Multiview Immersion in Hybrid Reality Environments

BY

ALESSANDRO FEBRETTI
M.S. Computer Science, University of Illinois at Chicago, 2008
B.S. Computer Engineering, Politecnico di Milano, 2006

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2017

Defense Committee:

Andrew E. Johnson, Chair and Advisor
Robert V. Kenyon
Luc Renambot
Leilah B. Lyons
Michael E. Papka, Argonne National Laboratory

To my wife Katy, for her patience and never ending encouragement. I started this journey alone: I'm so happy to end it with you by my side. And to Emilia: I can't wait to learn all the things you will teach me.

ACKNOWLEDGEMENTS

I would first like to thank Dr. Andrew (Andy) Johnson for inspiring me to pursue a Ph.D. and for being my Advisor during my entire graduate career. On all the research projects I worked on, Andy was always ready to provide guidance, while leaving me free to trace my path through the challenges I encountered. I could not have asked for a better mentor. I would also like to thank Dr. Luc Renambot, Lance Long and Arthur Nishimoto for the significant technical support they provided on both the CAVE2 hardware and software systems. The work presented in this dissertation would have been much harder without their help. I believe this is the greatest lesson I have learned from my time at the Electronic Visualization Lab. We help, challenge and mentor each other regardless of whether we are students, faculty or staff. I'm much better, professionally and as a person, thanks to the wonderful people I have met at EVL. Thank you all.

AF

CONTRIBUTION OF AUTHORS

This dissertation incorporates portions of previously published work by myself and other authors. All portions contained within in this dissertation represent my direct contributions to these publications. The prior publications included in this work are the following:

In chapters 1, 2, 3: *Febretti, A., Nishimoto, A., Mateevitsi, V., Renambot, L., Johnson, A., & Leigh, J. (2014, March). Omegalib: A multi-view application framework for hybrid reality display environments. In Virtual Reality (VR), 2014 iEEE (pp. 9-14). IEEE. © IEEE*

In chapters 1, 3: *Reda, K., Febretti, A., Knoll, A., Aurisano, J., Leigh, J., Johnson, A., Papka, M.E. and Hereld, M., 2013. Visualizing large, heterogeneous data in hybrid-reality environments. IEEE Computer Graphics and Applications, 33(4), pp.38-48. © IEEE*

In chapter 1: *Febretti, A., Nishimoto, A., Thigpen, T., Talandis, J., Long, L., Pirtle, J.D., Peterka, T., Verlo, A., Brown, M., Plepys, D. and Sandin, D., 2013, March. CAVE2: a hybrid reality environment for immersive simulation and information analysis. In Is&t/spie electronic imaging (pp. 864903-864903). International Society for Optics and Photonics.*

In chapter 1: *Febretti, A., Mateevitsi, V.A., Chau, D., Nishimoto, A., McGinnis, B., Misterka, J., Johnson, A. and Leigh, J., 2011, September. The OmegaDesk: towards a hybrid 2D and 3D work desk. In International Symposium on Visual Computing (pp. 13-23). Springer Berlin Heidelberg.*

Permission to use these works is detailed in Appendix C.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

API:	Application Programming Interface
AUV:	Autonomous Underwater Vehicle
CAMI:	Center for Advanced Molecular Imaging
CAVE:	CAVE Automatic Virtual Environment
CIERA:	Center for Interdisciplinary Exploration and Research in Astrophysics
CPU:	Central Processing Unit
DOF:	Degrees of Freedom
EEG:	Electroencephalogram
ENDURANCE:	Environmentally Non-Disturbing Under-water Robotic ANtarCtic Explorer
EVL:	Electronic Visualization laboratory
GB:	GigaByte
GPU:	Graphical Processing Unit
HDF5:	Hierarchical Data Format
HMD:	Head-Mounted Display
HRE:	Hybrid Reality Environment
IMM:	Immersive
IPC:	Inter-Process Communication
IRVE:	Information-Rich Virtual Environment
LIDAR:	Light Detection and Ranging
LCD:	Liquid Crystal Display
LOD:	Level of Detail
MPI:	Message Passing Interface
MVI:	Multiview Immersion
NOMVI:	Non-multiview immersive

LIST OF ABBREVIATIONS (continued)

OLED:	Organic Light-Emitting Diode
OS:	Operating System
OSG:	OpenSceneGraph
SAGE:	Scalable Adaptive Graphics Environment
TB:	TeraByte
UIC:	University of Illinois at Chicago
VR:	Virtual Reality
VRPN:	Virtual Reality Peripheral
VTK:	Visualization ToolKit
ZVTM:	Zoomable Visual Transformation Machine

SUMMARY

In the domain of large-scale visualization instruments, Hybrid Reality Environments (HREs) are a recent innovation that combines the best-in-class capabilities of immersive environments, with the best-in-class capabilities of ultra-high-resolution display walls. HREs create a seamless 2D/3D environment that supports both information-rich analysis as well as virtual-reality simulation exploration at a resolution matching human visual acuity. Collaborative research groups in HREs tend to work on a variety of tasks during a research session (sometimes in parallel), and these tasks require 2D data views, 3D views, linking between them, and the ability to bring in (or hide) data quickly as needed. Addressing these needs requires a matching software infrastructure that fully leverages the technological affordances offered by these new instruments.

In this dissertation, I detail the requirements of such infrastructure and outline the model of an operating system for Hybrid Reality Environments. I present an implementation of core components of this model, called Omegalib. I show how this implementation has been successfully used to create HRE visualizations for research, education and outreach. One key feature of Omegalib is its ability to support multiple applications running simultaneously on an HRE. This is a significant improvement compared to classic immersive systems, which normally support a single application at a time. HREs (and large displays in general) are ideal systems to support co-located collaboration, and the ability to run and control multi-application workspaces is key to effective work in this setting. However, enabling multiple *immersive* applications on a large display presents several challenges. I discuss our solution to these challenges and present an extension of Omegalib that supports fully dynamic, multi-view, multi-user immersion. I evaluate this new system, which I call Multiview Immersion (MVI) in a formal user study: two-user groups are asked to compare 3D sonar scans using MVI, single-view immersion and multiple views without immersion (simulating a standard display wall). My objective is understanding the effect of MVI on analysis effectiveness, view usage and collaboration patterns compared to alternatives. MVI appears to reduce analysis error rates for this sample task and makes it more likely for both users to remain engaged with the analysis task.

Intellectual Merit

The work outlined in this dissertation extends the state of the art in large display software infrastructures, solves several limitations in current systems (lack of immersion support in multi-window environments, single-application-only immersive environments) and provides an advanced platform for application development on Hybrid Reality Environments. Moreover, this work outlines a technique for describing co-located collaboration patterns with large displays, based on user gaze direction and communication activity.

Broader Impact

An operating system for Hybrid Reality Environments will accelerate research using these novel instruments in two ways. It will simplify and speed up the creation of domain-specific applications that make full use of HREs. And it will support co-located user groups tackling modern research, analysis or planning tasks requiring multiple heterogeneous data displays involving immersive and non-immersive views. The results of the user study presented in this work identify and describe features of co-located collaborative work within Hybrid Reality Environments that can be extended to large displays in general, and can guide the design of applications for these systems.

CHAPTER 1

BACKGROUND

Today, most scientific research involves the analysis of data of ever-increasing scale and complexity. This increase is driven by the continuing progress in the capabilities of scientific instruments, sensors and supercomputers, leading to the capture or simulation of events with a level of precision and resolution unseen in the past. However, our increased ability to capture data demands improvements in the ways we extract meaningful insights from it. The 2008 NSF Building Effective Virtual Organizations [1] and 2007 DOE Visualization and Knowledge discovery workshops [2] addressed this challenge, identifying the need for new visual approaches to data analysis. Large scale visualization instruments are among the available tools to tackle these new challenges. They are proven to enhance sensemaking for complex data [3]–[11] and are becoming increasingly essential to researchers, letting them transform raw data into discovery.

In particular, immersive systems are an attractive option for exploring 3D spatial data such as molecules, astrophysical phenomena, and geoscience datasets [12] On the other hand, display walls with their high resolution help interpret large datasets, offering both overview and resolution, and can be used to lay out a variety of correlated visual data and documents for collaborative analysis [3], [11]. These advanced visualization tools are often leveraged through the concentrated effort of interdisciplinary teams: scientists from different backgrounds whose work involves large, heterogeneous data sources.

1.1 Evolution of Large-Scale Visualization Instruments

The displays we typically use are part of a laptop, desktop or mobile device. These displays are based on LCD and more recently OLED technology and are getting both cheaper and higher-resolution. However, despite these improvements the resolution of most of these displays rarely exceeds 4 Megapixels. In a research environment, it is not unusual to tackle datasets with resolutions orders of magnitude larger than what is offered by traditional displays. These limitations become more evident when researchers need to analyze or compare multiple datasets simultaneously. A classical approach to work around these limitations is based on interactivity: the user selects or filters parts of the data in order to visualize a subset or a downsampled version of the original dataset. Panning and zooming on a map or large-scale image is an example we regularly encounter of subset visualization and downsampling. Switching between multiple views is another frequently used approach. These techniques are paired with a variety of interaction metaphors that depend on the input devices available to the user, including multitouch surfaces, gesture recognition and natural language processing [13]. Of course, well-designed interaction can increase the efficiency of accessing a large-scale datasets, but it inevitably places an additional burden on the user. Interaction is normally a conscious effort that increases the cognitive workload of a user. When dealing with simple data this effect can be minor, but it can potentially hinder complex analysis tasks [7]. Multi-monitor setups are a way to address resolution limits that are often employed in professional fields, and have been proven to increase the user's ability to process complex workloads [9][4]. The small form factor of these display setups still limits their use to an individual operator. If a larger team wants to cooperate on data analysis discussion or planning, doing so in front of a smaller display designed for a single user quickly becomes cumbersome. Team members resort to using their own separate machines, which limits collaboration, or they share a single machine output through a large room display or overhead projector, bringing back the issues of inadequate resolution in addition to limiting interaction to one user.

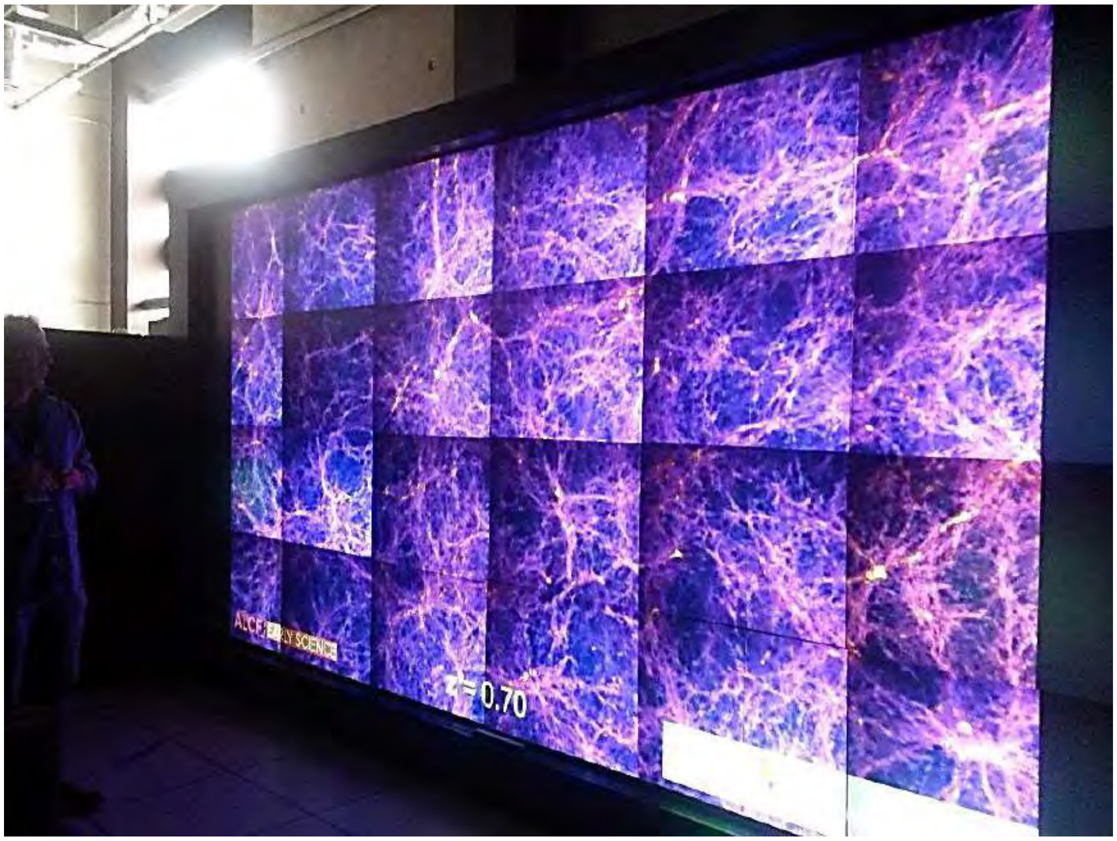


Figure 1. A rear-projected display wall at Argonne Visualization Lab

1.1.1 Display Walls

The introduction of larger digital display surfaces in scientific work predates the widespread adoption of LCD screens. At the beginning of the 2000s, multi-projector systems, called Powerwalls [14] were able to achieve resolutions of 10 to 20 Megapixels. Projectors were arranged linearly or in a grid, with overlapping areas carefully blended to approximate a seamless image spanning the entire projected area (Figure 1). The blending parameters would be readjusted regularly to account for projector shifts and changes in light or color output from each

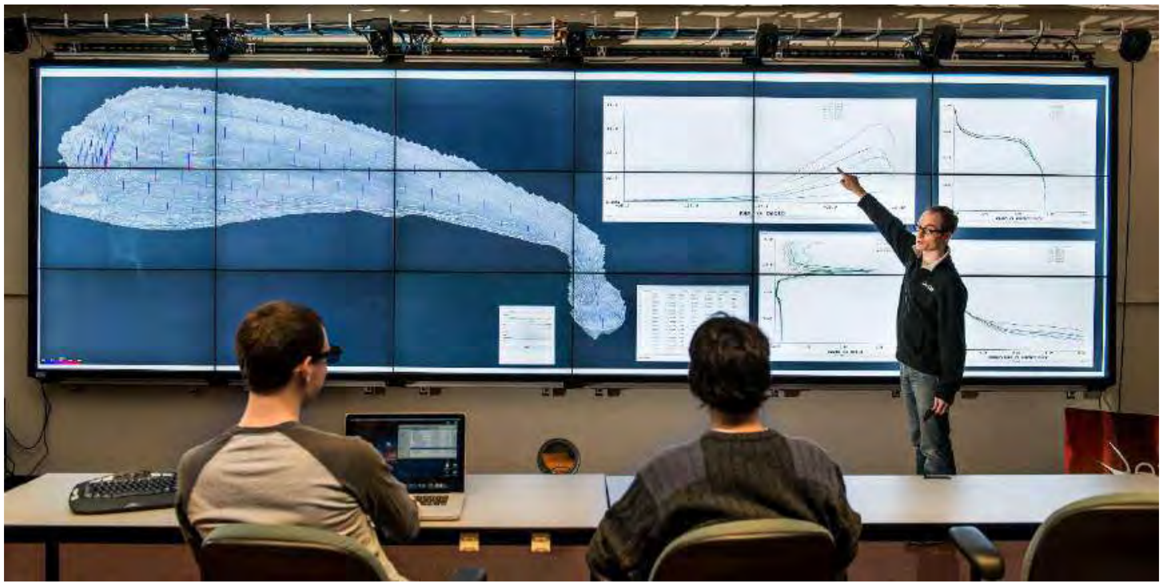


Figure 2. The cybercommons display wall at the Electronic Visualization lab, displaying multiple views from a water chemistry dataset. The Cyber-Commons wall is a 22x10-foot tiled display wall composed for 18 thin-bezel LCD displays with support for polarized, stereoscopic 3D at a total resolution of 19 Megapixels (8.5 Megapixels per eye)

projector. This calibration process was eventually automated using camera feedback and image processing techniques. Despite these advancements, multi-projector systems required constant maintenance to mitigate these issues and achieve good output quality. Since they utilized off-the-shelf projectors, they also suffered from low contrast ratio, limiting their use to dimly lit spaces. Commercial-grade projectors performed better but had a much higher initial cost and regularly needed expensive light bulb replacements.

More recently, tiled LCD displays have become the primary technology used for building large scale, continuous display walls [15] (Figure 2). Bezels, one major limitation of flat panels as display wall building blocks, have become extremely thin, to the point of being eliminated almost completely in OLED displays, so the edge blending techniques required by multi projection systems are unnecessary. Color and luminance are easily calibrated and stay consistent for a long time, reducing maintenance efforts and improving color perception for visualization. LCD displays are used in some of the largest display walls currently in use, including the highest-resolution system in the world as of 2016, the 1.5–Gigapixel Reality Deck at Stony Brook University [16], [17]. These systems allow researchers to analyze extremely large datasets at their native resolution or near to it, offering both detail and context across the entire dataset. Thanks to their diffusion it is more common to see them now integrated in the workflow of research teams.

1.1.2 Immersive environments

When computer-based Virtual Reality started to emerge in the late 80s, its most common implementation was the head-mounted display (HMD). Users would wear a helmet containing displays and special optics that rendered user-centered stereoscopic graphics. This technology

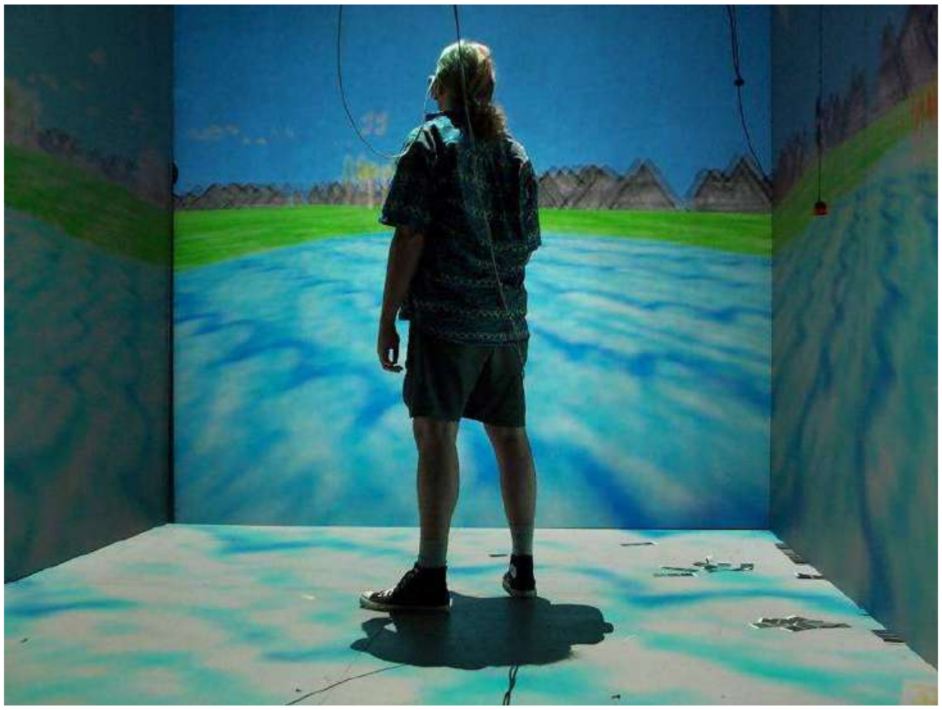


Figure 3. An image of the original CAVE system at the Electronic Visualization Lab.

produced impressive results at the time, introducing users to simulated worlds that, albeit primitive in terms of graphics, still offered a distinct sense of presence. However, besides the limits of real-time 3D rendering, a significant drawback of HMD technology was its isolating nature. Within a virtual environment, all a user could experience was the virtual environment itself. Other people, the real environment and the user's own body were excluded from it. Techniques to reproduce some of these features as artifacts of the VR scene were introduced but they could only partially replicate the real world within the virtual one. A revolutionary solution to this issue was, quite literally, to turn virtual reality technology inside-out with the creation in 1992 of the CAVE audio-visual environment [18]. The first CAVE consisted of a partial cube (three side faces initially then extended to include floor and ceiling) 10 feet at each side. A set of projectors synchronized with shutter glasses provided stereoscopic graphics, while a magnetic tracking system adjusted the projection to match the user head position and orientation. Up to this point, only head-mounted displays offered a form of VR limited to a single user. With the CAVE, a group of people could, at least partially, get a shared experience of immersion. Moreover, CAVE achieved through its multiple projectors a resolution of 4 Megapixels, an outstanding result at the time especially compared to the limited resolution of HMDs. Scientists could experience presence and embodied navigation within their dataset and they could easily manipulate features using a variety of tracked and non-tracked input devices. Researchers working with 3D spatial datasets such as astrophysical simulations, molecular structures and architectural models were some of the first to take advantage of this technology [12]. Being projector-based, CAVE suffered from the same issues we discussed with Powerwalls: poor color consistency, low brightness and contrast ratios. Alternative systems such as the C-Wall and later the GeoWall [19] simplified the design of CAVEs to limit them to a single 3D retro-projected surface using passive stereo technology. This led to a significant reduction in development cost and increased the flexibility of the system, since it could also be used as a standard projected display. However, total resolution remained limited at ~1 Megapixel.

Modern versions of the CAVE such as the StarCave [20], Allosphere [21][22] or 6-sided CORNEA [23] use better projector technology yielding resolutions up to 100 Megapixels per eye. But their enclosed nature, space required for projector setup and need of a dimly lit environment hindered their adoption in everyday scientific work limiting them to occasional use. Modern high-resolution CAVE variants are also still extremely expensive to setup and maintain.

1.2 **Hybrid Systems**

Due to their features, Immersive systems and display walls reach their maximum potential when they are used to visualize specific, and distinct, kinds of data. Display walls, with their high resolution, are excellent tools for visualizing large 2D datasets, abstract or multivariate data or for performing comparative analysis of multiple datasets. Immersive environments such as CAVEs are best used on 3D datasets and scientific visualization applications thanks to their ability to show user-centered stereoscopic 3D, naturalistic interaction and navigation., However, their applicability to information visualization, albeit tested [24][25] remains limited by their lower resolution.

What if we were to combine all these capabilities into a single, new display system offering the best features of both display walls and immersive environments in order to leverage the strength of both technologies at the same time? Of course, beyond the technical feasibility of such a display system, we first need to determine whether this hybrid display modality would be useful.

1.2.1 Benefits of Hybrid Visualization

The effectiveness of presenting data in different modalities has been the subject of previous research. 2D views have been found to be better when used to establish precise relationships between data, and for visual search ([26], [27]), while 3D is very effective for approximate 3D manipulation and navigation, especially with the use of appropriate cues, like shadows. In [28] it is suggested that combining both views leads to good or better analysis and navigation performance than using 2D or 3D alone. These findings are confirmed in [28], where in an air traffic control simulation 2D displays proved to be better for checking aircraft speed and altitudes while 3D was best used to perform collision avoidance

Bowman et al. have conducted extensive research on the advantages, challenges and best practices on hybrid 2D/3D visualization, proposing a taxonomy for Information-Rich Virtual Environments (IRVEs). In [29] the authors categorize visualizations depending on the physical and logical positioning of 2D and 3D displays, and in [24] they present a hybrid molecular visualization application for the CAVE system [18] that statically assigns 2D and 3D content to CAVE walls.

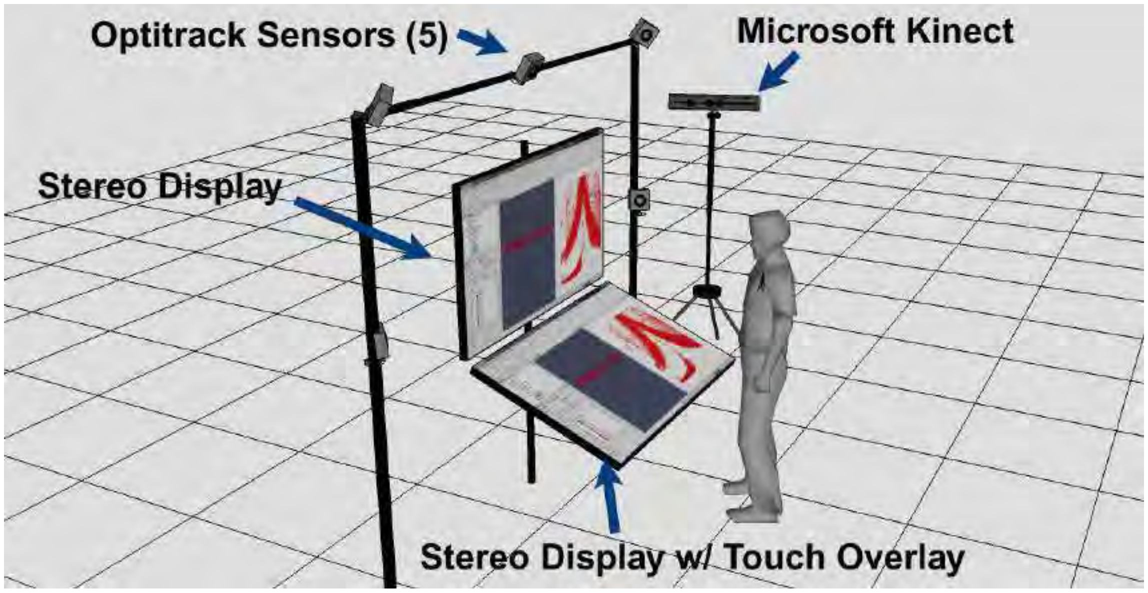


Figure 4 A concept illustration of the OmegaDesk: both displays support stereoscopic 3D. The bottom screen is touch-enabled. The user head and hands positions are detected by a set of marker-based or markerless tracking devices. The separate inputs make it possible to create hybrid interaction techniques with the system.

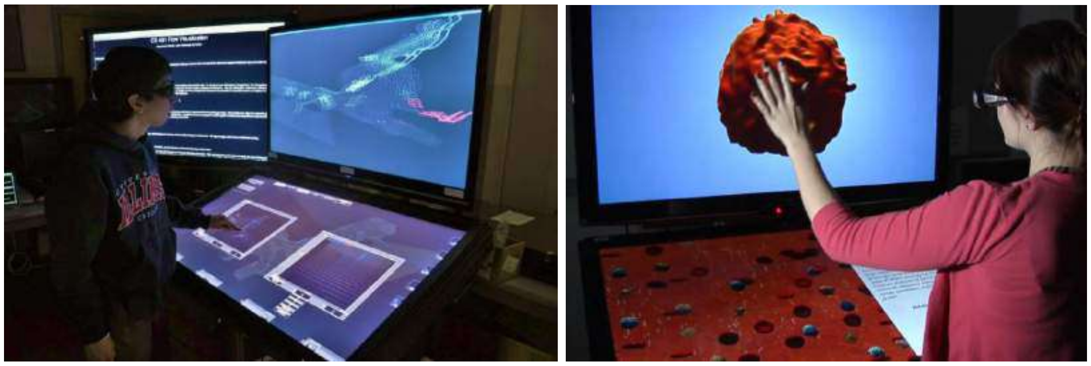


Figure 5. Two examples of OmegaDesk applications. On the left: a flow analysis tool visualizing water flow data from Corpus Christi Bay, Texas. The top screen displays a 3D view of the region and water streamlines. The bottom screen allows the user to create multiple scatterplots of the data. The user can 'Paint' regions of the scatterplots through touch, and have the corresponding streamlines displayed on the top view. On the right: an educational application illustrating the components of blood. The bottom panel is used to show a blood stream with flowing parts that the user can touch. Selecting one of the elements shows its 3D model on the top display. The user rotates the model through hand gestures.

1.2.2 <u>The OmegaDesk: A Hybrid Display Workspace</u>

In order to evaluate some of the uses of hybrid visualization, in 2010 the Electronic Visualization Lab designed and implemented a small-scale single user hybrid display system called OmegaDesk [30]. The OmegaDesk prototype is a hybrid, 2D/3D work desk that enables users to work seamlessly with 2D content (such as text documents and web browsers), as well as 3D content (such as 3D geometry and volume visualizations), and integrates multi-touch-sensitive surfaces (Figure 4). OmegaDesk consists of two high-definition stereo displays, one positioned horizontally in a 45-degree angle and another positioned vertically in front of the user. The bottom display is enhanced with a touch sensing overlay; additionally, the position of the user head and hands are tracked using either a marker-based or markerless tracking system. This combination of heterogeneous display and interaction devices made it possible to create hybrid applications that used immersive and non-immersive device features, as shown in Figure 5. Various operational modes could be envisioned on systems like the OmegaDesk depending on the nature of data to be displayed and user's needs (Table I).

Operational Mode	Potential Application Usage
Top 3D, Bottom 3D	Fully immersive mode. Ideal for applications that require navigation thru a virtual space or bringing 3D objects close-up for manipulations, etc.
Top 3D, Bottom 2D	3D Viewer mode. The vertical display is used to visualize 3D objects and worlds, while the horizontal display can be used to control aspects of the visualization.
Top 2D, Bottom 3D	'Bathtub' mode. The horizontal display is used to look at 3D data bottom-down, like looking at a fish tank from top and the vertical display is used to look at 2D projections or slices of the data.
Top 2D, Bottom 2D	Touch augmented desktop / cubicle mode. The vertical display is the wall of the cubicle while the horizontal display is like a giant iPad where document editing and manipulation can be performed.

Table I. Operational modes of the OmegaDesk

1.3 **Hybrid Reality Environments**

The technology driving tiled display walls is progressing at a fast pace, while costs per-display are significantly reducing. As we discussed, LCD and OLED displays with slim bezels are becoming common, increasing the effective area of display walls while reducing content-bezel overlap. Several manufacturers also introduced thin-bezel panels with built-in support for passive stereoscopic 3D [31]. Such recent advancements made it conceivable to merge the best-in-class capabilities of immersive Virtual Reality systems, such as CAVEs, with the best-in-class capabilities of Ultra-high-resolution Display Environments thus creating a conceptually new display instrument that blurs the line between immersive systems and display walls. I will refer to them as Hybrid Reality Environments (HREs). Hybrid Reality environments have five main characteristics:

1. A large, high-resolution display that approaches the sphere of influence and perception of a human [32], while providing a pixel density that comes close to matching the visual acuity of the human visual system.

2. Support for stereoscopic rendering for 3D datasets. The benefit of stereoscopy is not limited to 3D datasets however. 2D representations can greatly benefit from stereoscopy, providing an extra perceptual channel to encode additional data (such as time).

3. Support for naturalistic interactions: to offer a truly hybrid 2D/3D visualization platform, such environments should support a wide variety of interactions, including keyboard/mouse, 6 degrees-of-freedom joysticks, head tracking for a viewer-centered perspective, and voice-activated interfaces. The appropriate configuration is, of course, application dependent.

4. A space that encourages and facilitates collaboration for a co-located scientific team. The ability to solve complex research problems is quite often a group endeavor. A Hybrid Reality environment should offer a workspace analogous to a ‘digital war room’ [33], where multiple surfaces (including tables and walls) become active, functional tools for the research group’s work. The space should simplify the integration of digital and analog work artifacts, for instance allowing users to bring their own laptops, paper documents etc. and easily integrating them into the collaboration. This is more of an architectural / design objective than a technical one, but I still consider this essential to the success of HREs. A Hi-Tech work area that doesn’t provide an inviting space where scientists can comfortably sit together, analyze, and interpret data won’t be used often once its novelty wears off.

These characteristics synergize the capabilities of virtual reality and high-resolution tiled-wall displays, giving rise to a qualitatively distinct display environment that combines the best of two worlds. The fusion of these two formerly separate modalities results in a hybrid environment that is capable of rendering large volumes of data, while catering to the distinct visualization and interaction requirements of different classes of data including 2D, 3D, multidimensional, and temporal datasets.

1.3.1 The CAVE2 Hybrid Reality Environment

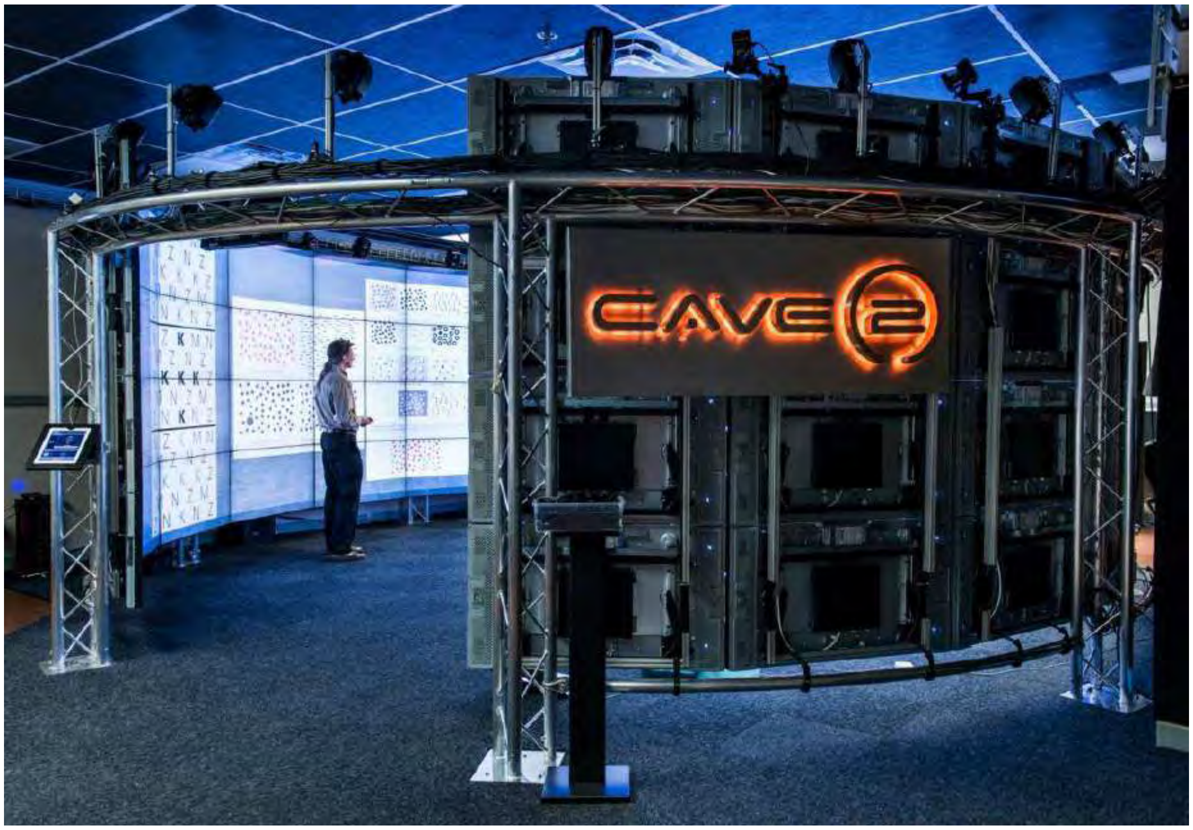


Figure 6. A picture of the implemented CAVE2 Hybrid Reality Environment. Compared to the projection-based original CAVE, CAVE2 uses passive stereoscopic LCDs offering increased resolution, easier color calibration

The CAVE2 system at the Electronic Visualization Lab was the first implementation of a full-scale Hybrid Reality Environment. Unveiled in 2012, the CAVE2 system is based on a cylindrical setup composed by 18 columns of 4 displays each. This arrangement provides a panoramic view of 320 degrees. Each 4-display column is driven by a separate computer in an 18-machine cluster, plus a head node. The 72 near-seamless passive displays have a resolution of 1366x768 pixels each and provide an aggregate resolution of 36 Megapixels per eye (74 MP total), almost 10 times the 3D resolution of the original CAVE. To an observer at the center of the system, this design provides a horizontal visual acuity of 20/20. CAVE2 uses an optical motion tracking system consisting of ten Vicon Bonita infrared cameras arranged in a circular configuration above the displays. Retroreflective markers are used to track the position and orientation of the primary viewer's head and the navigation controllers. As the main interaction device, multiple tracked wands can be used within CAVE2. The wands are built using off-the-shelf game controllers. The wands provide 8 digital and 3 analog input channels, plus 6-DOF tracking.

The success of the EVL CAVE2 led to its commercialization [34], and several installations, following the original design with minor variations, have been built since 2012. Hybrid Reality Environments with different geometries such as the planar Cyber-CANOE [35] and curved CALIT2 WAVE [36] have also been developed.

1.3.2 Benefits of Hybrid Reality Environments

The power of Hybrid Reality environments stems from their ability to support distinct visualization and interaction styles while simultaneously catering to different classes of data. This gives them an edge in modern scientific inquiries where several heterogeneous data sources must be collectively visualized and analyzed.

Modern scientific projects require the analysis of several datasets. These datasets often depict different phenomena and comprise a wide range of variables. Scientists not only want to look at one isolated variable in one dataset at a time; they often need to do integrative analysis where elements from heterogeneous data sources are linked and analyzed collectively in order to uncover hidden relationships and generate insights.

As discussed, a defining feature of HREs is offering a space that encourages collaboration. The benefits of co-located collaborative work have been analyzed in a variety of contexts like design [33], software development [37], and engineering [38][39]. Advantages of co-location include reduced communication friction (users more than 30 meters away are effectively remote [40]) and distributed cognition: teammates exploit features of the social and physical world as resources for accomplishing a task [41].

Effective visualization in large-scale environments is an area of ongoing investigation, and its requirements have been explored in the past. In terms of data size and format, some researchers need to analyze datasets whose size surpasses the resolution of even a large high-resolution display [42]. Others find a large display ideal for comparative visualization, displaying alternate versions of similar data like brain scans [43] or building models [39]. Lastly, large display surfaces can help aggregate heterogeneous data about a specific phenomenon [44]. This third scenario (heterogeneous multiple views) is the most common [45]–[47]. In terms of effectiveness, large high-resolution displays appear to suffer from diminishing returns once past a certain size and resolution, if their use is limited to single-user applications [48][49]. These observations underscore how one of the best uses for large high-resolution displays is as collaborative tools, dedicated to visualizing heterogeneous data or multiple views of the same dataset.

In [45], the authors note how researchers leverage large display surfaces to efficiently organize team work. The displays facilitate this by offering a physically suitable space to lay out large amounts of information. Part of the work is spent with each user individually working on a separate part of the display. Another part is dedicated to collaborative data consolidation and analysis, so everybody in the group can aggregate and discuss partial findings. During this kind of work, researchers often need to bring in new data and documents from their laptops or from the web: thus, a single pre-packaged application running on a large display rarely covers the full needs of a research group. Important factors for an efficient co-located collaborative space are, among others, the physical design of the space (space should support both the work of the entire team, and separate sub-teams), and its reconfigurability (both physical and at the software level).

1.4 **Towards and Operating System for Hybrid Reality Environments**

Naturally, hardware is only half of the equation. To fully take advantage of these novel technological affordances we need a matching software infrastructure that facilitates scientific work within HREs. This infrastructure should be tailored to the real-world needs of research teams, taking into account the way users interact with the instrument, with their applications and with co-located or remote members of their team.

So far, software development for display walls and immersive systems has followed independent paths. On display walls, the focus has been on supporting scalable ultra-high resolution visualization, enabling and managing multiple views of heterogeneous datasets and co-located collaboration. On immersive environments, the effort is to provide low latency viewer centered stereo, naturalistic interaction and remote collaboration.

What is now needed is a convergence in software design for display walls and immersive environments: The integration of display wall software, an immersive environment framework and additional components into an “operating system” for Hybrid Reality Environments to promote the creation of hybrid reality visualizations that make the full use of the capabilities of HREs. This operating system aims to solve two major challenges with alternative approaches.

1.4.1 Dynamic View Allocation

*A*lthough an HRE is capable of displaying 2D and 3D content at the same time, existing display software relies on static configurations describing how the physical display space should be split into 2D and 3D views. Multiple predefined configurations give the end users some flexibility, but require restarting and resetting all running applications.

Application users need access to a system that lets them easily and seamlessly manage display space and content. We can for instance consider a structural engineering task: Engineers want to compare different designs of a building, load a digital model, and explore it in full-scale in the Hybrid Reality environment. They then decide to compare two variants of the design side-by-side, splitting the available screen space in two. Different users from the team explore the structures separately, while discussing their design. The group then chooses to look at pictures of the target site: they hide one of the 3D visualizations, and use the now available screen space to share photographs. The team splits again, with one group discussing the site while another user navigates the building model. The user notices a flaw in the design and interrupts the rest of the team. He uses a virtual slicing plane to generate a section of the building that gets displayed on a separate 2D view. The team now brings up the alternate design again, observes the two 3D visualizations and a set of 2D sections, until they agree on one of the design variants. Before closing the meeting, they display this variant on the full display again, and mark a few points for future revision.

