also can be platforms on which to build AR-based analysis tools. For example, works exist with mobile phones allowing users to create and manipulate drawing-like visualizations [46, 47], augment existing displays with AR elements [48], and expand the plane of the display outwards in AR [49]. Smartphones have also been used as input devices for exploring 3D data visualizations in AR [50]. Tablets, used functionally as larger, more powerful phones, provide many of the same benefits. For example, Hubenschmid et al. [51] introduced a system which anchors AR visualizations to spatially-aware tablets. Finally, desktop monitors can serve as platforms and have been augmented through AR extensions [37] and 3D hand inputs [52, 53]. Tools also exist which utilize multiple devices to create networked AR visualizations using both desktop [37, 45] and large, high-resolution displays [29, 44, 54, 55].
In addition to the visualization of existing structures in augmented reality, new user interfaces have been developed to interact with distributed data in MDEs while retaining the benefits of AR [40]. For example, users might need to see relationships between contents of multiple displays [56] to understand how different pieces of data within the MDE are related. Visual links are one way to see such relationships, and Prouzeau et al. [34] have considered drawing such visual links in AR between multiple virtual displays while minimizing clutter and visual obstruction. Exploration has also been done to consider how to cluster related elements in AR for ease of interaction [32]. Allowing users to smoothly operate within these MDE/AR environments can increase users’ ability to understand large datasets [57], and interest in doing so is currently high among analysts [30]. Even in the most basic of MDEs, the ability to interact with visualizations in AR seems to improve an analyst’s ability to understand datasets [58, 59].

In summation, the large variety of platforms for AR experiences has given rise to a wide range of data analysis tools in both single-display and multi-display environments. These tools each leverage AR to support visual data analysis in numerous ways, such as new AR-based visualization techniques, new methods for interacting with visualizations using AR, or new ways to utilize AR and non-AR displays in parallel for visual data analysis.
CHAPTER 3

DESIGN CONSIDERATIONS

In this chapter, we will discuss the primary design considerations of AR-SA ViL. The design of AR-SA ViL’s AR visualization techniques is primarily informed by challenges that users faced while using prior visual analytics tools for multiple displays [4, 5, 7, 11, 60]. Users of these prior tools were found to encounter significant impediments when understanding relationships between information items and visualizations among separate displays. This hindered the integration of displays and devices into a single visual analysis space. The main design goal of AR-SA ViL was to address the existing impediments in understanding relationships between information across independent displays by using AR features such as overlays, object recognition, and tracking. Next, the design considerations and requirements we used to guide the design of AR-SA ViL and user interfaces are discussed.

3.1 Visualize Relationships between Information across Physically-separated Legacy Visualizations and Displays

In a multi-display environment, information can be spread across physically separated displays. However, analysts still need to compare, explore, and/or combine
data and information presented across these displays [4]. Each display in the multi-display environment may be driven by a different PC or mobile device, each containing various visualizations and information items. Since the contents of each display are physically separated in this manner, the cross-display relationships between these contents are not explicit.

AR-SA ViL can explicitly connect information items on separate displays using visual link representations which consist of nodes and edges that span multiple displays. These cross-display links create a single visual structure in AR which illustrates the connections between data items. AR-SA ViL employs a state-of-the-art AR headset in combination with traditional non-AR displays (e.g., tablets, desktop monitors) to form a hybrid display ecology where virtual content can be overlaid on and linked with traditional display content. The visual links represent relationships between data items on different displays, enabling users to organize and connect datasets in AR.

3.2 Create Virtual Representations of Analysis Targets in the AR Scene

Analysis tasks incorporate many varied information sources, such as visualizations, documents, and elements within visualizations (e.g., map markers, graph nodes). A critical task to the success of the AR-SA ViL system is creating virtual AR representations of these sources that can be connected via visual links.

We define the AR scene as the three-dimensional environment which combines real-world objects like displays with AR overlays. AR-SA ViL uses visual links to
draw connections between *nodes* that reside in the AR scene. We define two high-level categories of these nodes: screen-space nodes and AR-space nodes.

Screen-space nodes represent traditional displays and their contents, and they are further divided into a hierarchy of displays, windows, and entities (visual elements which follow windows), where entities reside inside windows and windows reside inside displays. This hierarchy is used so that the position of nodes need only be known relative to their parent node – *i.e.*, a window is positioned using its two-dimensional location on the display, and entities are positioned using their two-dimensional locations within the window. AR-space nodes exist outside of physical displays in the AR scene, and can be freely manipulated by the analyst to remain in any chosen physical location.

### 3.3 Support Foraging-oriented and Synthesis-oriented Tasks using Multiple Displays

AR-SAViL focuses on supporting two aspects of visual data analysis [7]: “foraging-oriented” and “synthesis-oriented” tasks. Next, these two aspects will be discussed.

Foraging-oriented tasks involve exploring a dataset to search for, collect, and filter information relevant to analysts’ inquiries [61]. In the absence of guidance, analysts manually “forage” for relevant information across all displays. AR-SAViL seeks to provide guidance for foraging-oriented tasks by drawing visual links between elements scattered across displays. In this way, analysts are steered toward information relevant to their inquiries, potentially assisting in the navigation of detailed information.
Synthesis-oriented tasks involve investigating a dataset to form and test hypotheses and make sense of relationships within the dataset [61]. The wealth of space afforded by a multi-display environment allows analysts to “externalize” their internal thought processes by visually organizing information relevant to their insights across multiple displays. However, analysts may still face difficulty making sense of the connections between information spread across displays. AR-SA ViL seeks to aid synthesis-oriented tasks by acting as a tool to integrate insights from information items and visualizations on different displays and form hypothesis. For example, by following visual links across displays, analysts can assign meaning to displays based on the contained information’s type or importance, and related information can be organized or clustered across multiple displays [62, 63].

3.4 Identify the Physical Locations of Independent Displays

The physical location of each display plays an important role in interacting and forming insights with their contained data [1, 62]. Additionally, drawing cross-display visual links requires awareness of the identity and accurate physical location of each display. However, there exist technical hurdles in correctly identifying displays and tracking their physical positions. Since displays can be diverse in respect to form factors (size, shape, style, and layout and position of major components), and different operating systems may be driving displays, designing systems which can integrate multiple displays into a single visual analysis space can be difficult. While some systems use external motion-capture systems to overcome these hurdles [8, 64], AR-SA ViL uses the positional tracking and computer vision features of an AR head-
set, such as the Microsoft HoloLens, to track each display’s physical location and orientation. AR-SA ViL uses computer vision to identify and locate displays based on their visual contents, and uses the positional tracking capability of an AR headset worn by the user to project the position of each display even if the display exits the user’s field of view.

3.5 Facilitating Display Discovery

Before AR-SA ViL’s visual links can be drawn, finding displays and ensuring awareness of each display’s contents is necessary. To accomplish this, AR-SA ViL must have information of each device’s connected displays and their properties, as well as the locations and metadata associated with each display’s contents. For the purpose of AR-SA ViL, we consider a “device” to be a machine (e.g., a workstation computer, tablet, or smartphone) which drives attached displays. AR-SA ViL places the discovery of information relating to a display and its contents into the environment where it is most readily available: the device driving the display.