We can observe this kind of work pattern in research groups in a variety of disciplines. Co-located collaboration often entails multiple phases that alternate full group work with individuals or sub-groups working in parallel: moreover the work may focus on a single issue, on different views into that single issue, or on independent issues [37], [38], [46]: the ability to easily re-configure the display space is fundamental to support such heterogeneous tasks [33].

1.4.2 Unified Interaction

Another issue we must address is consistency of input and interaction modalities. While it's possible to run classic display wall software and immersive software together in an HRE system, the 2D and 3D portions of the display often rely on inconsistent interaction schemes: for instance a pointing device may use absolute movement on the 3D portion of the display and relative movement on the 2D part. Even worse, different physical interaction devices may be required to control the 2D and 3D view. In cases when a single device is used, interaction may lead to conflicting results (i.e. navigating a 3D view moves 2D views).

An HRE operating systems also needs to provide an easy, yet powerful application programming interface (API) that developers can leverage to implement custom software. Immersive applications are often created to serve interdisciplinary research teams whose members may have limited programming experience. Applications also have a great variance in their complexity: some only need basic visualization and interaction with 3D models. Others require the implementation of complex and custom visualization techniques, need low-level access to the graphics API or need to use specialized visualization libraries. A way to satisfy these requirements is to provide a layered API, with multiple levels of access that offer a tradeoff between complexity and control.

1.5 **Multiview Immersion**

It is of course possible to use an HRE like a classic immersive system, or use it as a classic display wall, making use of the wide variety of frameworks currently available to support these systems. However, if we limit ourselves to leveraging existing tools, we lose the potential advantages HREs have by bringing these two modalities together. Joining those means enriching immersion with the support for multiple co-located views, applications and users. I envision a system that allows users to use screen space afforded by a large display in an optimal way, as determined by the visualization they are using and by their preferred collaboration pattern. I want to achieve this objective without sacrificing immersion: views should be freely resizable and movable without leading to breaks in immersion. I also want to let users choose who is in the 'driver's seat' for each view, and make negotiation of the active user as simple as possible. I will refer to these features as *multiview immersion (MVI).*

In order to fulfill these objectives, we need to address several technical challenges. How do we move and resize an immersive viewport on non-planar display geometries such as the one of CAVE2? How does navigation and head tracking work when an application viewport changes at runtime? Given the importance of framerate for immersive applications, how do we optimize hardware usage for multiple dynamic viewports (particularly when running on a cluster).

Finally, we need to define some metrics for success. We need to demonstrate multiview immersion is significantly better than alternatives (classic immersion and classic multiview 2D) for one or more realistic usage scenarios. Given a real or synthetic co-located visual analysis task, the purpose of multiview immersion (and arguably hybrid environments) would be less clear if it does not lead to any meaningful change in performance. Furthermore, we want to understand the effect of multiview immersion on the collaboration behavior of users. Are researchers more likely to work in parallel or engage in discussions when they can control their own immersive views?

1.6 **Outline**

The next chapter will present an overview of existing software frameworks for Display Walls and Immersive Environments, focusing on the features that would make them candidates for a Hybrid Reality Environment operating system. I will show how no existing solutions achieves all the objectives I set forth in section 1.4.

Chapter 3 describes my implementation of an operating system for hybrid reality environments, which I call Omegalib. The implementation was motivated by the need to support the OmegaDesk and CAVE2 systems at the Electronic Visualization lab, but has been since used in multiple systems and applications beyond EVL. I will discuss some of these use cases at the end of the chapter.

Chapter 4 introduces both the design and implementation of Multiview Immersion. The implementation details will focus on how this technique was integrated in Omegalib framework. but I will underscore how MVI is not bound to any specific framework. While Omegalib, by design, provides several functionalities that make the implementation of multiview immersion easier, they could be reproduces in other environments as needed. I conclude this chapter with an evaluation of system performance for multiple immersive application running on a cluster-display system, to validate my implementation.

In chapter 5 I will describe a user study plan to evaluate the effectiveness of multiview immersion compared to other techniques commonly used in large-scale displays (namely, multiple 2D views and a single immersive view). I will detail the research questions and hypotheses that motivate the study and present the tools and methods used to answer them.

Chapter 6 will discuss the results of our user study, analyzing both the quantitative and qualitative data collected during the study sessions. I will use these results to address the questions identified in chapter 6, and I will use the collected data to discuss some additional (and in a few cases surprising) findings about multiview immersion.

Chapter 7 will conclude this dissertation summarizing my work and results, and outlining a few future venues of research and applicability of mulitivew immersion.

CHAPTER 2

RELATED WORK

Existing software frameworks for large displays and immersive systems can be classified in a variety of ways. In [50], Chung et al. point out how a single taxonomy is insufficient to fully describe frameworks for cluster-based large display functionalities. They proposed a multi-faceted taxonomy that organized frameworks based on three dimensions: task distribution, input handling and programming model. In this review of prior work, I use a similar taxonomy, enriched with two additional dimensions: multiview support and immersion. The added dimensions allow us to better characterize different approaches of software frameworks on display walls and immersive environments.

The following review of current software frameworks organizes the existing approaches into several classes, based on a few defining characteristics shared by multiple solutions. Each class can be described by one or more dimensions of the taxonomy presented in Table 2: for instance, all the frameworks classified as Transparent Primitive Distribution Frameworks have the same task distribution model (Distributed Render) and the same programming model (No API). In general, frameworks in the same class can be seen as different solutions to the same problem (porting legacy applications to a large display, improving cluster render performance, supporting immersion, etc.). Although the classification presented in this work is not the only possible one, it helps in clustering similar approaches together and providing a better description of the state of the art.

Dimension	Taxonomy	Description
Task Distribution Model	Distributed application	An instance of the application runs on each cluster node. Head node takes care of synchronization and message passing. Reduces network usage, but duplicates part of the workload on all nodes.
	Distributed Renderer	Application runs on the head node. The head node distributes drawing primitives to the cluster nodes, which act as lightweight renderers. Requires more bandwidth but consumes less processing resources on the rendering nodes.
Input Handling	No Input Handling	Input handling is completely left to the application developer.
	2D Event Handling	The framework provides functionality for handling 2D-type input (mouse pointers, touch, etc.). The input is typically used to handle 2D content or interact with a classic 2D graphical user interface.
	3D Event Handling	The framework provides functionality for handling 3D-type input (tracked devices with six degrees of freedom, wands, 3D gesture interfaces etc.). The input is typically used to control manipulation and navigation in 3D environments, but can also be re-mapped to 2D input on virtual of physical surfaces.
	Hybrid / Extensible Event Handling	The framework supports both 2D and 3D input capability, usually through an abstracted input event syntax. The event handling machinery is extensible to new event types and to new logical or physical input devices.
Programming Model	Transparent / No API	The framework is designed as a transparent layer that handles application execution and / or rendering on the display cluster. It simplifies porting of legacy applications, usually at the cost of efficiency and / or flexibility.
	Low-level API	A minimal API is provided, usually consisting of a set of simple callback-style interfaces that developers use to implement application logic. And low-level primitive-based drawing. Makes it moderately easy to port legacy applications and gives developer some control over the environment, but provides no utility APIs (rendering, scene, input handing, etc.)
	Medium-level API	The framework provides one or more APIs that assist application development using higher-level concepts like a scene graph, user interface widgets, etc. Legacy applications cannot easily be ported.
	High-level API	In addition to the medium-level API, the framework provides access to higher level functionality like a scripting engine, visual scene editor, pre-made application modules and plugins, etc. Frameworks with high-level APIs may force a specific application model and sacrifice some flexibility in exchange for ease of creating new applications.
Multiview Support	No Multiview	Only one application at a time can run and control the entire display surface. Switching applications or display configurations typically requires restarting the framework runtime.
	Static Multiview	Multiple views (from the same or multiple applications) can share the display, but their viewports are fixed at configuration or startup-time. Viewports cannot be moved or resized.
	Dynamic Multiview	Multiple views can share the display and can be moved / resized freely like classic windows on a desktop.

Dimension	Taxonomy	Description
Immersion Support	No Immersion	The framework is only capable of displaying 2D views, or non-immersive 3D views (similar to 3D graphics on a standard display)
	Basic Immersion	The framework supports the display of stereoscopic 3D views.
	Advanced Immersion	The framework can display user-centered stereoscopic 3D views and supports advanced immersive capabilities like head tracking, tracked navigation/manipulation, positional audio, etc.

Table II. The multi-dimensional taxonomy used to describe the existing large display / immersive software frameworks.

2.1 **Transparent Primitive Distribution Frameworks**

Transparent Primitive Distribution Frameworks are designed mainly to port legacy applications to large displays, have a fully transparent API (matching the underlying graphics system API, like OpenGL) and use graphic primitives as the unit of distribution.

2.1.1 Chromium

Based on the earlier WireGL library [51], Chromium is a widely used transparent OpenGL framework for large display applications [52]. Chromium works by intercepting and replacing OpenGL library calls, so OpenGL applications can be executed through the Chromium runtime without modifications. When an OpenGL application runs on a cluster head node, Chromium intercepts OpenGL calls using Stream Processing Units (SPUs). SPUs are used to manipulate the stream of OpenGL primitives. On a tiled display configuration, the *tilesort* SPU, performs a sort-first partition of the geometry and passes the OpenGL command stream to the display nodes. Another Chromium component running on display nodes (crServer) receives the command stream and performs the actual rendering. A configuration server running on the head node, called the mothership, manages information about the configuration of the display and controls execution.

Chromium is a practical and simple tool for transparently running legacy OpenGL applications on a large display, but introduces significant overheads that make it unsuitable for dynamic, large data visualization. Intercepting rendering commands in the head node is an expensive operation, especially when OpenGL is used in immediate mode and primitives are forwarded on a vertex-by-vertex basis. Given its reliance on distributing graphics commands for each frame, Chromium's performance is often bound to network bandwidth availability. Additionally, since Chromium must intercept OpenGL function calls, it only work for applications using functions wrapped by the Chromium/OpenGL API. Since the OpenGL API is extensive and constantly evolving, several features typical of modern OpenGL implementations, such as Vertex Buffer Objects, Occlusion Queries and Programmable Shaders are not supported by Chromium.

2.1.2 ClusterGL

The ClusterGL middleware [53] uses a transparent wrapping approach similar to Chromium's, but uses several optimizations to reduce network usage such as multicast, frame differencing and compression. Frame differencing and compression make it possible to use intra-frame coherency to reduce data transfers by creating delta representations of the changes between frames. The tradeoff of this approach is the increased load on the head node due to computation of deltas. Benchmarks by ClusterGL authors show that, for most applications, ClusterGL outperforms Chromium applications. The performance difference increased with more complex scene geometries and more display nodes.

2.2 **Distributed Compositing Managers**

Distributed Compositing Managers make it possible to use a cluster-based display as a large seamless "desktop environment", where multiple applications can run in separate windows whose layout is controlled by the user. These frameworks usually expose a minimal API and use pixel streams as the unit of distribution. They also offer some form of 2D input support, to make window management possible.

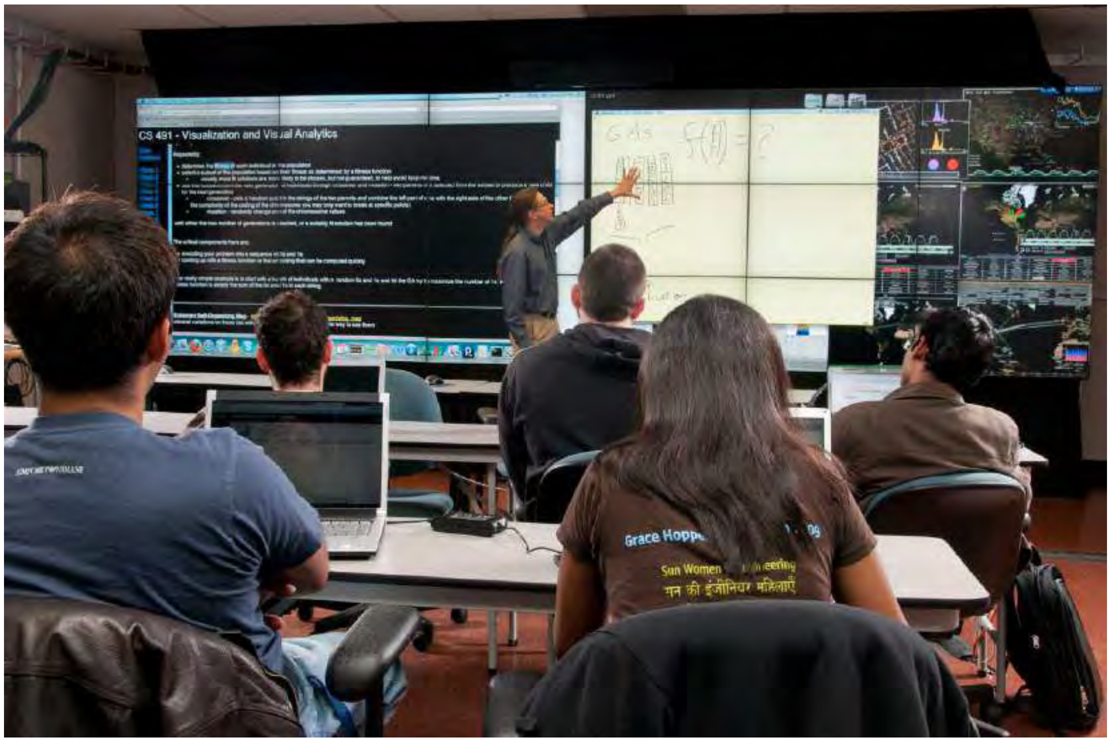


Figure 7. SAGE used for teaching on the EVL Cybercommons wall.

2.2.1 <u>SAGE</u>

The Scalable Adaptive Graphics Environment (SAGE) is a widely used middleware for driving multiview large scale displays. [54]. Prior to SAGE, software to drive tiled displays operated like DOS in the 1980s, running only a single application at a time, and occupying the entire wall. SAGE turns a tiled display wall into a multitasking environment that enables multiple users to maximally take advantage of the vast resolution and screen size and operate multiple applications simultaneously. It permits them to juxtapose multiple visualization and data products side by side for analysis. The SAGE implementation has three main components: the Free Space Manager, SAGE Receivers and the SAGE Application Interface Library (SAIL). The SAGE Receivers are the individual clients running on each display node and receiving pixel streams. The Free Space Manager is the distributed window manager that handles output viewports and routes pixel streams to the appropriate clients. Applications use SAIL to generate an additional frame buffer. Once drawing for a frame is done, the application is in charge of copying the frame data to the SAIL frame buffer. Pixels in a SAIL frame buffer are divided in blocks and streamed to the SAGE receivers on the nodes corresponding to the output viewport of the application on the tiled display. A single application could also be running on a distributed system, with separate instances generating separate parts of the frame buffer. The Free Space Manager takes care of routing and organizing the pixel streams accordingly, to generate a final consistent image on the display nodes. The SAGE distribution includes several standard applications like picture and video viewers, remove desktop streaming tools and mouse pointer sharing applications. SAGE support 2D input from multiple devices and users. For instance, it is possible to have two users controlling window placement on a display simultaneously, with one user interacting with windows directly through a touch overlay and another controlling them using the mouse pointer on his or her laptop. While SAGE integration requires applications to be modified to use the SAIL interface, several tools provided with the SAGE distribution make the environment easier to use. For instance, OpenGL applications can run on a SAGE system with minimal modifications, thanks to a wrapper layer that translates OpenGL buffer swaps into the appropriate SAIL commands.

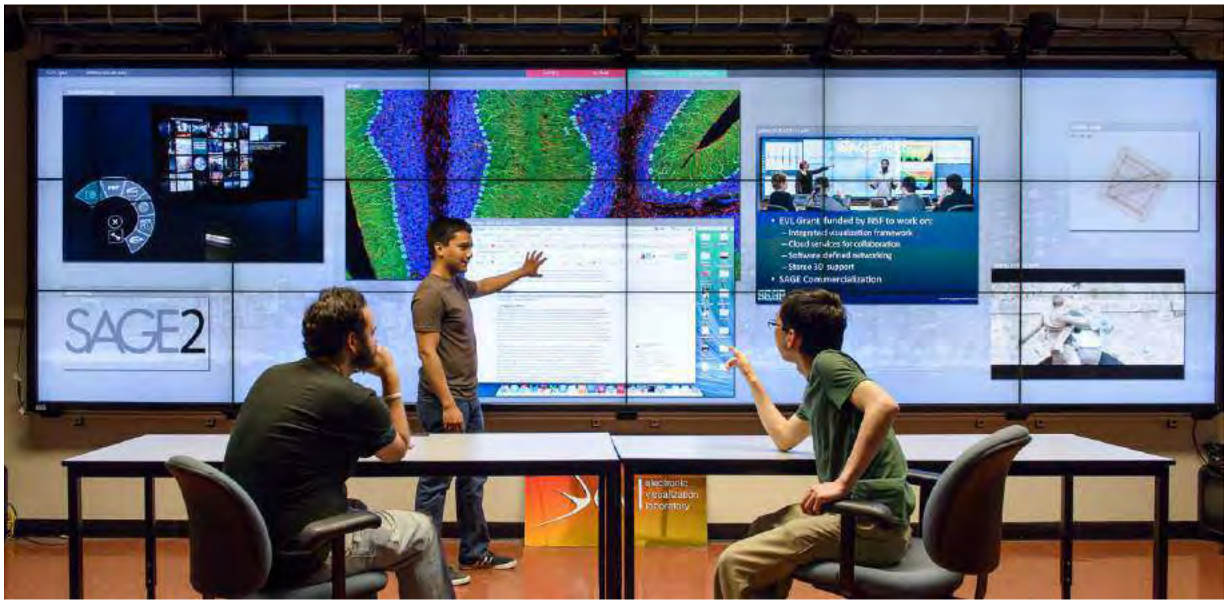


Figure 8. An example of a SAGE2 session running multiple applications including an image viewer, video player, remote browser and 3D model viewer. All applications are developed using Javascript, using standard web technologies.

2.2.2 SAGE2

The next version of SAGE maintains several of the successful features of its predecessor, but its implementation is radically different. The increased power and flexibility of web technologies made it possible to create a distributed tiled-display window manager running entirely inside a browser [55]. A central manager similar to the SAGE FSManager handles synchronization across browser instances running on separate machines, allowing web-based applications to run across an entire display wall (Figure 8). Applications are developed using Javascript, making it much easier to deploy and run them on a wide variety of systems. SAGE2 also supports advanced remote collaboration features, including synchronized application between tiled displays of different size, laptops tablets and any other device capable of running a browser [56][57].

2.2.3 DisplayCluster

DisplayCluster is an interactive visualization environment for cluster-driven tiled displays [44]. It provides a desktop-like windowing system similar to SAGE. In addition to managing window content based on pixel streams, DisplayCluster provides a built-in media viewer optimized for the display of ultra-high resolution images with dynamic zooming. Similar to SAGE, DisplayCluster provides a python-based scripting interface that can be used to program and control interaction.

2.2.4 Commercial Solutions

Research around tiled-display software is reaching a mature point. Together with the wider diffusion of large high-resolution displays, this favored the emergence of multiple commercial solutions for managing content on such platforms. Systems like Oblong Industries Mezzanine [58], Bluescape [59] and Multitaction Cornerstone [60] offer a polished content and collaboration platform for large displays. While they offer better technical support, security and reliability, these solutions do not differ significantly from their research counterparts in terms of features.

2.3 **Parallel Rendering Frameworks**

Parallel Rendering Frameworks can be used to create applications that make efficient use of a cluster rendering and computation resources. They tend to expose a low-level generic API that gives users a lot of control and flexibility on implementation. They may use a Distributed Application distribution model but they are typically flexible enough to let the developer control the details of distribution and parallelization. Parallel Rendering Frameworks can be used to create 2D or 3D applications.

2.3.1 Equalizer

Equalizer is a framework for scalable parallel OpenGL rendering [61]. It provides an API to create OpenGL multi-pipe applications for cluster-based large displays and immersive systems.

Equalizer provides a runtime environment that applications can use to distribute their execution. The runtime consists of a messaging layer, a basic display management layer capable of creating graphic viewports and managing contexts, and a multi-threaded set of callbacks that developers use to implement application logic and rendering across cluster nodes. While applications running on Chromium are still fundamentally running on a single node (up to the point of primitive generation), the graphical part of an Equalizer application is distributed across all the nodes that are in charge of rendering. These can be nodes plugged to an output display or dedicated machines in a rendering cluster. Therefore, Equalizer application can reduce computation steps and workload in the head node and instead of transmitting low-level graphics commands, the head node can send higher level of graphics calls to reduce network traffic.

A dedicated configuration server manages the utilization and load balancing of the display cluster resources. An extended configuration syntax makes it possible to define pixel-generating channels (the application rendering nodes) and output channels (the display nodes). Channels can be linked by compound trees that specify how frames should be generated, and can be used to define various task decomposition techniques for parallel rendering (sort-first, sort-last, tile load balancing, pixel compounds). To add Equalizer capabilities to an application, developers use an API akin to GLUT but finer-grained. The API consists of sub-classes that present abstraction of windows, displays, machines and single GPU units. Equalizer also offers several additional libraries that simplify the implementation of features often present in distributed graphical applications, such as a serialization library, multi-threading primitives and logging. Since it leverages the OpenGL API on the host system, Equalizer supports all modern OpenGL features.

2.3.2 CGLX

The Cross-Platform Cluster Graphics Library (CGLX) provides a minimally invasive OpenGL programming model aimed at supporting high-performance rendering on a display cluster [62]. Nodes in the display cluster maintain each an independent OpenGL context, and use a custom communication layer provided by CGLX to exchange synchronization messages. CGLX exposes most of the standard OpenGL and GLUT APIs, and works by intercepting and re-implementing some draw callbacks and view-related OpenGL functions.

Nodes in a display cluster using CGLX can be connected to multiple displays. CGLX uses multiple thread to make efficient use of the CPU and GPU resources of each node. CGLX lets developers control individual rendering thread, giving users fine-grained control over the visualization. A tool included in the CGLX distribution provides a graphical interface to simplify display configuration (a process typically done through manual editing of configuration files). With this tool, developers can configure the display system by adjusting various display parameters including size of bezels, resolution of displays, and arrangement of the display array. The tool also allows to preview an application switching between different configurations. The framework can handle input event streams from multiple input servers and handles a variety of 2D input devices An extension of the framework called CGLXTouch adds support for multitouch displays such as tabletops and hand-held displays [63].

2.3.3 MPIglut

Similar to CGLX, MPIglut supports running applications on cluster based large displays providing an OpenGL/GLUT-compatible API. Unlike CGLX, which uses its own communication layer, MPIglut uses the standard Message Passing Interface (MPI) library for network communication [64]. While fully transparent approaches like Chromium work by intercepting function calls at runtime, with MPIglut users need to recompile OpenGL code and explicitly link it against the MPIglut library.

2.3.4 FlowVR Render

While Chromium distributed raw primitives to rendering nodes, FlowVR [65] uses a sort-first parallel rendering algorithm: FlowVR distributes resources in a higher level format based on assets (vertex buffers, textures, shaders, shader properties) and simple draw commands using those assets. This distribution model reduces bandwidth requirements, simplifies the sort-first step (which works on vertex buffer bounds instead of filtering single OpenGL commands) and exploits the GPU's shader capabilities. A FlowVR application is made of two separate components: a Viewer and a Renderer. The Viewer component run the application logic, creates assets and issues draw commands to be distributed to Renderers. The Renderer component receives assets and commands from Viewers and is in charge of rendering content destined to a specific tile. Developers can select the appropriate rendering strategy based on the kind of application they need to support. Strategies change based on how the Renderer and Viewer modules are distributed across nodes in the rendering cluster.

2.4 **Distributed Scene Graphs**

Distributed Scene Graphs are a "specialized" subset of Parallel Rendering Frameworks that specifically target 3D application development. Distributed Scene Graph frameworks expose a medium or high-level API and may include support for immersion or 3D input.

2.4.1 OpenSG

The OpenSG framework implements a scene graph interface that explicitly supports multithreading and multi-node distribution [66], [67]. Each display node maintains a copy of the scene graph in a serializable data format based on *fields* and *field containers.* Any change in the scene graph (node transformations, visual property changes, node additions / deletions) can be represented as one or more field changes in a set of containers. Each display node runs a process which synchronizes the scene graph based on the change set forwarded from the head node. User applications can also synchronize custom data by implementing additional classes that use the field container specification. Due to the overhead in synchronization and update of the distributed scene graph, very dynamic visualizations may not run efficiently with OpenSG. To mitigate this drawback and reduce network bandwidth, change sets can be compressed before transmission and forwarded using multicast. Both sort-first and sort-last parallel rendering algorithms are supported. OpenSG is built on top of the OpenGL API: applications can access the underlying OpenGL implementation directly, offering finer control over rendering when required.

2.4.2 Garuda

Garuda is built on top of the OpenSceneGraph (OSG) toolkit [68], a widespread scene graph rendering and utility library [69]. Similar to what Chromium does for OpenGL, Garuda enables legacy OSG applications to run on cluster-based display without the need to recompile or alter their code. Garuda takes care of detecting and synchronizing changes in the scene graph using multicast communication to the display nodes. The framework replaces the standard OpenSceneGraph cull draw and swap operations with specialized versions that optimize tile-based frustum culling [70].

2.5 Wall Application Frameworks

Wall Application Frameworks are designed to simplify the creation of 2D applications for large, cluster-driven displays. They may offer some multiview capability like Distributed Compositing

Managers, but sacrifice some of their flexibility to provide a higher-level (and possibly domain-specific) API.

2.5.1 jBricks

jBricks is a Java framework for 2D drawing on large scale cluster driven display walls [71]. jBricks supports interaction with heterogeneous input devices and integrates the 2D graphics rendering API of ZVTM [72], a powerful toolkit for developing information visualization applications. jBricks requires small changes to a single-display ZVTM program, mostly related to initialization and callback management. Several standard 2D graphic primitives supported by Java (bitmaps, text, widgets) are available in jBricks, embedded in objects called *glyphs.* Glyphs are distributed through JGroups multicast communication, and visualized on viewports that span one or more tiles in a display wall. Viewports can also contain other viewports to increase the flexibility of visualization layouts. Each display node performs 2D culling on the viewport area within its tile or tiles. jBricks also provides a specialized input server called jBIS, which supports a variety of 2D input devices such as tablets and game controllers, and can connect to other input data sources using the TUIO protocol [73].

2.5.2 Shared Substance

Shared Substance is a middleware for developing flexible interactive multi-surface applications [74]. One of the objectives of shared substance is to make it possible to create dynamically reconfigurable environments where software and interaction is distributed across heterogeneous devices (display walls, smartphones, multitouch tables), and where hardware and software features can be added and removed at runtime. Shared substance utilizes a tree representation of applications, display and input resources. Nodes represent data, while the connections between nodes represent message exchange paths between different components in the environment. Nodes support the connection to *facets*, which represent the programmable entry-points of the application setup. Facets can be used to run scripts, launch applications, setup data sharing points and so on. Adding, removing and modifying nodes or facets in the three reconfigures the application or its execution environment at runtime. Multiple shared substance processes running on different machines, can be linked together through a discovery process and can share their node/facet trees through a process akin to mounting a drive or filesystem in Linux. Mounted trees can be replicated locally, or can exist as proxies that exchange messages with the remote tree location.

The authors of shared substance show how the proposed model can be used to implement both a distributed scene graph and synchronize multiple instances of a non-distributed application. The Shared Substance middleware is created in python. It should be underscored that although Shared Substance uses a high level language and programming concepts, it does not expose a high-level API in the sense discussed in this chapter. Shared substance provides an environment to launch applications, coordinate their execution and share data/input, but developers are in charge of choosing the appropriate tools or libraries to develop the applications themselves.

2.5.3 MediaCommons

MediaCommons is a middleware designed to simplify the creation of complex visualization applications for tiled display walls [75]. MediaCommons makes it possible to display and arrange 2D content on a display wall, but does so without using the pixel streaming approach of SAGE. Rendering happens directly at the visualization nodes, with replicated *renderable data objects* that synchronize their content and are aware of the boundaries of the local display. MediaCommons' approach is similar to the one use in jBricks and partially DisplayCluster, but offers the additional capability of displaying 3D data, since the middleware is based on OpenSceneGraph. A limitation of this approach compared to distributed compositors like SAGE, is that rendering for all viewports happens within a single process: a resource-intensive view with a slow framerate would slow down the framerate of the entire wall display.

2.6 **Immersive Application Frameworks**

Immersive Application Frameworks are aimed at creating 3D applications for immersive environments. These frameworks support advanced immersion features (as discussed in table 2) and are designed to handle 3D input. Immersive Application Frameworks expose heterogeneous APIs. Some have low-level APIs made of simple callbacks, designed to be as generic as possible. Others implement full scene graph APIs or ad-hoc visualization pipelines to allow for the rapid development of immersive applications in specific areas.

2.6.1 CAVELib

CAVELib was originally developed as a dedicated API for the original CAVE system [76], [77]. The initial implementation of CAVELib was targeted at shared-memory system, but it has been extended to support fully distributed display clusters. At startup, an application is replicated and runs a separate instance on each node. As in other distributed application frameworks, application logic and rendering tasks are split and run in separate threads. CAVELib supports immersive interactive 3D application features like head tracking and off-axis projection. More advanced 3D functionality like navigation and object manipulation is left for application developers to implement. Although CAVELib only provides an OpenGL API for drawing; it supports integration with higher-level libraries such as Performer and OpenSceneGraph.

2.6.2 FreeVR

FreeVR is an open source library for the creation virtual reality applications [78][79]. Like other VR frameworks, FreeVR abstracts interaction and display systems, supporting several common hardware interfaces out of the box. This allows FreeVR to be used with existing virtual reality systems, while making it possible to expand support to future environments. The FreeVR API is also designed to be compatible with CAVELib, simplifying the translation of legacy VR applications to this platform. Compared to some alternatives discussed in this section, FreeVR is a lower-level solution: It only provides basic hardware abstraction (akin to CAVELib). Higher level functions such as a scene graph, object manipulation, physics and collision detection are expected to be provided through the integration with other third-party libraries. Integrations commonly used with FreeVR are SGI's Performer, OpenSceneGraph (OSG) and the Delta3D system [80]. In this regard, FreeVR sits in the middle between basic libraries such as CAVELib and complete framework like VR Juggler or CalVR, offering a balance between flexibility and ease of use. The FreeVR API is in C and no bindings to other languages are not part of the core distribution.

2.6.3 VR Juggler

The VR Juggler virtual reality toolkit [81] is based on a virtual platform framework. The virtual platform is constructed of two main components: the draw manager and the kernel. Every application acts as an application object in the form of a C++ object and the manager and kernel provide the application with interfaces to the graphics API and hard- ware devices respectively. The kernel executes application objects and handles communication between distributed managers. Each manager provides an abstraction layer on top of input devices, displays, networking, and windowing systems. Developers add support for new devices by creating a new manager through the VR Juggler API. Managers can be added or removed from the system at run time. Developers can create applications combining the VR Juggler API with existing graphics libraries such as OpenSceneGraph and VTK [82],. Compared to other frameworks, VR Juggler is more effective in maintaining a stable frame rate since it replicates the full application across nodes. This makes it less sensitive to network issues or single-node bottlenecks. To further tune performance, VR Juggler offers a debugging and analysis tool called VjControl. This tool collects performance data for an entire multi-node system like frame rendering times, GPU stalls etc.

2.6.4 CalVR

CalVR (Figure 9) is a virtual reality middleware system based on OpenSceneGraph [83]. CalVR implements similar core functionality present in other immersive environment frameworks like CAVELib FreeVR and VR Juggler and adds a menu system, multiple user interaction and

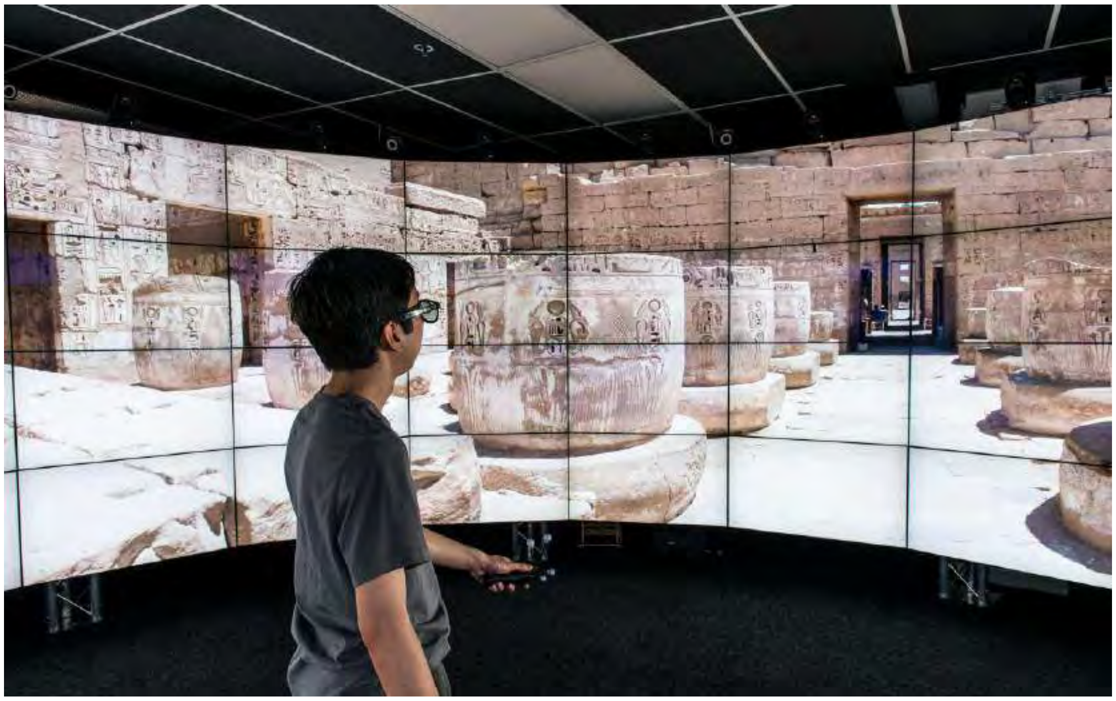


Figure 9. CalVR displaying a stereoscopic panorama inside the CAVE2 Hybrid Reality Environment

remote collaboration support. Applications are programmed using the OpenSceneGraph C++ API, plus several additional classes exposed by CalVR itself that handle 3D object manipulation, input and menu support. CalVR applications are implemented as plugins that can be loaded and unloaded at runtime. CalVR supports rendering stereo views for multiple users [84], but does not support fully separate views displaying independent immersive applications.

2.6.5 AVANGO

AVANGO is a distributed scene-graph framework [85] and flexible display configurations for different types of cluster-based large displays. AVANGO uses a data distribution architecture similar to OpenSG, but is based on OpenSceneGraph and the Python scripting language. Python can be used to create entire applications through scripting, making the system easier to access for less technical developers. AVANGO integrates modules that implement utility functions for VR applications like 3D menu support, display setup and input device configuration.

2.7 Game Engines

Game Engines such as Unity3D and Unreal Engine, offer some scalability and immersion support. Using third-party extensions, they can be adapted for larger cluster-based systems. They are designed to be used by artists so many of their capabilities are superfluous for scientific work, while other necessary features visualization (volume rendering, heterogeneous file formats, large-scale data support) are missing.

Name	Distribution Model	Input Handling	API	Multiview Support	Immersion Support
Chromium, ClusterGL	Render	No Input Handling	No API	No Multiview	No Immersion
SAGE	Render	2D	Low Level	Dynamic	Basic Immersion
DisplayCluster	Render	2D	Low Level	Dynamic	No Immersion
Equalizer	Both	2D	Low Level	Static	Advanced Immersion
CGLX	Application	No Input Handling	Low Level	Static	No Immersion
MPIglut	Application	No Input Handling	Low Level	No Multiview	No Immersion
FlowVR Render	Render	No Input Handling	Low Level	No Multiview	No Immersion
OpenSG, Garuda	Render	2D	Low + Medium Level	No Multiview	No Immersion
jBricks	Application	2D	Medium Level	No Multiview	No Immersion
Shared Substance	Application	Hybrid/Extensible	Low Level	Dynamic	No Immersion
MediaCommons	Application	2D	Low + Medium Level	Dynamic	No Immersion
CAVELib, FreeVR, VR Juggler	Both(Default=Application)	3D	Low Level	No Multiview	Advanced Immersion
CalVR	Both(Default=Application)	Hybrid/Extensible	Low + Medium Level	No Multiview	Advanced Immersion
AVANGO	Application	3D	High Level	No Multiview	Advanced Immersion
HRE OS	Both (Default=Application)	Hybrid/Extensible	Low + Medium + High Level	Dynamic	Advanced Immersion

Table III. A summary of the reviewed software frameworks and middlewares, categorized based on the multi-dimensional taxonomy presented in this chapter. Features marked in green represent desirable features in a Hybrid Reality Operating System, based on the requirements described in the introduction. Note: frameworks identified as having both a renderer and application distribution model typically implement a distributed application approach (an instance of the application runs on each cluster node), but their API is flexible enough to decouple application logic from rendering when desired. Frameworks with a multi-level API expose optional higher-level interfaces that can be skipped when the programmer needs more control over the implementation.

2.8 **Review**

A summary of the reviewed software frameworks for large displays and immersive environments can be found in Table III. The last row of the table enumerates the features for a Hybrid Reality Environment Operating System. These features follow from the requirements we discussed in sections 2.4 and 2.5, and are detailed in the following sections.

2.8.1 Input Handling

The HRE OS should handle both 2D and 3D input devices, events and interaction techniques. Depending on the execution environment, an application should be able to switch input devices without significant decrease in functionality or interaction efficiency. Since HRE systems exist at the intersection between 2D display walls and 3D immersive environments, HRE applications could make use of both 2D devices (mouse pointers, touch) and 3D devices (wands, markerless tracking systems, etc.). Choosing 2D interaction over 3D or vice-versa should only depend on application requirements and not on inherent limitations of the operating system. Given the variety of input devices that need to be supported, the input handing model should also be generic and extensible, allowing developers to integrate new devices as needed.

2.8.2 API Level

As discussed in 2.4, a multi-layer API is a way to support developers with a broad range of requirements for their applications. A high level API provides a simple and quick access path to the HRE system. Lower level APIs provide flexibility at the cost of increased complexity.

2.8.3 Multiview Support

In the background chapter, through examples and references to previous work, we also discussed the importance of letting users arrange visualizations on a large display as they see fit. Supporting this capability in an HRE operating system depends on two requirements. First, an interaction model should be provided to let users re-arrange content. Second, the operating system should use the layout information to optimize the use of rendering and computational resources in the HRE cluster.

2.8.4 Immersion Support

The HRE operating system should support all the advances immersion features present in state-of-the-art immersive application frameworks such as multiple stereo rendering modes, user-centered stereo, head tracking, etc. Additionally, given the desire to support multiple HRE users simultaneously, the operating system should offer some form of multi-user immersion.

2.8.5 Distribution Model

Ideally, the HRE OS should be flexible enough to support both a renderer-based and application-based application model. This can be achieved through application level replicated execution, if the operating system API is powerful enough to support asymmetric code execution and message passing between display and application nodes. In other words, user code should be ‘aware’ of its execution environment and have access to a low level API that supports implementing a distributed rendered model, if so desired.

2.9 **Summary**

Of the features I outlined, multiview support, immersion and hybrid input handling represent the core foundation needed to build a multiview immersive system, as we will see in Chapter 4. A flexible API and distribution models are not strictly needed: they are enriching features that increase the flexibility and usefulness of my implementation. In particular, a flexible API aligns with the need I identified in 1.4 to provide different layers of access to a framework, catering to different categories of users and developers.

The summary table illustrates how no existing software framework supports all the requirements for the HRE operating system described above. In particular, frameworks that support some form of multiview do not support immersion, and vice versa.

In the next chapter I will describe our implementation of an operating system for hybrid reality environments that addresses the requirements described in Chapter 1 and formalized in this chapter. This system is the foundation upon which Multiview Immersion is implemented, as I will detail in Chapter 4.

CHAPTER 3

OMEGALIB: AN OPERATING SYSTEM FOR HYBRID REALITY ENVIRONMENTS

During the development of the OmegaDesk hybrid workspace in 2010, we identified the need for a software framework to drive this novel platform. We planned for this framework to support interaction through the heterogeneous input technologies offered by OmegaDesk, including multitouch, marker-based and markerless tracking. Moreover, we required support for multi-display rendering and various stereoscopic modalities.

After an initial exploration of available frameworks, it became clear no single solution would support all our requirements, so we set out to design our own. At the same time, it was evident multiple software packages offered subsets of features that we found desirable. As we wanted to avoid re-engineering existing solutions, we chose to design a framework that would act as a set of abstractions and adapter interfaces to a variety of input, display and visualization libraries exposing their functionalities in a transparent way to OmegaDesk applications. This allowed us to select the libraries we found useful without committing to them, leaving the door open for replacement as new input or display technologies became available. It also made it possible to use complex libraries like the Visualization Toolkit [82] on devices that would normally not support them, with minimal changes to visualization applications code. We realized this design principle was useful beyond OmegaDesk, and we evolved our framework, called Omegalib to support our next large hybrid display environment, CAVE2.

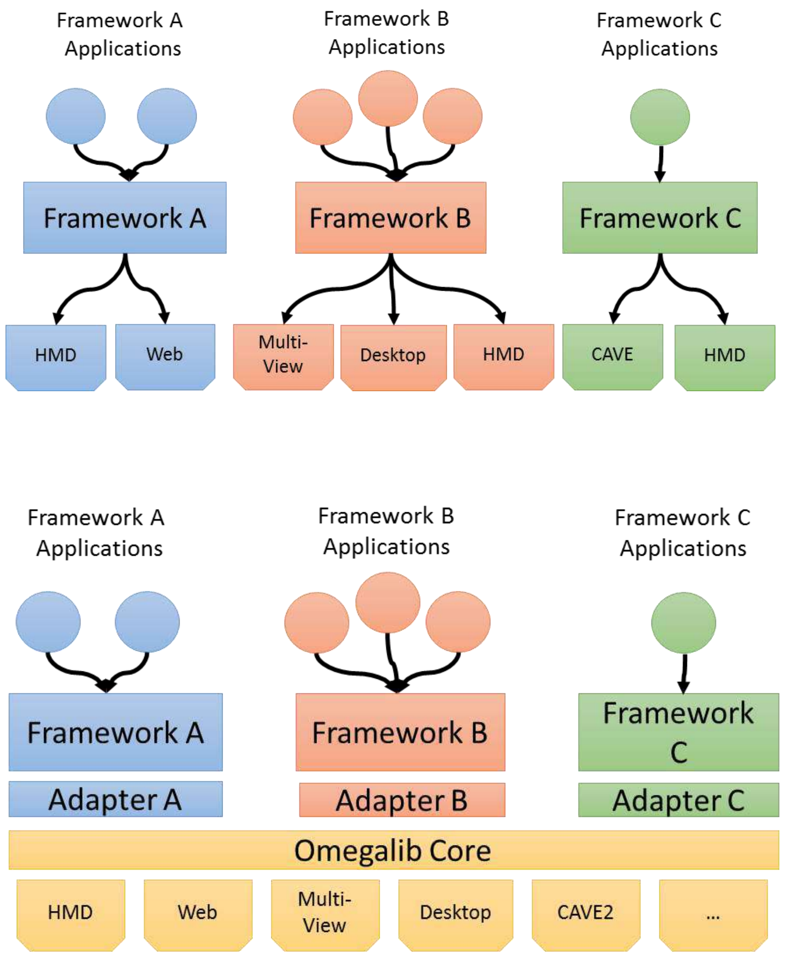


Figure 10. (Top) The current landscape of visualization frameworks/toolkits: development is compartmentalized. (Bottom) Omegalib represents an integrated visualization ecosystem. Frameworks interface with the Omegalib core using lightweight adapter modules. The core provides a common messaging facility to tie applications together. Support for specific display/interaction devices is implemented once and is accessible to all frameworks.

As Figure 10 (right) shows, Omegalib acts as an abstraction layer between visualization hardware and user applications. Applications use the high-level graphics or visualization framework of their choice. Through a series of lightweight adapter modules, each framework interfaces with the Omegalib core, which in turn abstracts both the display and input capabilities of target hardware, simplifying porting applications across heterogeneous systems. This design principle is widely used: however, its implementation in Omegalib takes place at a higher level. Instead of abstracting basic concepts like graphical primitives or simple input events, we abstract full 3D objects, object hierarchies and logical input devices. This allows us to create simpler, lightweight adapters that leave the original frameworks in control of most of the rendering. This makes it comparatively easy to enable new frameworks and their applications to work with Omegalib. In the next section we will outline the HRE operating system model that Omegalib is based on. In section 4.2, we will discuss the Omegalib implementation, including details of the adapter layer implementation.

3.1 **Design**

Figure 11 illustrates the conceptual structure of the Omegalib system, including its components and the connections between them and the Hybrid Reality Environment hardware. This structure derives both from the requirements we identified for an HRE OS and from best practices we observed in prior works including VR toolkits, distributed rendering systems and display wall control software. The system consists of three main components: a runtime layer, an input manager and an application controller / message broker.

The runtime represents the core of the HRE OS model: it implements both the hardware abstraction and adapter layers I presented in the previous section. Applications interact with the rest of the system through this runtime, either using one of the high level framework connected to it or through some utility API exposed by the runtime itself (for instance to receive input, control the application state, interact with the abstract 3D scene etc.). The runtime controls the underlying graphical hardware, which can be a single computer, rendering cluster or streaming system. Control of graphical hardware is delegated to back-end adapters called Display Systems, as I will discuss in section 3.2.1 One separate instance of the runtime exists for each application running on the system

The input manager handles input streams from heterogeneous devices, and performs a first level of input processing, for instance merging input data coming from separate physical systems into logical input devices. The input manager routes input data to one or more applications, as determined by the Controller. I will detail our implementation of the input manager (called Omicron) in section 3.3.

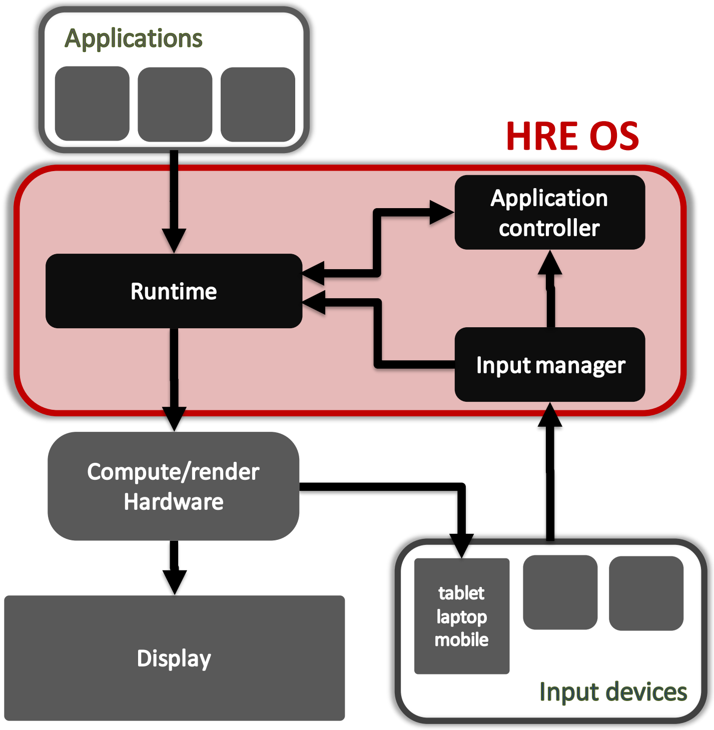


Figure 11. The high-level structure for a multi-view operating system for Hybrid Reality Environments. The core components of the operating system are: the distributed application runtime; a controller and inter-process communication (IPC) manager that handles application lifetime and communication between application instances; a distributed input manager capable of handling heterogeneous devices.

The Controller and inter-process communication component orchestrates the execution of multiple applications on the system and handles their lifetime. The controller offers a central messaging system that applications can used to exchange data with each other, making it possible to coordinate and link multiple visualizations. I discuss this functionality in 3.4. This component also handles the flow of data from the input manager to applications, routing different input devices to applications based on application-user associations. This feature is fundamental to the implementation of Multiview Immersion and will be detailed in Chapter 4.

The three HRE OS components I outlined share one important characteristic: they are loosely coupled and designed to run independently from each other, in separate processes or even on separate machines. This guarantees the flexibility needed to support a wide range of hardware, from single computer systems to cluster installations with dedicated input, control and rendering units. When this flexibility is not needed, all components can also run within a single process as a normal application.

3.2 **Runtime**

The default execution and rendering mode for Omegalib is replicated execution: each node in a display cluster runs an instance of the target application, while a master instance synchronizes the updates and screen buffer swaps across the nodes (Figure 12). Omegalib also makes it possible to control execution on a node-by-node basis (the API offers functions to check on what node the application is executing), and it supports the synchronization of custom data objects between master and slave instances.

At the front end, Omegalib integrates visualization frameworks and tools using an abstract scene graph and pluggable render passes. The abstract scene graph makes it possible to decouple object interaction techniques from concrete object representations. A concrete object can for instance be an OpenSceneGraph node [68], a Visualization Toolkit (VTK) Actor [82], or some other entity provided by one of the Omegalib front-ends. Concrete objects can be attached to nodes of the scene graph, which then represent the interface between user code and the underlying objects. Users can therefore control object transformations, visibility, hierarchy, etc. through a unified API that is independent from the library used to draw the objects on screen. Each front-end provides a render pass that can be attached to the Omegalib rendering system, and that takes care of drawing concrete objects of a specific class. Users needing low level drawing control can implement their own render pass and obtain a GLUT-like callback interface that can be used to execute custom OpenGL code. Render passes can be prioritized and can perform 2D or 3D drawing. For instance, a high priority 2D rendering pass can be used to overlay 2D content on top of an interleaved-stereo 3D scene.

Event Services implement input event processing. While they are part of the Input Manager component, they can also be run as part of the runtime. This is useful for instance to allow a 'master' display in a cluster installation to be controlled through the local keyboard and mouse, while also processing input data from a remote input server, making it possible to dedicate a separate machine as the hub for all input devices in an HRE installation. A service may implement support for a specific category of physical input devices (like a keyboard or mouse controller), or it can aggregate and reprocess events generated by other services. Event services run exclusively on the master instance of a distributed application. The Omegalib runtime takes care of serializing and distributing events to slave instances.

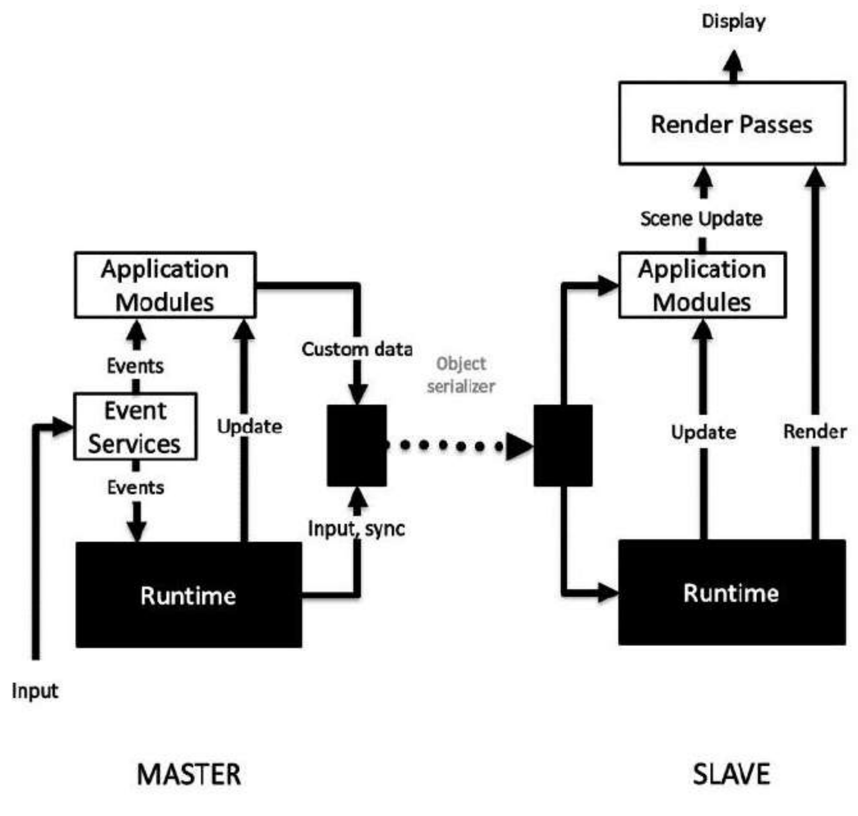


Figure 12. An overview of the communication flow of a distributed Omegalib application. In this example, the Master node runs in headless mode, and only takes care of application and input event processing. © 2014 IEEE

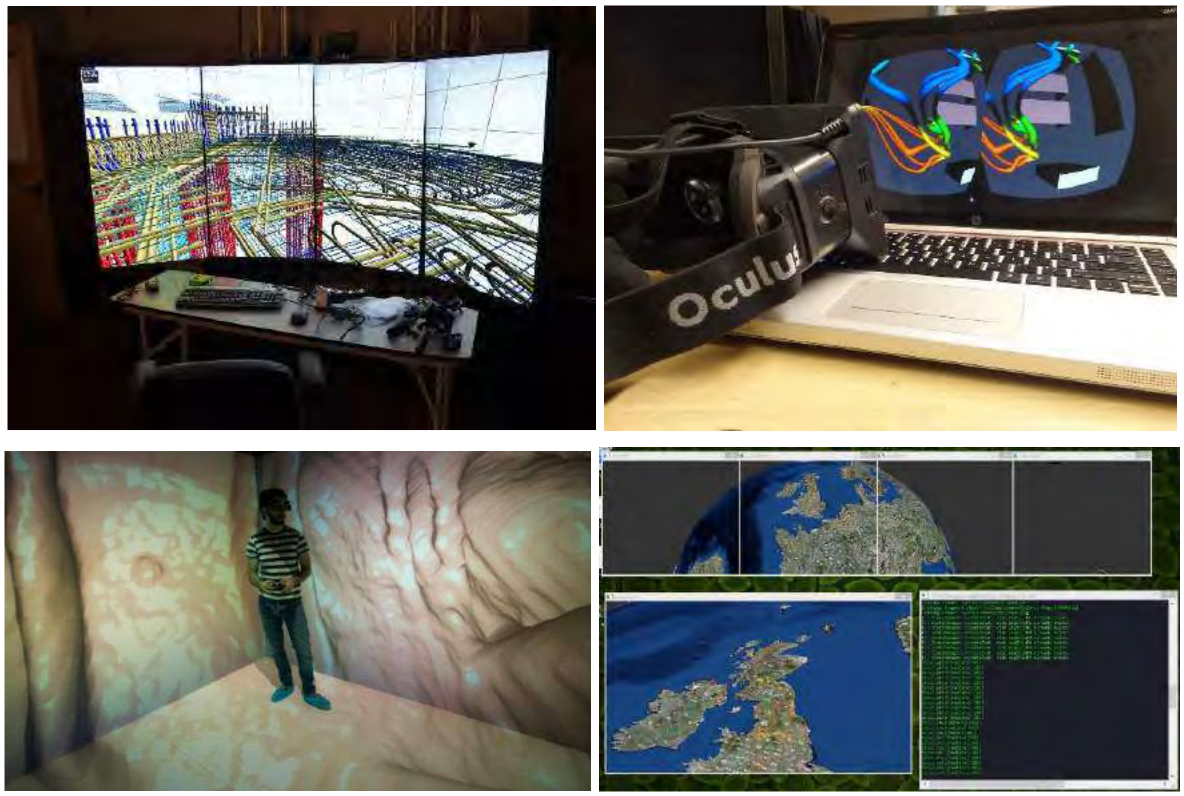


Figure 13. Examples of Omegalib adapter module and display system use. Each picture shows how various third-party graphical libraries can leverage Omegalib abstracted display system to run on a variety of hardware. Top Left: an OpenSceneGraph 3D model viewer on a multi-screen curved display. Top Right: a flow visualization using Vtk on the Oculus DK1 head mounted display. Bottom Left: a virtual colonoscopy application inside the Stony Brook CAVE (image courtesy Charilaos Papadopulos). Bottom Right: OsgEarth running in an emulated 4-screen CAVE environment.

Application Modules contain the core of an Omegalib application or adapter logic. Modules can receive input events, update the abstract scene graph (or other custom scene representations), and communicate with other modules. Modules run on both master and slave instances of Omegalib regardless of whether they display content or not.

Render Passes implement all the functionality needed to render scenes to cluster tiles or to secondary image streams. Application developers typically do not need to implement render passes, unless they need to perform custom low-level drawing operations. Render passes are implemented by adapter modules: they forward draw context and issue draw commands in the correct format used by the third party library. Render Passes are executed only on application nodes that need to perform rendering operations. For instance, a headless master configuration will not run render passes on the master application instance.

3.2.1 Display Abstraction

Omegalib abstracts display support at two levels. At the lowest level, *display system* implementations connect the runtime to different display hardware setups, such as head-mounted displays, cluster-based display systems or standard desktop computers. A display system is managed through a *display configuration*, an internal structure that fully describes a display system's physical geometry and capabilities. The configuration can be specified manually, or it can be built automatically by *display configuration builders*. The display configuration builder is the second level of the display system abstraction. It makes it possible to describe a display system's geometry though a simplified set of parameters. For instance, a display wall can be configured using a `PlanarDisplayConfig` object with a specific number of display tiles, a tile resolution and real-world position of the display center. A semi-cylindrical display such as CAVE2's can be configured with a `CylindricalDisplayConfig` together with a cylinder radius, center, number of tiles etc.

Display configuration builders also offer additional functionality like optimized conversion from pixel to real-world coordinates. This is faster than generic solutions since it can leverage knowledge of the implicit screen geometry. Display configuration builders are a central part in the implementation of Multiview Immersion, as will be detailed in chapter 4. Both display systems and display configuration builders can be implemented through the Omegalib module system (3.2.5), allowing for the creation of a reusable library of common hardware setups.

The combination of display systems, the abstract scene graph and render passes is the foundation that allows third-party visualization frameworks and graphical libraries to be adapted to run on the heterogeneous platforms supported by Omegalib: a few examples of this are illustrated in Figure 13.

3.2.2 Panoptic Stereo

Since Omegalib is aimed at supporting co-located collaborative groups, it is fundamental to provide stereo-viewing capabilities to multiple users in the system. Because traditional CAVEs use viewer-centered perspective, one effect that typically occurs is that when the tracked user looks 180 degrees away from a given stereo screen, stereo reverses. While this is acceptable for a single user, given they are no longer looking at the screen, it poses a problem for other team members, especially in a larger cylindrical environment such as CAVE2.

To solve this issue, Omegalib supports *panoptic stereo*. Techniques similar to Panoptic stereo have been implemented in the past to support multi-user stereo without generating multiple views [86]. When Panoptic stereo is enabled, the system tracks the main viewer’s position, but generates head orientation information based on each display plane normal. This way, the stereo frustum is projected outward to each display panel, preserving stereo separation for each user in the system. An additional benefit of Panoptic stereo is its independence from frame latency: When users turn their in place, stereo will look correct regardless of the application frame rate, leading

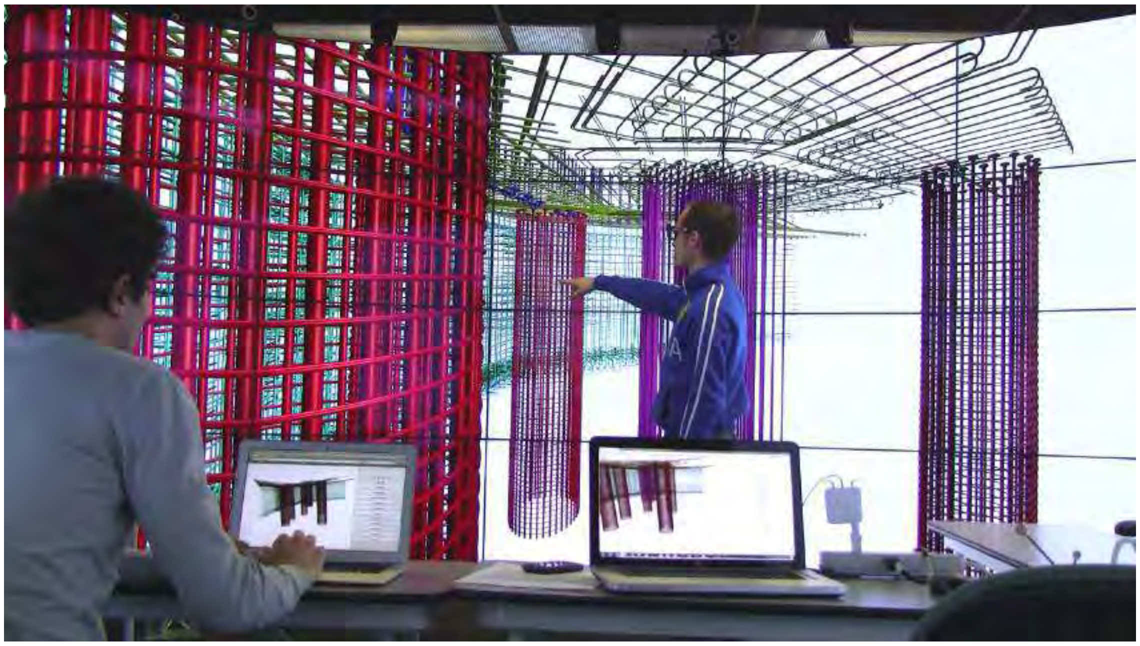


Figure 14. Example of personal secondary view streaming during collaborative analysis of a structural engineering model (data courtesy Falko Kuester). Each user controls a separate view running on his laptop. These views are rendered by the HRE cluster driving the main display, and are streamed to the user's browser. While viewpoint are independent, other view options (like enabled sub-components of the 3D model the 3D model) are synchronized and their changes are reflected in all views.

to a much more comfortable viewing experience. This comes at the price of a slightly incorrect overall projection, particularly when users rotate their head sideways. Panoptic stereo can be turned on and off at runtime.

3.2.3 Streaming to secondary Views

Another approach to multiview support is based on streaming additional 2D or 3D views to a *secondary device* like a laptop, smartphone or tablet. These devices can act both as secondary display surfaces for visualization output and as input sources to the application (Figure 14)

Personal secondary views (PSVs) are appealing for a number of reasons. In a multi-user setting, PSVs create a private visualization space that each user can control to visualize information relevant to his own analysis task. PSVs can also act as a sort of augmented-reality tool: A smartphone-based PSV can be moved around by a user within a virtual environment and provide additional data about objects that fall within the view frustum [87]. From a technology standpoint, it makes sense to use laptops, smartphones or tablets as PSV channels. PSVs implemented on these devices can be used as a view and a source of input to the application. Touch screens offer a suitable surface for 2D interaction tasks like system control and brushing that are difficult to perform precisely in an immersive environment [45]. Moreover, since PSVs are decoupled from the main visualization environment, they can be used to facilitate remote collaboration: users outside of the environment can connect to it, receive a view of the data and interact with it in real time as users present in the system see the changes in the main and local secondary views. This is a particularly interesting aspect, since the size and cost of state-of-the-art virtual reality environments limits their access for everyday scientific workflows.

3.2.4 Dynamic Configurability

As mentioned in the introduction, one of the objectives of Omegalib is the creation of user-defined and reconfigurable workspaces: areas of the display that are dedicated to a 2D or 3D application, that can be re-defined while the application is running and can be independently controlled.

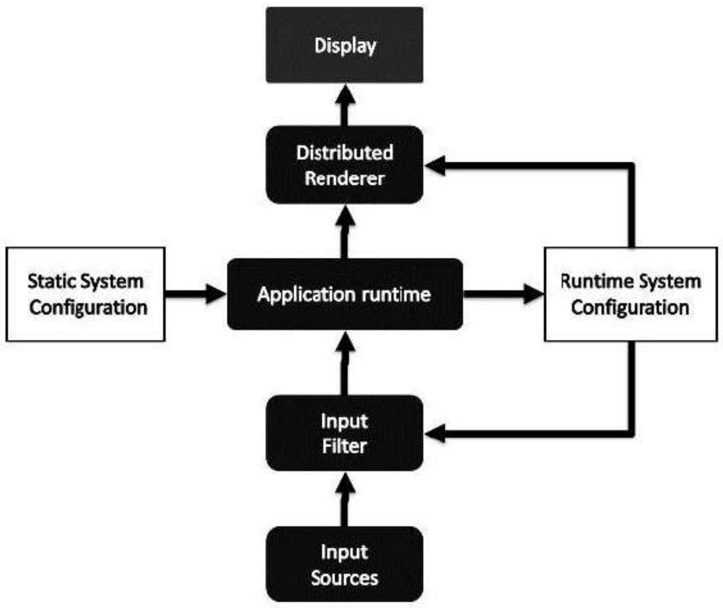


Figure 15. The application uses launch-time setting and user interaction to generate a runtime system configuration that determines the application workspace area. This configuration controls rendering resource allocation and input filtering. © 2014 IEEE

To support this feature, we let users specify an optional 2D display region as a startup argument to an Omegalib application. The application runtime uses this information to find the subset of nodes in the rendering cluster that drive displays in the selected region: it then adjusts the system configuration to launch the application only on the identified target nodes. Multiple applications can be launched on the system in this fashion. The runtime also manages the networking setup, making sure that each application instance uses an independent set of ports to establish communication between the application master and the slaves.

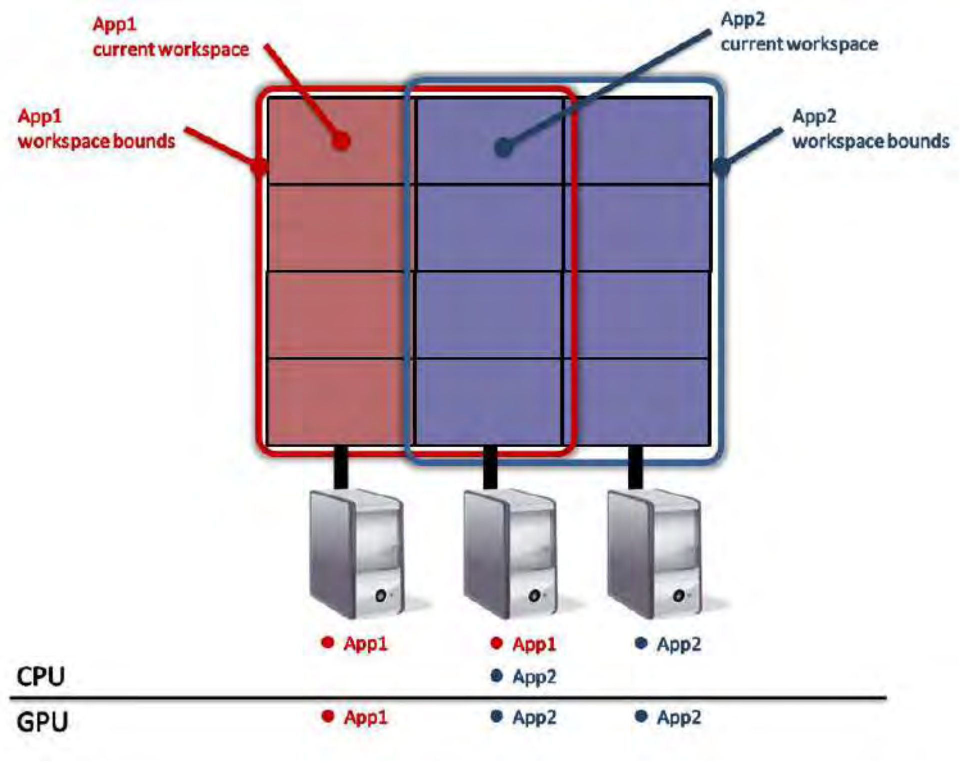


Figure 16. An example of dynamic workspace configuration for two running applications. Each machine controls one display column consisting of four tiles. Application 1 is launched on the left and center column. Application 2 is launched on the center and right columns. Applications workspaces overlap on the central column. In this example, the runtime configuration allocates the central column to Application 2. As shown in the bottom rows, both applications are running slave instances on the center column machine, but only the active application is assigned to the GPU. © 2014 IEEE

To increase the flexibility of workspaces, we also let users expand or shrink the visible area of each workspace within the bounds specified at run-time. When a user shrinks a workspace, the runtime disables rendering on the nodes whose tiles are not covered by the workspace active area (Figure 16). While GPU resources are de-allocated, the application remains available on the machine: if the user later decides to expand the workspace again, rendering on the inactive tiles can be reset almost instantaneously.

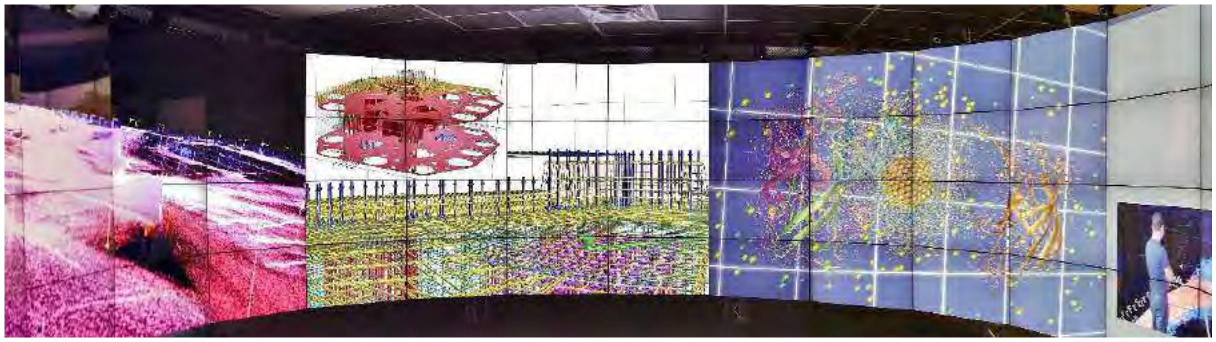


Figure 17.Omegalib running multiple workspaces in the CAVE2 Hybrid Reality Environment.

When launching multiple applications, it is therefore possible to run applications on overlapping regions of the display system: the display space shared by multiple regions can be associated to any of the overlapping applications at runtime. This double level of dynamic configuration (launch-time and runtime) provides a good level of control over cluster resource usage versus workspace flexibility. On one end of the spectrum, applications can be launched on non-overlapping regions, optimizing cluster resource allocation (each node is dedicated to a single application) but giving up dynamic display allocation. On the other end, applications can be launched on fully overlapping regions covering the entire display space: in this case the workspaces have the full runtime flexibility, at the cost of sub-optimal cluster resource allocation (each node needs to keep one active instance of each application).

3.2.5 Module System

Omegalib's software infrastructure is fully modular: a lightweight core provides abstractions of display systems, input devices and visualization sources. Domain users extend Omegalib through *programmed adapter modules,* or interfaces, that streamline data integration from different, commonly used packages and leverage the capabilities of advanced software and displays. The module ecosystem is decentralized, giving research communities full control over the modules they create, while allowing them to easily share, link and reuse modules from other sources, making it easy to deploy a tailored visualization system to researchers.

3.2.6 Interpreter

Although the main Omegalib API is in C++, most of it functionality is exposed to a scripting interface: the Omegalib runtime embeds a python interpreter that can be used to launch standalone script applications, or can be used to control running applications through a runtime console. Other than facilitating development access to non-technical users, scripting support has the added advantage of simplifying application portability. A script application can run on a personal computer or on a HRE without requiring recompilation or reconfiguration. This makes it possible for scientists to work on a visualization script on their own computer, save it to a flash drive, plug the drive into the HRE system and re-launch the visualization during a collaborative research session.

3.3 **Input abstraction**

Omegalib Input services are implemented though an external library called Omicron. Omicron was initially developed as part of Omegalib, before being split in order to simplify its development and integration with other tools. Omicron input services are designed to abstract both 2D and 3D input devices using a unified event format. Input services can be integrated directly in an application, or they can be serialized and sent over the network to one or more clients. The input event structure is described in Table IV. Omicron input event structure Input services can also be connected to each other to generate composite input events: for instance, tracking data from a motion capture input service can be merged with input data from a game controller service to create wand events (that is, events from a device capable of 3D pointing, Figure 18). The Omicron implementation currently supports the following input event types:

- *Pointer* events are used for devices that generate 2D input data, such as mice, multitouch surfaces or 3D devices pointing towards a surface.
- *Keyboard* events are generated by physical or logical keyboards. These events have no position or orientation.
- *Controller* events are generated from gamepads and similar devices. Controller events from multiple simultaneous devices are supported. Events of this kind include data from digital buttons, analog axes and triggers and gyroscopes or accelerometers embedded in the device.
- *Motion Capture* events are generated by devices capable of tracking 3D position, orientation, or both including marker based systems like VICON or Optitrack cameras and markerless systems like the Microsoft Kinect. Motion Capture events can also track multiple joints within a single event, which is useful for representing full-body tracking data.
- *Wand* events are a hybrid between pointer, motion capture and controller events. They have a 3D position and orientation but also include buttons, axes etc. coming from a controller device.
- *UI* events are generated by graphical user interfaces and represent button clicks, sliders, input from text boxes etc.
- *Brain* events are experimental events generated from devices capable of capturing EEG data. These events contain several analog values representing detected brain wave values.
- *Speech* events are generated by devices capable of speech recognition (such as the second-generation Microsoft Kinect).

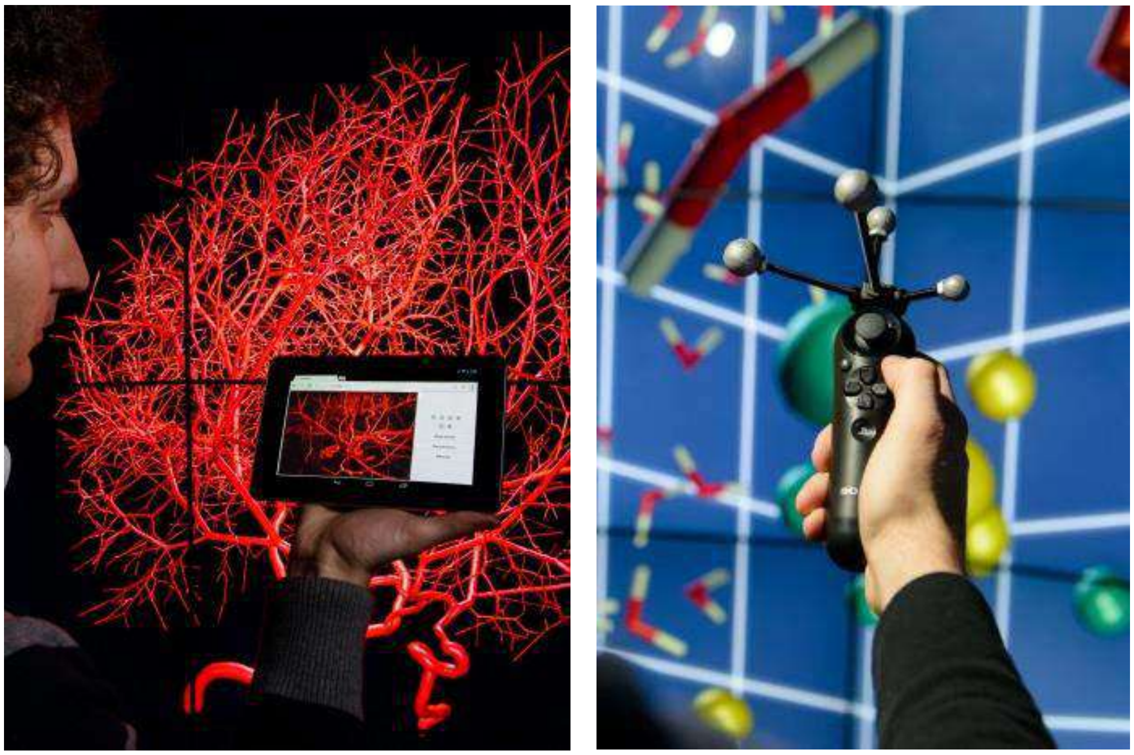


Figure 18. Examples of interaction devices supported by Omegalib: a tracked wand controller and the Porthole web interface running on a tablet. Porthole work as an input-output device and displays a personal secondary view streamed from the rendering cluster.

Field name	type	Reserved Size (bytes)	Description
timestamp	Unsigned integer	4	the event timestamp in milliseconds using the standard unix time
serviceType	enumeration	4	The class of service that generated this event (Pointer, Keyboard, Controller, …)
serviceId	Unsigned short	2	The unique identified of the physical or logical device that generated this event
userId	Unsigned short	2	The unique identified of the user associated with this event
type	enumeration	4	An enumerated value giving general information about the natire of this event (like a button being pressed or released, a pointer moving, a generic update, etc.)
flags	Bit field	4	Additional Boolean data associated with this event (such as the state of buttons on a gamepad, modifier keys etc.)
position	3D vector	12	The 3D position of this event. For 2D events, the first 8 bytes are used
orientation	quaternion	16	The 3D orientation of this event. For 2D events, the first 4 bytes are used
extraData	variable	64K max	A variable size field that can be used to store additional data attached to this event
extraDataType	enumeration	4	Indicates the type of data stores in the extra data field (such as integer or floating point arrays, 3D vectors, text, etc.)
extraDataItems	Unsigned int	4	The number of elements stored in the extra data array
extraDataMask	bitfield	4	A generic bit field used to specify flags about extra data items. For instance it can be used to mark valid/invalid entries in the extra data field to optimize network transmission.

Table IV. Omicron input event structure

3.4 **Message Passing**

I have so far discussed two of the software components of the HRE operating system model presented in Figure 11: the application runtime and input abstraction. The third and final component is the controller and Inter-process communication (IPC) manager. The primary purpose of the controller is to manage the execution of applications, providing users with an interface to start and stop applications manage their workspace areas and exchange messages between them. The controller (whose Omegalib implementation is called Mission Control) runs as a server to which applications connect once started. Applications connected to a Mission Control server can exchange messages with each other (typically script commands), or receive messages from third party software through an external interface. Mission Control therefore allows multiple views in the HRE to communicate with each other, for instance in order to coordinate the information they display. Since views run as separate processes in the cluster, their frame rates are independent: this is a desirable feature when one of the views is computationally or graphically intensive, while others require real-time interactivity.

Protocol	**Transport**	**Use**
Omicron	TCP + UDP	Input
Mission Control	TCP	Messaging and data exchange
Porthole	Websocket	Messaging and view streaming

Table V. A summary of the core Omegalib protocols, their transport layer and purpose.

Table summarizes the format and purpose of Omicron, Mission Control and Porthole, the three protocols Omegalib implements to support input, messaging and view streaming respectively. In the previous sections, we described how each is used in different areas of the

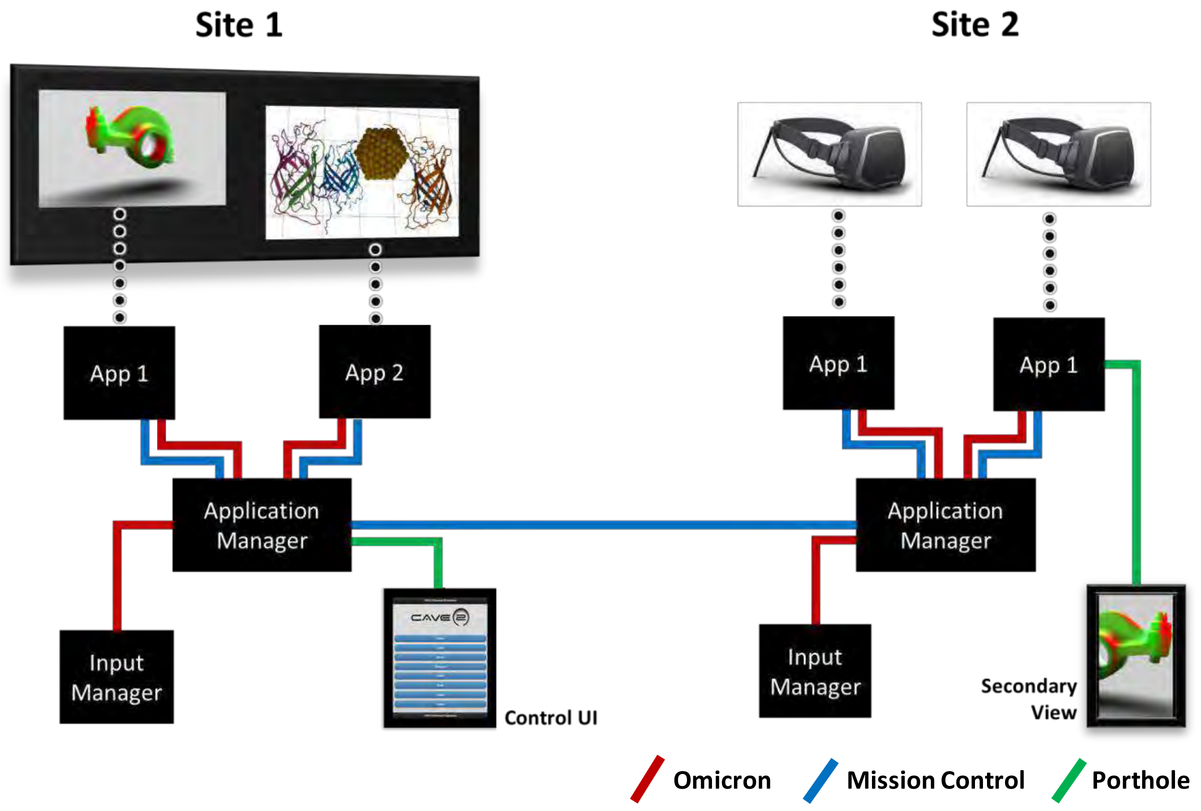


Figure 19. An example of an Omegalib multi-application setup illustrating how the input, messaging and streaming protocols interoperate. The setup has two sites, running two applications. Application 1 is replicated on both sites and its view can be synchronized using Mission Control. On Site 2, Application 1 is also streamed to a local 'operator' view for users not wearing a head mounted display. On site 1, a web interface is used to launch and control applications on the shared display wall.