AR-SA ViL uses a desktop application, named the display application, which connects to the AR device over a local network and provides information about a device’s displays and all windows and entities managed by that device. The display application initiates the addition of new displays to the AR-SA ViL system and provides continuous updates relating to the visual contents of each tracked display.
3.6 Reduce Visual Clutter when Drawing Numerous Visual Links

AR-SA ViL’s visual links directly connect information across displays and visualizations. As the number of the links is increased, users may begin facing difficulty perceiving links and information on different displays due to visual clutter. This clutter can result in confusion due to degradation of visual clarity as links may cross each other and obscure other visual elements. To alleviate this problem, links can be “bundled” together, combining multiple links in order to reduce visual clutter.

AR-SA ViL bundles links involved in one-to-many and many-to-many relationships by instead routing links through an intermediate AR node (named a proxy node) that can be freely manipulated by the user. Figure 3.1 illustrates an example of bundling links representing two one-to-many relationships through the use of a proxy node. Since users can freely manipulate these proxy nodes, links can be rerouted by the user to create physical layouts for links in three dimensions.

3.7 Support the Use of Multiple Devices

When using multiple devices for visual data analysis, information is often scattered across each device. To support AR-SA ViL’s visual links, the AR device must be aware of information present across all devices in the system. With a single device, gathering this information is trivial since all information can be shared within a single application. However, when multiple devices are used, additional infrastructure is necessary to allow devices to communicate and share information.
To this end, AR-SA ViL creates a peer-to-peer event-driven network consisting of the AR device and all connected devices. Through this network, information can be freely exchanged. Different views can broadcast and receive in this network any information necessary to support the visualization of cross-view relationships. This allows for the creation of modular visualization systems that can operate indepen-
dently on any device in the AR-SAViL system. We believe these design considerations will allow AR-SAViL to aid users in effectively performing visual data analysis and sensemaking tasks on multiple displays.
CHAPTER 4

CROSS-DISPLAY VISUAL LINKS

In this chapter, we describe the relational visualizations of AR-SA ViL, which are designed to visually connect relevant visual objects across separate displays in support of visual analysis and sensemaking tasks in multi-display environments. The visual link representations can facilitate the creation of a visual analysis workspace by visualizing the relationships between information and guiding the user’s attention to scattered, but relevant, information across different screens, all while maintaining independent workspaces on each separate display. AR-SA ViL’s visual links representation consists of nodes and links.

4.1 Nodes

In AR-SA ViL, links are created to connect two nodes, which are defined as objects in the AR scene that can be connected via visual links. We categorize nodes into two high-level categories: a) screen-space nodes, which are presented on the display and represent displays and their contents, and b) AR-space nodes, which exist in 3D space. Figure 4.1 shows both of these node types in an example where the
Figure 4.1: Screenshot showing screen-space (a-c) and AR-space (e-d) nodes: a) Display, b) Window, c) Entity, d) Proxy, e) Bicluster

contents of three displays are connected via visual links. The specific types of both screen-space and AR-space nodes are described next.

4.1.1 Screen-space nodes

There are three primary classes of screen-space nodes: displays, windows, and entities. Each of these node types is illustrated in Figure 4.2. These nodes form parent-child relationships, where displays are parents of windows, and windows are parents of entities. Conversely, entities are children of windows, and windows are children of displays. Children will follow their parent’s motion, for example, if a window is moved, any entities that are children of that window will move in the same manner. These three node types are described next.
Figure 4.2: Diagram showing the categorization of screen nodes into displays, windows, and entities.

**Display** nodes represent an entire display. Their positions are derived from the observed or projected positions of real-world displays. The specific process for finding display positions is described in Section 5.1.2. **Window** nodes are contained within displays and represent each tracked window shown on that display, such as a visualization or text document. Their positions are derived from the positions of the displays that contain them and information about their screen-space position provided by the device. **Entity** nodes are contained within windows and represent any visual elements which follow a window, such as elements within visualizations or text within documents. Their positions are derived from the positions of their parent windows and information about their position within their parent windows.
4.1.2 AR-space nodes

AR-space nodes, in contrast to screen-space nodes, reside within the three-dimensional AR scene, and can be positioned by the user to any physical location. There are two types of AR-space nodes: proxy nodes and bicluster nodes. These two node types are described next.

**Proxy** nodes are created to reduce visual clutter by combining links involved in one-to-many and many-to-many relationships. In one-to-many relationships, one node connects to multiple other nodes, resulting in many links converging into a single screen location. In many-to-many relationships, multiple nodes share relationships with multiple other nodes, potentially causing many links to cross. In both cases, links may obscure screen elements and create visual clutter. Proxy nodes help address visual clutter by acting as a movable intersection point for many visual links. By connecting nodes to an intermediate proxy node rather than connecting nodes directly, the total area obscured by links can be reduced. An example of links connecting to proxy nodes is shown in Figure 4.3.

**Bicluster** nodes extend SightBi’s [12] relationship-views, which visualize data relationships between two views, into AR. Each bicluster node represents a relationship between subsets in separate data sets, such as an observed relationship between a group of individuals and a set of locations. Collections of bicluster nodes are generated using data about the relationships between two datasets. The bicluster nodes appear similarly to a bar chart with one bar for each involved data set, each representing the number of nodes represented in from that set, e.g., the number of people
and the number of locations involved in the relationship. When a bicluster node is activated via user input, links are drawn between bars and each node representing an item in their subsets, e.g., one bar will connect to each person in the relationship, and the other will connect to each location. An example of bicluster nodes is shown in Figure 4.3, where links connect to two activated bicluster nodes.

4.2 Links

AR-SAViL’s visual links draw direct connections between nodes. If a large number of information items are connected with straight lines, displays and visualizations may be obscured due to overlapping lines. To improve clarity, the links are drawn using cubic Bézier curves. Since visual links can be drawn between any of the previously mentioned types of nodes, we define seven different seman-
tic classes of links: entity-to-entity, window-to-window, display-to-display, entity-to-window, window-to-display, entity-to-display, and cross-space. These seven link classes are described next.

**Entity-to-Entity** links are drawn between entities. An entity-to-entity link represents a relationship, such as a node in a graph (which represents a person) linking to markers on a map (representing that person’s known locations).

**Window-to-Window** links represent a relationship between two windows. This link may be used for higher-level relationships, such as illustrating that a text document contains information used to construct a chart, *e.g.*, if the chart counts the frequency of keywords in a document.

**Display-to-Display** links represent a relationship between the contents of two entire displays (*e.g.*, if all the contents of both displays are semantically related). If there are two displays which include several relevant documents about a key person, both displays can be connected with this link. This type of a link can be drawn to show the occurrence of multiple entities presented in documents or visualization views on displays.

**Entity-to-Window** links represent a relationship between an entity and a particular window. This may be used to illustrate a connection between a data item and a window containing information about that item. For example, an entity representing a business in a bar chart could connect to a window containing a chart representing that business’s stock history.

**Window-to-Display** links represent a relationship between a window and the entire contents of a display, for example if an entire display is populated with
data related to a single document, a link may be drawn between the display and the document window.

Similar to Window-to-Display, **Entity-to-Display** links represent a relationship between a single entity and an entire display, for example if a graph node representing a person links to a display containing that individual’s profile, background, records, etc.

**Cross-Space** links connect an AR-space node to a screen-space node, for example connecting an AR-space proxy node to a screen-space node representing a pie chart.

These seven link classes are each illustrated in Figure 4.4.
CHAPTER 5

THE AR-SAVIL SYSTEM

The goal of AR-SAViL is to create a visual analysis workspace combining an AR headset, like the Microsoft HoloLens, and one or more displays, using visual links to illustrate relationships between information scattered across displays. In this chapter, we discuss the AR-SAViL system, including its architecture and details about the process of drawing visual links across displays in AR.