HRE Operating System Model. These protocols are designed to offer complementary features and work together to build multi-application visualization ensembles, where views can be distributed on heterogeneous displays (including HREs) both local and remote. As with the rest of the Omegalib implementation, the protocol suite is designed for flexibility instead of focusing on a specific structure for applications or the underlying hardware infrastructure. This allows Omegalib to support a wide variety of visualization system designs, from simple desktop analysis tools to render on-demand web-based visualizations to multi-device, multi-site systems like the one exemplified in Figure 19. In the next section, we will present some real-world examples that make use of this flexibility, with a particular focus on multi-view applications.

3.5 **Use Cases**

Omegalib has been used in several projects both at the Electronic Visualization Laboratory and at other institutions. This section presents a selection of these use cases that exercise several features of Omegalib. I contributed directly to the development of these applications, while others are the result of independent work of other research groups.

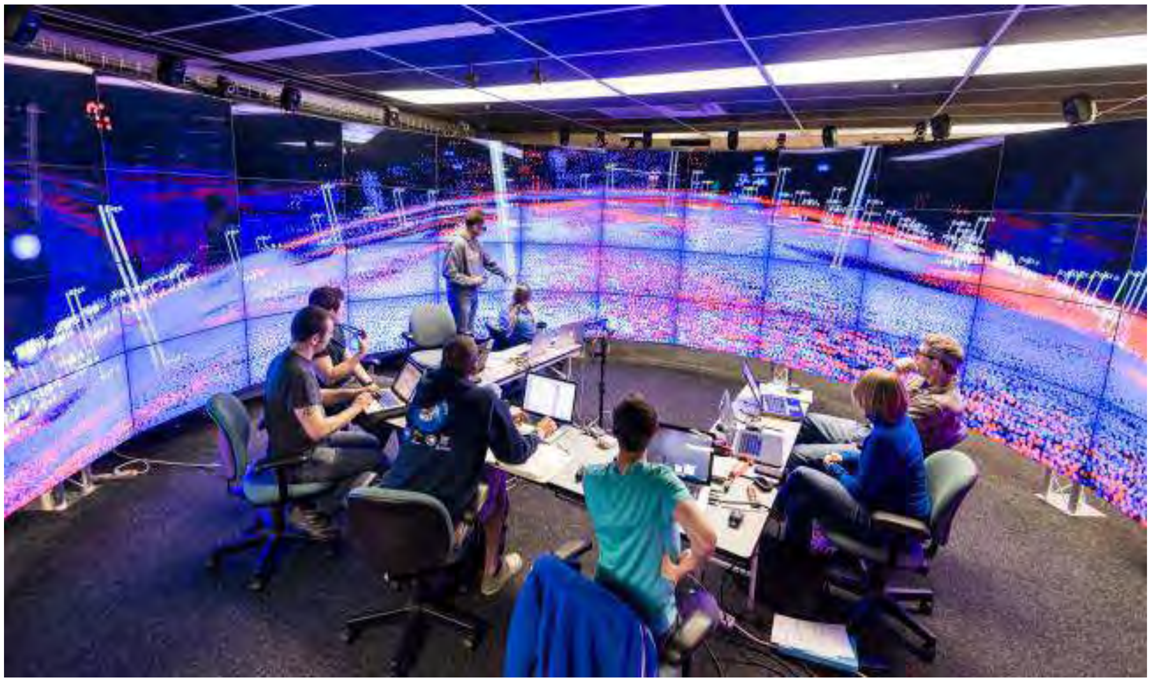
3.5.1 ENDURANCE

To evaluate the effectiveness of Omegalib in supporting co-located collaborative work, I used it as the development platform for a geo-science application for the visualization and analysis of sonar data. This application was first used during a two-day meeting of the multidisciplinary team working on the NASA ENDURANCE project.

The McMurdo Dry Valleys, located within Victoria Land in Antarctica, are one of the world's most extreme deserts, and represent the largest ice-free region in the continent. The unique conditions in the valleys are in part caused by winds reaching speeds of 320 kilometers per hour, evaporating all water and ice in the environment. Some of the lakes in the Dry Valleys rank among the world's most saline. One of them is Lake Bonney, a perennially ice-covered lake at the end of Taylor Glacier. Since the Dry Valleys are one of the terrestrial environments closest to Mars and to some of Jupiter's moons, they are considered an important source of insights into possible forms of extraterrestrial life. For this reason, NASA funded the Environmentally Non-Disturbing Under-water Robotic ANTarctic Explorer (ENDURANCE) project. ENDURANCE is an autonomous underwater vehicle (AUV) designed to map the geometry, geo-chemistry and biology of Lake Bonney in three dimensions.

The underwater vehicle operated during two Antarctic summer seasons (2008 and 2009). Each mission entailed a set of deployments of the AUV, for a total of 45 dives. For each dive, the AUV operated depending on 3 distinct science objectives: water chemistry profiling, bathymetry scanning, or glacier exploration. For the purpose of bathymetry reconstruction, the source data consisted of about 200 million distinct sonar range returns, plus navigation data and AUV attitude information at 0.2 second intervals [88]. From these sonar points, a high-resolution 3D model of the lake bottom was generated, comprising approximately 200K vertices and 400K faces. Over the course of the full two-day ENDURANCE meeting, the research group had to complete multiple tasks: discuss new vehicle designs for a future mission, analyze mission logs and cross-reference them to water chemistry readings, and generate a new 3D map of the lake based on the collected sonar data.

As shown in Figure 20**,** the team used the display in different configurations during the meeting. During the initial evaluation of sonar data, the entire display was dedicated to a 1-to-1 scale, immersive visualization of the sonar point cloud. This visualization allowed the team to identify issues in the data, compare depth measurements from different data sources and iterate through data collected for each mission. Later in the meeting, the 3D workspace was shrunk to make space for additional 2D views representing satellite imagery from Lake Bonney and different versions of the lake bathymetry represented as a contour map. One of the 2D views was controlled by an Omegalib script running a VTK pipeline and was linked to the 3D view.



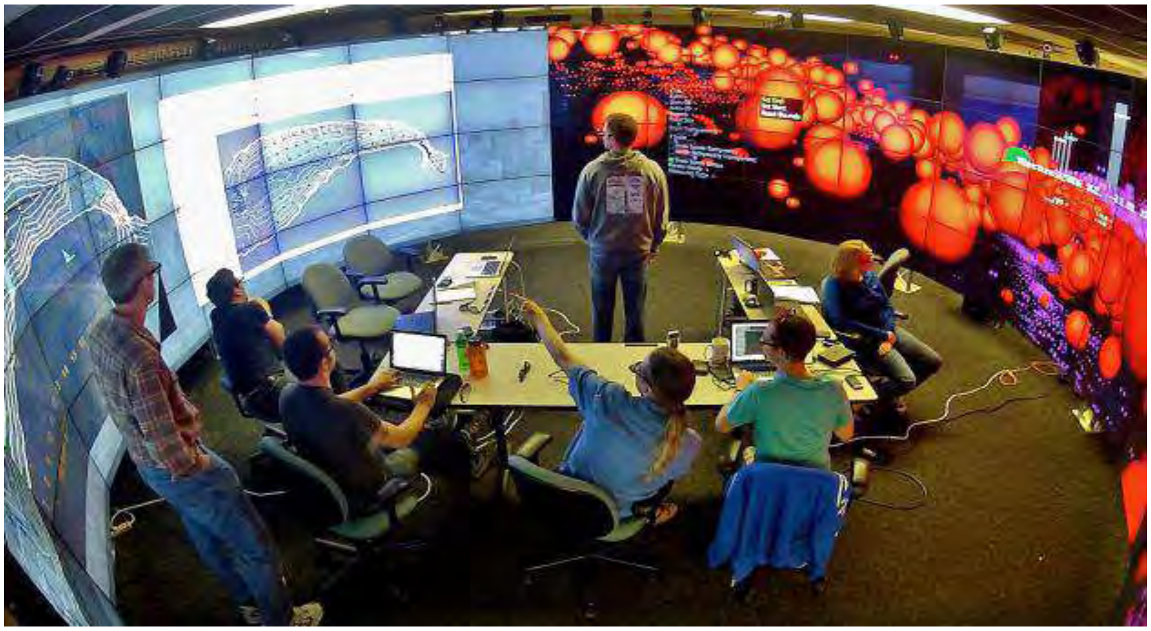


Figure 20. Two photographs taken during a co-located collaborative meeting in CAVE2. On the top, an Omegalib immersive visualization is running on the full display. On the bottom, CAVE2 is split into two workspaces to display additional 2D views. Switching between the two modes can be done at runtime, without resetting running applications. © 2014 IEEE

As researchers picked points in the immersive environment (effectively making ‘virtual measurements’ of the lake depth at points they deemed relevant), the 2D view would update, regenerating the contour information to take the new points into account. A researcher could use the hand-held interaction device (a tracked game controller) to navigate the 3D view, pick depth points or rearrange and resize the 2D views by simply pointing in the desired direction on the screen. Other users could control the view arrangement and content from their laptops. The workspace allocation could also be controlled by the hand-held device (using an on-screen menu) or from one of the laptops.

3.5.2 Developers sharing CAVE2

Another advantage for multiple workspace support is specifically targeted at application developers. As many other large scale display environments, HREs like CAVE2 are expensive and offer limited availability: they are often a highly contended resource, with multiple application developers scheduling access to the system to make sure their work does not conflict with others’. Emulators and smaller system replicas help, but are not a perfect substitute.

For instance, estimating the performance of an application in an emulated environment is complex, due to the difference in hardware and display geometry between the two environments. Thanks to runtime workspace configuration, multiple developers can use an HRE system concurrently. I observed this usage pattern multiple times in the CAVE2 system. Developers join a work session and negotiate display allocation with others, so that each developer has a section of the display exclusively available to him/her. Developers then use a command line switch to start their application on their workspace. Developers occasionally ask others to control the full display for a brief time, usually to test a specific feature.

3.5.3 Firefly

Northwestern University Assistant Professor Claude-André Faucher-Giguère leads the galaxy formation group at the Center for Interdisciplinary Exploration and Research in Astrophysics (CIERA). His team researches galaxy formation and evolution through numerical simulations. These simulations generate thousands of timeframes, each with more than 20-million data points representing the spatial distributions and properties of gas, stars, and dark matter. The team wants to analyze the data on desktop workstations and on large displays for more in-depth data analysis. A new visualization tool, Firefly [89], was developed for this team.

Firefly leverages Omegalib scalability features to run on desktop machines, a cluster-based 3D display wall (the CAMI 3D wall described in the next section) and stream to remote users (Figure 21**,** bottom). The Python interpreter integrated in Omegalib allows the research group to easily modify the application, and integrate it with their existing python-based pipeline. Several modules developed to support this application such as point cloud rendering, HDF5 and numpy data loading, are now available to other Omegalib applications. Faucher-Giguère's team is now extending Firefly in collaboration with an astrophysics team at CALTECH, and plans to use some of the visualizations for outreach at the Adler planetarium.

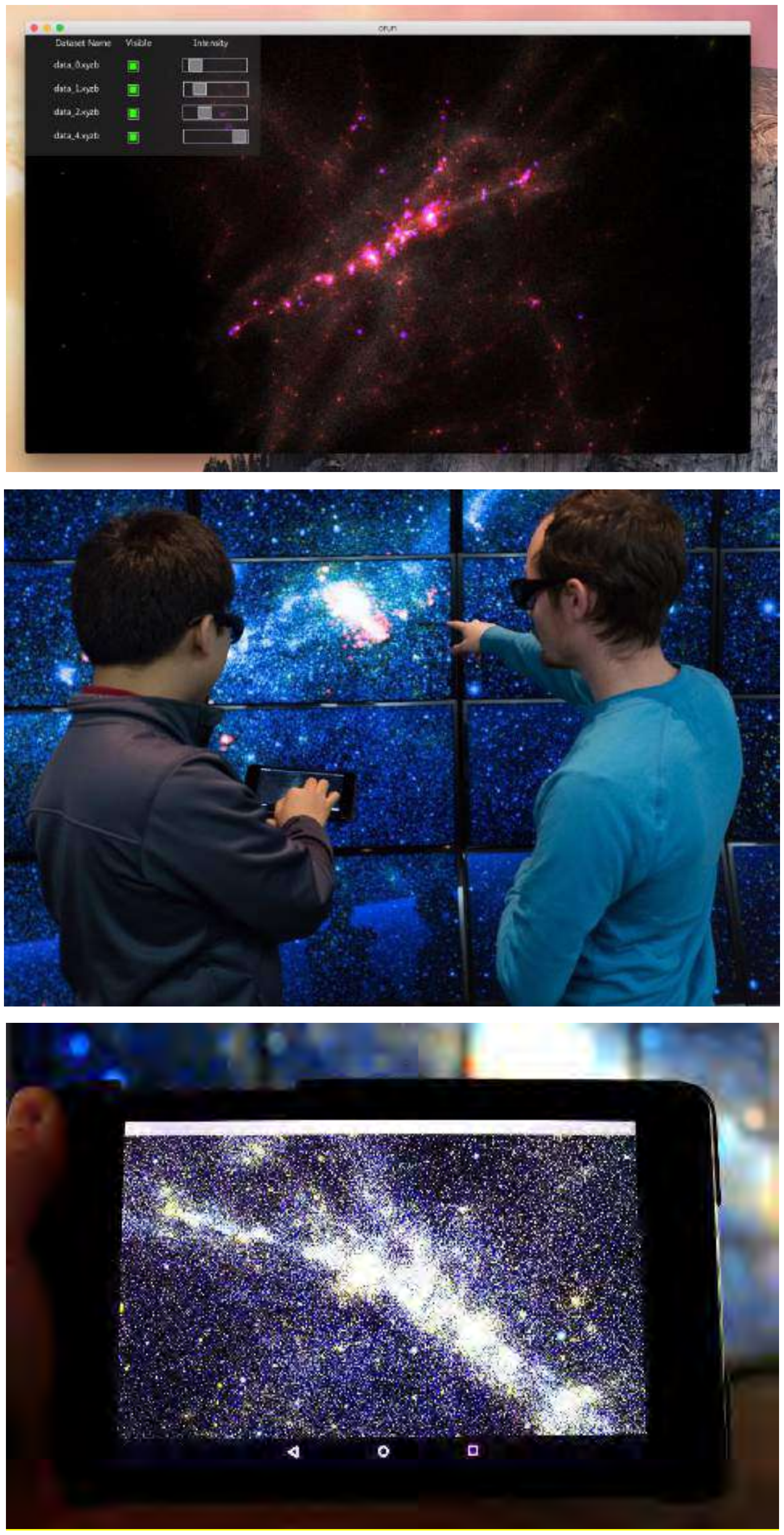


Figure 21. An Omegalib-based galaxy visualization runs on a cluster-driven 3D display wall (center), a desktop (top), and streams to a browser on a tablet (bottom). Omegalib supports a wide spectrum of display/interaction systems without modifications to its source code. Data from the Northwestern University GALFORM group.

3.5.4 CAMI Wall

The Center for Advanced Molecular Imaging (CAMI) at Northwestern University hosts a 50-megapixel 3D display wall, driven by a cluster of 5 multi-GPU machines. The display is used to present data generated by this center's instrumentation such as volumetric scans of animal specimens, confocal microscopy, and 3D molecular structures of organic compounds or drugs under development. The CAMI wall runs a set of Omegalib applications that function as simple viewers for 3D models, volumes, movies etc.

Applications are launched and controlled using an IPad interface running inside a browser. All the viewer applications are pre-loaded and running simultaneously on the wall thanks to Omegalib's dynamic configuration capabilities. This makes it possible to switch immediately from one demonstration to the next without having to wait for each application to load. The system is developed entirely in python making it much simpler to maintain compared to the previous software stack used on the wall.

3.5.5 Outreach and Education

Omegalib and Python were used to teach the basics of videogame programming to 5 students (aged 10-15) during volunteer mentoring for the Spark program[1] (Figure 22). In particular, we used the Mission Control protocol to interface a game running in CAVE2 with an Arduino controller, allowing students to create a tangible game controller interface.

[1] Spark is a mentorship program targeted to 7th to 9th graders, designed to help students in underserved communities succeed in high school and college. The program connects students to mentors in various workspaces based on their skills and interests.

A similar session was offered to a group of 14 middle school students during an interactive session organized by Northwestern University's Center for Talent Development. Omegalib has also been used at EVL as the project platform for CS528 (Virtual Reality) and CS526 (Scientific Visualization), taught by this Author's Advisor.

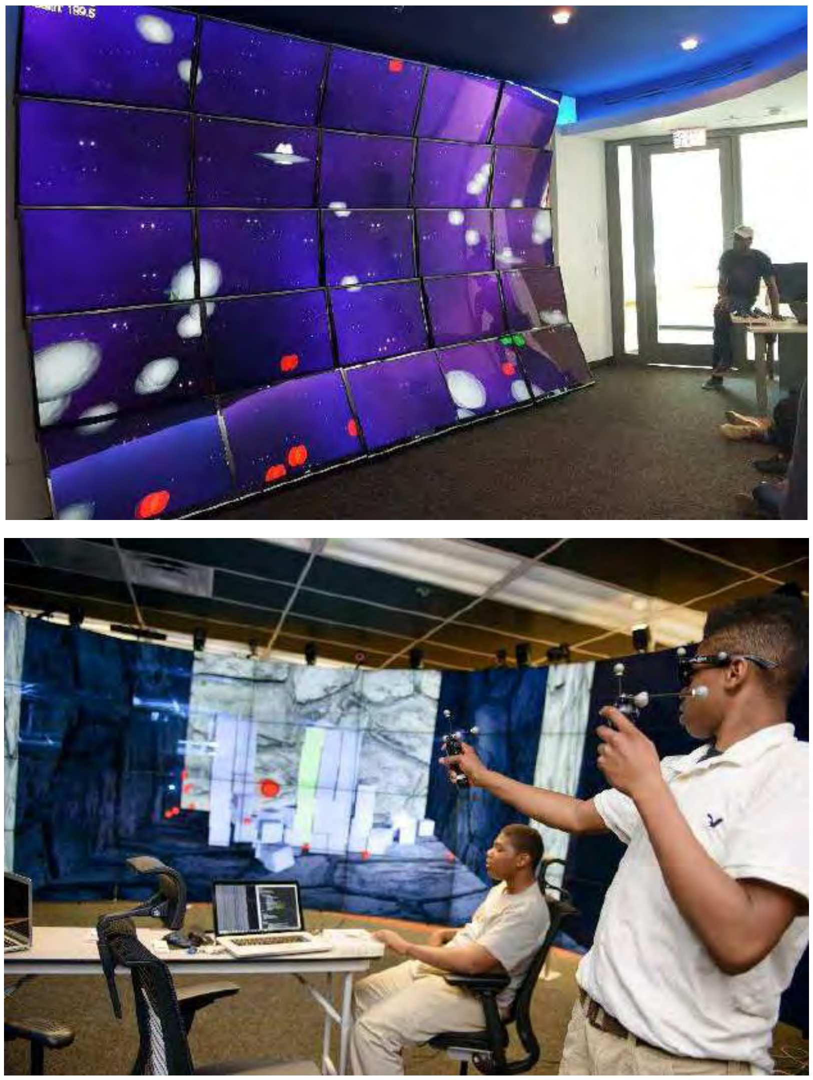


Figure 22. Omegalib used for outreach and education activities. Game design class for the center for talent development class at Northwestern (top), Spark at UIC (bottom).

3.5.6 Other Contributions and Project Status

At the time of writing, Omegalib is used as application development platform for several display installations worldwide. Teams have used Omegalib to create applications for other CAVE2 systems, display walls, projection-based displays and classical CAVEs. Third party contribution to Omegalib include support for additional display systems and stereo modes, integrations with other visualization tools like Houdini [90] and LavaVu [91] and new streaming encoders [92].

The entire project including core and optional modules is open source and hosted on GitHub [93], [94]. An online wiki offers hosts a user guide for display setup and application development, including over 100 pages of reference material. To assess project usage outside of EVL, we have been collecting the following statistics from GitHub since August of 2015:

1. Monthly downloads of OSX and Windows binaries: 65, source code: 35
2. Monthly unique IPs accessing downloads: binaries: 20, source code 30
3. Monthly wiki visitors (unique by day): 140
4. Total modules: 47, with 20 under active development (new commits since January 2016)
5. Core, Module and Patch Authors: 20

In August 2016, we surveyed current Omegalib users, including eight institutions (Monash University, University of Technology Sydney, Brown University, UIC, the University of Hawaii at Manoa, University of Maryland Baltimore County, and Northwestern University). The Omegalib features rated most useful were the dual API (C++ and Python) and the modular rendering capabilities. Desirable improvements included streaming, more support for consumer HMDs, and improved documentation/tutorials.

3.6 **Performance Evaluation**

As described in section 4.2.4, Omegalib supports static and dynamic workspace configurations. In fully dynamic setups, application viewports can be customized at runtime, to occupy any non-overlapping portion of the display. The tradeoff of dynamic workspaces is the need to replicate and synchronize multiple running applications across all nodes running a tiled display.

The working hypothesis is the following: for typical immersive application workloads, a dynamic workspace setup (i.e. replicating applications on all nodes) consumes additional system resources but has a small effect on application performance. I justify this hypothesis based on the following assumptions: 1) like most modern graphic frameworks, Omegalib typically operates a few high priority non-graphics threads. High priority non-graphics threads include scene database update, logic/physics and network processing. 2) mid-tier machines used in current generation cluster displays have 8-32 CPU cores, enough to allocate multiple aforementioned workloads on disjoint cores. 3) Baseline network usage for replicated immersive applications mostly consists of input, synchronization and low frequency geometry transfers. Average per-frame usage is typically 1-10Mbps, orders of magnitude less than consumer-grade wired network connectivity available today.

Based on these assumptions, I identify the graphical processing unit (GPU) as the most likely bottleneck for Omegalib applications running on a modern cluster-based HRE. Omegalib allocates the GPU (and graphics-processing CPU threads) to a single application on each node, depending on the workspace associated to that node’s tiles. Therefore, I expect applications to achieve similar frame rates, regardless of workspace configuration (static/dynamic).

To determine the performance effect of a dynamic workspace setup, I evaluated the system and application behavior of an experimental setup running three workspaces with distinct applications, in static and dynamic workspace mode. The experiment was carried out on the

Name	**Bottlenecks**	**GPU use**	**Threads**
ENDURANCE	GPU	High	3-6
Video Player	CPU, NET	Low	3-32
Physics Demo	CPU, GPU	Medium	3-7

Table VI. Summary of applications used in the performance evaluation experiment. The bottlenecks column indicates system components most likely to limit application frame rate. The GPU use column qualitatively indicates how much aggregate GPU resources (processing, memory, and bandwidth) are used by the application. The threads column indicates the typical amount of CPU threads used by the application.

computer cluster running the CAVE2 system. This cluster consists of 36 machines with 16 2.9GHz Xeon E5 cores, 64GB memory, 2TB local storage and 20 GB/s connectivity between nodes and storage server, and NVIDIA GTX 680 graphics with 2GB dedicated memory. Each machine (and graphics card) drives two display tiles with 1366x736 pixels of resolution each. The choice of a three-workspace setup was motivated by empirical observations of the usage pattern of the CAVE2 cylindrical display area. In typical research sessions, observed the display being typically split into one, two (left, right) or three (left, center, right) work areas, each one dedicated to a specific task. This observation will also drive the user study design in Chapter 5. In the experiment, the most demanding configuration was replicated, with three applications running on a 6x4-tile section each. Around 30 Omegalib applications and examples were available at the time of writing: I chose three that would stress different system components (Table VI).

The ENDURANCE application is a point cloud data visualization tool supporting the real-time display and query of large point cloud datasets (5-200M datapoints). The application makes heavy use of vertex, geometry and pixel shaders to efficiently filter and rasterize points as shaded spheres. The application frame rate depends on the number of point batches within the view frustum.

The Video Player application plays back pre-rendered stereo videos with arbitrary frame sizes (typically using the same resolution as the tiled display, ~75MPixels in CAVE2). Each player slave process loads a dedicated frame stream as jpeg images from networked or local storage, and uses several worker threads to decode images fast enough to support 60fps playback.

The Physics Demo application demonstrates Bullet physics integration with Omegalib and displays a spinning hollow cube containing several hundred box shaped rigid bodies (300 in our experiment). The spinning cube is used to ensure all rigid bodies remain active throughout the simulation, keeping a constant load on the physics engine.

Applications were scripted to execute a repeatable experiment in the dynamic and static workspace conditions: Mission Control messaging was used to inject script commands into the three applications at scheduled times, ensuring a consistent collection of application and system performance data.

For the static workspace condition, each application ran on a fixed portion of the CAVE2 display. In the dynamic workspace condition, applications were all started on the entire display, and after startup were configured to use the same display section as in the static workspace condition. Visually, both experiments rendered the same content on the CAVE2 display, but in the dynamic workspace condition each display cluster node was running a replica of each application, two in update-only mode and one in update and display mode.

During experimental runs, application frame rates were logged at 1-second intervals, and system network, CPU, GPU and memory usage on all 36 CAVE2 nodes were recorded, again at 1-second intervals. System performance data was aggregated by display section, using the left-center-right tile allocation described previously.

Figure 24 summarizes the results of the aforementioned experimental runs. As predicted, the framerate trends for all three application are similar for static and dynamic workspace setups. Average frame rates are between 1 and 3 fps less for a fully dynamic workspace setup, an acceptable tradeoff for the flexibility of runtime workspace resizing and reallocation. System performance data also matches our expectations. For instance, network usage in the dynamic workspace condition is higher on all three display sections, since the biggest user of network resources (the Video Player application) is now running on the entire display. A similar trend can be observed for CPU usage. It is worth noticing that it would be trivial to optimize network usage, since each application instance is aware of its display status. A “smarter” video player implementation could stop frame buffering on inactive display nodes, bringing network usage to very similar levels to the static workspace condition. I did not introduce this optimization in our experiment, since I was interested in evaluating the behavior of applications unaware of dynamic workspace optimizations, to determine how much the Omegalib runtime could do without requiring application modification.

These performance observations confirm the feasibility of multiview techniques using the Omegalib default execution model (replicated application logic with on-demand rendering). The multiview immersion technique I will present in the next chapter does partially depend on this execution model (see section 4.2.1, dynamic application viewports). This is the reason I considered the performance evaluation above necessary, even if its result were expected and not especially surprising. I verified the technological foundation for multiview immersion is sound. Chapters 5 and 6 will assess its usefulness.

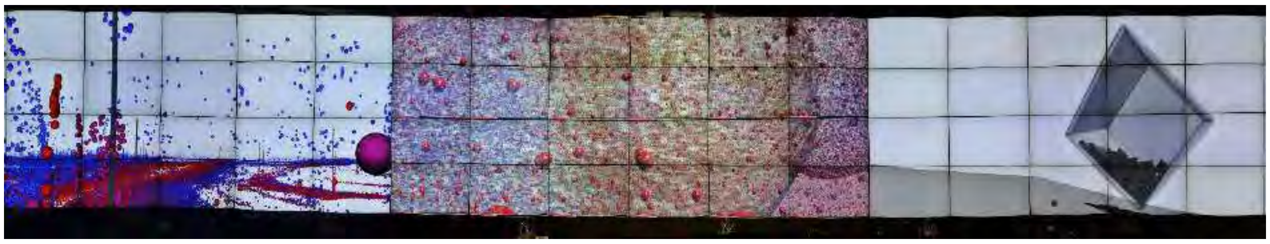


Figure 23. An unwrapped view of the cylindrical CAVE2 display running the workspace setup used for evaluation. The three applications are, from left to right, ENDURANCE, Video Player, and Physics Demo.

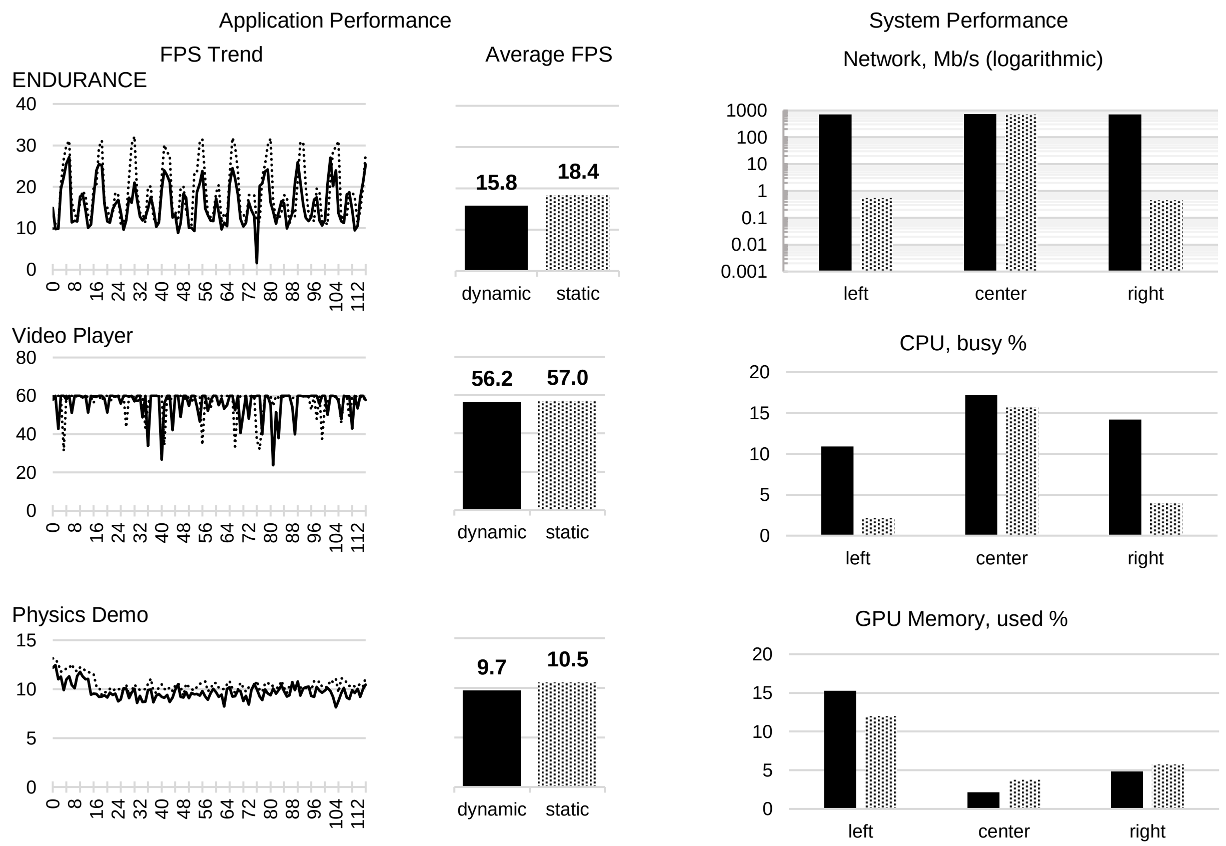


Figure 24. Application and system performance results from the multiview evaluation experiment. Charts on the left show frames-per-second (fps) trends over ~2 minutes, and fps averages for the three test applications, in the dynamic and static workspace setups. Charts on the right summarize cluster system performance for three components (Network, CPU, GPU) for distinct areas of the display (left, center, right) corresponding to the active workspace areas of the three test applications.

3.7 **Summary**

In this chapter, I described the implementation of Omegalib including its core components, module support and dynamic reconfigurability features. I presented uses cases of the framework showing examples of how Omegalib is used on several large-scale display systems in both national and international laboratories. One of these use cases, ENDURANCE, focused on how a research group collaborating within CAVE2 alternated between different display configurations, including both immersive and non-immersive views. The experience with the ENDURANCE team allowed me to identify several potential areas of improvement of the current HRE OS implementation. Two major observations will drive the improvements described in the next chapter. First, in a research team where expertise is distributed across participants, different researchers will lead different parts of the collaboration. It's therefore desirable to allow people to easily take control of any application running on the system, without requiring the exchange of physical input devices. Second, as users rearrange views, they tend to do so in predefined ways based both on the task at hand and the display geometry. For instance, splitting the display in half was quite common, as was having a central view share the screen with two side views offering ancillary information. We want to make sure both these features (distributed control, view management) are as intuitive as possible in our system, while working as expected with fully immersive content. In the next chapter, I will present an improved version of the HRE OS addressing these requirements. As a whole, I will refer to these improvements as *Multiview Immersion*.

CHAPTER 4

MULTIVIEW IMMERSION

4.1 Requirements

In Chapter 3, I observed how both classic immersive application frameworks and display wall software environments have desirable features for a hybrid environment software infrastructure. In particular, tiled display window managers like SAGE and SAGE2 have the ability to freely layout (and resize) application viewports and support direct interaction from multiple users. Another major advantage is application frame rate decoupling: since applications run separately from the compositor software, they do not influence each other's responsiveness.

Omegalib supports dynamic reconfiguration of views, as we have seen in Chapter 3, but compared to tiled display window managers it does so with several limitations. Resizing and view movement is limited to tile boundaries, and does not adjust frustum, view or navigation as a function of view position. These adjustments are not needed for non-immersive content. However, immersive content needs to be rendered according to several additional parameters, such as the geometry of the projection surface and its relation to the user's head and any tracked interaction device. In classical virtual reality software supporting a single, static viewport these properties are commonly hard-coded in a configuration file. Correct dynamic viewer centered immersion requires all these parameters to be dynamic. The rendering system must receive information about the physical position of the 2D viewport on the display surface in order to recompute the off-axis projection frustum. Furthermore, if the viewport is resized then the renderer also needs to re-adjust the eye separation factor accordingly.

In the Omegalib implementation, this requirement can be tackled with additional information flow from the application controller to the runtime. In detail, the controller needs to communicate the position of a 3D window in screen coordinates. The application runtime has knowledge of the display geometry, and can convert the 2D window corners into real-world 3D points. These points can then be used in conjunction with the eye position to generate a new frustum for the off-axis stereo projection.

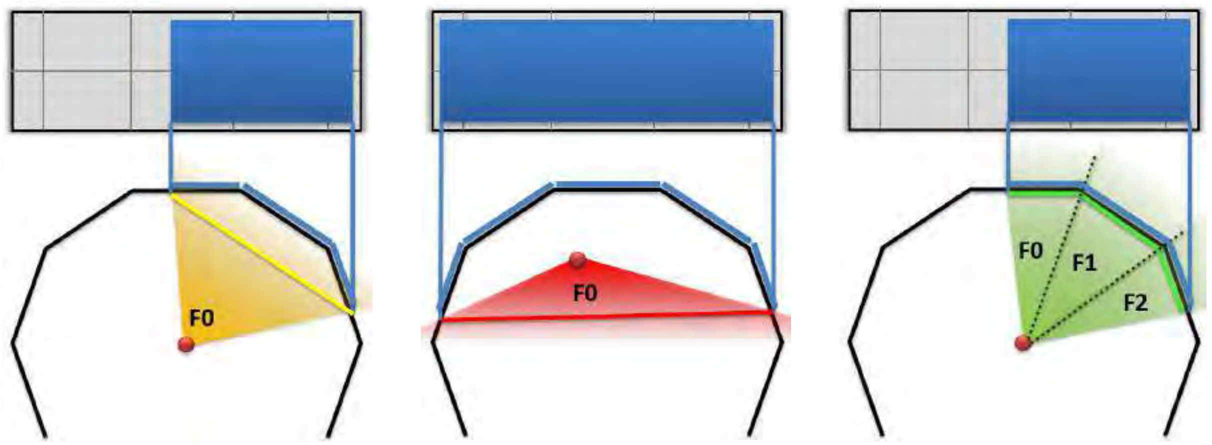


Figure 25. Examples of correct and incorrect frustum generation for movable stereo windows. On the left, a projection frustum generated by the window corners (yellow) does not match the actual display surface (blue segments). The resulting projection can go from slightly warped to completely incorrect as shown in the center example, where the combination of head position and window size leads to a frustum that is inverted w.r.t. the physical display surface. On the right, an example of correctly generated frustum set.

It should be noted that generating a new off-axis frustum for a dynamic viewport is not trivial, if one makes no assumptions on the screen geometry. As mentioned in the previous paragraph, an intuitive way to create this frustum is taking the corners of the new viewport, projecting them on the display and using those points to determine the frustum plane. Although this simple technique works well for planar displays (like classic display walls), it leads to incorrect projection results on arbitrary display surfaces, like the CAVE2 cylindrical display or other displays with non-planar form factors such as the CALIT2 Wave [36] or dome displays. Figure 25 (left and center) illustrates the issue: on non-planar displays, a single frustum generated using the window 3D corners will not match the actual display projection surface. To address this issue, we need to generate a set of *per-tile frusta*, each one corresponding to one of the physical display tiles, as shown in Figure 8 (right). The implementation details of viewport movement and frustum generation will be presented in sections 4.2.1 and 4.2.2.

Another important feature of the HRE operating system is the ability of multiple users to interact with the system, using a variety of input devices depending on task requirements. Furthermore, users should be able to seamlessly switch between application interaction (i.e. navigating within a view), window interaction (i.e. re-arranging or resizing windows) and system interaction (i.e. starting and stopping applications).

In the case of wands, it is desirable to let users perform 2D and 3D interaction using the same device, and have multiple users in the system control different windows. For this feature to work, it is important to create an association between a wand and a specific window, in a sense letting a specific application 'acquire' input from a specific input device. Subsequent interaction does not depend on the user continuously pointing towards the application viewport, which may be intuitive or unpractical for tasks like navigation. This is also similar to the traditional 'active window' desktop metaphor, where the last window that has been clicked on by the user is the one with input focus, even after the mouse pointer leaves the window area. A key difference in the HRE context is that multiple interaction devices may be present, and each application should support association with any number of devices. Details of 2D/3D input support are discussed in section 4.2.4.

4.2 **Implementation**

The implementation of multiview immersion is divided into four runtime functionalities, whose implementation touches different components of the HRE OS model as illustrated in Figure 26:

1. *Dynamic viewports* that can be moved and resized at the pixel level on a tiled display. The viewport boundaries determine the parameters of off-axis projection frusta for each display tile. This ensures the stereoscopic projection stays consistent as the viewport changes.
2. *Transformation adjustments* based on viewport and display geometry. These adjustments reconcile the immersive viewport reference frame with the tracked user(s) reference.
3. *Navigation adjustments* that keep 3D navigation consistent when using tracked input devices like wands. They take the corrections computed by transformation adjustments and apply them to navigation transforms.
4. *Hybrid interaction and routing* of multiuser input / tracking data to applications. Hybrid interaction allows a single physical device to be used both as a 3D and 2D interaction controller based on context. Routing distributed the input data from multiple devices to views based on user/view/device association.

The following sections will describe the implementation of each of these functionalities. While most aspects of the implementation are generic, specific parts require explicit knowledge of the approximate display geometry (planar, cylindrical, etc.). The implementation is designed to allow for these geometry-specific functions to interface with the rest of the system through the use of *display configuration builders*. In the next sections, I will show examples of geometry-specific functions, giving a complete description of formulas for the CAVE2 cylindrical display geometry. I will also outline how formulas could be implemented for other display geometries.