5.1 System Description

AR-SAViL is composed of two distinct applications: (a) the AR application which runs on the AR headset and is the primary user interface, and (b) the display application which runs on devices other than the AR headset to provide information to the AR application about displays and support visualization content. The display application hosts a server to which the AR application connects, and the display application provides continuous updates and visualization information to the AR application over a local network connection. The AR application uses the information provided by the display application from each device to discover displays, screen contents, and metadata about how different nodes should connect. Together, the applications
collaborate to support the synthesis of relational information and the creation of cross-display visual links.

The AR application also facilitates the creation of a peer-to-peer (P2P) network amongst all devices. This P2P network is used to enable visualization interactions between devices, enhancing the ability of the system to handle analysis tasks. This network can be used, for example, to let visualizations broadcast the data items they possess and the data items the user is requesting so that, in turn, other visualizations can submit their relevant information to the AR-SAViL system. Another use case is the creation of views which “control” other views, even on other devices, such as creating detailed navigation controls. The AR application may also access this P2P network via the coordinator to support visualizations in the three-dimensional AR space.

Figure 5.1 illustrates the architecture of the AR-SAViL system. In short, the AR headset runs the AR application, which connects to devices each running the display application, and all devices are connected to a P2P network.

The most vital job shared by both applications is the discovery of displays and their contents. Once all visual elements have correctly-positioned virtual representations (screen-space nodes), links can be drawn between these nodes as needed. The steps involved in the process of discovering, positioning, and maintaining these screen-space nodes to draw links are as follows:

1. Connect AR and display applications

2. Register displays
3. Collect screen contents

4. Determine screen contents’ positions in AR space

5. Calculate relations between nodes

6. Draw links

The following sections detail the processes and algorithms involved for each of these steps.

5.1.1 Connect the two applications (Display and AR)

In this step, the AR application initiates a network connection to the display applications on different displays/devices, and they subsequently exchange general information, such as the number of displays connected to the companion’s device. A
Table 5.1: Information encoded in the connection QR code

<table>
<thead>
<tr>
<th>Field</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>localIP</td>
<td>The local IP address of the device</td>
</tr>
<tr>
<td>port</td>
<td>The port this device is listening on</td>
</tr>
<tr>
<td>p2pPort</td>
<td>The port that should be used if this device coordinates the P2P network</td>
</tr>
<tr>
<td>p2pServerUp</td>
<td>Set to “true” when this device is the P2P coordinator</td>
</tr>
<tr>
<td>id</td>
<td>Unique ID for this device</td>
</tr>
<tr>
<td>numDisplays</td>
<td>Number of displays connected to this device</td>
</tr>
</tbody>
</table>

QR code is well-suited for this process, as one can encode both the network address of the display’s device and whatever other general information is required, such as the number of displays, port, and P2P server status. The display application builds a QR code which encodes a JSON string containing the device information preceded by a header identifying the code. Table 5.1 shows the different properties encoded within each QR code.

The AR device’s built-in cameras are used to detect and decode QR codes. When a valid QR code is read, the AR application opens a new network connection to the display application. Multiple devices may be connected to the AR application in this way; for example, a user may connect her primary workstation computer and a tablet computer to her AR device simultaneously in order to visualize the links between them. An illustration of this process is shown in Figure 5.2, where the AR application detects a QR code and uses the decoded information to connect to the
5.1.2 Register displays and devices

In this step, the display application initiates a display registration sequence which allows the AR application to discover and track a connected display. Computer vision is used to identify displays based on their visual contents and to determine the physical location and orientation of displays. This is done using image targets, which are images that the AR application can search for to recognize and locate real-world...
objects. These image targets can be premade or created at runtime using screenshots of a display’s visual contents. The full registration sequence described next.

First, the display application sends a notification message to the AR device with information about the display being registered. This includes identifying information about the display, its resolution, and its physical width. The physical width is required to ensure correct real-world placement of the virtual display node.

Next, the display application renders a full-screen image which is known to the AR application. This image is a random array of black and white blocks and acts as an image target. An example of one of these image targets is shown in Figure 5.3. The AR application and display application share a collection of these image targets which are generated ahead of time, and the display application selects one based on the aspect ratio of the display. It should be noted that any image matching the display’s aspect ratio would be permissible for this step. Accordingly, displays can be alternatively registered using a screen capture of the display’s contents if the screen’s contents are sufficiently recognizable to the AR application.

Next, the AR application recognizes the image target and creates a virtual screen-space display node at the display’s perceived location. Computer vision is used to track the display and fix its position in the three-dimensional AR space. Next, the AR application sends a confirmation to the display application that the display has been recognized and is registered in the AR-SA ViL system.

Finally, at this point, the AR application can maintain the position of the display using the AR headset’s built-in tracking capabilities, even when the display is out of the user’s field of view. However, the accuracy of tracking may degrade over
time or as the user navigates the AR space due to imperfection in the AR headset’s tracking capabilities. For example, we observed that if the user’s environment has poor lighting conditions, or if the AR headset’s cameras are obstructed, then tracking quality may degrade. To counteract this, after the display is recognized, the display application periodically sends screenshots of each display’s contents to the AR application. These screenshots are used to rebuild the image target if tracking is lost, avoiding the need to repeatedly display the registration image for calibration.

This registration process allows displays to be recognized and tracked by the AR application without the need for external trackers or other equipment. By simply having an awareness of the visual display contents, the AR application can dynamically track displays in physical space. An important note about the structure of this system is that it relies on two components: the AR headset’s built-in tracking
capabilities and an AR image tracking system. When both systems are active, the image target tracking may malfunction due to screen glare, window movement, or other obstructions and cause displays to lose their proper positions. When this occurs, obvious deviations in AR overlays can be observed by the user. To counteract this, when the virtual display node is correctly positioned, the image target tracking may be turned off by the user to avoid competition between the two systems, as well as to conserve system resources. The user may again enable image target tracking if the display’s position must be calibrated due to a loss of tracking accuracy.

5.1.3 Inform AR application of screen contents

After at least one display is registered, the display application may provide information about that display’s contents. At a minimum, the AR application must know the following properties of an on-screen visual object: (a) its position in screen space (its $x$ and $y$ positions in pixels within the display), (b) its width and height in pixels, and (c) its unique identifier. It is also useful to distinguish between at least three hierarchical categories of objects: displays, windows, and entities. Displays represent the highest category; they contain windows, which in turn contain entities. These categories are represented in the different classes of screen-space nodes. Also included in each object is information about that object’s relationships with other objects, described as a list of related objects’ unique identifiers.

The display application manages visualizations on the device and sends information about each window and entity to the AR application over the local network connection. This registers the window or entity in the AR-SAViL system. Whenever
Figure 5.4: Diagram illustrating display content reporting. The display application observes the position of each window, and each window reports any contents it wishes to register as entities. All information is sent to the AR application.

A new window is opened, metadata is transferred to the AR application as a JSON string, and this information is updated whenever a window is resized or moved within the display. An example of object metadata for an entity representing Canada on a map is shown in Figure 5.5. If a window is closed, then a notification to remove the corresponding window node and its contained entities from the AR-SA ViL system is sent to the AR application. Each window is responsible for independently reporting its contained entities and their updates since window contents are isolated from the display application and the needs of each view can differ. The process of sending window information is illustrated in Figure 5.4. Here, three windows report the positions of contained entities to the display application, and the display application sends information about those entities and windows to the AR application. A rate limiting system is used to prevent the AR application from being overwhelmed with a large volume of update requests.
5.1.4 Determine the AR position of screen content

Once the bounds of objects are known in the screen space, this information can be used in conjunction with the known bounds of the display in the AR scene to place nodes in augmented reality. To accomplish this, the known screen bounds of an object must be transformed to some equivalent bounds within the AR scene. Since the AR application is aware of the display’s resolution, and the ratio between the width and height is equal for both the screen-space node and the real-world display which it represents, this task can be accomplished using a straightforward mapping.
process, described below:

\[
\frac{\text{Screen units}}{\text{Screen size}} \times \text{AR size} = \text{AR units},
\]  

(5.1)

where screen units are described in pixels (e.g., a position within a display or the width of an on-screen object), and the screen size is the width or height of the display in pixels. The AR size represents the width or height of the AR node representing the display, expressed in the units of the AR application’s world. The idea is to map the unit scale from that of the display space (pixels) to that of the AR scene (meters).

Performing this process for the \(x\) position, \(y\) position, width \(w\), and height \(h\) of a visual object will give its AR bounds relative to its parent display. First, the screen position must be corrected, since screen objects are positioned by their top-left corners, while AR nodes are positioned by their center point. If the object has a screen-space parent, such as an entity which is a child of its parent window, the parent’s screen-space position \((x_p, y_p)\) is added to the child’s position:

\[
x_c = x + \frac{w}{2} + x_p, \quad y_c = y + \frac{h}{2} + y_p.
\]  

(5.2)

Next, this corrected position value is normalized to the range \([0, 1]\) in each dimension by dividing these values by the display’s resolution \((w_d, h_d)\), where \((0, 0)\) corresponds to the top-left corner of the display, and \((1, 1)\) corresponds to the bottom-right corner. The height and width are similarly normalized to the range \([0, 1]\), where a value of 0 corresponds to zero size, and a value of 1 corresponds to the entire height
or width of the display:

\[ x_n = \frac{x_c}{w_d}, \quad y_n = \frac{y_c}{h_d} \] \hspace{1cm} (5.3)

\[ w_n = \frac{w}{w_d}, \quad h_n = \frac{h}{h_d}. \]

With these normalized position values, the three-dimensional position and size of the node relative to its parent can be determined by multiplying with the display node’s bounds \((w_{dn}, h_{dn})\), correcting again for the fact that AR objects are positioned from their center points rather than the top-left corner:

\[ x_a = w_{dn}x_n - \frac{w_{dn}}{2}, \quad y_a = \frac{h_{dn}}{2} - h_{dn}y_n \] \hspace{1cm} (5.4)

\[ w_a = w_{dn}w_n, \quad h_a = h_{dn}h_n. \]

An important detail is that the Y axis is flipped when moving from screen space to AR space, since in screen space, Y increases in the downward direction, and in the AR scene, Y increases in the upward direction.

An algorithm for this process is shown in Figure 5.6, where information about an object’s screen-space bounds, \(S_b\), the resolution of the display, \(R\), and the size of the display node representing that display, \(D_s\), are used to update the position and size of a screen-space node \(N\). Each node keeps track of the last screen-space bounds used to update its position and size. Nodes also have a “local” position and scale which represent the position of the node in the AR scene relative its parent (in this case, the display containing the node). The algorithm recurses by calling itself for all children of the updated node. If a child updating could cause a subsequent update to its parent, then the algorithm would infinitely recurse. However, children
**Input:** Screen-space node $N$, screen bounds $Sb$, screen resolution $R$, display node size $Ds$. Bounds objects are defined as $(x, y, width, height)$

**function** `ScreenToAR(N, Sb, R, Ds)`

```plaintext
N.Sb ← Sb
Sb.x ← Sb.x + \(\frac{Sb.width}{2}\)  \(\triangleright\) Move from top-left to center
Sb.y ← Sb.y + \(\frac{Sb.height}{2}\)
if $N$ has a parent $P$ in screen space then
  Sb ← Sb + P.Sb
end if
Nb ← (0, 0, 0, 0)  \(\triangleright\) Normalize screen bounds
Nb.x ← \(\frac{Sb.x}{R.width}\) \(\frac{Sb.y}{R.height}\)
Nb.y ← \(\frac{Sb.width}{R.width}\) \(\frac{Sb.height}{R.height}\)
Ap ← (0, 0, 0)  \(\triangleright\) Calculate 3D AR coordinates
Ap.x ← \(Ds.width \times Nb.x - \frac{Ds.width}{2}\)
Ap.z ← \(\frac{Ds.height}{2} - Ds.height \times Nb.y\)
N.localPosition ← Ap  \(\triangleright\) Set position relative to parent
Asize ← (0, 0, 0)
Asize.x ← \(Ds.width \times Nb.width\)
Asize.z ← \(Ds.height \times Nb.height\)
N.localScale ← Asize
for all children $C$ of $N$ do
  ScreenToAR($C, C.Sb, R, Ds$)
end for
end function
```

**Figure 5.6:** Position a screen-space node according to its screen coordinates

of a node can never be higher than their parent in the node hierarchy, *i.e.*, a window will never contain a display, and an entity will never contain a window or a display. Thus, a child updating can never cause its parent to also update, and the algorithm terminates after all the node’s children have updated.

Using this approach, visual objects can be placed in their correct location without the need to visually recognize and track them using techniques such as computer vision or motion capture. Accordingly, if a visual object moves or changes size,
the companion application need only inform the AR application of the change, and
the rendering process can be repeated to reposition the object. During this process,
if the visual object happens to be a window containing one or more entities, these
entities inherit the transformation of the parent window, rather than the companion
application sending updates for each individual entity.

5.1.5 Determine relationships between different nodes

Once multiple objects are tracked, the relationships between these objects can
be evaluated to determine which objects should be linked across displays. These
relationships can be provided by the display application, included with the object
metadata as previously described. Relational data can also be obtained from other
sources, such as a database or data files on the AR device. Relationships are identified
using the unique identifiers of objects, however they could be found using keywords or
tags. For example, a chart entity representing a business named Technology, Inc. may
explicitly report to the AR-SA ViL system that it is related to the node “technologyInc-
stock,” which represents a chart of the business’s stock history. Bicluster nodes state
relations using keywords, and the AR application will search active nodes for matches
to those keywords to determine relationships. Users may also define relationships by
selecting multiple nodes. Every individual relationship is a one-to-one relationship
between two nodes, however each node can be involved in any number of relationships.
Thus, one-to-many and many-to-many relationships are also possible.
Table 5.2: Locations for each Bezier control point

<table>
<thead>
<tr>
<th>Control Point</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$A$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$(M.x, A.y, M.z)$</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$(M.x, B.y, M.z)$</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$B$</td>
</tr>
</tbody>
</table>

Figure 5.7: Example of links drawn between nodes as Bezier curves

5.1.6 Draw the AR visual links across registered displays

Once relationships have been determined, related nodes are linked using cubic Bezier curves between node positions $A$ and $B$ with control points $P_0, P_1, P_2,$ and $P_3$. The positions of each control point are defined in Table 5.2, where the midpoint $M$ is defined as $\frac{A+B}{2}$. Figure 5.7 shows multiple visual links being drawn as Bezier curves between nodes.
Links are continuously updated to connect nodes as they move in AR space according to user movement or input. If a node is removed from the system, then any links connecting to that node will also be removed.