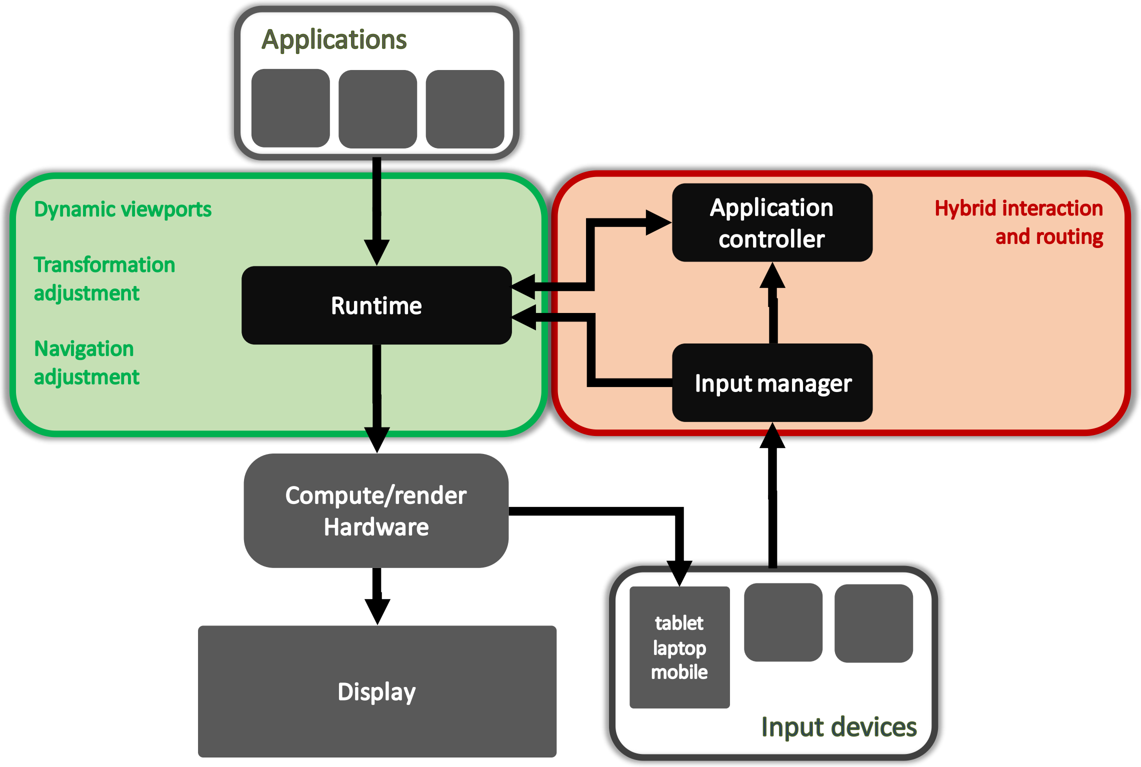


Figure 26. A review of the HRE OS model, showing where the multiview immersion features described in this chapter are placed relative to the HRE OS components. All features except for hybrid interaction and routing are implemented in the application runtime.

4.2.1 Dynamic viewports

We already discussed how Omegalib supports launching multiple applications in Chapter 3, and how applications can be assigned an area of the display either at startup or dynamically. However, application viewport movement was limited to tile granularity: viewports could only be moved and resized along tile boundaries (with tiles possibly spanning multiple displays)

To fully support multiview immersion, I extended Omegalib to support the arbitrary movement of windows. I considered leveraging an external distributed windowing system such as SAGE / SAGE2 to control the actual window manipulation, with Omegalib only responsible of generating and sending the pixel streams to the appropriate endpoints. While this solution is viable, I chose to use the OS windowing system directly to implement window movement. Using the OS windowing system allows Omegalib applications to be overlaid on top of any other distributed window manager, together with other immersive frameworks, or even without a display management system at all, since all that's needed is the host operating system windowing functionality.

To implement dynamic viewports on top of OS windowing support, Omegalib takes into account where an application viewport is on the global display space (that is, set of tiles that make up the display system), and it adjusts windows on each tile accordingly. For the remainder of this section, and to keep terminology consistent with the implementation, the application viewport will be referred to as the *canvas*. The region of pixels covered by the canvas on the global display space is called the *canvas rect*. The intersection between a tile's pixels and the canvas rect is the area on that tile that our application should render to. I refer to this area as the *active tile rect* to indicate that this is the only space on a tile that our application will output to. The active tile rect can therefore can be used as the position and size of the operating system window on each tile.

When the user moves or resizes the application canvas, the runtime recomputes the intersection between the canvas and the tiles, to generate a new set of active tile rects. The difference between the new and old active tile rects is then used to generate a set of commands to control the operating system windows on the display cluster. The commands for each tile are generated as follows:

1. If the new and old tile rects are the same, do not modify the current window.
2. If the new tile rect has a different position, size or both, move or resize the current window accordingly.
3. If the new tile rect is empty and the old one wasn’t, hide the current window and disable rendering on this tile’s node.
4. If the new tile rect is not empty while the old one was, show a window on this tile, set its position and size to the new tile rect and enable rendering on this tile’s node.

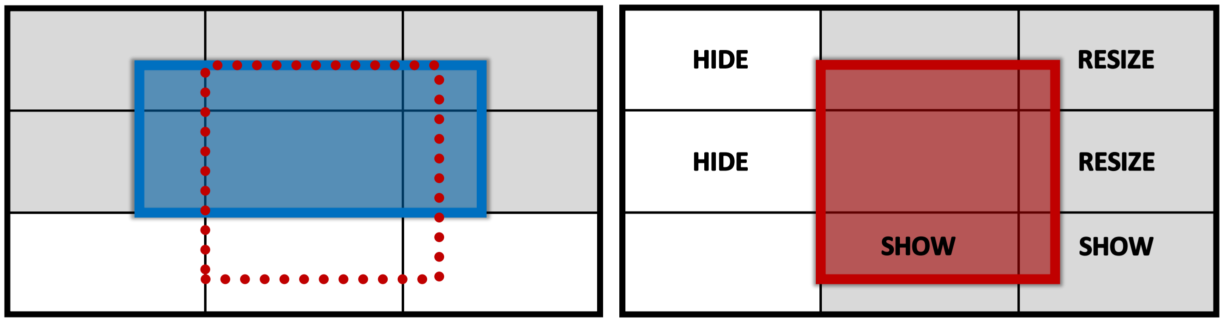


Figure 27. An example of canvas movement and relative commands issued on each tile. On the left, the original canvas rect is in blue. On the right, the new canvas rect is shown. The center and center-top tiles do not have any commands issues to them since the local shape of the canvas rect did not change.

An example of window movement and the generated windowing commands is illustrated in Figure 27. In the Omegalib implementation, the tile intersection and command generation process is not centralized. Each node is notified of an application canvas change, and takes care of generating the list of windowing commands needed for its local tiles. This is not done for performance reason as much as practicality: having each node control local tile windows results in a much simpler implementation of messaging as it limits the data exchanged between the master and display nodes to the application canvas rect.

Another piece of information computed together with the active tile rect is the *active canvas rect.* The active canvas rect contains the same information as the active rect but in canvas coordinates. In other words, this is the area covered by a tile's active rect within the current canvas. The active canvas rect is used to compute the offset of any 2D graphics drawn by the application. It also makes it possible to optimize 2D drawing. If a 2D object's area does not intersect with a tile's active canvas rect, it won't be visible on that tile and can be skipped by the tile's rendering process.

After an application canvas has been modified, we need to make sure the view rendered within this canvas is also adjusted to take into account the new canvas size and position. As explained in section 6.1, a fully immersive viewport behaves differently than a standard 2D 'window': when the viewport moves the content within won't move with it, as the viewport is just a portal into the VR world. While this behavior is correct if we look at it from a strict definition of immersion, this is rarely what's expected or desired by users. If a user is observing an object in the center of a window, and moves the window to another position on the screen, they expect the object to still be centered in the window's new position. We therefore need to implement this capability for the immersive applications supported by Omegalib.

Implementing a fully immersive application canvas depends on two complementary features: transformation adjustment and navigation adjustment.

4.2.2 Transformation adjustment

The transformations used to render the VR scene need to be extended. Mathematically, these transformations are represented by 3D affine matrices. These matrices are a function of the observer navigational position and orientation (usually called the camera transform), the viewport frustum, and in the case of immersive graphics, the observer's eye position relative to the physical position of the viewport. This information is used to compute two transforms: the *view transform*, which converts the positions, orientations and scales of objects from the VR scene into the camera frame of reference; and the *projection transform* which converts the objects' coordinates into the projective space used by the viewport. The projection transform used in immersive graphics is different than the one used by non-immersive 3D graphics as it also takes into account the observer's eye position to generate an off-axis projection. To support a dynamic application canvas, the projection transform computation was modified to account for the fact that, on each tile, the physical corners of the frustum did not necessarily correspond to the corners of the tile. the *active tile rect* discussed in 6.2.1 is used to adjust the frustum corners for a tile. Given a tile's resolution $\mathbf{r}\in\mathbb{N}^2$ in pixels, a tile's top, bottom, left corners ($\mathbf{t}\in\mathbb{R}^3$, $\mathbf{b}\in\mathbb{R}^3$, $\mathbf{l}\in\mathbb{R}^3$) in real-world coordinates and an active rect's position and size $\mathbf{p}\in\mathbb{N}^2$ $\mathbf{s}\in\mathbb{N}^2$ in pixels, the local frustum corners ($\mathbf{t'}\in\mathbb{R}^3$, $\mathbf{b'}\in\mathbb{R}^3$, $\mathbf{l'}\in\mathbb{R}^3$) are:

$$\boldsymbol{l}' = \boldsymbol{l} + (\boldsymbol{b} - \boldsymbol{l})\alpha + (\boldsymbol{t} - \boldsymbol{l})\beta$$

$$\boldsymbol{t}' = \boldsymbol{l} + (\boldsymbol{t} - \boldsymbol{l})\gamma$$

$$\boldsymbol{b}' = \boldsymbol{l} + (\boldsymbol{b} - \boldsymbol{l})\delta$$

Where $\alpha, \beta, \gamma, \delta$ are frustum coefficients defined as follows:

$$\alpha = \frac{p_x}{r_x}$$

$$\beta = 1 - \frac{p_y + s_y}{r_y}$$

$$\gamma = 1 - \frac{p_y}{r_y}$$

$$\delta = \frac{p_x + s_x}{r_x}$$

This new frustum is used to compute a per-tile off-axis projection matrix that accounts for the position and size of the application canvas.

To adjust the view transform, we consider the movement of the application canvas in the real-world frame of reference, and use this information to compute an inverse transform that, when applied to the standard camera transform, makes the VR scene ‘follow’ the canvas movement. It’s important to underscore that there isn’t a unique way of ‘following the canvas movement’. For instance, on a cylindrical display such as CAVE2 we probably just want to rotate the camera to adjust for canvas movement, while on a planar display we might want to translate it. If the canvas

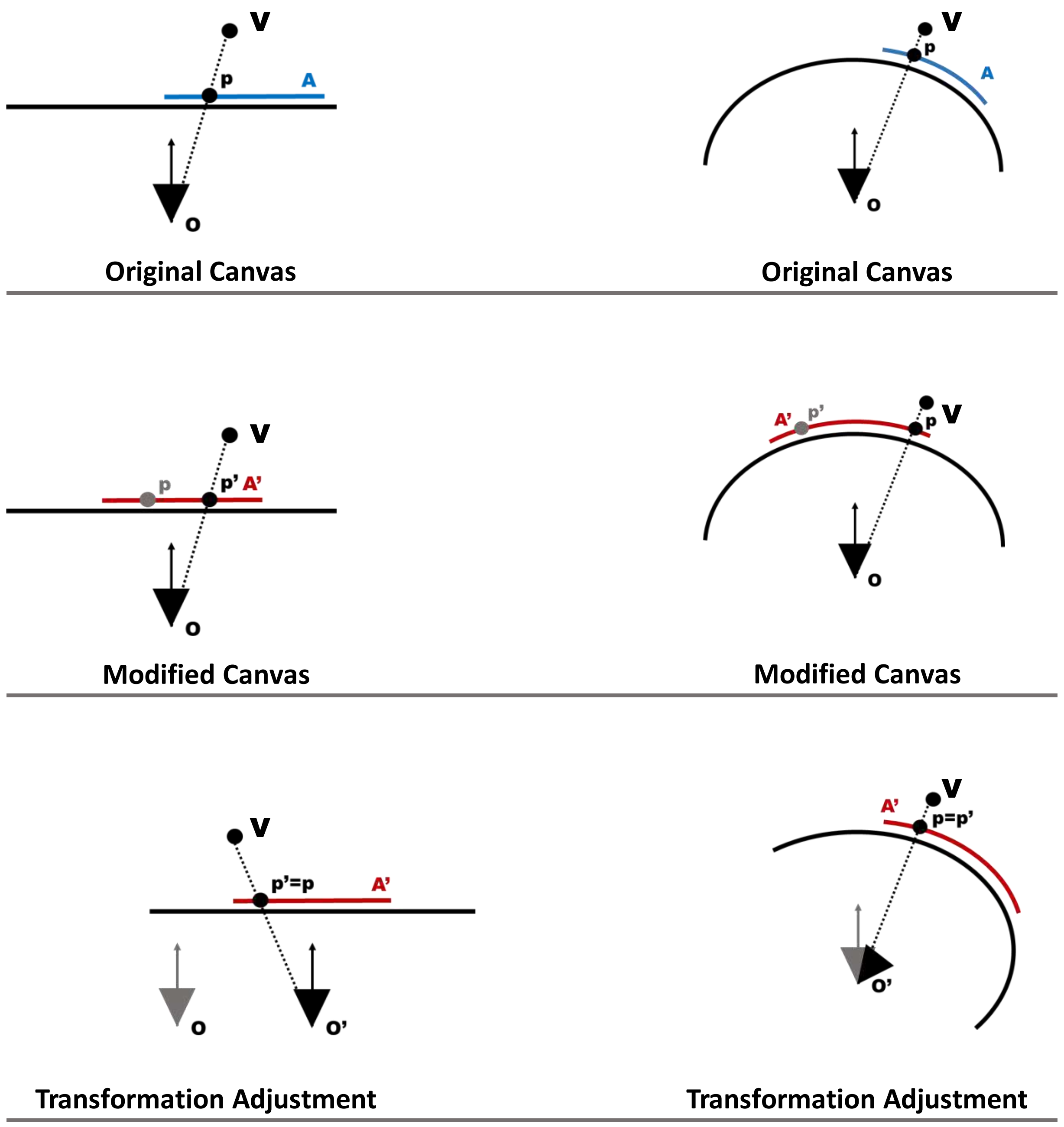


Figure 28. Example of view adjustment for planar and cylindrical displays, illustrated in 2D for simplicity. The top pictures show the initial canvas position **A** in blue, and a projected point **p** on them, based on the position of a VR point **V** and the camera reference frame **O** (black triangle). The middle picture shows the modified canvas **A'** in red: the original projected point and new one **p'** are now different.In the bottom picture, a transformation adjustment is applied to the camera so the new reference frame **O'** re-aligns the old and new projection points.

gets resized, we might want to rescale the VR scene to keep the same objects in the view, or we might choose to ignore scaling corrections in favor of a better immersive experience.

View transform adjustments represent a tradeoff between a fully immersive experience and intuitive manipulation of a two-dimensional canvas. For this reason, the Omegalib implementation provides the necessary support for view transform adjustment, without enforcing a specific kind of view adjustment. The Omegalib display configuration object (which contains the full description of the display geometry) has been extended to include a *canvas transform*, represented as a canvas offset vector $\boldsymbol{o} \epsilon \mathbb{R}^3$ and canvas orientation quaternion $\boldsymbol{q} \epsilon \mathbb{R}^4$. This transform is combined with the camera navigational position and orientation $\boldsymbol{p} \epsilon \mathbb{R}^3$, $\boldsymbol{r} \epsilon \mathbb{R}^4$ to compute a *canvas-adjusted view matrix* $\boldsymbol{V} \in \mathbb{R}^{\boldsymbol{4x4}}$ as follows:

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \vec{\boldsymbol{0}} & 1 \end{bmatrix}$$

Where:

$$\vec{\boldsymbol{0}} = (0,0,0)$$

$$\boldsymbol{R} = \mathcal{M}(\boldsymbol{rq})^T$$

$$\boldsymbol{t} = -\boldsymbol{R}(\boldsymbol{p} + \boldsymbol{o})$$

and the function $\mathcal{M}: \mathbb{R}^4 \rightarrow \mathbb{R}^{3x3}$ converts a quaternion to a 3x3 rotation matrix. When the canvas rect changes (for instance due to the user moving the application on the display) Omegalib generates a canvas change event. This event can be processed by Omegalib modules, user scripts or display configuration builders to update the canvas offset vector and orientation. For instance, the cylindrical display configuration builder used by CAVE2 systems handles the canvas change event and updates the canvas orientation. The new canvas orientation rotates the camera in the direction opposite to the rotation of the canvas center. This ensures the VR scene view follows the position of the application canvas on screen.

CAVE2 canvas transformation

The detailed calculation of the canvas orientation for a CAVE2 cylindrical configuration is the following. Assume we have a function $\mathcal{P}: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ that converts a pixel position into real world 3D coordinates. For the sake of brevity I won't define that function here, but different implementations are possible, either using an approximate geometric description of the display or by using the actual tile positions (Omegalib uses the latter strategy). Given the application canvas position $\mathbf{p}$ and size $\mathbf{s}$ as defined previously, the real-world position of the canvas center is:

$$\boldsymbol{c} = \mathcal{P}(\boldsymbol{p} + \frac{\boldsymbol{s}}{2})$$

I assume the neutral direction for the system (that is, the original orientation of a view, that would require no orientation adjustment) is the negative Z axis. This corresponds to the commonly used reference frame used by computer graphics. The angle between the neutral center and the new canvas center is:

$$\alpha = \operatorname{acos}(\frac{-\boldsymbol{c}_z}{\sqrt{\boldsymbol{c}_x^2 + \boldsymbol{c}_z^2}})$$

This angle is the rotation around the Y axis that converts the center of a canvas in the neutral orientation (the negative Z axis) to the center **p** of our current canvas. This orientation can finally be converted to the canvas orientation quaternion **q:**

$$\boldsymbol{q} =< imag: \left[0, \sin\left(\frac{\alpha}{2}\right), 0\right], real: \cos(\frac{\alpha}{2}) \ >$$

It should be noted again that this calculation of a canvas transform is just one of many options, even on a cylindrical display. Omegalib users can override this implementation with their own version. An Omegalib module can offer a library of display configuration builders with canvas transform presets for various display geometries. It is also possible to disable the view adjustment at runtime, simply by setting the canvas offset to zero and canvas orientation to identity on canvas rect changes. This will switch the canvas back into ‘fully immersive mode’ making the VR view float with the viewport.

Name	Description	Type	Context
Canvas Rect	The position of this application's viewport on the display system	2D Rect in global display system pixel coordinates	Global
Active Tile Rect	The area of each tile ***covered*** by the canvas rect	2D Rect in tile pixel coordinates	Tile
Active Canvas Rect	The area of each tile ***relative*** to the canvas rect	Rect in canvas pixel coordinates	Tile
Canvas Transform	The adjustment of the VR reference frame to account for canvas rect change	Quaternion + 3D position	Global
Camera Transform	The camera position and orientation in the VR reference frame	Quaternion + 3D position	Global
View/Projection Transforms	The transformation from VR coordinates to local display coordinates (normalized)	4x4 Matrix	Tile

Table VII. A review of the data elements involved in the implementation of Multiview Immersion. The table lists the format and purpose of each element, and whether the element's context is global (i.e. one per-application) or tile (i.e. the element has a different value for each tile used by the application).

4.2.3 Navigation adjustment

User navigation (that is, camera movement) needs to be adjusted as well to account for canvas changes. Navigation adjustment poses similar challenges to the view corrections discussed in 4.2.2: there is no unique solution to this: possible solutions depend on the changes we applied to the view transforms, and on the type of controller users are operating to input navigation commands. Consider a tracked 6DOF wand controller. The real-world tracked movement of the wand is used to determine the direction and speed of movement in the VR world. This navigation technique assumes that the real world and VR scene reference frames are the same. If we re-orient the VR view to follow the application canvas, that assumption is broken: for instance, a user moving the controller ‘forward’ relative to the viewport would not be navigating in the expected direction. On the other hand, if we use an untracked controller such as a console gamepad, the direction the controller is pointing to becomes irrelevant: as is common with these devices, pushing the forward button or analog stick on the controller always means ‘move forward’ regardless of the orientation of the controller.

This challenge can be addressed in a manner similar to the one discussed in 4.2.2: Omegalib does not enforce a specific navigation adjustment technique. The Omegalib camera object (representing the navigational observer) can be extended with a Camera controller: an object in charge of processing user input from a specific device and turning it into navigation commands for the camera. The camera controller can, optionally, take into account the current canvas transform to adjust the navigation commands. For instance, the tracked wand camera controller uses the canvas orientation to adjust the navigation direction and solve the inconsistency problem we discussed above. Developers or additional camera controllers can similarly extend their controllers to use this information. Figure 29 summarizes the implementation of canvas, view and navigation corrections presented in 4.2.1, 4.2.2, 4.2.3.

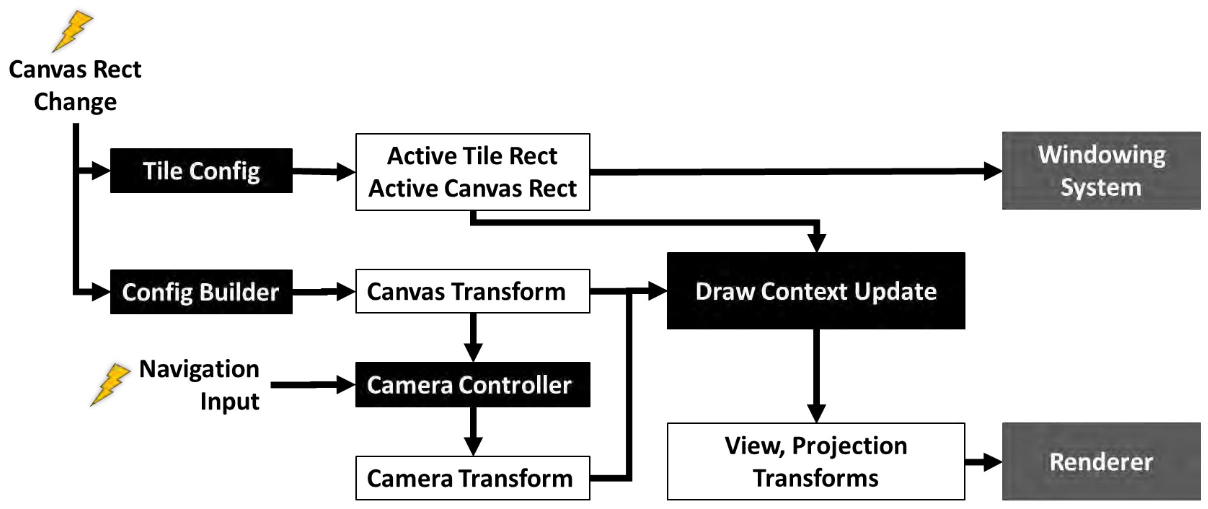


Figure 29. Multiview immersion inputs and transformations flow. The black boxes represent data transformation steps. The white boxes are data structures. Gray boxes represent the runtime subsystems consuming the data. Canvas change and navigation input are user-controlled events.

4.2.4 Hybrid interaction and routing

So far I detailed how application canvas changes can be used to adjust the immersive view rendered by that application. One final aspect to consider is 2D interaction. While most of the interaction with immersive views is going to involve the VR scene itself, we can expect most applications to have at least some form of 2D input required. This could be to control a classic 2D user interface overlaid on top of the 3D scene. We need some function that allows users to choose which application they are controlling, when multiple users are collaborating within the environment. Moreover, users need an interface to move and resize the application canvases on the display.

The classic way of controlling application viewports on a desktop computer is the standard window: users ‘click’ on a window using a 2D pointing device like a mouse or touchpad to focus on it. A focused window received the user input from other devices (like a keyboard) until the user selects another window. Windows be resized and moved using the pointing device. On newer operating systems, keyboard shortcuts allow windows to be arranged in commonly used layouts such as full-screen, left display half, right display half. Our objective is to offer similar functionality for immersive applications and multiple users.

To add support for multiuser 2D input, I modified Omicron, the input management library used by Omegalib. Omicron already supports 2D input device, but the objective was extending 3D input devices like tracked wands, to make them expose 2D and 3D interaction data simultaneously. For instance, a wand can be turned into a 2D pointer by intersecting a ray with the wand origin and orientation with the display geometry. This 2D pointer information is embedded in the input events generated by the wand: this way, client applications can consume wand event as either 3D inputs (for VR navigation) or 2D inputs (for user interface control).

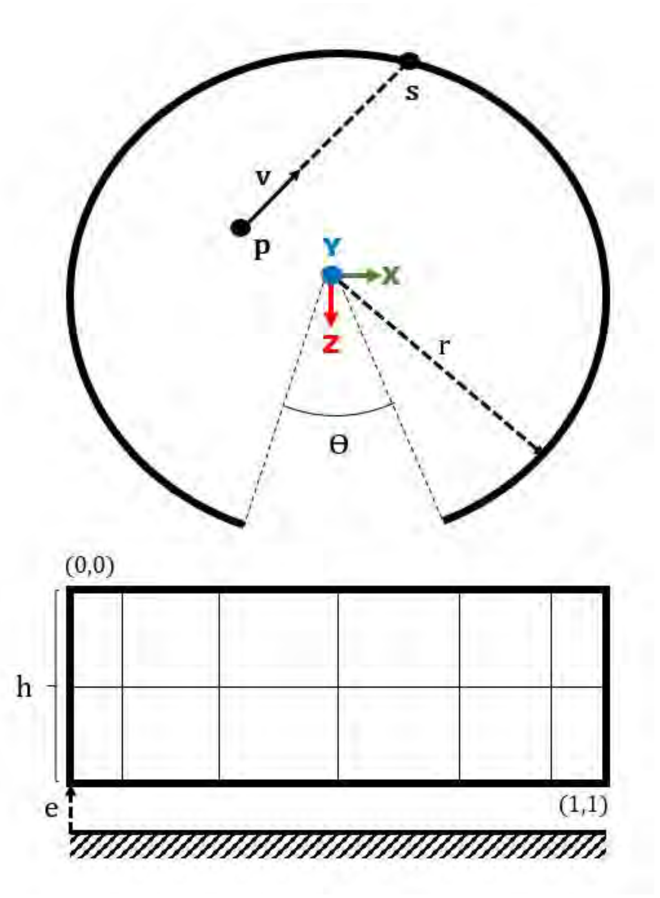


Figure 30. Ray-to-pointer conversion in the CAVE2 system.

CAVE2 ray-pointer conversion

As an example, let's consider a 3D ray to 2D pointer conversion for the CAVE2 system. The CAVE2 display can be approximated by a cylindrical segment, as represented in Figure 30. The cylindrical segment is incomplete and missing an arc angle θ, which corresponds to the CAVE2 entrance opening. Let us assume the cylinder axis is the y axis of our tracking system reference frame (the same reference used by the ray). The cylindrical segment offset from the ground plane is *e*. This value can be set to zero if the displays reach all the way to the floor. We want to compute an intersection ***s*** in normalized display coordinates, between the display and a ray with origin ***p*** and direction ***v.*** We calculate the ray direction on the ground plane *A,* ray coefficient *B* and normalized ray origin *C* as:

$$A = \boldsymbol{v}_x^2 + \boldsymbol{v}_z^2$$

$$B = 2(\boldsymbol{v}_x\boldsymbol{p}_x + \boldsymbol{v}_z\boldsymbol{p}_z)$$

$$C = \boldsymbol{p}_x^2 + \boldsymbol{p}_z^2 - r^2$$

The ray-display intersection in normalized world coordinates (1 = r, the cylinder radius) ***u*** is:

$$\boldsymbol{u} = \mathbf{v}\frac{-B + \sqrt{B^2 - 4AC}}{2A} + \boldsymbol{p}$$

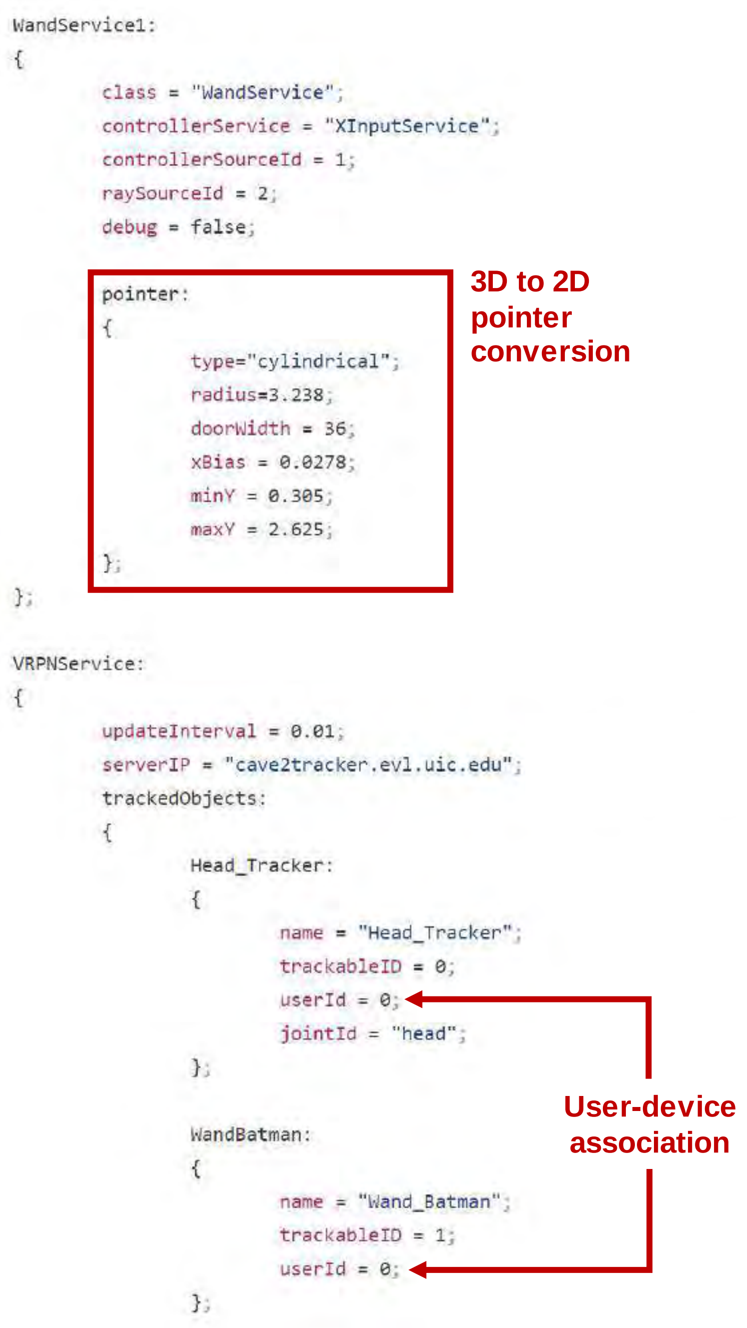
and the intersection in normalized display coordinates can be calculated as follows:

$$\boldsymbol{s} = \begin{bmatrix} \dfrac{-\ atan2(\boldsymbol{u}_x, \boldsymbol{u}_z) +\ \theta}{4\pi - \theta} \\ \dfrac{\boldsymbol{u}_y - e}{h} \end{bmatrix}$$

Another functionality missing from Omicron was user-device association. If two users are collaborating in CAVE2, each wearing a set of tracked stereo glassed and each using a wand, we need to know which wand is controlled by each user. For instance, if a user is interacting with a specific immersive view, we want that user's head tracking information to be connected to the view. All the devices owned by a specific user need to share the same user identifier.

To add this feature, the Omicron event definition was extended to include a user ID field. This field is set in the Omicron input configuration file, for each device associated with a particular user. To avoid breaking compatibility with existing Omicron clients, this new field is integrated in into the existing event protocol payload. The service id field, used to identify the logical input device generating the event was truncated from 32 to 16 bits. The additional 16 bits are dedicated to the user id field. Legacy code accessing the full 32-bit service id still gets a unique identifier for the device, made of the device tag (lower 16 bits) and associated user id (upper 16 bits). The new 16 bit fields still allow for 65'536 distinct user ids and input devices, more than enough for any expected usage scenario.

These new features can now be used to intuitively control immersive application windows. If users want to take turns controlling a particular application, they do not need to share a single wand and set of tracked glasses.



```
WandService1:
{
        class = "WandService";
        controllerService = "XInputService";
        controllerSourceId = 1;
        raySourceId = 2;
        debug = false;

        pointer:
        {
                type="cylindrical";
                radius=3.238;
                doorWidth = 36;
                xBias = 0.0278;
                minY = 0.305;
                maxY = 2.625;
        };
};

VRPNService:
{
        updateInterval = 0.01;
        serverIP = "cave2tracker.evl.uic.edu";
        trackedObjects:
        {
                Head_Tracker:
                {
                        name = "Head_Tracker";
                        trackableID = 0;
                        userId = 0;
                        jointId = "head";
                };

                WandBatman:
                {
                        name = "Wand_Batman";
                        trackableID = 1;
                        userId = 0;
                };
```

Figure 31. Excerpt from an Omicron input configuration, showing the newly added features that support input for multiview immersion. The 3D to 2D pointer conversion section shows an example of converting data from a tracked device into a 2D pointer, using information about the display geometry. The user-device association field attached a unique user id to a set of tracked glasses and a wand, so they can be handled consistently by client applications

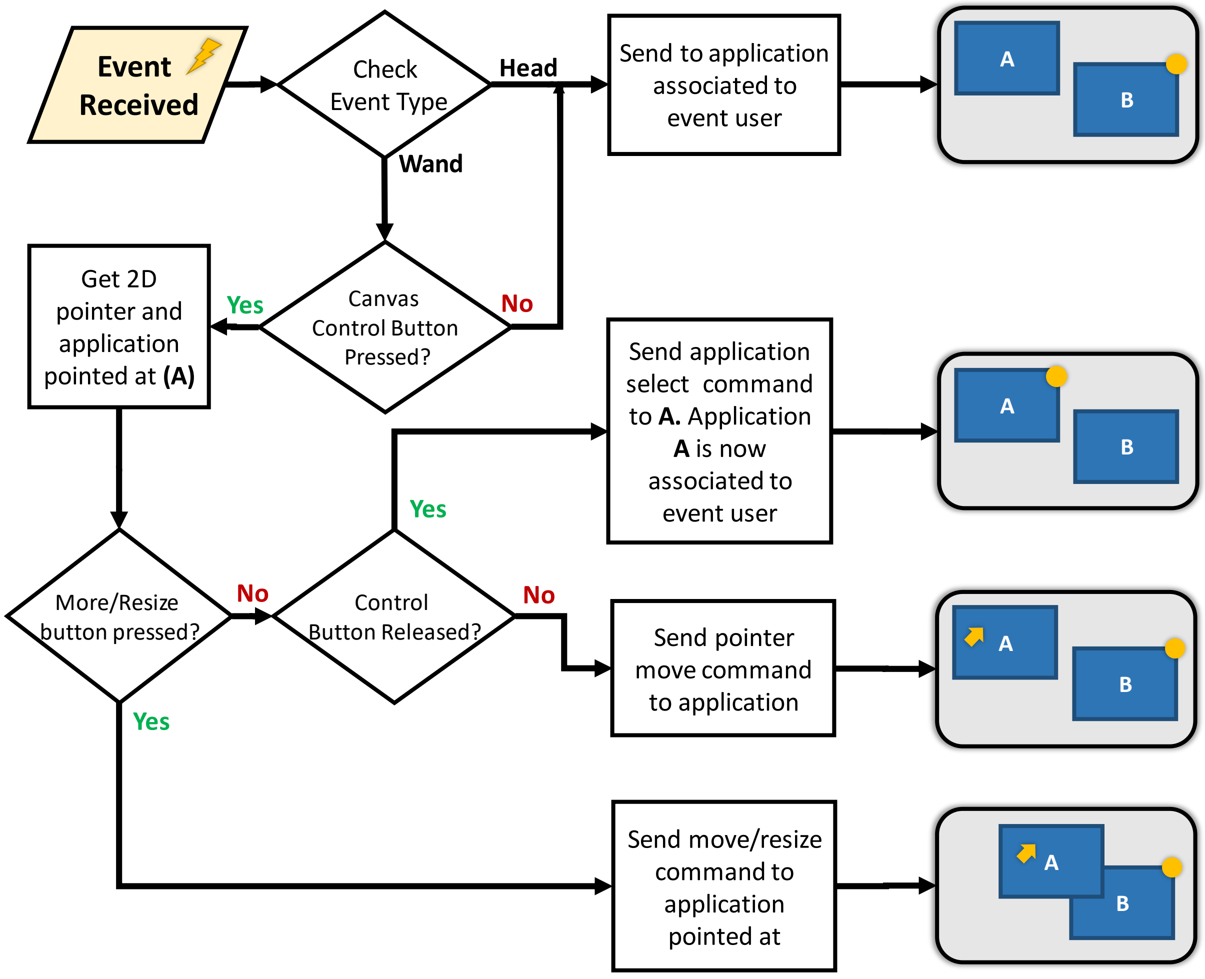

Event Received
Check Event Type
Head
Send to application associated to event user
A
B
Wand
Get 2D pointer and application pointed at (A)
Yes
Canvas Control Button Pressed?
No
Send application select command to A. Application A is now associated to event user
Yes
More/Resize button pressed?
No
Control Button Released?
No
Send pointer move command to application
Yes
Send move/resize command to application pointed at

Figure 32. The processing flowchart for wand and head tracking input events. The screens on the right show the results of the event processing on a two-window setup. The round yellow mark indicates which window is associated with the user (i.e. receiving head tracking and 3D input). The yellow arrow indicated which application is receiving 2D input.

A user can point the tracked wand towards one of the available viewports and "click" on it using one of the wand buttons to select it. Head tracking and Input from that user will now be routed to the selected application. Each application has viewport has a border whose color changes based on the active user. This lets users quickly distinguish applications they are controlling. It is also possible to point to a specific viewport and use wand movements to move or resize it, similar to how clicking and dragging a window border allows window resizing on a desktop. Finally, windows can be arranged in predefined layouts using key combinations on the wand or an external interface like a tablet. In CAVE2, the system supports three layouts (single fullscreen app, two apps side-by-side or three apps left-center-right) but more combinations can be easily added through scripting.

Since the window selection and manipulation code is not essential to multiview immersion support, it has been implement it in a separate Omegalib module [95]. Keeping this code separate from the Omegalib core makes it is easier to experiment with alternative window control techniques, besides the classic desktop window metaphor implemented in the current module. For instance, I implemented a simplified window control scheme for CAVE2 that allows users to place windows into pre-determined ‘slots’ on the display, similar to the Aero Snap functionality of Microsoft Windows 7+ and split screen mode of OSX El Capitan (Figure 33). Given the larger screen estate, this implementation provides an extended set of slots compared to the desktop operating system implementations. This alternative window control implementation was used in the MVI user study (see 7.1).

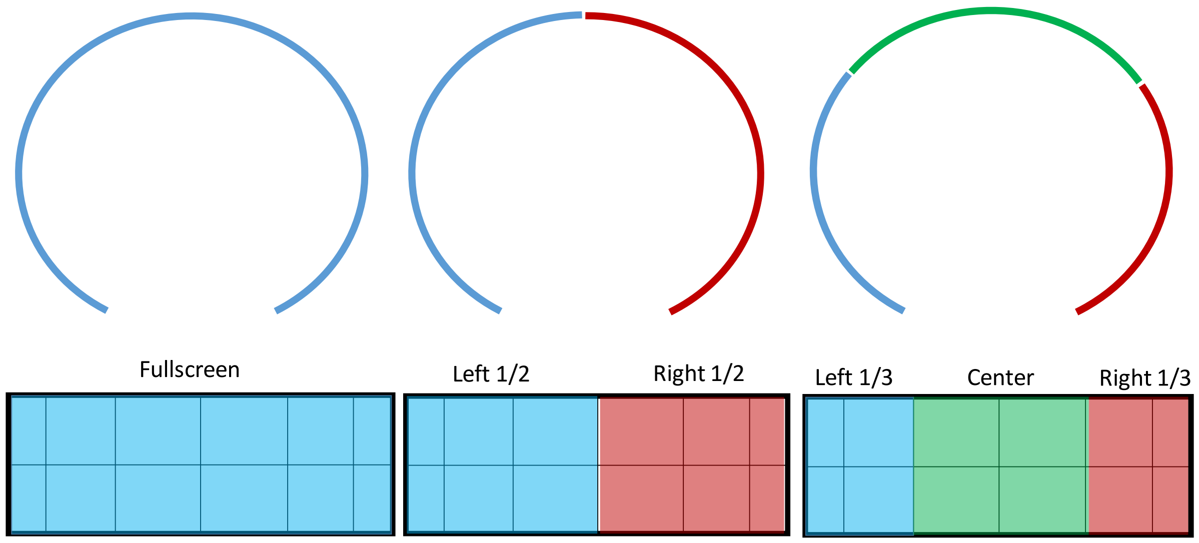


Figure 33. The six view presets supported by our alternative window control implementation. Users are still allowed to adjust windows manually if they wish, but these presets offer convenient shortcuts to common display space configuration. This technique will be used in the MVI user study.

4.3 **Summary**

In this chapter, I presented a detailed description of the principal features of a multiview immersion system, based upon the requirements and challenges discussed in previous chapters. I discussed an implementation of these features within the Omegalib framework, focusing on support for dynamic viewports, view and navigation correction, multi-user input routing and layout control. In the next chapter, I will outline the user study plan to validate and assess the value of multiview immersion for co-located collaborative work. I will identify several research questions I plan to answer, and use them as guidelines for the experiment design.

CHAPTER 5

EVALUATION

Measuring the success of Multiview Immersion involves both human and machine factors. On the machine side, it is important to assess the performance of the software infrastructure and implementation I presented. Section 3.2.4 underlined the importance of resource allocation across the cluster when multiple applications are running. In 3.6, I also presented an evaluation of performance of multiple Omegalib applications running in a hybrid reality environment, showing how a dynamic view setup performs similarly to statically allocated views. Since the Multiview Immersion implementation I detailed in the last chapter is based on the dynamic workspace technique, the application performance and system resource usage of multiview immersion are unchanged from the results discussed at the end of chapter 3.

Regarding the human side of the evaluation, I want to understand if Multiview Immersion provides any significant benefits (objective or subjective) compared to classic, non-hybrid display use modalities. The alternatives I consider are the 2D multi-view use typical of display walls and the single-view, single-user immersion of systems such as the CAVE. Our overarching hypothesis (which I will detail and break down in this chapter) is that MVI provides measurable benefits compared to alternatives, given a suitable task. In other words, I don’t expect MVI to be better than multiview 2D for the analysis of abstract, 2D or very simple 3D data. I am also not interested in the performance benefits of MVI over single view immersion for non-collaborative use. This isn’t to say that MVI could not provide potential advantages in this setting: I am just not focusing on this particular aspect. The premise of this work is that large-scale, high resolution display environments (including HREs) are an ideal platform for *co-located collaboration*. I therefore focus my evaluation to this setting: multiple users participating in an analysis task involving a 3D dataset of moderate-to-high complexity, where the analysis involves measuring or comparing properties of part of the dataset in relation to others or to the whole. This kind of analysis is common to a variety of real-world 3D analysis tasks. Some examples are understanding the spatial location of artifacts in a 2D excavation site, determining the size and position of a tumor relative to other structures in brain MRI scans, interpreting LIDAR data and so on.

For this evaluation, I considered two alternative approaches. The first option was to evaluate the infrastructure in a real-world collaboration setting, similar to the ENDURANCE meeting presented in section 3.5.1. Sessions of those meetings could make use of the improved infrastructure described in this proposal. However, depending on the nature of these meetings, acquiring consistent quantitative data may be difficult, and this alternative would be oriented more towards a qualitative observational study.

The second option involves to validating the MVI technique in a controlled user study. The requirements of this study are the following:

1. The study should simulate a co-located collaborative work session, since this is the principal usage mode of large scale HREs, and the proposed software infrastructure specifically supports it. Therefore, testing sessions should involve at least two users working together in a problem solving or visual analysis task.
2. A subset of data used in the study should be designed to allow users to be trained quickly enough to interpret it, and solve sole simple task based on it. Tasks could involve identifying features in the data like geographical points, boundaries or outliers. The task should also be designed to encourage manipulation of the dataset and 3D navigation

3. The study should require the use of multiple, inter-related 2D and 3D views of data. The 3D data should be dense or spatially complex, to encourage the use of stereo cues and physical navigation around the data.
4. The layout of views, and the creation/destruction of views should be determined by users as part of the experiment.
5. Users should use wands, tracked glasses or remote pointers to control views independently, to favor the use of different collaboration patterns.

Given the desire to collect quantitative data on the effectiveness of MVI, the controlled experiment was considered a better choice. Following the requirements listed above, the following sections will detail a controlled experiment design that emulates with good accuracy typical usage patterns of a large scale Hybrid Reality Environment.

5.1 **Research Questions and Hypotheses**

My central research question can be stated as follows: given a representative 3D analysis task, does Multiview Immersion provide any benefits to co-located collaboration, compared to classic display usage techniques? I consider both objective and subjective benefits. An objective benefit would be increased *effectiveness* of MVI versus alternatives. We can further break down effectiveness into *efficiency* (i.e. time to completion for a task) and *precision* (i.e. the amount of errors in the analysis result).

My hypothesis around objective benefits is that a suitable 3D data analysis / quality assessment task carried out by two collaborating users will be completed more efficiently, precisely or both in the MVI condition compared to alternative conditions. Efficiency and precision can be assessed in different ways but I concentrate on task time-to-completion and number of errors respectively.

Regarding subjective benefits, my hypothesis is that users will find MVI more pleasant to use than alternatives. To be more specific, I expect users to be more ‘engaged’ with the analysis task in the MVI condition: this can be measured by observing the communication and collaboration patterns during the experiment and also by gathering data through post-experiment questions.

An additional research question is centered on the presence of multiple views and the nature of interaction in this system. Since we are giving users the option to control and rearrange views, we also expect users to engage with this feature without having it become a hindrance to their main analysis objective. Clearly, if users were to never engage with multiple views we would have invalidated one of the core driving principles behind MVI. Moreover, if rearranging multiple immersive views is so cumbersome as to consume significant analysis time, we would need to rethink our interaction design.

One realization following from this research question is that setting limits to the extent to which users can customize the number of views and their arrangement can be beneficial, if that doesn’t have an impact on the analysis’ effectiveness. My main objective is understanding the value of multiview immersion, independently of the interaction techniques chosen to control the views themselves. This is also the motivating factor for keeping the view control implementation separate from the rest of the MVI implementation, as explained at the end of 4.2.4. At the end of that section, I described an alternative implementation for view control using presets. I will use that implementation in the user study, in order to focus my analysis to the benefits of multiview immersion regardless of the specific interaction scheme used to control views.

My hypothesis regarding multiple view interaction is that different user groups will choose different view layouts depending on the way they choose to approach the task and how they prefer to collaborate. I also hypothesize that the time users spend rearranging views will be limited to avoid conflicting with their actual task.

5.2 **Materials**

The user study was based around a collaborative quality assessment task of 3D sonar data. 15 groups of two users completed the same task using three different subsets of the data under three different conditions simulating classic immersion (3D) classing multiview (2D) and multiview immersion. The next section will describe the general system setup. In 5.2.2 and 5.2.3 I will present the dataset and application used for this study. They both derive from the work done for the NASA ENDURANCE project (see 3.5.1).

5.2.1 Setup

The study was carried out in CAVE2. I used two sets of tracked glasses and two wands, one for each user in a user study session. A table and two chairs were placed close to center of CAVE2. A computer was used to control the views, and additional material related to the experiment dataset was be made accessible: Users were provided with paper forms to fill out with their data quality assessment. For a review of material used in this study refer to *Appendix A.* One camera and microphone were installed to record the session.

5.2.2 Dataset

Point cloud data from NASA ENDURANCE was selected as the dataset for the user study. The dataset consists of ~50 partially overlapping sonar scans (dives) of an Antarctic lake, plus additional measurements of lake depth at several points in the lake. All of the dives were accessible to the users during the experiment, but a pre-determined subset was part of the actual quality-assessment tasks. Three groups of dives were selected, one for each of the three conditions, with 5 dives per group. Each group of dives contained dives with one or more ‘defects’. Typical issues present in dives are:

- Noise above or below the correct measured depth
- Noise or artifacts around the correct measured depth
- Missing data
- Misaligned data

Each set of 5 dives were designed to contain approximately the same number of defects overall. These defects were present in the original data and were not introduced specifically for this study.

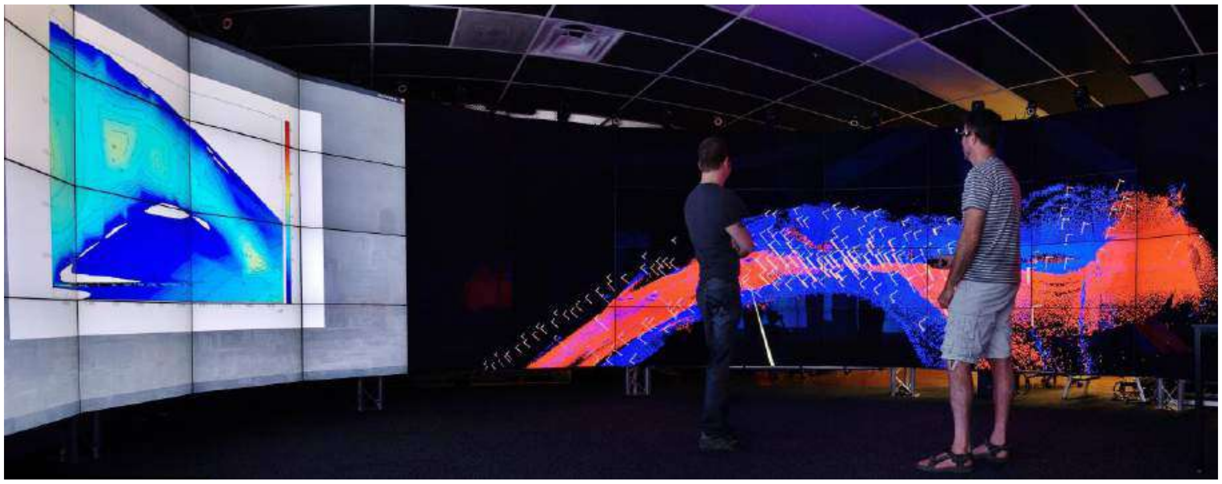


Figure 34. An early version of the IceCloud application being used by two members of the ENDURANCE team to validate the quality of sonar point cloud data. The version used in the study added support for marking active dives and the 2D dive timeline overlay, as explained in 5.2.3

5.2.3 The IceCloud application

The users had access to an application to view and analyze the point cloud data in CAVE2 (Figure 34). The application allows users to freely navigate around the data using a tracked 3D wand. 6DOF navigation was available in all conditions. Navigation speed was fixed, and users could reset the view to the center of the lake if desired. The 3D model scale was fixed to 1:1 and could not be changed by participants. Each user was able to use a separate wand to control one view of their choosing (views available depended on the experimental condition, detailed in the next section). Users were able to select which dives were visible, from a list of all the available dives (15 total). Moreover, users were allowed to mark one dive per view as the active dive for that view. The active dive was displayed in a different color on all views. The list of available dives, indicating which dives were visible and active, was displayed on all views. Users could scroll through the dives using the directional arrows on the wand, and use two buttons to toggle visible and active dives. A detailed description of available controls is presented at the end of Appendix A.

A 2D plot of the dive depth profile was available inside a 2D overlay. Users were able to scroll through time on the 2D profile and have a specific area of the active dive outlined in the main 3D view (Figure 35). In experimental conditions where multiple views are available, the active dive selection was synchronized between all views. Finally, users could access a top-down view mode. In this mode, the camera changed to display the entire lake from above, and both stereo and head tracking were disabled for the view. The main purpose of this view was to allow users to get an overview of all currently visible dives.

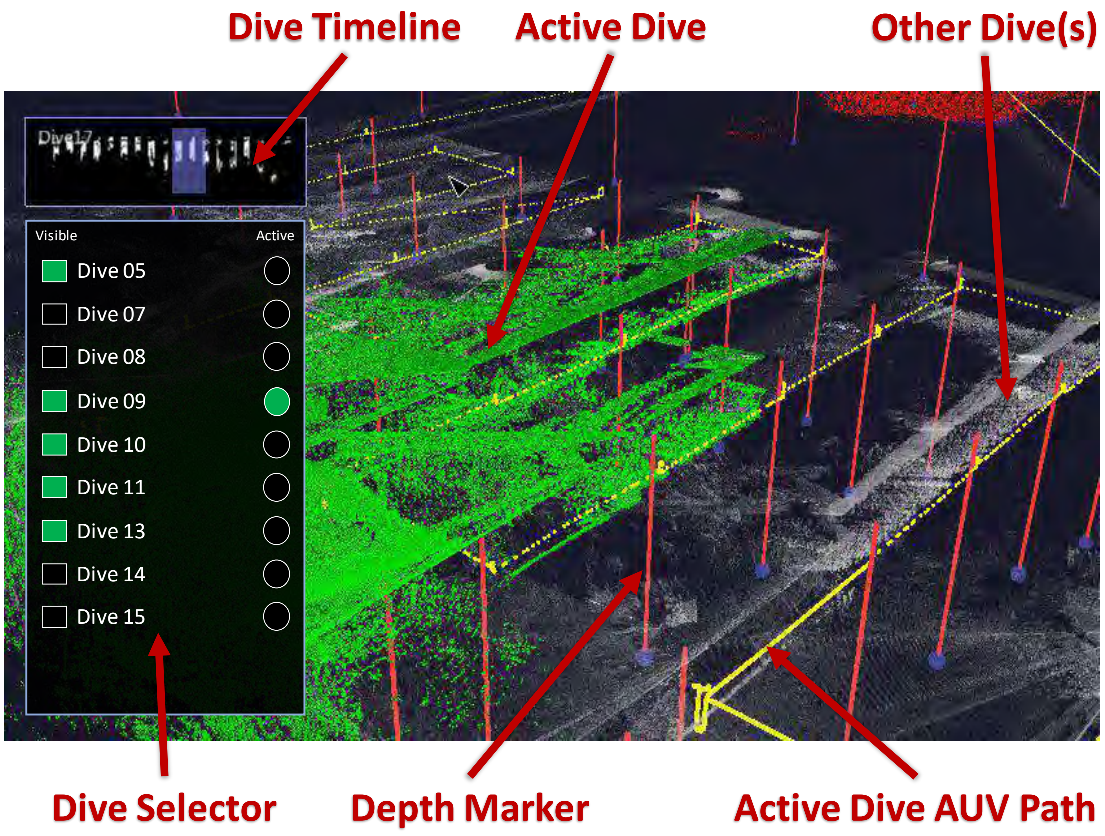


Figure 35. Screenshot of the desktop version of IceCloud, showing one active dive in green. Other, not selected dives are rendered in white. For the active dive, the path followed by the ENDURANCE vehicle is rendered as a yellow line. The orange columns represent high-precision depth measurement locations. The scatterplot on the top-left corner represents a timeline of the active dive with recorded depth on the y axis. The dive selector GUI can be used to toggle dive visibility and select the active dive.

5.3 **Methods**

Groups of 2 participants were asked to complete a series of analysis and quality evaluation / classification tasks on 3D data (point clouds), under 3 conditions:

1. *Multiview 2D (2D) Condition:* The display supports multiple views with 2D features only (no stereo rendering, no head tracking), both users are allowed to control and re-arrange views. This condition simulates a 'standard' multi-user 2D display walls
2. *Single View Immersive 3D (3D) Condition*: The display runs a single view with 3D features enabled. A single is tracked and allowed to control navigation. Users can allowed alternatively take control of the view using a button on their input devices (wand). This condition simulates a standard single-view immersive display environment, except that both participants can use their input device to negotiate control instead of having to pass a single device around.
3. *Multiview Immersive (MVI) Condition*: The display supports multiple 3D views. Both users are allowed to control and re-arrange views.

The order of conditions changed for each user group in the study to adjust for learning effects. Given the limited number of research subjects I planned to recruit (30 divided in 15 groups), I were not able to test all possible permutations of experimental conditions to obtain statistically significant comparisons between them. Since our objective is comparing MVI to alternatives, I chose to have the MVI condition as the first one for half of our experiments and the last one of the rest. This allowed us to compare task error rates for MVI to the aggregate alternatives.

To elicit collaboration and force an efficient use of time, users were given a limited amount of time to complete each task. Preliminary observations let us determine 20 minutes as a good limit, pushing users to concentrate on the task while being long enough to expect most groups to complete it. This was a hard time limit: once time expired the test was stopped and participants had to submit their quality assessment in its current state. As the study facilitator, I kept users informed of how much time was left for the current task.

5.3.1 Task

For each condition users were given a set of 5 predetermined dives from the ENDURANCE dataset. A dive is a single mission for the ENDURANCE vehicle, covering a portion of the lake with a sonar scan. Users were asked to judge the quality of each dive compared to other dives in the set. The comparison was based on the amount of noise found in the dive, its alignment to other dives and high-precision depth measurements, and the size of the area covered by the dive. Users were asked to order the five dives from best to worst, using a paper form. After reaching an agreement on the ordering, users would ask the facilitator to stop the task. If the 20 minute time limit expired, the facilitator ended the task and collected the forms in their current state.

5.3.2 Use study structure

1. The two users gathered in the CAVE2 room and completed the initial consent and media release forms.
2. The users were given a brief background on the data they were about to analyze.
3. The facilitator introduced users to the visualization application and its basic controls.
4. The users were allowed to freely use the application for about 10 minutes. During this phase, the screen was split in half, so each user was be able to experiment independently.

5. The users started the quality assessment task for each of three conditions (pre-randomized).
6. During the user study the facilitator provided briefing, training and debriefing. During the actual task, the facilitator was present in the CAVE2 environment with the users and was available to answer technical questions the users had on usage of the application. The facilitator also provided overall information on the data the users were analyzing but did not answer questions that related directly to the user task (i.e. if part of a dive was a defect, or if a specific dive quality was higher/lower than another).
7. During debriefing users completed a questionnaire to assess the usability and ease of collaboration for each of the three conditions. Users were also elicited to provide additional unstructured comments about the features of each condition.

In conditions where multi-views are supported (2D and MVI), users were given access to up to three separate views of the data, and were allowed to re-arrange the views on the display. View control was performed through a web interface. The web interface let users choose one of 6 pre-defined ‘slots’ for each view: Left or right 1/3 of the display, left or right half of the display, center of the display, full display (with a 7th option to hide the view). While the MVI implementation lets users freely arrange views on the display, this feature was disabled for the user study, to limit interaction complexity.

5.3.3 Participants

I recruited participants by posting announcements to the UIC graduate student mailing list. I also directly recruited faculty, students and staff from the Electronic Visualization Laboratory (lab members are known to utilize HREs regularly in their work). Additionally, I looked to recruit scientists from the pool of collaborators who are familiar with this work and who actively use HREs or other large-scale displays in their research. Potential participants were recruited using E-mail (the recruitment letter is attached in Appendix A). The study took place at the Electronic Visualization Laboratory at UIC. Details on participants will be listed at the beginning of Chapter 6.

5.4 **Data Collection**

Audio and video was recorded for the full study excluding briefing and debriefing. The application automatically collected user tracking data (user position, gaze direction, input). Users were asked to keep the tracked glasses on their heads at all times, even when in the 2D condition). The application also collected data on view arrangement, user-view assignment, view location, etc. User tracking and view tracking were saved in two separate files. For each user study, a new file was generated for each condition. File names contained information on the user group id, condition, condition permutation, dive set.

5.5 **Summary**

In this chapter I presented my research questions and hypotheses on the benefits of Multiview Immersion compared to other classical alternatives (single view immersion and 2D multiview systems). Based on our research questions I outlined a user study designed to test our hypotheses and collect additional data for potential post-hoc analysis. In order to be as realistic as possible, the user study leverages a real-world dataset derived from the NASA ENDURANCE project which I described in chapter 3. The next chapter will outline the results of this study.

CHAPTER 6

RESULTS

The user study sessions took place between March and May 2016. As described in the previous chapter, I recruited 30 participants for this study: 18 were students or staff at the Electronic Visualization Laboratory. 8 Were graduate and undergraduate UIC students from outside EVL. 4 were people external to UIC, recruited directly by the Author. 20 participants were already familiar with CAVE2. The other 10 received an additional introduction to the system, to let them familiarize with the platform before the actual study. All but 4 users received a basic stereo perception test before the start of the experiment: simple shapes were drawn at different distances on the CAVE2 display and users were asked to identify the closest to them. All participants passed this test. 4 users skipped this test due to technical issues: these particular users have extensive experience with CAVE2: I have a reasonable expectation they would have performed similarly to other participants in the test. All experiments were carried out according to the plan described in 6.3.2, without outstanding variations. All groups were able to complete the three quality assessment tasks within the allotted 20 minutes. No incomplete answers were submitted. The average task completion time was 15 minutes. As one would expect, task time decreased for the three tasks (16.6 minutes for the first task, 14.6 for the second, 13.5 for the third).

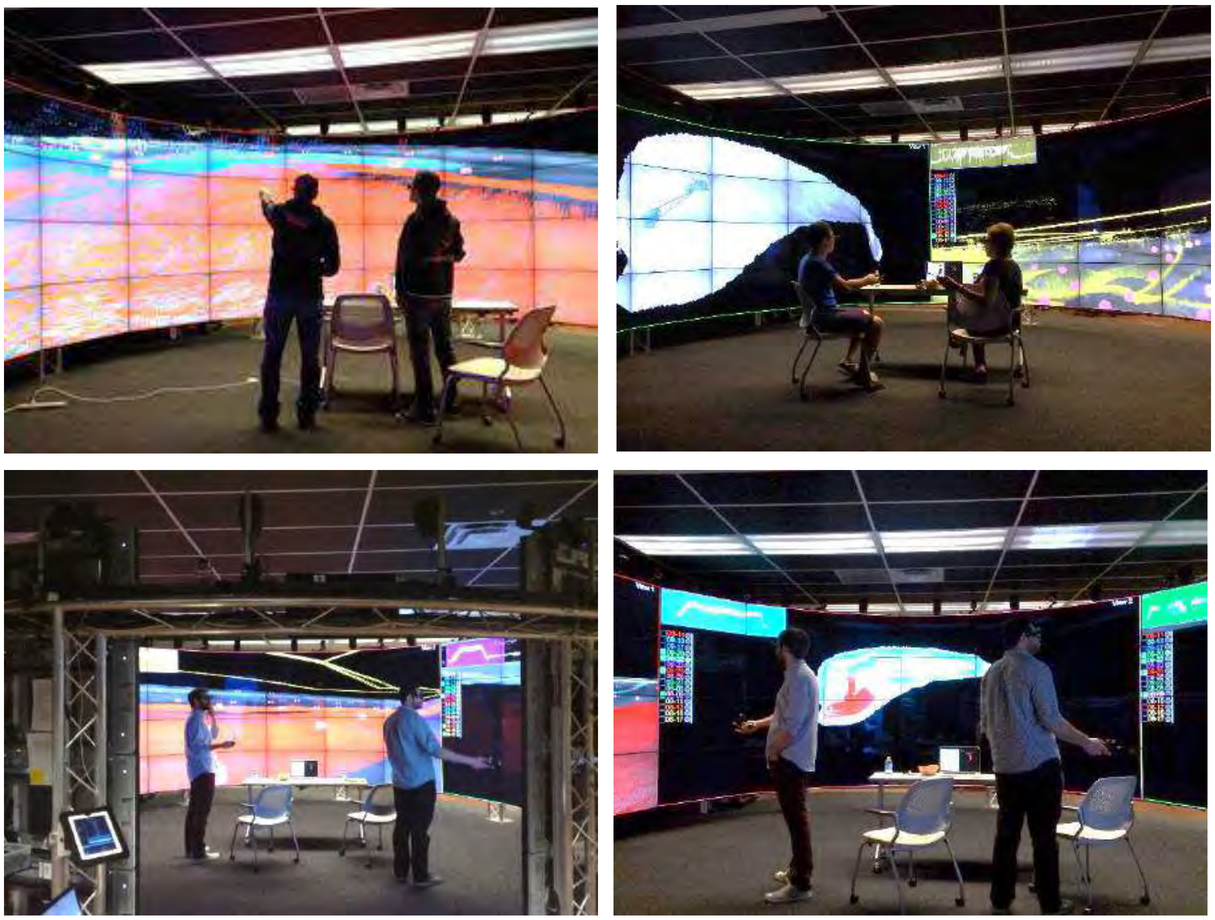


Figure 36. Pictures from the user study, showing example of different user collaboration styles and view setups. Top left: users discuss over a single full screen view. Top right: users look at two side by side views (one 2D top-down view and one immersive). Bottom left: users collaborate on a side view with one of them controlling navigation. Bottom right: users are working in parallel on two immersive views on the sides of the screen (not visible), with a central shared view showing the 2D top-down view of the lake.

6.1 **<u>Multiview Immersion vs other techniques</u>**

As described in the user study design, I used the results of the user study task to understand how Multiview Immersion compares to single view 3D and multiview 2D in supporting the collaborative analysis of 3D data. Specifically, I planned to compare the amount of errors participants committed for each technique, and how long it took them to complete the assigned task. The hypothesis I set forward to test was that participants would commit fewer errors in the

Multiview Immersion condition compared to the alternatives, spend less time completing the task, or both. For the sake of brevity and consistent with the previous chapter, the rest of this chapter will refer to the three experimental conditions using the following abbreviations:

- Multiview Immersion: *MVI*
- Multiview view 2D: *2D*
- Single view 3D: *3D*

6.1.1 Task error score

The error score for each participant team / condition was measured comparing the ordered sequence of dives submitted by the team to a reference sequence established while assembling the dive sets for each task. The three dive sets used in the experiment were assembled to have a similar difficulty. Each set of five dives consisted of two dives that could have easily been identified as the best and worst of the set, plus three intermediate dives whose sorting was more challenging. The dive sets were designed to contain a similar number of features that required analysis (like larger/smaller covered areas and sequences with noise above and below the lake reference surface).

To ensure that reference sequences of dives were fair and clearly identifiable, a pilot user study with two users was performed using the same conditions of the real study with the following exceptions: time limit on task completion was removed and two participants were directed to be as precise as possible in ordering the dives. If the pilot users' dive sequences were identical or very similar to the reference ones, we could confirm the sequences were fair and the tasks were presented correctly. The pilot study participants submitted three dive sequences that exactly matched the reference sequences.

To assess the error score for each team / condition pair I measured the element-by-element distance of the two sequences. Given a dive set of 5 dives d**,** the submitted sequence $s \in \mathbb{N}^5$ and reference sequence $r \in \mathbb{N}^5$are orderings of d: for instance $r_1 = 2$ means that the first (i.e. best) dive in the reference sequence is the second one in the dive set d. The error score for s compared to r is: $e = \sum_{i=1}^{5} |r_i - s_i|$. Computing the error score this way allows us to compare the number of quality assessment errors in a sequence, giving larger errors more weight.

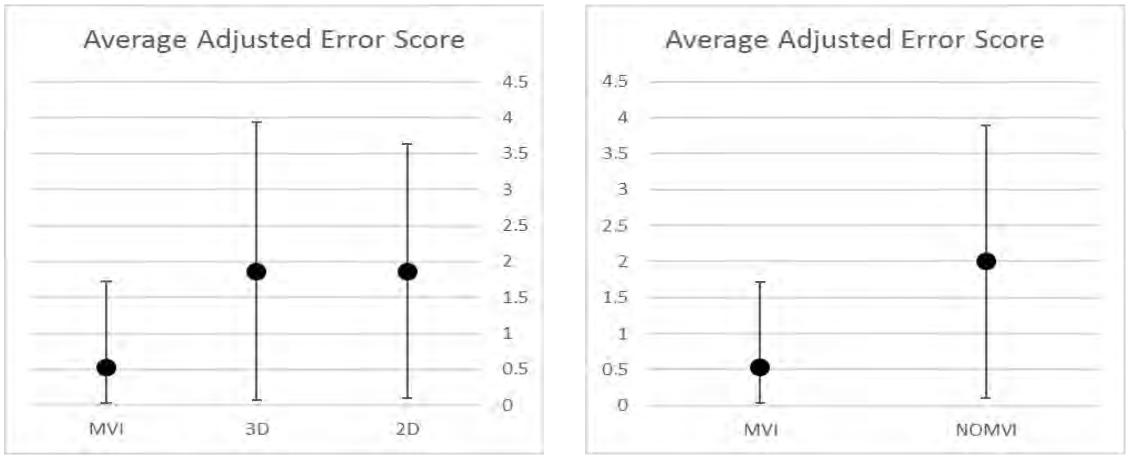


Figure 37. Left: the adjusted error score for the three experimental conditions. Right: the adjusted error score for aggregated 2D, 3D vs MVI. Correcting for baseline error within groups reduces the error standard deviation, especially for MVI.

The 3D and 2D conditions were merged to avoid learning effects in the analysis of data. During the experiment the MVI condition was first and last half of the time. I did not have enough research subjects to randomize all three of the conditions a sufficient number of times to achieve a strong enough comparison between groups. Therefore, this analysis compares the performance of the MVI condition against the *average performance of alternatives* (2D and 3D). A significant statistical result would mean MVI is better than this average, but I could not specify whether MVI is better than both, or if just one of the two alternative condition drives this effect. On the other hand, observing the error score distribution of the 2D and 3D conditions in figure 38 left we can observe they are quite similar so we can reasonably assume a test against their average would hold for both conditions, given a large enough experimental sample. I used a paired t test to compare the means of the MVI and non-MVI samples. Two-tailed P(T<=t) is 0.048, just below the significance threshold.

It is worth pointing out that the standard deviations of the error score samples for the three conditions are quite large. This can be explained by the variation of average error scores between user study groups: some participants performed better or worse *on all three conditions* than others. In this analysis what matters most is how much the error score changed depending on experimental condition *within a group*, while comparing the error scores across groups wasn't as meaningful. This is why a paired t-test was used in the statistical analysis. To better illustrate the error score variation between categories, I also computed an adjusted error score, subtracting the minimum score for a group to the three conditions in order to isolate the error score variation from each group error baseline. The average and standard deviation for the adjusted error scores are presented in Figure 38.

6.2 **Display Usage and Collaboration**

Another observation I planned to carry out was on participant's use of multiple views in the 2D and MVI conditions. Specifically, I collected data on how many views were active at any point during the experiment, and whether users were interacting with or looking at them. To reconstruct view usage behavior, I used the tracking data from user's stereo glasses, converted the user position and gaze direction to a point on the display and cross-referenced it with the view layout active at that time. The gaze to screen point conversion uses the same logic presented in 4.2.4 while discussing hybrid interaction.

View usage varied widely from group to group. While some participant teams used all the available views in the 2D and MVI conditions, others limited their view usage to one view, even when multiple views were available on the display. Based on this observation it is worth investigating whether the number of effective views used by participants has any effect on their error rate or task completion time.

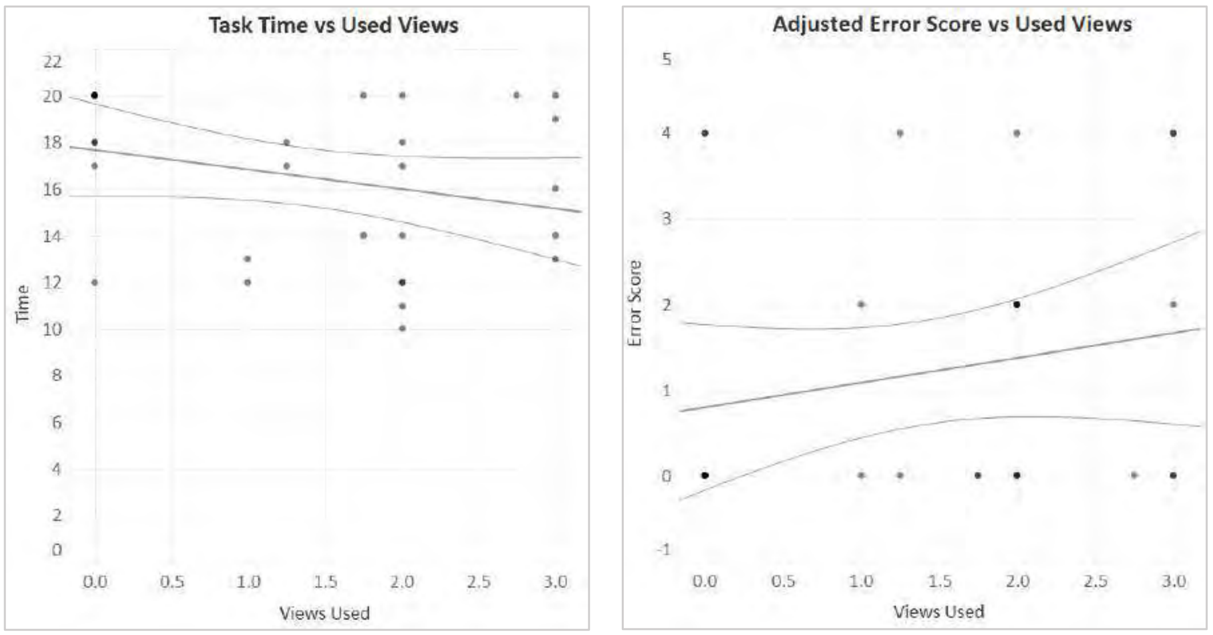


Figure 38. Correlation between the average number of used by a group and task completion time (left) and error score (right). The lines represent the linear regression result, which is not significant in both cases. For task completion time, R^2=0.08, p=0.13. For adjusted error score, R^2=0.04, p=0.28.

Figure 38 illustrates the results of this analysis: it appears the number of views used by a team has no correlation to that team's performance. This result is somewhat counterintuitive. 2D and 3D error averages and standard deviations are very similar, suggesting the effect of immersion alone on performance is small. Since error score analysis suggests MVI is better than 2D and 3D average performance, one would expect the other major differentiating feature (multiple views) to be the main driver of improvement on the data analysis task used in the study. One possible explanation is that while immersion alone and multiple views alone don't have a measurable impact on error scores, *the presence of both* does**:** MVI might provide an improvement to precision that can't be trivially assessed as the aggregate improvement from multiple views and immersion.

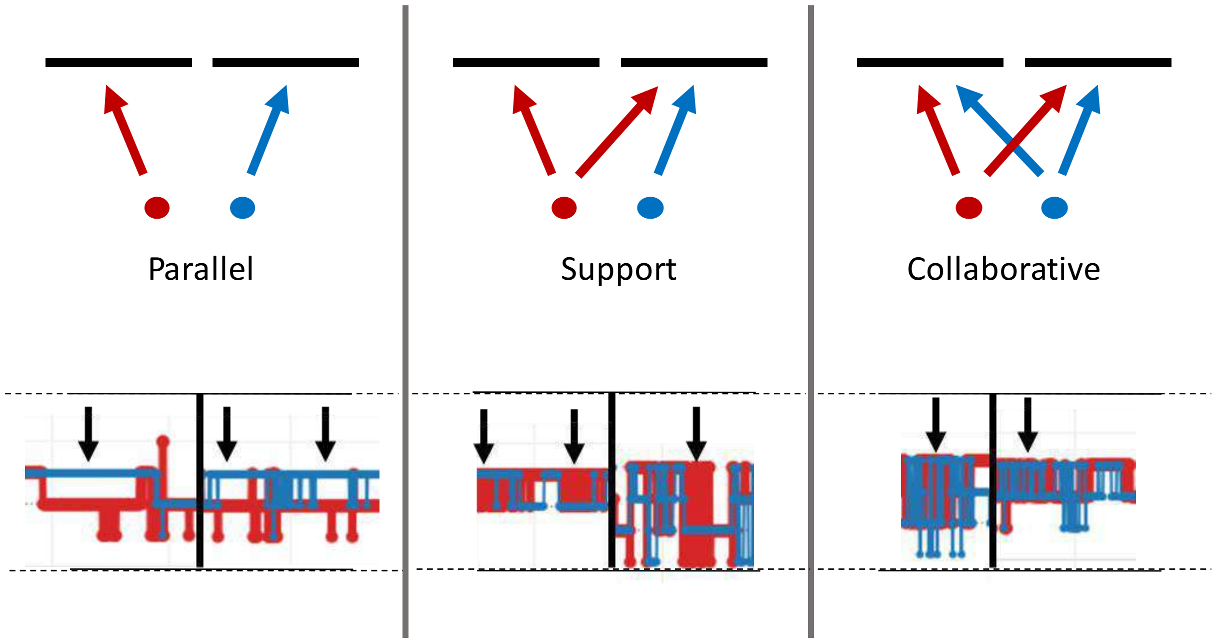


Figure 39. View usage patterns observed during the user study in the 2D and MVI conditions. The top row shows a diagram of each pattern, illustrating the relation between users and the views they are observing. The bottom row shows examples of how each pattern appear in the view usage logs. The parallel pattern (left) shows as two separate lines. The support pattern (center) shows as one line (red in the example) jumping between two views while the other is mostly static. The collaborative pattern shows as both lines frequently moving between two (or more) views.

6.2.1 View Usage Patterns

One hypothesis formulated before the user study stated that a team's use of views and display space allocation would be dependent on the current task. For instance, users might operate a different a different number of views on the display depending on whether they are in the initial phase of a task (getting an initial sense of the dives' structure) in the central phase (carrying out the quality assessment task) or in the final phase (synthesizing and reviewing the results). Based on the user logs and observations during the user study, I identified three main multiview usage patterns (Figure 39).

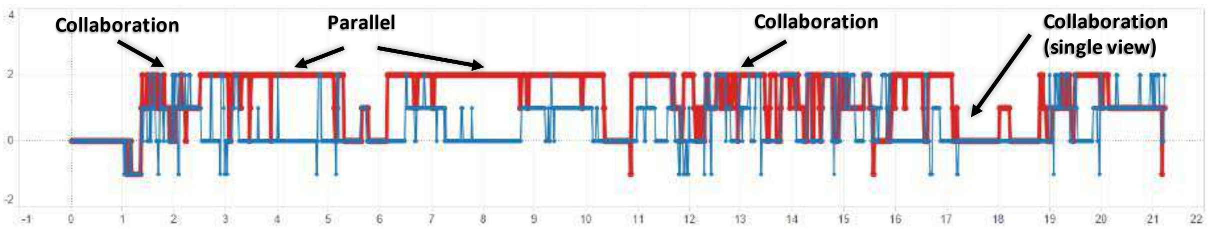
In the *Parallel* pattern, the users are acting independently, with each user interacting exclusively with one view. In the user logs, this pattern can be identified by two separate lines on the log. This pattern was used by teams that split the task into separate components that could be worked on independently.

In the *Collaborative* pattern, users are observing together one or more views, switching frequently their gaze direction. In the user logs this pattern is identified by user lines frequently jumping between views, often at the same time. This pattern was used either at the end of a task, when reviewing the quality assessment before submission, or throughout the task for teams that choose not to split the task into separate components.

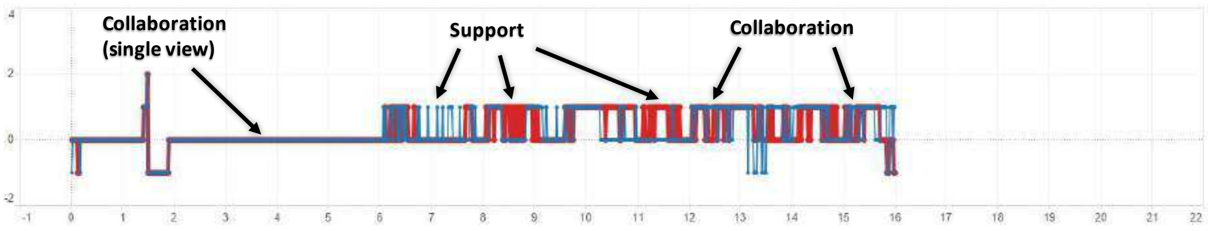
In the *Support* pattern, one user interacted almost exclusively with one view, while the other switched between multiple views. In the user logs, this pattern is identified by one flat user line, with another jumping frequently between views. This pattern is a hybrid between the Parallel and Collaborative patterns, with the users doing partially independent work but one user (the one observing multiple views) in charge of synthesizing results regularly instead of waiting until the end of the task.