When two screen-space nodes are connected, the connections pass through proxy nodes, which are created dynamically as connections are drawn. When a connection is being made, the first proxy connected to the source node is selected if one exists. If not, the first proxy connected to the destination node is selected if one exists. If a proxy is still not found, one is created. The relationship is added to the proxy, and both screen-space nodes are connected to the proxy node. The process for connecting screen-space nodes is shown in Figure 5.8. In this algorithm, the Connect function is called to instruct AR-SA ViL to create a visual link between two nodes, and the first proxy node attached to a node (if one exists) is assumed to be readily available. “Active nodes” refers to any node which is available to be connected by visual links; in our implementation this means any node that is registered in AR-SA ViL.

For the case of bicluster nodes, a different strategy is used. Since biclusters comprise a relationship view between multiple datasets, bicluster nodes have a list of relations for each dataset they connect. Each of those relations is a keyword which can be used to discover other nodes in the AR-SA ViL system. After finding all nodes matching the bicluster’s relations, links can be drawn between the bicluster node and each screen-space node. This process is shown in Figure 5.9. As in Figure 5.8, a Connect function exists to connect two nodes via visual links. “Active biclusters” in our implementation refers to any bicluster node enabled in the AR scene and selected
Input: A list of nodes $Nodes$, each containing a list of related nodes $Relations$

function $ESTABLISHCONNECTIONS(Nodes)$

for all active nodes $A$ in $Nodes$ do
    $P \leftarrow$ First proxy node connecting $A$, otherwise $null$
    for all active nodes $B$ in $A.Relations$ do
        if $P$ is $null$ and $B$ is connected to a proxy then
            $P \leftarrow$ First proxy node connecting $B$
        else
            $P \leftarrow$ New proxy node between $A$ and $B$
        end if
        CONNECT($A,P$)
        CONNECT($P,B$)
    end for
end for
end function

Figure 5.8: Connecting screen-space objects

Input: A list of bicluster nodes $BC$, each containing a list relations for each dataset

function $CONNECTBICLUSTERS(BC)$

for all active biclusters $B$ in $BC$ do
    for all datasets $D$ connected by $B$ do
        for all keywords $K$ in $B.Relations[D]$ do
        $N \leftarrow$ Any node matching $K$, otherwise $null$
        if $N$ is not $null$ then
            CONNECT($B,N$)
        end if
        end for
    end for
end for
end function

Figure 5.9: Connecting bicluster nodes to related screen-space nodes

by the user. Keyword matching is implemented by matching the unique ID of a node against a regular expression constructed using the keyword.
5.2 Implementation

The AR application was created in C# using the Unity game engine [65] (version 2020.3.34f1), utilizing Vuforia [66] for AR tracking features and the Mixed Reality Toolkit [67] for AR interfaces, components, interactions, and gestures. An implementation of the Socket.IO protocol for Unity [68] based on the websocket-sharp [69] library was used for networking, and the ZXing.NET [70] library was used for QR code recognition and decoding.

The display application was created in NodeJS using the Electron framework [71], augmented with the electron-window-manager [72] package. Socket.IO [73] was used for network communications, and the socket.io-p2p [74] package was used to manage the peer-to-peer network. For visualizations, D3.js [75] was used. Electron uses multiple processes, including the main process which coordinates the application and renderer processes which are Chromium web views that act as user interfaces. In AR-SAViL, the main Electron process is responsible for managing network communications as well as the various views on the device. Electron also provides inter-process communication infrastructure, which allows different windows managed by the application to communicate, and this functionality is leveraged to implement the passing of screen object metadata from renderer processes to the main process, as well as to exchange other information (e.g., display properties, network adapter properties) or trigger events, such as sending messages in the P2P network.

In summary, the AR-SAViL system consists of two applications: the AR application, which runs on the AR headset, and the display application, which runs on
each other device in the system. The two applications connect over a local network and form a peer-to-peer network connecting all devices in order to exchange information. Display registration and tracking is supported using computer vision to identify and position displays based on their visual contents, allowing displays to be tracked using only the AR headset. The display application collects metadata about screen contents and then sends this metadata to the AR application, which in turn positions and sizes screen-space nodes in the AR scene to match the visual contents of each display. Then, once relationships between nodes are known, visual links are drawn to represent those relationships in the AR scene.
CHAPTER 6

SENSEMAKING TOOL

To demonstrate and explore the capabilities of the AR-SAViL system, a collection of sensemaking tools was developed. These tools allow visualizations and documents displayed within a web view to be seamlessly integrated with the AR system by connecting screen elements with visual links and leveraging user hand gestures to explore data. A collection of views was created based on previous work from SightBi [12] to explore the Sign of the Crescent dataset [76]. The user can use the information provided in these views and the connections and elements drawn between views by the AR-SAViL system to perform analysis of the represented data.

These views include map, graph, and bar chart views, as well as a document viewer. Additionally, three collections of user-manipulable bicluster nodes were added to explore relationships between the map, graph, and bar chart data. Each of these views were augmented to register their contained elements with the AR-SAViL system and draw visual links in the three-dimensional AR scene. To support the sensemaking tool, we use AR-SAViL’s peer-to-peer network system to facilitate cooperation between all the views and create an interactive analysis space. In this chapter, we will discuss each of these sensemaking tools.
6.1 Dataset Views

The Sign of the Crescent dataset [76] consists of a series of documents which contain references to locations, names, and organizations, and we provide three visualizations [12] as views which illustrate these references. When visual elements within these views are registered with the AR-SA ViL system, a unique ID is assigned to them based on which view they come from and what data item they represent. These unique IDs are used to explicitly state relationships between nodes from each view so that links can be easily drawn between them. In this section, we will discuss the map, graph, and bar chart views, as well as a document viewer and AR tools, all of which use data from this common dataset.

6.1.1 Map View

The map view features a global map populated with geographically-distributed markers. In Figure 6.1 (a), the entire map is shown, while in Figure 6.1 (b), the user has zoomed in and selected New Jersey. Each of these markers represents a significant map location in the underlying dataset. The map is generated using D3.js with additional behavior added to integrate the visualization with the AR-SA ViL system. The user can explore the map by panning around with the mouse and zooming in and out with the scroll wheel or touch gesture. If the user shift-clicks on a map marker, related documents are opened in separate views.

Initially, the map view does not add any elements to the AR-SA ViL system. Once the user hovers over or clicks (toggles) a map marker, however, the registration
Figure 6.1: Screenshots of the Map view
process begins. At that time, the dataset is explored to construct a list of related persons and organizations for that location. The position and size of the marker is also obtained from the DOM, and this information is sent to the main display application process to register the map marker with the AR-SAViL system. Using the information of the marker’s location on the display and its relations, the AR application can create a screen-space entity node to draw links between map markers and other visual elements. Whenever the user navigates the map, the updated position and size of all registered markers is also sent to the AR application to update the virtual screen-space node positions. When the user moves the mouse out of a map marker element (if that element is not toggled on), a request is generated to remove that marker’s entity from the AR-SAViL system. Unique IDs are assigned to map entities in the form of “sbloc-{location name}.”

6.1.2 Graph View

The graph view (shown in Figure 6.2) features an undirected graph containing various persons of interest and their relationships. This graph is created from the same underlying dataset as the other views discussed in this chapter. The graph is a D3.js visualization like the map view, and features similar zooming and panning controls. Like the map view, shift-clicking an element in the graph view will open documents related to the selected person of interest.

The integration of the graph view with the AR-SAViL system follows the same pattern as the map view. When the user hovers over a graph node, the relationships between the represented person and various map locations and organizations are
calculated and the bounds of the object within the DOM are found. This information is sent to the AR-SAViL system so that the graph node can be represented as a virtual screen-space entity node. Clicking a node toggles it on and off, which allows multiple nodes to be active at a time. Navigation of the graph view triggers a process which updates the positions of all active graph nodes in the AR-SAViL system. If the user moves her mouse out of a graph node that is not toggled on, then the node’s corresponding entity is removed from the AR-SAViL system. Unique IDs are assigned to graph entities in the form of “sbname-{person name}.”

Figure 6.2: Screenshot of the Graph view. The user has selected Clark Webster.
6.1.3 Bar Chart View

The bar chart view (shown in Figure 6.3) features a D3.js bar chart representing the frequency of various organizations in the underlying dataset. By default, the bars are ordered according to reverse alphabetical order, however they can be instead sorted by frequency using a checkbox in the top left corner. Bars can be selected and deselected with a mouse click, and shift-clicking in the bar chart view will also open any documents related to the selected organization.

When a user hovers over a bar, it is registered in the same way as the map and graph view to the AR-SAViL system. Its relations to map locations and persons of interest are calculated and sent along with its DOM bounds, and a corresponding virtual screen-space entity node is created in the AR application. Moving the mouse outside of the bar will remove its entity from the AR-SAViL system if the bar is not currently selected. Unique IDs are assigned to bar entities in the form “sborg-{organization name}.”

6.1.4 Document Viewer

Users can also explore plain text documents using the document viewer to investigate the context behind the data visualized in each other view. The user may highlight text by making a text selection and right clicking. User-generated highlights can be cleared by right-clicking on the highlighted text. When text is highlighted, all occurrences of that text in the entire document are also highlighted. The document view will also highlight keywords based on the user’s selections in other views. User-
Figure 6.3: Screenshots of the Bar Chart view

(a) The organizations are presented in reverse-alphabetical order

(b) The organizations are sorted according to frequency
generated highlights are shown with an orange color, while highlights generated based on the user’s outside selections are shown with a blue color. Figure 6.4 shows a document where the user has highlighted the text “University of Virginia,” and other keywords have been highlighted based on the user’s selections in other views.

When a highlight is created, the DOM bounds of the highlight are found and an entity node is registered with the AR-SAViL system. If a user-generated highlight is cleared via a right click, then that highlight’s entity node will be unregistered from the system. Unlike other views, the document view may feature multiple entities which represent the same data item. Therefore, unique IDs are assigned to highlight entities based on the document’s internal ID, the highlighted text, and the position of the highlighted text within the document. For example, the text “Mario” highlighted in the document “m10” starting at character 15, the unique ID for this highlight’s
entity would be “sbdoc-Mario-m10-15.” Since this ID is not predictable for any given keyword, other views cannot reliably state relations to entities spawned from the document view. To solve this, highlight entity nodes’ relations are set to any elements from other views containing the highlighted keyword. For example, the highlighted “Mario” text will simply connect to entities with the IDs “sbloc-Mario,” “sbname-Mario,” or “sborg-Mario.” This allows highlights to connect to elements in any of the three other views that match the contained text. Additional consideration is also made to handle acronyms which may not be used between views, such as “NYC” for “New York City” or “VA” for “Virginia.”

6.1.5 Biclusters and AR Navigation

To support the discovery of relationships between different views [12], AR-space nodes representing precomputed biclusters between each pair of views (i.e., map-graph, graph-bar, map-bar) are provided. These bicluster nodes are generated in a collection, such as the one shown in Figure 6.5 (a), and arranged according to their similarity (how strongly their represented relationships overlap). Each bicluster node is created with two lists of related keywords, each relating to one of the connected views (e.g., a node in the map-graph collection will have a list of locations and a list of names). Connections are made between these nodes and entities generated from other views by matching these keywords with the unique IDs of those entities. These collections are encoded in a JSON format which stores meta information including the ID for each node, its position, its relations, and the size each bar should be drawn at.
Biclusters do not immediately create links, but when “focused” on through hand gestures such as finger proximity/pointing or when activated through AR selection gestures, they are marked as “active” and links are drawn as described in Chapter 5. Bicluster nodes (as well as proxy nodes) can be repositioned using standard AR manipulation gestures such as pinching and dragging. Collections can be toggled through buttons found on a menu attached to the hand. When the user presents a palm-up hand, the menu is displayed, shown in Figure 6.5 (b), and the contained buttons can be pressed using another hand. When a collection is toggled on, it is positioned in front of and slightly above the user to ensure all nodes are created at a comfortable position.

6.2 Peer-to-Peer Protocol

Using the aforementioned strategies, links are drawn between any entities which the user selects. However, this alone is insufficient, because the AR-SAViL system would not provide user guidance toward related nodes that they have not selected. This could be solved through schemes which register all entities up front, however this results in numerous superfluous nodes being generated in the AR application, degrading performance, especially when the positions of nodes must be updated. A different strategy is needed to preserve system performance while still enabling an interactive analysis space.

To this end, all of these elements implement a standard protocol used to request the submission of information from other views. This protocol includes two messages: a JOIN message and a LINK message. The LINK message is used to
(a) A collection of bicluster nodes; the user has selected one using a pointing gesture.

(b) The menu used to toggle the presentation of each collection.

Figure 6.5: Screenshots of augmented reality-based features
broadcast information about what data items a particular view would like other views to register. Every view is given some persistent name, such as “loc” for the map view and “ar” for the AR application. When a view broadcasts a LINK message, it includes its name as well as the elements it is requesting from each other view. An example of a LINK message is shown in Figure 6.6. The JOIN message is broadcast by a view when it initializes, and acts as a request for other views to broadcast LINK messages. Views also maintain sets of IDs for their own outgoing requests and entities requested by other views.

When a user selects an element, the selection’s calculated relations are added to the view’s outgoing requests and a LINK message is broadcast. If the user deselects an element, any of its relations that are no longer needed (i.e., relations shared with no other selected elements) are removed from the set of outgoing requests, and the LINK message is broadcast.
message is rebroadcast. Views also broadcast an empty LINK message when closing to inform other views that they may unregister the sender’s requested elements.

When a view receives a LINK message, it will add the incoming requests to a set of requests from that message’s sender. Simultaneously, it will remove any requests from the set that were not requested this time. Then, the view will submit to the AR-SA ViL system any requested entities which it can provide and remove entities that are no longer being requested. This allows visual links to be drawn in AR between elements even if the user did not manually select them through interaction with only a single view.

Using this protocol, all views can dynamically request and provide any entities necessary to guide the user toward related nodes via AR visual links. This allows visualizations distributed across displays, devices, and platforms to be uniformly integrated into a seamless AR sensemaking tool utilizing the AR-SA ViL system.
CHAPTER 7

USAGE SCENARIO

To demonstrate a usage scenario for the AR-SAViL system, the sensemaking tools described in Chapter 6 were utilized to analyze and explore a dataset and investigate details associated with a fictitious terrorist plot. The analyst utilized the Microsoft HoloLens 2 shown in Figure 7.1, a see-through AR head-mounted display which is capable of mixed reality renderings. The HoloLens 2 provides advanced real-time inside-out tracking features as well as hand and eye tracking technology which enables intuitive physical interaction gestures (e.g., tapping, dragging, pinching). AR-SAViL’s AR application ran on the HoloLens 2 while display applications ran on separate traditional display-attached devices. The HoloLens 2 cameras, tracking, and gestures are used to navigate the AR application’s interactive analysis space in real time. Utilizing on-screen visualizations and documents alongside AR-SAViL’s AR visual links and bundling features, the analyst is guided through connections between separated entities across devices, displays, and views, and eventually uncovers the nature of the plot. The views and data used based on the Sign of the Crescent dataset [76] were adapted from the SightBi [12] system to utilize AR-SAViL’s AR visual link and peer-to-peer collaboration system.
The analyst's workspace consists of many scattered devices with varied platforms and form factors, such as a traditional desktop, an all-in-one device, and a tablet computer. To make sense of information spread across all these devices, she uses the AR-SAViL cross-display visualization system. She is running the AR application on the Microsoft HoloLens 2 and connects to each device by pointing her view toward QR codes shown in the display application, shown in Figure 7.2 (a).