Examples of view usage logs are shown in Figure 40 and 41 with annotated instances of the patterns listed above.

(1)



(2)



(3)

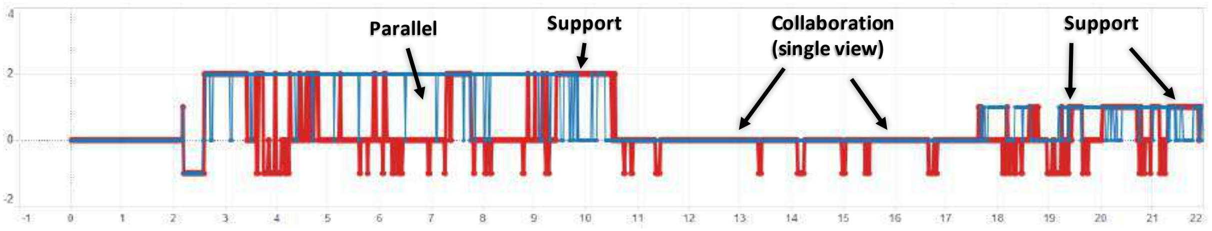


Figure 40. Examples of full view usage logs (the full list of logs can be found in Appendix B). The X axis is time in minutes. The Y axis represents the view ID (0-3). View -1 indicates the user is looking down (normally checking the user dive document, writing notes etc.).

1) Users start collaborating on one, then multiple views. Between minutes 4-10 they mostly work in parallel. Minutes 12-16 are collaborative, followed by a brief parallel session, then collaboration on a single view, likely to review results.

2) Users spend minutes 0-6 collaborating on a single view. The rest of the session is spent on two of the three available views either in the collaborative or support pattern. This work structure has been observed in groups that choose a streamlined approach at the beginning (a single view), then add more views to the system only when/if needed. This is different from group 1, which allocated 3 views on screen right from the start.

3) Participants spend minutes 0 – $3^{1/2}$ collaborating on a single view (0, documents, 2). This is followed by mostly parallel work up to minute 11. In this case, similar to group 1, users choose to split the task between them and started analyzing different dives on separate views. Minutes 11-18 are spent collaborating on a single view, likely to synthesize the outcome of parallel work. Minutes 18-20 are dedicated to reviewing the results before submission, checking them on multiple views. Note that this recording proceeds after minute 20, but data after20 is a recording artifact and not included in analysis.

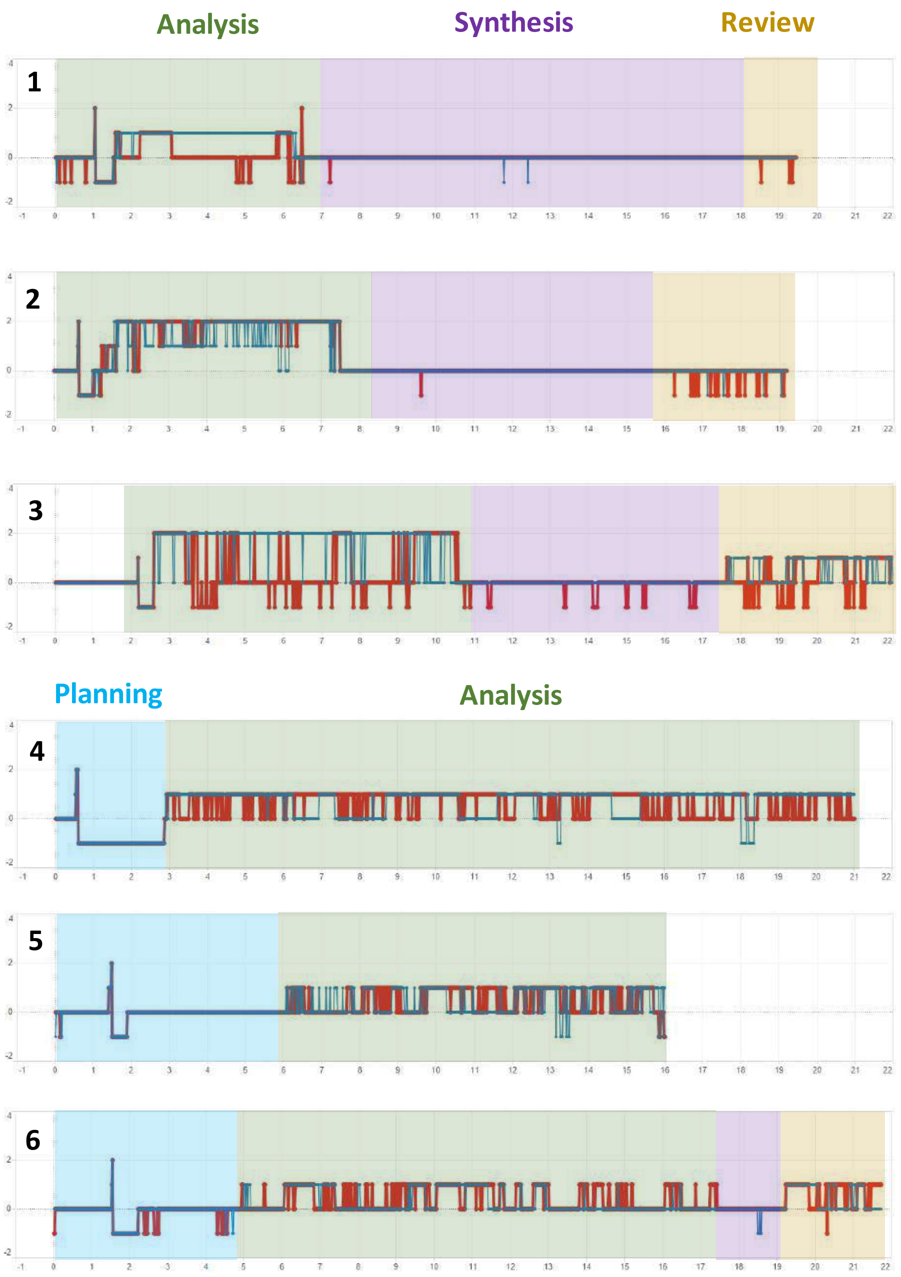


Figure 41. Examples of view usage patterns appearing in multiple experiments. Examples 1, 2, 3 show a common structure: an analysis phase (using multiple views) is followed by a synthesis phase where users consolidate their individual work. The review phase is spent recording the results and double-checking against the visualization. Examples 4, 5, 6 show a different structure: an initial planning phase involves both participants looking at the full dataset together using a single view, and deciding how to proceed with in-depth analysis. In particular, example 6 shows a view usage log where all four phases can be identified.

6.2.2 Collaboration Patterns

While the view usage pattern described in the previous section are useful in characterizing how participants interacted with multiple views during the user study, they don't fully capture the patterns of activity and interaction *between* users. For this reason, in addition to the gaze direction data I used the participant conversation records and intra-study observations to identify main patterns of collaboration between the two users, beyond their interaction with the display. These patterns are defined in part by the view usage patterns described above, but they also consider conversation between users, and whether users are just observing views or actively interacting with them. Conversation activity was recorded at1-minuteintervals, and identified simply by the presence of verbal communication between users, regardless of content. Given the simplicity of this coding scheme, I did not consider it necessary to check for inter-rater reliability on the coding results. Based on data from the user study, I identified four major collaboration patterns. These patterns are summarized in Table VIII and are the following:

1. In the *Joint Work* pattern both users are collaborating on the same task. Users are actively discussing the task and might be either both interacting with views (Collaborative view pattern) or only one of them is while the other is providing additional insights (Support view pattern)
2. In the *Parallel Activity* pattern users are working independently. This corresponds to the Parallel view pattern. While in parallel activity, there is little to no conversation going on between the users. Both users are interacting with views.
3. In the *Single User* pattern, one user is actively observing and interacting with the views, while the other user appears to disengage from the task.

4. In the *Planning* pattern the users are choosing how to organize the task's work (either at the beginning or some other time during the task itself). In this pattern users are actively engaged in conversation, and are typically deciding how to use the display space (i.e. how many views to have on screen, who will control them etc.). One user is in charge of the actual display configuration, with the other user's input. The view usage pattern is collaborative, as both users are looking at all the available views until planning is complete.

	VIEW USAGE PATTERN	**ACTIVE USER**	**CONVERSATION**
JOINT WORK / DISCUSSION	Collaborative, Support	One or Both	Yes
PARALLEL ACTIVITY	Parallel	Both	No
SINGLE USER	Support	One	No
DISPLAY ORGANIZATION	Collaborative	One	Yes

Table VIII. Summary of the identified collaboration patterns. Each pattern is classified based on the way the display is used (based on the view usage patterns discussed in 7.3.1), how many users are actively controlling views, and whether users are engaged in discussion.

One initial observation based on this data related to one of the study hypotheses: users would spend a limited amount of time organizing views, and this time would not be significantly higher in the MVI condition compared to the classic multiview 2D condition. On average, users spent 39 seconds or 4.3% of the average task time (15 minutes) organizing views. The time dedicated to organizing views was actually slightly lower for MVI compared to the 2D condition: 30 seconds vs 48 seconds respectively.

As an additional post-hoc analysis, I analyzed how collaboration patterns were distributed across the three experimental conditions, to establish if the presence of multiple views, immersion or both had any effect on the way teams collaborated. Specifically, I want to address the following post-experiment questions:

1. Is the single user pattern (that is, user engagement) affected by the availability of multiple views or immersion?
2. Are users more likely to work in parallel if multiple views are enhanced with immersion?
3. Do users lean more towards collaborating on the same task vs working independently in any of the conditions?

Figure 42 shows the single user pattern frequency for MVI non-MVI (2D+3D) conditions and for 3D vs multiview (2D + MVI) conditions. Both comparisons are statistically significant (one tailed t-test MVI vs NOMVI p=0.03, 3D vs MV p = 0.046). We can conclude that while immersion alone has a positive effect on user engagement, reducing the rate of single user activity, multiview immersion has an even more significant impact, reducing single user activity from an average of ~17% to ~5%.

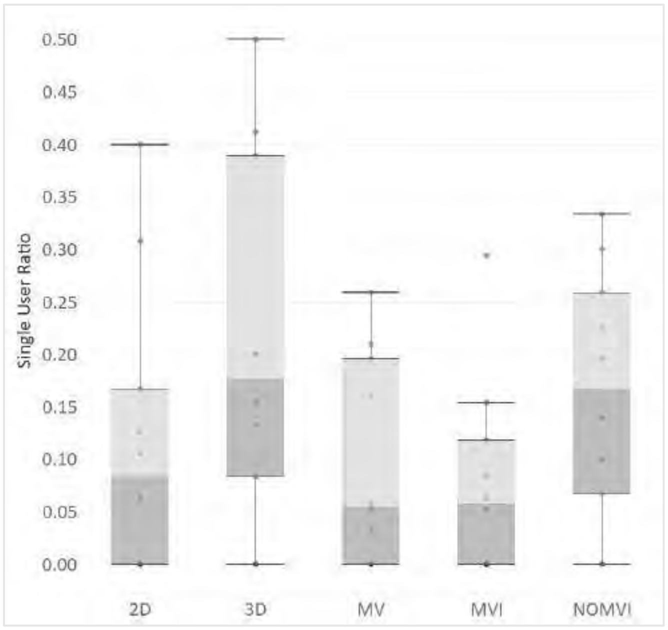


Figure 42. The frequency of occurrence of single user pattern in 3D, MV (2D+MVI), MVI, NOMVI (2D+3D) conditions. Since this pattern in undesirable (it suggest one user is not engaged with the task), high values of single user pattern occurrence are negative

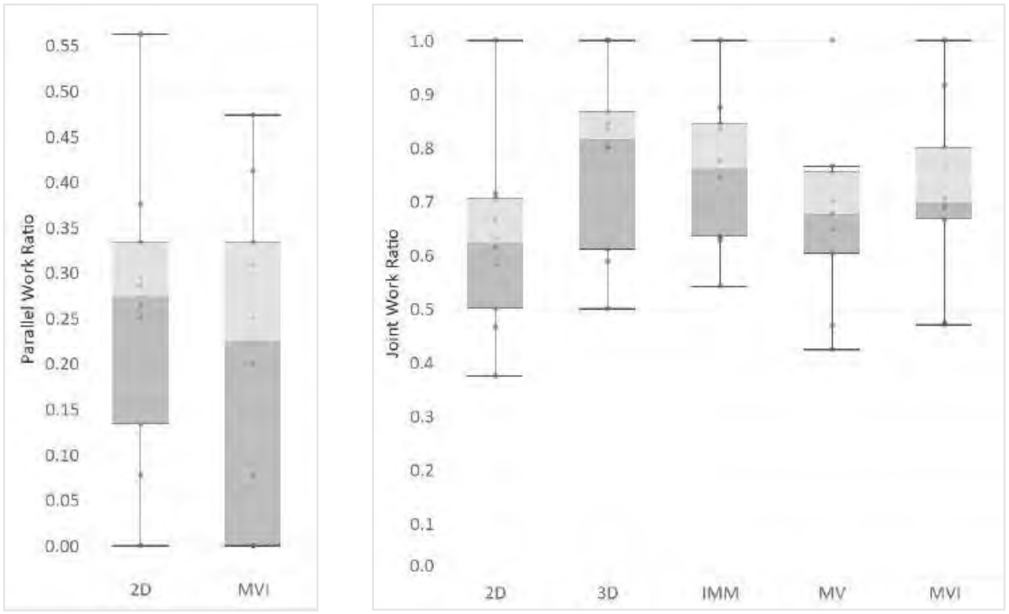


Figure 43. Left: The frequency of parallel work for the 2D and MVI conditions (3D is excluded as parallel work is not considered for a single view). Right: frequency of collaborative work for 2D, 3D, Immersive (3D+MVI) and multiview (2D+MVI) conditions.

Figure 43 left shows parallel work pattern frequency for the 2D vs MVI conditions. In this case, the 3D condition was excluded since parallel work was not possible with a single view. The difference is not statistically significant: immersion does not appear to determine how much teams choose to work in parallel. On the other hand, the frequency of collaborative work significantly improves for immersive (3D + MVI) conditions vs 2D (p=0.001) and for the 3D condition alone vs multiview conditions albeit less significantly (p=0.027). These results are shown in Figure 43 right: immersion appears to improve the rate of collaboration between users.

To summarize these results: both immersion alone and multiview immersion appear to have a positive impact on co-located collaboration. They increase the likelihood that both users in a two user team remain active and work jointly on an analysis task, with multiview immersion having a larger impact on this. An interpretation for this result is that while immersion alone increases the engagement of users with the data, the addition of multiple independently controllable views makes it possible for each users to ‘own’ a part of the visualization and independently control it if desired. This avoids a typical drawback of classical immersive system, where a single user can be in charge of controlling the visualization, establishing an imbalance in the activity between the users and making it more likely that one of them will temporarily lose focus on the task.

6.3 **Survey**

The survey was divided in three sections. The first consisted of 12 questions asking the user to agree or disagree with a statement using a 10-point likert scale. The questions were designed to elicit an evaluation of the collaborative, immersive and multi-view features used in the study. The questions were the following:

1. Having multiple views (both 2D and 3D) helped me in the analysis of the ENDURANCE data.
2. Looking at a 3D immersive view that the other user was in control of (with navigation and/or head tracking) made me uncomfortable.
3. Placing views and choosing which view to interact with was easy.
4. With a single immersive view, time controlling the application was split evenly between me and the other user.
5. Synchronization of active dives between multiple views helped in the analysis.
6. I wanted to be able to re-arrange views freely instead of using pre-configured options (center, left, right, etc.).
7. 3D immersion (stereo view and head tracking) helped me understand the structure of the data.
8. The ability to re-arrange and resize views was useful.
9. When multiple views were available, it was difficult to decide how to best use them.
10. 3D immersive views made me uncomfortable.
11. Having multiple views (both 2D and 3D) facilitated splitting work between me and the other user.
12. I wanted to use more than three views.

The second section asked the user to rank the three techniques used during the study (Single View Immersion, Multiview 2D and Multiview Immersion) based on five criteria:

1. Ease of use
2. Usefulness in splitting tasks evenly
3. Collaboration
4. Enjoyability
5. Overall favorite

Users were allowed to rank two techniques with the same number, if they felt they were comparable for any of the criteria above. This section of the survey was designed to assess how each technique performed compared to the others based on the experience of each participant. The final section of the survey consisted on one optional, open question asking users for any additional observation on the study and on the Multiuser interaction technique.

6.3.1 Results: Questionnaire

To analyze questions of the first part of the survey they were grouped in three categories based on the specific aspect of the user study they covered. All the questions were cross-cutting and did not refer to one specific technique. For each category, questions were designed to outline one *positive* and *negative* aspect of that category. By aggregating the results of these questions, it was possible to assess the subjective value users gave to collaboration, immersion and interaction with multiple views for the user study.

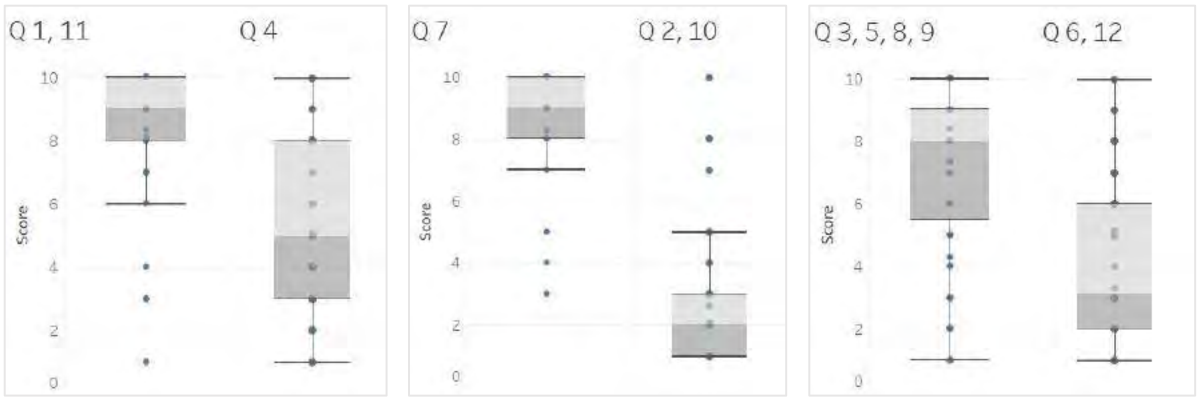


Figure 44. Results from the exit survey. Questions are grouped based on topic. Left: collaboration Center: immersion. Right: Interaction with multiple views. Higher scores are better.

For *collaboration*, the underlying question was: do multiple view help organizing and splitting the work between users (questions 1, 11) or is a single view sufficient? (Question 4). Collaboration in a single view was scored as 5 on average, while multiple views were scored at 8.2 on average. The complete score breakdown is shown in Figure 44 left. The difference is statistically significant ($p < 0.001$)

For *Immersion*, the underlying question was: does immersion help analyzing the data (question 7) without causing discomfort to the user? (Questions 10, 2). The results are shown in figure 44 center. Immersion helpfulness was scored on average at 8.3, while discomfort was scored at 2.3. The difference is significant ($p < 0.001$) although in this case the two questions address different aspects of immersion, so a statistical comparison between them isn't as meaningful as looking at the scores separately.

For *interaction with multiple views*, the underlying question was: were multiple views easy to use and beneficial (3, 5, 8, 9) or were they too limited? (Questions 6 12). The beneficial/easy to use score average was 7, while multiple views being limited scored an average of 4.23. These results are shown in Figure 44 right. The difference is statistically significant (p < 0.001).

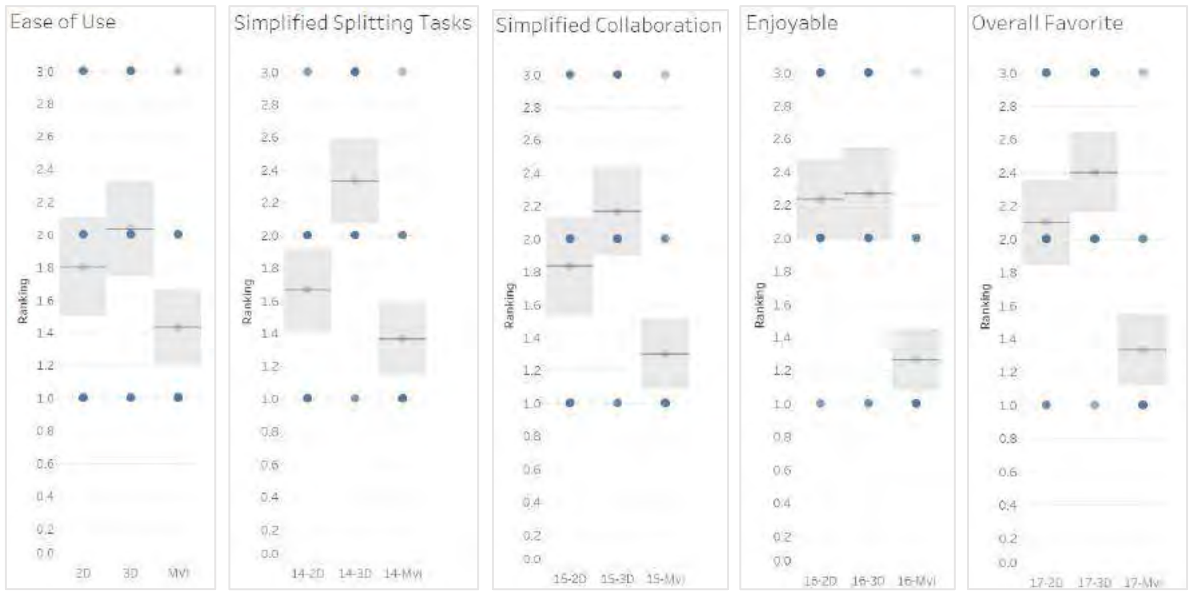


Figure 45. Technique ranking results. Gray bands are 95% confidence intervals around the average rank. Lower values are better.

6.3.2 Results: Technique Rankings

As explained in section 5.4, the second part of the exit survey asked participants to rank the three experimental conditions (2D, 3D, MVI) based on a set of criteria: Ease of use, usefulness in splitting tasks and easing collaboration, enjoyability and overall favorite technique. Figure 45 reports the average rankings for each technique on the criteria listed above. Based on these results, I wanted to test three hypotheses focusing on ease of use, usefulness and general likeability. The tests were carried out using a multi-factor ANOVA with a post-hoc t-test between samples.

For *Ease of use***,** I expected MVI to perform at least as well as the 2D and 3D techniques. In other words, I wanted to ensure users' subjective assessment was that MVI was no more difficult to use than the alternatives. This measure complements the objective measure of planning time analyzed in 6.2. MVI actually ranked better than the 2D and 3D techniques (1.4 vs 2 and 1.8 respectively). The difference in ranking between 2D and MVI is statistically significant (p <0.1).

For *Usefulness in splitting tasks evenly* and *Collaboration* we expected MVI and 2D to rank better than 3D. In particular, the hypothesis was that multiple views in 2D and MVI would be useful in splitting tasks (parallel work), and would also be potentially beneficial for collaboration (joint work), with MVI being overall better than 2D. We saw in section 6.2.2 how, somewhat surprisingly, parallel work did not increase when multiple views were available. This objective measure is confirmed by the user technique rankings: multiview 2D scored worse (2.3) than either 3D or multiview immersion (1.6, 1.3). The difference between 2D and 3D, MVI is statistically significant (p < 0.1). In conclusion, the presence of multiple views didn't increase the amount of parallel work between subjects in our objective measure. Moreover, participants considered it actually *worse* than the alternatives. In the collaboration ranking, the collaboration patterns data we discussed in 5.3.2 suggests that immersive techniques were more beneficial to collaboration than non-immersive multiview 2D. The technique rankings confirm this, with 2D scoring worse (2.1) than 3D MVI (1.8, 1.3). The difference is statistically significant, but only between MVI and 2D (p <0.1).

For *Enjoyability* and *Overall Favorite* we expected MVI to rank better than both the 2D and 3D. As these are both subjective measures, there is no objective data discussed in previous section that we can compare this result against. On these two rankings MVI performed significantly better than the alternatives. For enjoyability, MVI scored an average 1.3 vs 2.3 and

2.2 of the 2D and 3D techniques respectively. For overall favorite, MVI scored 1.3 vs 2.4 and 2.1 for 2D and 3D. Both these comparisons were significant with ($p<0.1$).

It should be noted that there is a possibility of implicit bias in the technique ranking results, as a significant portion of participants, being from the Electronic Visualization Lab, was familiar with the background of this research. The purpose of the study was stated to be a comparison of different techniques for collaboration in large immersive displays, and Multiview Immersion was never explicitly indicated as the Author's proposed technique. That said, it can't be excluded that participants (even the ones external to EVL) built an understanding of which technique was really being tested, potentially biasing their ranking in questions 13-17.

On the other hand, it is remarkable how well subjective and objective measures agreed in this study. It's important to point out that we are not suggesting that collaboration in multiview immersion is better than 2D multiple views *in general*. Our user study involved a specific analysis task that involved a 3-dimensional dataset with multiple components and moderate-to-high complexity. We can extrapolate our results to similar tasks. In these scenarios, the ability of multiview immersion to provide both stereoscopy and the ability to look at data from multiple viewpoints provides measurable advantages compared to 'classic' alternatives.

6.4 **Summary**

In this chapter I discussed the results of a comparative user study, aimed at measuring the effectiveness of MVI, 2D and 3D display usage in a quality assessment task of 3D data. While the time to complete tasks was not different between conditions, MVI appears to provide an advantage in the precision of analysis, lowering the error score from 3.7 in 3D, 2D to 2.5 in the MVI condition. I also observed how different user groups exhibited different view usage and collaboration patterns in their approach to the task, noting how both immersion alone and MVI can improve the quality of collaboration between users. On the other hand, the presence of immersion does not appear to increase the amount of parallel work between users. Based on observations during the study, it is more likely that users will determine how to work on the task during the initial planning, and this choice depends more on participant’s preference than the experimental condition. The full breakdown of user study results is presented in Appendix B.

CHAPTER 7

CONCLUSION

Collaboration has long been an empowering, often fundamental element in scientific endeavors. Its importance is not new, but it is arguably increasing. Part of this is due to the sheer complexity of modern research datasets. Part is due to the potential of teams from different disciplines tackling a common problem together or looking at it from a different perspective. Hybrid Reality environments, with their ability to adapt to widely different analysis modalities can be powerful tools in modern scientific work, as long as we provide suitable ways to integrate them with researchers' work.

This dissertation built on these assumptions to derive two challenges which drove the entirety of this work. First, I argued that effective collaboration in Hybrid Reality Environments requires a conceptually new software foundation to support this kind of work. The HRE OS model and Omegalib are my proposal to address this challenge. I then argued that co-located research groups working with heterogeneous visualizations demand flexibility: the display system must accommodate to the data and work patterns of users, not the other way around. Multiview Immersion (MVI), the central part of this dissertation, is my approach to address this second challenge (with a specific focus on HREs). Omegalib and MVI represent together the main research line of my Ph.D. studies.

Validation followed different approaches for Omegalib and MVI. For Omegalib, I relied on observations of its use in different groups and for different visual applications (3.5). MVI was evaluated through a formal user study (Chapters 5, 6). A significant portion of this dissertation is dedicated to the user study design and discussion. Validating MVI by showing some measurable advantage on a reasonable analysis task was, in my view, fundamental to justify the need for

Multiview Immersion as a concept, not just my particular implementation. I also approached this study believing a validation of MVI is important in assessing the value of Hybrid Reality Environments in general, as it puts one of their key features to the test.

While my MVI implementation was built upon my work on Omegalib, I should underscore how MVI itself does not depend on Omegalib, or even the HRE operating system model. Omegalib is an ideal environment to support MVI, but not necessarily the only one. It could be desirable to build some (or all) the features of MVI on top of other large-display software frameworks, possibly with some tradeoffs.

7.1 **Limitations**

One potential limitation of this work is in some specifics of the MVI implementation. I chose not to define a unified solution for transformation or navigation adjustments as described in Chapter 4. While I believe this choice makes sense from a practical standpoint, one could argue this is less elegant than a truly generic solution. I investigated this possibility, but could not identify a suitable candidate for a truly generic solution without introducing a large amount of complexity to the system. For instance point-matching algorithms [96] could be used to compute a view adjustment transform from canvas real world corners: however these work for rigid transformations: as we saw, on a cylindrical display these would not work correctly without major modifications. I believe my solution strikes a good balance between simplicity and effectiveness, but further investigation on alternative techniques could provide suitable solutions to this challenge.

There are also areas of the user study that could warrant further exploration. As we saw in 6.2.1, the effectiveness result (adjusted error score) for MVI vs alternative techniques were positive, but they were right at the threshold of significance. A larger comparative study to further qualify differences between MVI, 3D and 2D display usage can strengthen this result, and help identify important differentiating factors between those techniques.

Finally, I noted at the end of chapter 6 that the user study results can only be generalized at most to analysis tasks similar to the one implemented in the study. However, it is not difficult to envision a variety of real-world tasks similar to the one we used: analysis of 3D MRI scans to identify peculiar structures, assessment of noise and features in LIDAR point cloud data, analysis of complex structural engineering models are a few examples.

7.2 **Applicability beyond HREs and Further Research Areas**

Hybrid Reality Environments in their current incarnation, depend on the availability of one specific display technology: thin-bezel, passive stereoscopic flat displays. We should consider the distinct possibility that, depending on market demand, these displays might become more difficult to procure, especially with the particular qualities needed for a large scale installation. It’s not completely unlikely for projector-based systems to regain importance in the large-scale immersive display space. On the other hand, we might see the introduction of even more advanced technologies such as bezel-free, OLED surfaces that could be tiled and shaped to the desired form factor, and could offer passive (or active) stereo support.

I believe this uncertainty represents a potential advantage for Omegalib, as it is designed specifically to be agnostic to display sizes, geometries and stereo technologies. As discussed in chapter 3, Omegalib also supports consumer head-mounted displays, and has been successfully used with systems such as the Oculus Rift for several research projects [97][98].

A potential which was touched upon in this dissertation is support for multiple views beyond HREs. In 3.2.3 we discussed using Omegalib to stream views to secondary devices. This technique can be extended beyond local devices to create datacenter or cloud instances of Omegalib applications which stream visualizations to devices on demand (view-as-a-service). Some work has been done in this area both for the Firefly application (3.5.3) and others.

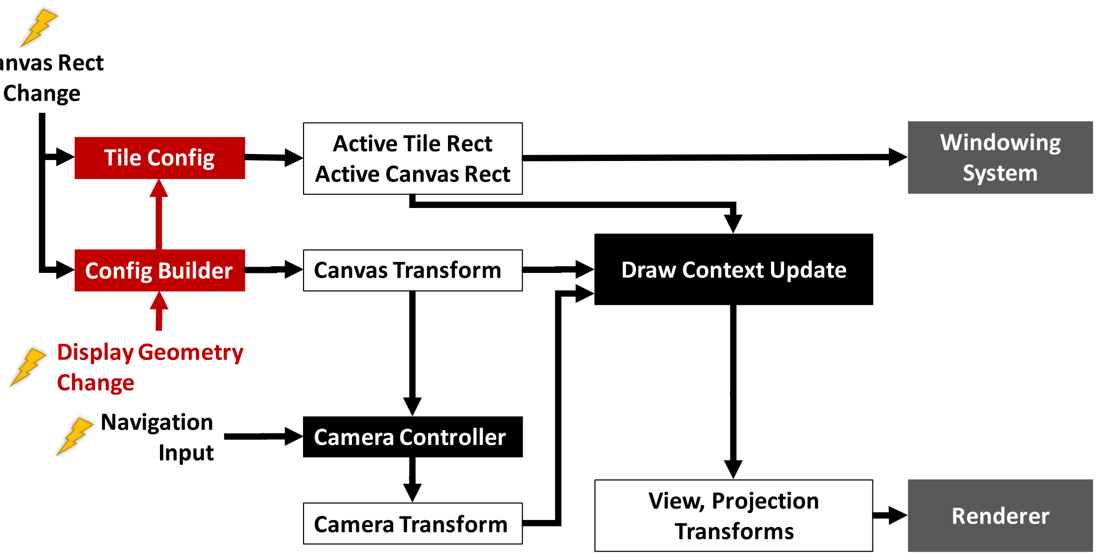


Figure 46. The transformation flow, extended to support variable display geometries.

Another area of applicability and potential research involves variable display geometries. Large display vendors such as Mechdyne commercialize systems where the displays themselves can be rearranged at runtime, such as CAVE systems where the sides of the CUBE can be unfolded to go from enclosed space to flat display. Conceptually, MVI would able to support movable display surfaces, using an alternative display configuration builder which adjusts transformations based both on canvas positions and changes to the display. Figure 46 shows how this configuration builder would be integrated in the transformation flow I discussed in 4.2.3

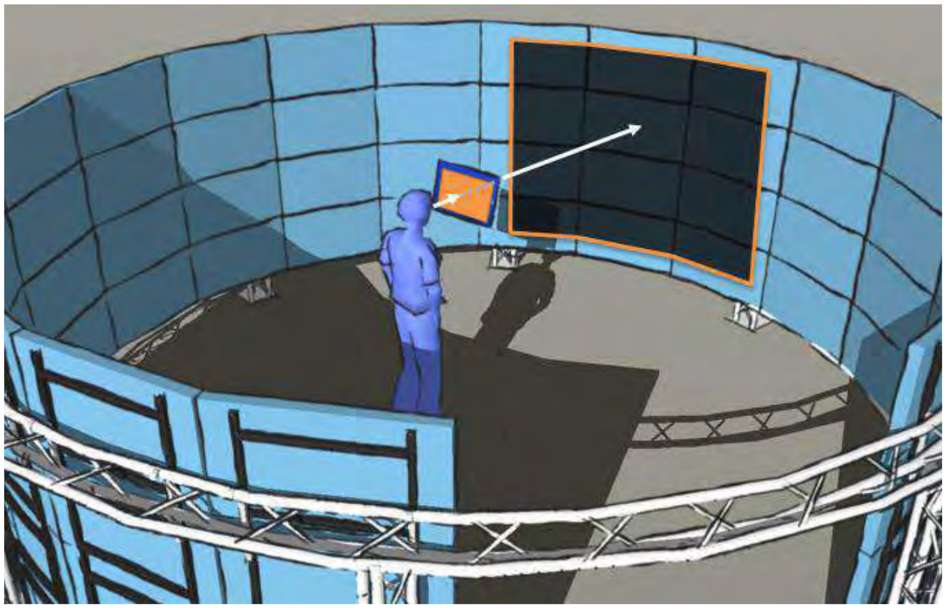


Figure 47. An example of device – application association. By tracking the tablet screen position, the application canvas position and the user's gaze direction, we can link an application to the device and advanced interface on the user's device.

Finally, besides the opportunity for extended user studies of MVI, an area deserving further exploration is the study of other interaction modalities and cross-display integrations between Hybrid Reality Environments and other systems. As an example, we could consider using a tablet device or augmented-reality display as a secondary input/output tool. This device can be associated to a specific application interactively through combined tracking of the device itself and the position of application viewports on screen (Figure 47). As users hold a device in front of them (or simply look toward an application using a head-mounted AR system), the runtime will identify which application can be seen through the user's device, based on its tracking data, size and user's head position/orientation. By touching a button or performing a gesture on the personal device, the user could associate it to an application, and get a personalized application interface. The device could then be used as a see-through interface providing additional information on specific areas of the visualization, it could be used as an annotation tool, or it could provide fine-grained input controls to the application. Techniques similar to the aforementioned ones have been investigated in the past ([87], [99]–[102]), but never in the context of a multi-view immersive system or hybrid reality environment.

REFERENCES

[1] J. V. R. C. Johnson, R. Ross, S. Ahern, J. Ahrens, W. Bethel, K. L. Ma, M. Papka and J. T. H. W. Shen, "Visualization and knowledge discovery", in *DOE/ASCR workshop on visual analysis and data exploration at extreme scale*, 2007.

[2] K. A. L. J. Cummings, T. Finholt, I. Foster, C. Kesselman, "Beyond being there: A blueprint for advancing the design, development, and evaluation of virtual organizations", 2008.

[3] C. Andrews, A. Endert, and C. North, "Space to think: large high-resolution displays for sensemaking", *Proceedings of the 28th international conference on Human factors in computing systems*, 2010.

[4] C. Andrews and C. North, "Analyst's Workspace: An embodied sensemaking environment for large, high-resolution displays", *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 123–131, Oct. 2012.

[5] R. Ball and C. North, "The effects of peripheral vision and physical navigation on large scale visualization", *Proceedings of graphics interface*, pp. 9–16, 2008.

[6] C. N. R. Ball, "Analysis of user behavior on high-resolution tiled displays", *Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction (INTERACT '05)*, pp. 350–363, 2005.

[7] R. Ball and C. North, "Effects of tiled high-resolution display on basic visualization and navigation tasks", in *CHI'05 extended abstracts on Human factors in Computing*, 2005.

[8] X. Bi and R. Balakrishnan, "Comparing usage of a large high-resolution display to single or dual desktop displays for daily work", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*, pp. 1005–1014, 2009.

[9] G. Czerwinski, T. Smith, Regan, B. Meyers, G. Robertson, and G. Starkweather, "Toward characterizing the productivity benefits of very large displays", *Interact*, pp. 9–16, 2003.

[10] D. S. Tan, D. Gergle, P. Scupelli, and P. R., "With similar visual angles, larger displays improve spatial performance", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*, pp. 217–224, 2003.

REFERENCES (continued)

[11] B. Yost, Y. Haciahmetoglu, and C. North, "Beyond visual acuity: the perceptual scalability of information visualizations for large displays", *Proceedings of the SIGCHI conference on Human Factors*, 2007.

[12] C. Cruz-Neira *et al.*, "Scientists in wonderland: A report on visualization applications in the CAVE virtual reality environment", in *Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium*, 1993, pp. 59–66.

[13] Y. Sun, J. Leigh, A. Johnson, and S. Lee, "Articulate: A semi-automated model for translating Natural Language queries into meaningful visualizations", *Proceedings of 10th International Symposium of Smart Graphics (SG), Lecture Notes in Computer Science*, 2010.

[14] D. Schikore, R. Fischer, and R. Frank, "High-resolution multiprojector display walls", *IEEE Computer*, 2000.

[15] J. Leigh *et al.*, "Scalable Resolution Display Walls", *Proceedings of the IEEE*, pp. 1–15, 2012.

[16] C. Papadopoulos, K. Petkov, and A. Kaufman, "The Reality Deck-an Immersive Gigapixel Display.", *IEEE computer graphics*, 2015.

[17] C. Papadopoulos, B. Laha, and A. Kaufman, "Interacting with Mixed Reality Systems", *Death of the Desktop:*, 2014.

[18] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The CAVE: audio visual experience automatic virtual environment", *Communications of the ACM*, vol. 35, no. 6, pp. 64–72, Jun. 1992.

[19] A. Johnson, J. Leigh, and P. Morin, "GeoWall: stereoscopic visualization for geoscience research and education", *IEEE Computer Graphics and Applications*, 2006.

[20] T. a. T. DeFanti *et al.*, "The StarCAVE, a third-generation CAVE and virtual reality OptIPortal", *Future Generation Computer Systems*, vol. 25, no. 2, pp. 169–178, Feb. 2009.

[21] T. Höllerer, J. Kuchera-Morin, and X. Amatriain, "The allosphere: a large-scale immersive surround-view instrument", *Proceedings of the 2007*, 2007.

REFERENCES (continued)

[22] X. Amatriain and J. Kuchera-Morin, "The allosphere: Immersive multimedia for scientific discovery and artistic exploration", *IEEE*, 2009.

[23] "The CORNEA immersive virtual environment. King Abdullah University of Science and Technology", *http://kvl.kaust.edu.sa/Pages/CORNEA.aspx*.

[24] N. Polys and C. North, "Snap2Diverse: coordinating information visualizations and virtual environments", *SPIE 5295, Visualization and Data Analysis 2004*, 2004.

[25] D. Raja, D. Bowman, J. Lucas, and C. North, "Exploring the benefits of immersion in abstract information visualization", *Proceedings of the Immersive Projection Technologies Workshop*, 2004.

[26] H. Smallman and M. S. John, "Information availability in 2D and 3D displays", *IEEE Computer Graphics and Applications*, 2001.

[27] R. Springmeyer, M. Blattner, and N. Max, "A characterization of the scientific data analysis process", *Proceedings of the 3rd conference on Visualization*, 1992.