The analyst opens several views on different displays via the display application to organize information spatially. She arranges different visualizations onto different displays, including a map, graph of persons, bar chart of organizations, and documents. She examines the visualizations, making notes about relationships between individuals in the graph, which geographic regions have more markers, and
which organizations are more frequently mentioned from the document data. How-
ever, the relationships between these views are unclear.

To make sense of the relationships between these entities, she begins utilizing the AR-SA ViL system. She registers all the displays in her workspace so they and their contents can be tracked in the augmented reality environment. To do this, for each display, she provides its physical width and instructs the system to initiate the registration procedure. The user interface for this process is shown in Figure 7.2 (b). The AR application recognizes the visual contents of each display, such as via a registration target shown in Figure 7.2 (c). After the AR application recognizes each display, she closes the registration target if it is opened (as shown in Figure 7.2 (d)) and she locks the display into place by disabling image tracking for that display and continues to the next, until all displays in her workspace are registered.

Now that the AR application can locate each display, view, and entity in physical space, she begins inspecting elements in each view. As she examines each element, the AR-SA ViL system collects information from each device and draws links between displays to direct her attention to related elements. She notes in the bar chart view that the organizations Al Qaeda and Taliban appear frequently in the documents and selects these elements in the bar chart view. AR-SA ViL broadcasts requests for related elements to all other devices and draws links between these related entities. When many elements share relations, links are combined via proxy nodes to represent one-to-many and many-to-many relationships and reduce visual clutter. After links are drawn, she arranges proxy nodes to route links into a comfortable and organized layout, as shown in Figure 7.3.
She takes note of individuals and locations linked to these organizations. She notices that these organizations are linked to Afghanistan, and requests AR-SAViL to find entities related to this location, and requests documents related to the location to be opened. Figure 7.4 shows Afghanistan being connected with links to related documents and graph nodes.

She arranges documents and views onto different displays to organize information, and links are drawn to connect these separated elements. Within each document,
Figure 7.3: Connections between bar, map, and graph views based on user selection of two organizations

Figure 7.4: Connections between map, graph, and multiple documents after user selects Afghanistan
Figure 7.5: Connections between graph and relevant documents for four selected individuals

text matching Afghanistan and any other related elements are highlighted, and the analyst combines all known information thus far to identify several persons of interest, including Abdul Ramazi, Clark Webster, Faysal Goba, and Muhammed bin Harazi, among other names. She opens documents related to these individuals on separated displays and follows cross-display links to better understand them. This is shown in Figure 7.5, where four selections in the graph view connect to text in six different documents.

The analyst wishes to better understand how these entities are related, and to get a sense of which relationships are significant, and thus requests AR-SAViL to show biclusters for each pair of views (e.g., Figure 7.6 (a) where the nodes surrounded by a dotted circle are bicluster nodes). She checks how similar bicluster nodes link to the noted entities and closes in on a more focused set of useful nodes. She requests AR-SAViL to open documents related to these interesting nodes on different displays.
and investigates the relations between elements (as shown in Figure 7.6 (b)). After reading the document text with the guidance of visual links between displays, she notes suspicious transactions to and from Abdul Ramazi, an alias used by former Taliban associate Muhammed bin Harazi, with funds being transferred to a Mukhtar Galab and Hani al Hallak. Galab is shown to have a reservation for AMTRAK Train #19 paid for by Faysal Goba, and al Hallak manages a carpet shop which was found to hold cartons containing C-4 explosive. The analyst surmises that Abdul Ramazi provided funds for this C-4 and that AMTRAK Train #19 may be the target for a terrorist attack.

We interviewed an expert on augmented and virtual reality business applications from a local university to better understand the advantages, disadvantages, and potential applications of AR-SAViL. The expert has 12 years of experience with usability testing and data analysis. The expert used AR-SAViL on the Microsoft HoloLens 2 to explore the Sign of the Crescent dataset [76] utilizing the workspace shown in Figure 7.7 consisting of a Windows laptop with an attached high-definition television and an iMac. The registration process of devices and displays was completed ahead of time, and the expert was instructed on how to interact with the AR-SAViL system by making selections in legacy visualizations, dragging AR nodes through physical space, and using the palm menu to add and remove elements from the analysis space. Since the expert had prior experience with the HoloLens 2, the expert quickly intuited the available interactions. After the expert used the system for approximately 15 minutes, a 15-minute interview was conducted to understand the expert’s thoughts on our system.
Figure 7.6: Various AR-SAViL cross-display visualizations
The expert responded positively overall to the system and gave feedback about which aspects performed well and which aspects could be improved. When asked what the expert liked about the system, the expert responded, “I think, for someone who knows the [HoloLens 2], it was very intuitive... visually, I thought [the links were] easy to follow.” The expert also felt the system was very novel and usable, and stated that it could assist with “cognitive overload” which can occur in applications where information across numerous displays must be quickly synthesized, such as drone...
pilots who must monitor information on many displays and decisively determine when to take corrective or defensive actions.

When asked which aspects the expert did not like, the expert noted that some links did not align precisely with their destinations due to inaccurate display calibration, “I could figure out [the link destinations] based on looking at multiple points... but that was a little confusing.” The expert also noted a delay which occurred when drawing links, during which time AR elements would temporarily disappear as the AR application processes information, occasionally interrupting their usage of the system.

When asked how the system could be improved, the expert noted that “the [proxy nodes] were all clustered” between the three displays and suggested that the system would perform best in environments where displays are more spread out so that the three-dimensional nature of the links could be better exploited. For the bicluster nodes, the expert was unsure of how to utilize the view and suggested adding labels to provide additional context to the user. Adding a method to control views entirely in AR for certain common actions like zooming and panning was also suggested. Another suggestion was the addition of a button or other control to globally clear all selections and remove any links, which would allow the user to quickly clean their workspace and begin investigating other elements.
CHAPTER 8

DISCUSSION AND CONCLUSION

We believe that AR-SAViL provides a strong framework to enable augmented reality data analysis in many-display, multi-device environments, however our implementation has some key limitations in terms of usability, and additional design considerations are necessary to enhance the system’s overall capabilities. We identified several key areas in which AR-SAViL under-performs and consider future improvements that can be made to the system to improve its utility.

8.1 Discussion

Firstly, an important design goal for AR-SAViL was to achieve tracking of physical displays in the virtual AR space without the need for permanent external trackers to be installed in users’ workspace. This design goal coincided with the need to be aware of the contents of each display, and consequently, the display application was developed which provides screenshots or alters display contents to assist the AR application in this purpose. The underlying image tracking subsystem provided by the Vuforia library proved to perform decently well for this purpose when displays are in the user’s view, but quickly malfunctioned when displays were out of view. The
result is that the accuracy of tracking would degrade, or displays would be removed from view if their contents could not be readily recognized for any reason, such as obstructions, glare, or movement. Another inherent limitation of this image tracking based approach is the reliance on the visual contents of each display to calibrate its position. If the system relies on screenshots to recalibrate tracking and the display is not well-populated with visual elements, then it is unlikely that any provided screenshot will assist in identifying that display.

To address the image tracking problem, a means of disabling this image tracking behavior was added so the user can “lock” the display in place once its position is calibrated. At this point, the spatial awareness system of the HoloLens proved largely sufficient to keep the display fixed in space. The result of this, however, is a new usability problem: the user must be responsible for chaperoning the accuracy of tracking and deciding whether to lock the display or allow the image tracking system to recalibrate its position. We believe that this would be confusing for most users and shows that more work is needed to solve this problem. Ideally, the AR application could identify the health of each display’s tracking, such as by detecting user motions or actions, and automatically take corrective action. Consideration is also needed to determine if any guidance or automated action can be taken to ensure screen contents are trackable (i.e., ensure each display has trackable content during calibration) without becoming intrusive to the user’s workspace. We believe future work should place a large focus on this issue, as accurately and stably tracking displays is fundamental to the success of AR-SAViL.
Another usability issue appeared when switching target AR devices from an Android phone to the HoloLens. In order for the HoloLens to accurately size and position displays with the image tracking system, the precise physical width of the display must be known. This means that, while the user need not install external trackers in her environment, the user must measure the width of each display as precisely as possible to ensure the display is correctly tracked. We observed during development that if the display is reported undersized, then the HoloLens will assume the display is closer than reality, and if the display is reported oversized, it will assume the display is farther away. This adds an additional step to the initial display registration process which we believe would be tedious for the user. A simple solution to this issue was to simply save the physical width of each attached display the first time the user enters it, and in future work, we believe that this saved information could also be used to automate the registration process when the display application connects. Additionally, because each display’s model can be internally identified by the display application, future work could consider constructing a database which is aware of the exact physical width of many popular displays, potentially avoiding the need for some users to measure at all.

We believe that in future work, another enhancement which could be made is to use features of the HoloLens that could potentially allow the AR application to remember the physical positions of each display within the user’s environment. Since many displays are rarely moved, this would ideally allow users to skip display registration in many cases. Experimentation could also be done to determine if the physical width of each display can be determined accurately only in AR, such as
through a “virtual tape measure” which measures the actual physical distance between two points set in AR.

When AR-SA ViL connects screen-space nodes, it will automatically generate manipulable proxy objects which can collapse many shared relations to pass through one central point. This is helpful to reduce visual clutter and to allow users to reroute links at will in three dimensions, however such bundling schemes by definition remove explicitness in the connections between nodes. As such, when numerous nodes are connected in a many-to-many relationship, we observed during development that discerning the relationships between constituent nodes becomes difficult. For example, if the user selects the entire United States on a map, that one node could relate to every single individual from the United States in another view (one-to-many). If the user then also selects Virginia, she might expect an additional one-to-many relationship to be drawn between Virginia and all individuals from only Virginia, but this relationship would instead be combined into the existing US relation (creating a many-to-many relationship), which would potentially degrade the semantic clarity of the links. To address this in future work, we believe a more intelligent bundling algorithm could be developed which considers such cases, and we have also considered that user controls could be added which allow the “expansion” proxy nodes into only one-to-many connections.

The AR-SA ViL node and linking systems are currently fundamentally based on explicitly stated unique object IDs, and we believe that this introduces limitations on how the system can grow and adapt to certain problems and workflows. For instance, if a user decides to open the same view on two displays, the system will simply assume
that the view has moved to a different display when the second copy is opened, since both views would share the same unique ID. There are schemes to circumvent this problem within the same framework, however we believe that different strategies for deriving relations should be considered moving forward. In future work, we may investigate the development of strategies such as a tag-based system wherein objects which share tags are connected or natural language processing-based strategies which connect nodes based on observed semantic relationships.

Other miscellaneous limitations of the AR-SA ViL system include the assumption of flat, rectangular displays, which disqualifies some common display form factors, such as curved monitors or other displays with a third dimensional element to their form. The system also assumes the existence of a common local network infrastructure existing between all involved devices, which we found in development can be troublesome in some network environments. Based on feedback from the interviewed expert in Chapter 7, investigation should be done to determine whether users prefer doing operations mostly in AR space, as opposed to switching interaction modes between traditional devices and using AR gestures. If this is found to be true, then future work should investigate further expanding the capability of AR-SA ViL to manipulate display contents using augmented reality controls.

Though we identified these issues during our own testing of the AR-SA ViL system, a deeper user study has not been done to thoroughly evaluate its efficacy for visual analysis and sensemaking tasks. Future testing will be necessary to ascertain the true strengths and weaknesses of AR-SA ViL and guide further enhancements toward developing the system into a more mature tool for visual data analysis.
8.2 Conclusion

Modern workspaces can simultaneously utilize a wide variety of devices and displays, from small form-factor devices like smartphones to large projector displays. Supporting visual analysis and sensemaking tasks in such a multi-display environment has been a challenge using only on-screen visual elements.

Our AR-SA ViL system provides a novel solution to the visualization of cross-display relationships using an AR headset to draw visual links in physical space, supporting visual analysis and sensemaking tasks in such environments. Our display application enables an image tracking approach wherein displays on many devices can be identified and positioned in the three-dimensional virtual space without the need for external trackers or other equipment, as well as enabling awareness of the positions of relevant screen contents through communication via a local network. We utilize bundling techniques to reduce visual clutter and enable the manual rerouting and arrangement of links in physical space, and address challenges in coordinating cross-device visualizations through a peer-to-peer infrastructure which enables collaboration between independent views.

With respect to our research questions, our AR visual link approach has shown promise in effectively guiding users to related information across displays. The visual links in combination with our proxy objects allow users to reroute links to create three-dimensional layouts, and users can follow links between displays to support visual analysis and sensemaking tasks. Our display tracking solution based on an awareness of visual display contents shows promise in supporting the observation of the physical
locations and orientations of varied displays. Even so, more work is needed to perfect this technique in terms of usability and precision so that it performs optimally in a broader range of conditions. AR-SA ViL is appraised to provide an intuitive and usable AR analysis ecosystem for navigational and sensemaking tasks in multi-display environments. Finally, the cross-device peer-to-peer network in combination with the information reporting capabilities of the display application enable the automated collection of information. An expert found that links drawn using this information were easy to follow, potentially supporting more efficient analysis of data scattered across many displays.

We believe that AR-SA ViL acts as an effective augmentation of existing visual data analysis workflows on traditional displays by expanding the practicality of utilizing numerous displays. Our sensemaking tools provide a framework based on existing visualizations which can potentially be adapted or expanded for a variety of sensemaking tasks, integrating multiple displays and environments into a single ecosystem where information is collected and visually connected. When elements across displays can be directly linked in the user’s physical space, understanding relationships across many displays may becomes simpler and analysts may leverage more devices and displays, organizing information and increasing the breadth of their visual analysis space. AR-SA ViL shows promise in enabling analysts to simultaneously consider a much larger information set than would be possible using traditional displays alone, potentially increasing the efficiency of visual analysis and sensemaking tasks.
REFERENCES


87
APPENDIX A

SCREENSHOTS OF Bicluster Collections

Figure A.1: Name-Location Biclusters
Figure A.2: Name-Organization Biclusters