[28] M. Tory and A. Kirkpatrick, "Visualization task performance with 2D, 3D, and combination displays", *Visualization and …*, 2006.

[29] N. F. Polys and D. A. Bowman, "Design and display of enhancing information in desktop information-rich virtual environments: challenges and techniques", *Virtual Reality*, vol. 8, no. 1, pp. 41–54, Jun. 2004.

[30] A. Febretti *et al.*, "The OmegaDesk: towards a hybrid 2D and 3D work desk", *Advances in Visual Computing*, pp. 13–23, 2011.

[31] T. a. DeFanti *et al.*, "The future of the CAVE", *Central European Journal of Engineering*, vol. 1, no. 1, pp. 16–37, Nov. 2010.

[32] C. Andrews, A. Endert, B. Yost, and C. North, "Information visualization on large, high-resolution displays: Issues, challenges, and opportunities", *Information Visualization*, 2011.

[33] M. Nieminen, M. Tyllinen, and M. Runonen, "Digital War Room for Design", *Lecture Notes in Computer Science*, pp. 352–361, 2013.

REFERENCES (continued)

[34] "CAVE2 Mechdyne", *https://www.mechdyne.com/hardware.aspx?name=CAVE2*.

[35] "Cyber-CANOE", *http://www.hawaii.edu/news/2014/08/18/cyber-canoe-to-explore-worlds-of-data-in-3-d/*.

[36] "New WAVE Display Technology Rises at UCSD", *http://www.calit2.net/newsroom/article.php?id=2273*.

[37] S. Teasley, L. Covi, M. Krishnan, and J. Olson, "How does radical collocation help a team succeed?", *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, 2000.

[38] D. Angelo, G. Wesche, M. Foursa, M. Bogen, and D. d'Angelo, "The Benefits of Co-located Collaboration and Immersion on Assembly Modeling in Virtual Environments", *Advances in Visual Computing*, pp. 478–487, 2008.

[39] T. C. Hutchinson, Æ. F. Kuester, and T. H. Æ. R. Chadwick, "A hybrid reality environment and its application to the study of earthquake engineering", *Virtual Reality*, pp. 17–33, 2006.

[40] T. Allen, "Managing the flow of technology: Technology transfer and the dissemination of technological information within the R&D organization", *MIT Press Books*, 1984.

[41] E. Hutchins and L. Palen, "Constructing meaning from space, gesture, and speech", *NATO ASI Series F Computer and Systems Sciences*, 1997.

[42] R. Wilhelmson, P. Baker, R. Stein, and R. Heiland, "Large Tiled Display Walls and Applications in Metereology, Oceanography and Hydrology", *IEEE Computer Graphics And Applications*, pp. 12–14.

[43] D. Vohl *et al.*, "Large-scale comparative visualisation of sets of multidimensional data", *PeerJ Computer Science*, vol. 2, p. e88, Oct. 2016.

[44] G. P. Johnson, G. D. Abram, B. Westing, P. Navr'til, and K. Gaither, "DisplayCluster: An Interactive Visualization Environment for Tiled Displays", *2012 IEEE International Conference on Cluster Computing*, no. Figure 1, pp. 239–247, Sep. 2012.

[45] M. Beaudouin-Lafon, "Lessons learned from the wild room, a multisurface interactive environment", *23rd French Speaking Conference on Human-Computer Interaction*, 2011.

REFERENCES (continued)

[46] R. Jagodic, "Collaborative Interaction And Display Space Organization In Large High-Resolution Environments", *Ph.D. Dissertation*, 2012.

[47] A. Bornik, R. Beichel, and D. Schmalstieg, "Interactive editing of segmented volumetric datasets in a hybrid 2D/3D virtual environment", *Proceedings of the ACM symposium on Virtual reality software and technology*, 2006.

[48] B. Yost, Y. Haciahmetoglu, and C. North, "Beyond visual acuity: the perceptual scalability of information visualizations for large displays", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*, pp. 101–110, 2007.

[49] C. Papadopoulos, S. Mirhosseini, I. Gutenko, K. Petkov, A. E. Kaufman, and B. Laha, "Scalability limits of large immersive high-resolution displays", in *2015 IEEE Virtual Reality (VR)*, 2015, pp. 11–18.

[50] H. Chung, C. Andrews, and C. North, "A Survey of Software Frameworks for Cluster-Based Large High-Resolution Displays", *IEEE transactions on visualization and computer graphics*, pp. 1–20, 2013.

[51] G. Humphreys, M. Eldridge, and I. Buck, "WireGL: a scalable graphics system for clusters", *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 129–140, 2001.

[52] G. Humphreys, M. Houston, and R. Ng, "Chromium: a stream-processing framework for interactive rendering on clusters", *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002*, 2002.

[53] B. Neal, P. Hunkin, and A. McGregor, "Distributed OpenGL rendering in network bandwidth constrained environments", 2011.

[54] L. Renambot, A. Rao, and R. Singh, "SAGE: the scalable adaptive graphics environment", *Proceedings of WACE*, vol. 9, no. 23, 2004.

[55] T. Marrinan, J. Aurisano, and A. Nishimoto, "SAGE2: A new approach for data intensive collaboration using Scalable Resolution Shared Displays", *Collaborative*, 2014.

[56] L. Renambot *et al.*, "SAGE2: A collaboration portal for scalable resolution displays", *Future Generation Computer Systems*, vol. 54, pp. 296–305, 2016.

REFERENCES (continued)

[57] T. Marrinan, "Data-Intensive Remote Collaboration using Scalable Visualizations in Heterogeneous Display Spaces", University of Illinois at Chicago, 2016.

[58] O. Industries, "Oblong Mezzanine", *http://www.oblong.com/mezzanine*.

[59] "Bluescape", *https://www.bluescape.com/*.

[60] "Multitaction Cornerstone", *https://www.multitaction.com/products/mt-cornerstone-sdk*.

[61] S. Eilemann, "Equalizer: A scalable parallel rendering framework", *IEEE transactions on visualization and computer graphics*, vol. 15, 2009.

[62] K.-U. Doerr and F. Kuester, "CGLX: a scalable, high-performance visualization framework for networked display environments.", *IEEE transactions on visualization and computer graphics*, vol. 17, no. 3, pp. 320–32, Mar. 2011.

[63] K. Ponto, K. Doerr, and T. Wypych, "CGLXTouch: A multi-user multi-touch approach for ultra-high-resolution collaborative workspaces", *Future Generation Computer Systems*, 2011.

[64] M. Snir, S. Otto, and D. Walker, *MPI: the complete reference*. 1995.

[65] J. Allard and B. Raffin, "A Shader-Based Parallel Rendering Framework", *IEEE Visualization 2005 - (VIS'05)*, pp. 17–17, 2005.

[66] D. Reiners, G. Voß, and J. Behr, "OpenSG: Basic Concepts".

[67] M. Roth, G. Voss, and D. Reiners, "Multi-threading and clustering for scene graph systems", *Computers & Graphics*, 2004.

[68] R. Osfield and D. Burns, "Open Scene Graph", *http://www.openscenegraph.com*, 2004.

[69] P. Harish and P. Narayanan, "Garuda: A scalable tiled display wall using commodity PCs", *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 864–877, 2007.

REFERENCES (continued)

[70] P. Harish and P. Narayanan, "Culling an object hierarchy to a frustum hierarchy", *Computer Vision, Graphics and Image Processing*, pp. 252–263, 2006.

[71] E. Pietriga, S. Huot, M. Nancel, and R. Primet, "Rapid development of user interfaces on cluster-driven wall displays with jBricks", *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, no. June, 2011.

[72] E. Pietriga, "A toolkit for addressing hci issues in visual language environments", *Visual Languages and Human-Centric Computing*, 2005.

[73] M. Kaltenbrunner and T. Bovermann, "TUIO: A protocol for table-top tangible user interfaces", *Proceedings of the The 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*, pp. 1–5, 2005.

[74] T. Gjerlufsen, C. Klokmose, and J. Eagan, "Shared substance: developing flexible multi-surface applications", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 3383–3392.

[75] J. Mangan, D. Srour, and F. Kuester, "MediaCommons: A Scalable Abstraction for Tiled Display Environments", *sc13.supercomputing.org*.

[76] M. Szymanski, "CAVELIB Support For PC Visualization Clusters", *Advanced Imaging*, 2004.

[77] C. Cruz-Neira, D. Sandin, and T. DeFanti, "Surround-screen projection-based virtual reality: the design and implementation of the CAVE", *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 1993.

[78] W. Sherman, "FreeVR", 2005.

[79] W. Sherman, D. Coming, and S. Su, "FreeVR: honoring the past, looking to the future", *IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics*, 2013.

[80] P. McDowell and R. Darken, "Delta3D: a complete open source game and simulation engine for building military training systems", *The Journal of Defense …The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, pp. 143–154, 2006.

REFERENCES (continued)

[81] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira, "VR Juggler: A virtual platform for virtual reality application development", *Virtual Reality, 2001. Proceedings. IEEE*, pp. 89–96, 2001.

[82] "The Visualization Toolkit", *http://www.vtk.org/*.

[83] J. Schulze, A. Prudhomme, P. Weber, and T. A. Defanti, "CalVR: an advanced open source virtual reality software framework", *SPIE 8649 The Engineering Reality of Virtual Reality*, 2013.

[84] J. Schulze and D. Acevedo, "Democratizing rendering for multiple viewers in surround vr systems", *IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 77–80, 2012.

[85] R. Kuck, J. Wind, K. Riege, and M. Bogen, "Improving the avango vr/ar framework: Lessons learned", *Workshop Virtuelle und Erweiterte Realität*, 2008.

[86] T. Holtkämper, S. Scholz, A. Dressler, A. Manfred, and M. Bogen, "Co-located collaborative use of virtual environments", *Proceedings AAPG Annual Convention and Exhibition.*, pp. 1–6, 2007.

[87] P. George, "Nomad devices for interactions in immersive virtual environments", in *IS&T/SPIE The Engineering Reality of Virtual Reality 2013*, 2013.

[88] K. Richmond *et al.*, "Sub-Ice Exploration of an antarctic lake: results from the Endurance Project", in *17th International Symposium on Unmanned Untethered Submersible Technology (UUST11)*, 2011.

[89] "GALFORM Firefly", *http://galaxies.northwestern.edu/firefly/*.

[90] "UTS Data Arena Virtual Machine", *http://www.dataarena.net/all_tutorials/houdini-tutorial-series/getting-started-with-houdini-engine/*.

[91] O. L. Kaluza, "LavaVR: LavaVu for VR Systems", *https://github.com/mivp/LavaVR*.

[92] "ffenc: FFMPEG H.264 encoder for Omegalib", *https://github.com/cruxeon/ffenc*.

[93] "Omegalib Github Repository", *https://github.com/uic-evl/omegalib/*.

REFERENCES (continued)

[94] "Omegalib Modules project".

[95] "Omegalib Multiview Immersion Module", *https://github.com/omega-hub/mvi*.

[96] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-D rigid body transformations: a comparison of four major algorithms", *Machine Vision and Applications*, vol. 9, pp. 272–290, 1997.

[97] Q. Sun, L. Wei, and A. Kaufman, "Mapping virtual and physical reality", *ACM Transactions on Graphics (TOG)*, 2016.

[98] A. Ye *et al.*, "A Virtual Reality Exploration of Brain Connectivity Metrics using Dimensionality Reduction", in *BIOLOGICAL PSYCHIATRY*, 2015, vol. 77, no. 9, p. 328S--329S.

[99] M. Tsang and G. Fitzmaurice, "Boom chameleon: simultaneous capture of 3D viewpoint, voice and gesture annotations on a spatially-aware display", *Proceedings of the 15th annual ACM symposium on User interface software and technology*, vol. 4, no. 2, pp. 111–120, 2002.

[100] R. Aspin and K. H. Le, "Augmenting the CAVE: An Initial Study into Close Focused, Inward Looking, Exploration in IPT Systems", *11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'07)*, pp. 217–224, Oct. 2007.

[101] N. Kukimoto and J. Nonaka, "Scientific visualization in collaborative virtual environment with PDA-based control and 3D annotation", *JSME International Journal Series B Fluids and Thermal Engineering*, vol. 48, no. 2, pp. 252–258, 2005.

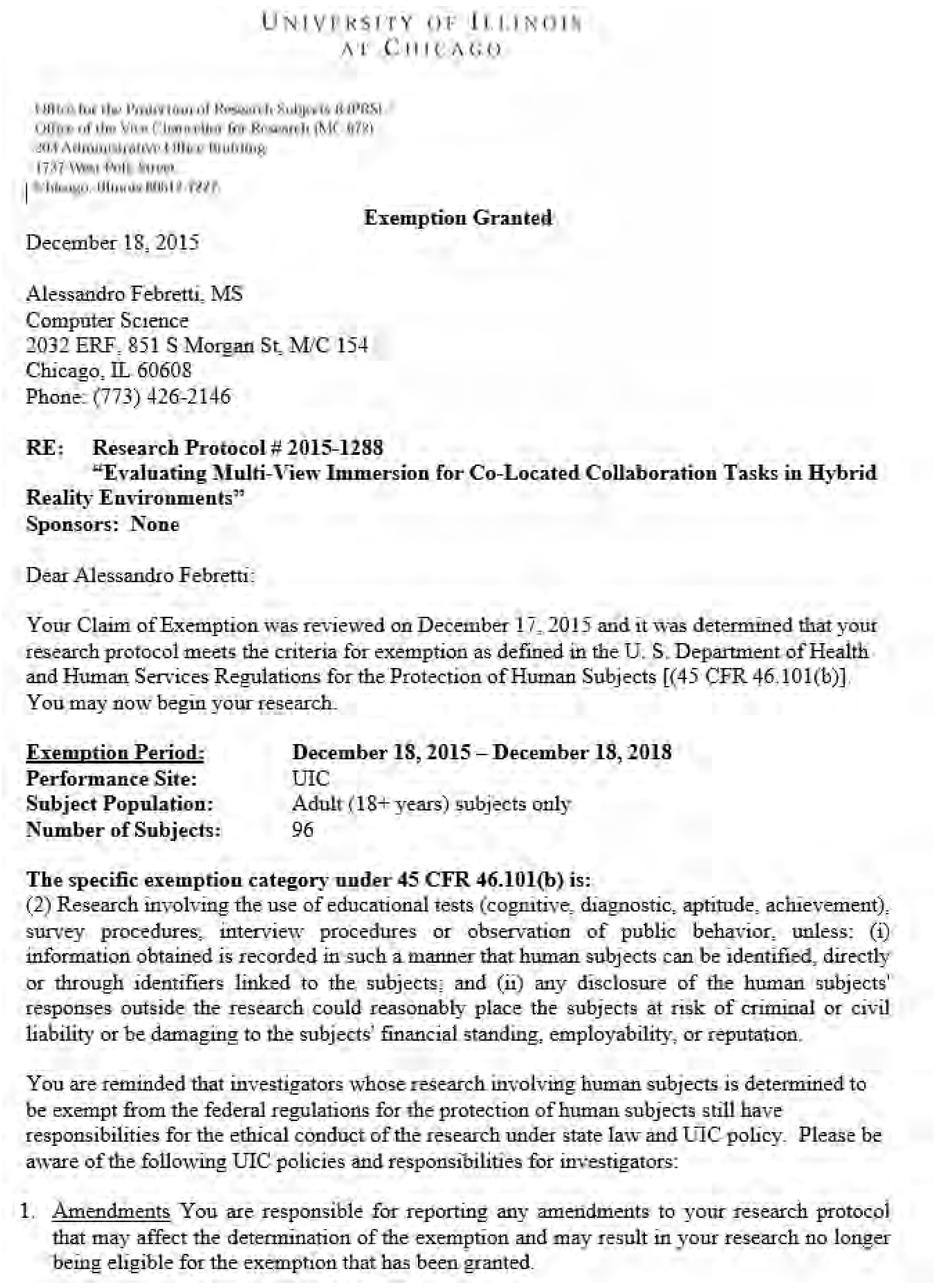
[102] F. Berwein, "Superimposed Displays", 2009.

APPENDICES

APPENDIX A

User Study Material

IRB Approval



UNIVERSITY OF ILLINOIS
AT CHICAGO

Office for the Protection of Research Subjects (OPRS)
Office of the Vice Chancellor for Research (MC 672)
203 Administrative Office Building
1737 West Polk Street
Chicago, Illinois 60612-7227

Exemption Granted

December 18, 2015

Alessandro Febretti, MS
Computer Science
2032 ERF, 851 S Morgan St, M/C 154
Chicago, IL 60608
Phone: (773) 426-2146

RE: Research Protocol # 2015-1288
"Evaluating Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments"
Sponsors: None

Dear Alessandro Febretti:

Your Claim of Exemption was reviewed on December 17, 2015 and it was determined that your research protocol meets the criteria for exemption as defined in the U. S. Department of Health and Human Services Regulations for the Protection of Human Subjects [(45 CFR 46.101(b)]. You may now begin your research.

<u>Exemption Period:</u>	**December 18, 2015 – December 18, 2018**
Performance Site:	UIC
Subject Population:	Adult (18+ years) subjects only
Number of Subjects:	96

The specific exemption category under 45 CFR 46.101(b) is:
(2) Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, unless: (i) information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; and (ii) any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

You are reminded that investigators whose research involving human subjects is determined to be exempt from the federal regulations for the protection of human subjects still have responsibilities for the ethical conduct of the research under state law and UIC policy. Please be aware of the following UIC policies and responsibilities for investigators:

1. <u>Amendments</u> You are responsible for reporting any amendments to your research protocol that may affect the determination of the exemption and may result in your research no longer being eligible for the exemption that has been granted.

APPENDIX A (continued)

2. Record Keeping You are responsible for maintaining a copy all research related records in a secure location in the event future verification is necessary, at a minimum these documents include: the research protocol, the claim of exemption application, all questionnaires, survey instruments, interview questions and/or data collection instruments associated with this research protocol, recruiting or advertising materials, any consent forms or information sheets given to subjects, or any other pertinent documents.
3. Final Report When you have completed work on your research protocol, you should submit a final report to the Office for Protection of Research Subjects (OPRS).
4. Information for Human Subjects UIC Policy requires investigators to provide information about the research protocol to subjects and to obtain their permission prior to their participating in the research. The information about the research protocol should be presented to subjects in writing or orally from a written script. When appropriate, the following information must be provided to all research subjects participating in exempt studies:
   a. The researchers affiliation; UIC, JBVMAC or other institutions,
   b. The purpose of the research,
   c. The extent of the subject's involvement and an explanation of the procedures to be followed,
   d. Whether the information being collected will be used for any purposes other than the proposed research,
   e. A description of the procedures to protect the privacy of subjects and the confidentiality of the research information and data,
   f. Description of any reasonable foreseeable risks,
   g. Description of anticipated benefit,
   h. A statement that participation is voluntary and subjects can refuse to participate or can stop at any time,
   i. A statement that the researcher is available to answer any questions that the subject may have and which includes the name and phone number of the investigator(s).
   j. A statement that the UIC IRB/OPRS or JBVMAC Patient Advocate Office is available if there are questions about subject's rights, which includes the appropriate phone numbers.

Please be sure to:

→Use your research protocol number (listed above) on any documents or correspondence with the IRB concerning your research protocol.

We wish you the best as you conduct your research. If you have any questions or need further help, please contact me at (312) 355-2908 or the OPRS office at (312) 996-1711. Please send any correspondence about this protocol to OPRS at 203 AOB, M/C 672.

Sincerely,

Charles W. Hoehne, B.S., C.I.P.
Assistant Director, IRB #7
Office for the Protection of Research Subjects

APPENDIX A (continued)

Recruitment Material

Evaluating Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments

Volunteers are needed for a research study being conducted by researchers at the University of Illinois at Chicago (Protocol 2015-1288)

The objective of this study is evaluating techniques for supporting collaborative work inside a large-scale immersive environment. As a participant, you would interact with scientific data on a large 3D display as well as communicate with other participants. After the trial is completed, you would undertake a short survey.

You must be over 18 years old to participate in this study.

A trial for this study will take you approximately 2 hours to complete, divided into three sections. The study will be audio recorded and videotaped for data analysis. Data collected through this study will remain confidential.

If you are interested, please send e-mail to: afebre2@uic.edu

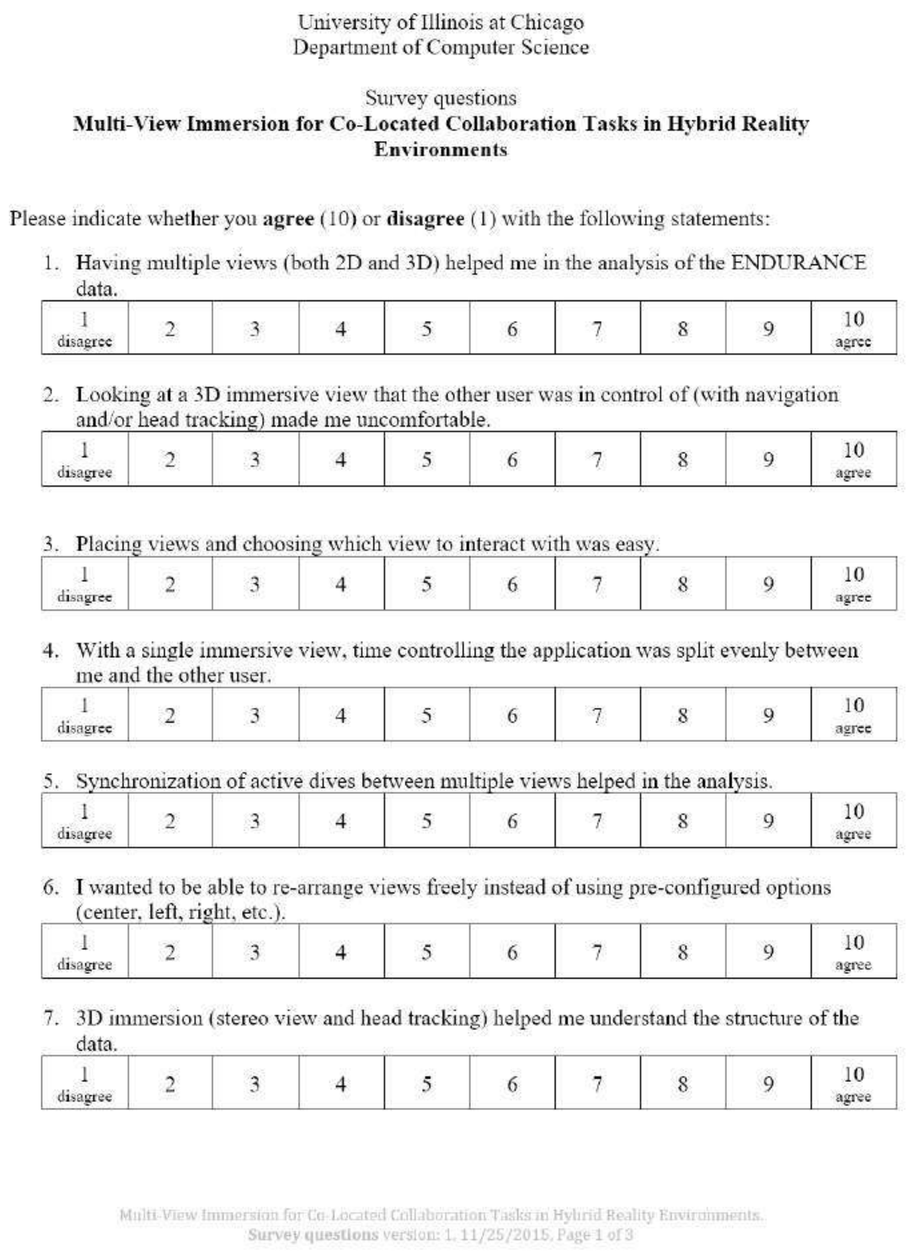
Questions and concerns regarding this research should be directed to:

Principal investigator: Alessandro Febretti
afebre2@uic.edu
Phone: 773-426-2146

Faculty sponsor: Andrew E. Johnson
ajohnson@uic.edu
Phone: 312-996-3002

APPENDIX A (continued)

Survey



University of Illinois at Chicago
Department of Computer Science

Survey questions
Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments

Please indicate whether you **agree** (10) or **disagree** (1) with the following statements:

1. Having multiple views (both 2D and 3D) helped me in the analysis of the ENDURANCE data.

1 disagree	2	3	4	5	6	7	8	9	10 agree

2. Looking at a 3D immersive view that the other user was in control of (with navigation and/or head tracking) made me uncomfortable.

1 disagree	2	3	4	5	6	7	8	9	10 agree

3. Placing views and choosing which view to interact with was easy.

1 disagree	2	3	4	5	6	7	8	9	10 agree

4. With a single immersive view, time controlling the application was split evenly between me and the other user.

1 disagree	2	3	4	5	6	7	8	9	10 agree

5. Synchronization of active dives between multiple views helped in the analysis.

1 disagree	2	3	4	5	6	7	8	9	10 agree

6. I wanted to be able to re-arrange views freely instead of using pre-configured options (center, left, right, etc.).

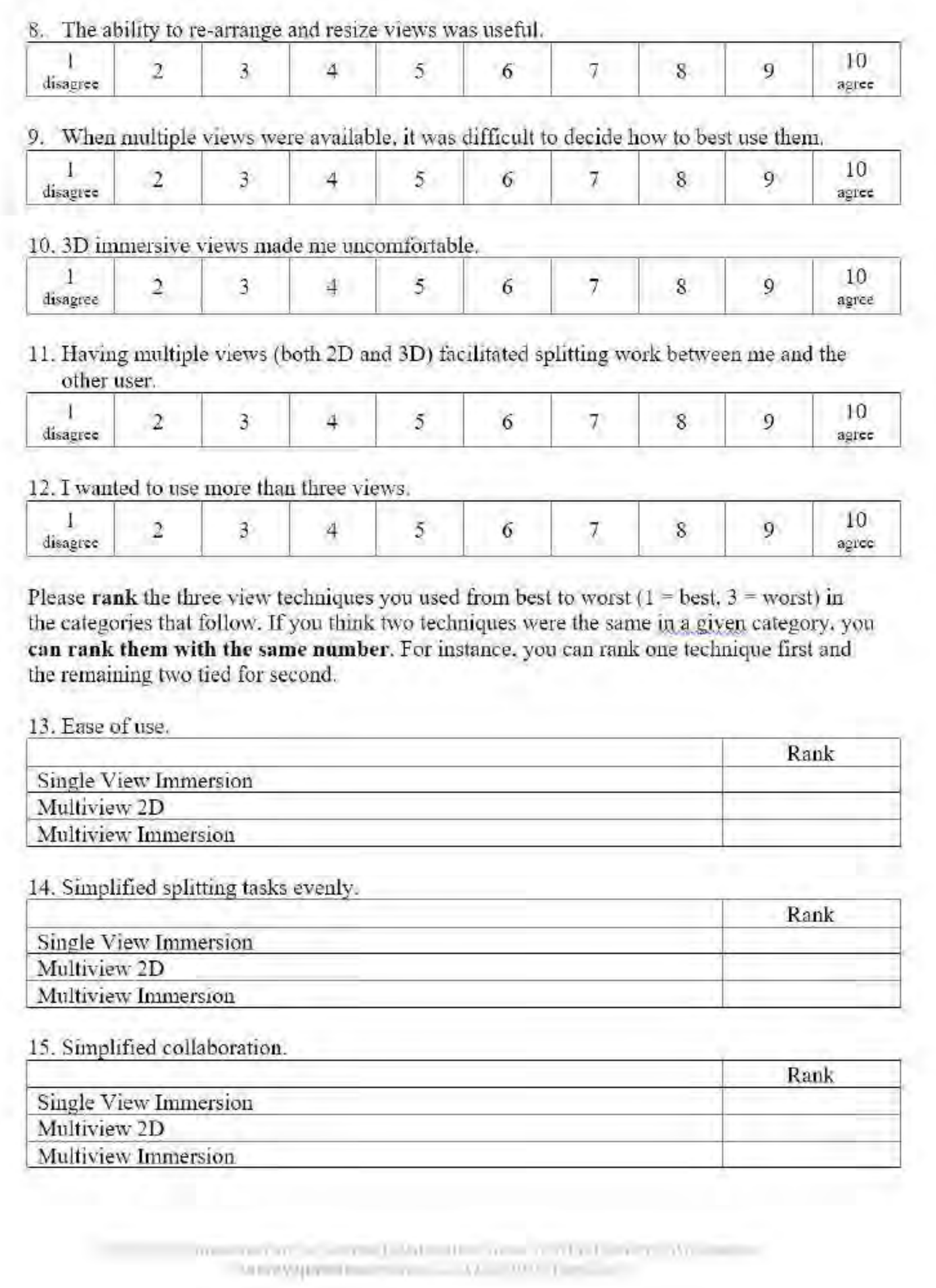
1 disagree	2	3	4	5	6	7	8	9	10 agree

7. 3D immersion (stereo view and head tracking) helped me understand the structure of the data.

1 disagree	2	3	4	5	6	7	8	9	10 agree

Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments.
Survey questions version: 1, 11/25/2015, Page 1 of 3

APPENDIX A (continued)



8. The ability to re-arrange and resize views was useful.

1 disagree	2	3	4	5	6	7	8	9	10 agree

9. When multiple views were available, it was difficult to decide how to best use them.

1 disagree	2	3	4	5	6	7	8	9	10 agree

10. 3D immersive views made me uncomfortable.

1 disagree	2	3	4	5	6	7	8	9	10 agree

11. Having multiple views (both 2D and 3D) facilitated splitting work between me and the other user.

1 disagree	2	3	4	5	6	7	8	9	10 agree

12. I wanted to use more than three views.

1 disagree	2	3	4	5	6	7	8	9	10 agree

Please **rank** the three view techniques you used from best to worst (1 = best, 3 = worst) in the categories that follow. If you think two techniques were the same in a given category, you **can rank them with the same number**. For instance, you can rank one technique first and the remaining two tied for second.

13. Ease of use.

	Rank
Single View Immersion	
Multiview 2D	
Multiview Immersion	

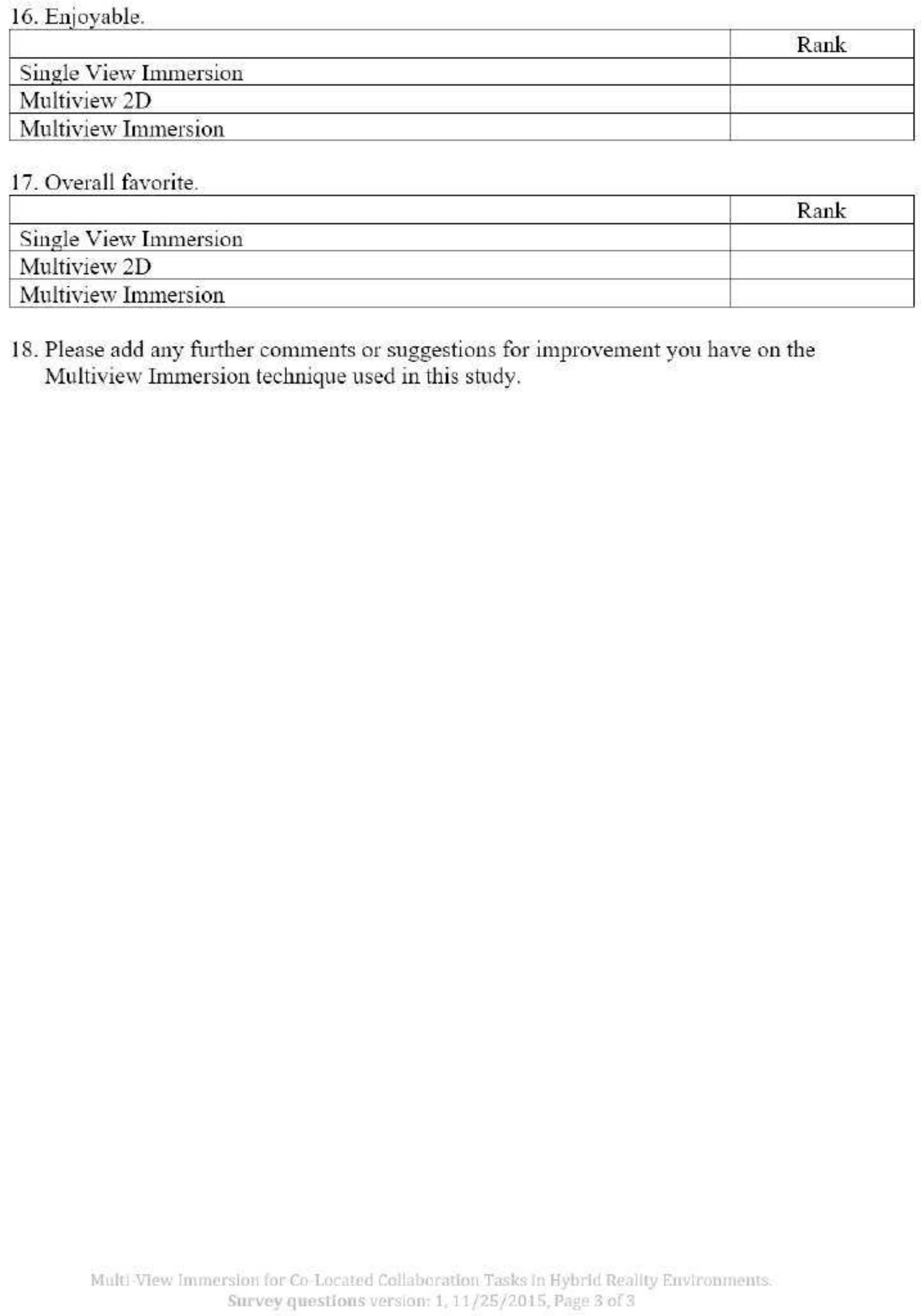
14. Simplified splitting tasks evenly.

	Rank
Single View Immersion	
Multiview 2D	
Multiview Immersion	

15. Simplified collaboration.

	Rank
Single View Immersion	
Multiview 2D	
Multiview Immersion	

APPENDIX A (continued)



16. Enjoyable.

	Rank
Single View Immersion	
Multiview 2D	
Multiview Immersion	

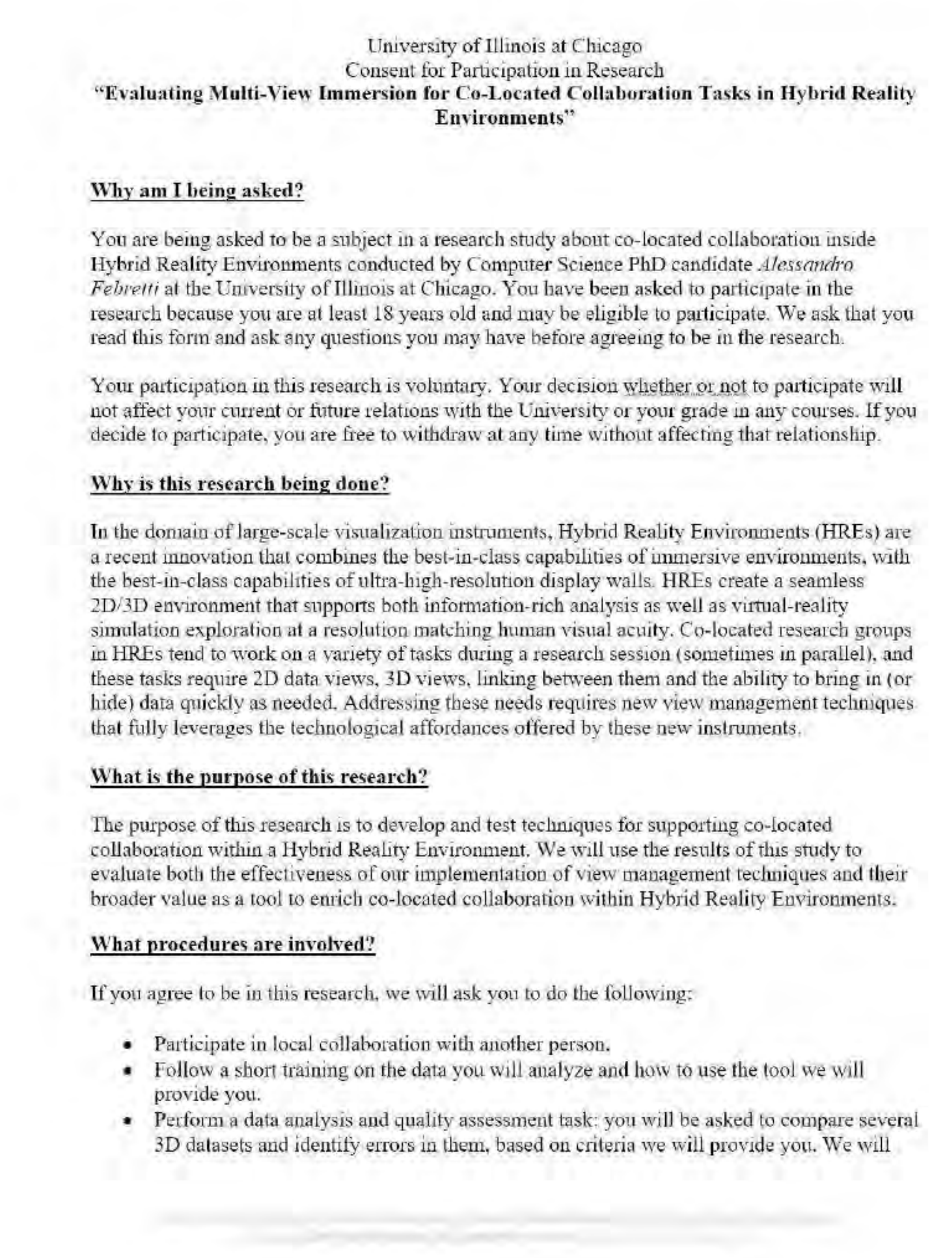
17. Overall favorite.

	Rank
Single View Immersion	
Multiview 2D	
Multiview Immersion	

18. Please add any further comments or suggestions for improvement you have on the Multiview Immersion technique used in this study.

APPENDIX A (continued)

<u>Consent Form</u>



University of Illinois at Chicago
Consent for Participation in Research
"Evaluating Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments"

<u>Why am I being asked?</u>

You are being asked to be a subject in a research study about co-located collaboration inside Hybrid Reality Environments conducted by Computer Science PhD candidate *Alessandro Febretti* at the University of Illinois at Chicago. You have been asked to participate in the research because you are at least 18 years old and may be eligible to participate. We ask that you read this form and ask any questions you may have before agreeing to be in the research.

Your participation in this research is voluntary. Your decision whether or not to participate will not affect your current or future relations with the University or your grade in any courses. If you decide to participate, you are free to withdraw at any time without affecting that relationship.

<u>Why is this research being done?</u>

In the domain of large-scale visualization instruments, Hybrid Reality Environments (HREs) are a recent innovation that combines the best-in-class capabilities of immersive environments, with the best-in-class capabilities of ultra-high-resolution display walls. HREs create a seamless 2D/3D environment that supports both information-rich analysis as well as virtual-reality simulation exploration at a resolution matching human visual acuity. Co-located research groups in HREs tend to work on a variety of tasks during a research session (sometimes in parallel), and these tasks require 2D data views, 3D views, linking between them and the ability to bring in (or hide) data quickly as needed. Addressing these needs requires new view management techniques that fully leverages the technological affordances offered by these new instruments.

<u>What is the purpose of this research?</u>

The purpose of this research is to develop and test techniques for supporting co-located collaboration within a Hybrid Reality Environment. We will use the results of this study to evaluate both the effectiveness of our implementation of view management techniques and their broader value as a tool to enrich co-located collaboration within Hybrid Reality Environments.

<u>What procedures are involved?</u>

If you agree to be in this research, we will ask you to do the following:

- Participate in local collaboration with another person.
- Follow a short training on the data you will analyze and how to use the tool we will provide you.
- Perform a data analysis and quality assessment task: you will be asked to compare several 3D datasets and identify errors in them, based on criteria we will provide you. We will

ask you to repeat this task three times, with different datasets. The duration for this part of the study will be approximately 1 hour.
- Undertake a short survey after you finish all three methods.
- During the study, you will be video and audio taped. We also ask that you try to keep talking out loud while trying to solve the task. Try to say anything that goes through your head. You will wear 3D-tracked glasses during each task. We ask you to keep the glasses on for the duration of the study.
- If you have questions during the study, please feel free to ask the researcher at any time. You may also take breaks betweek tasks during the study.

Approximately 30 subjects will be involved in this research at the University of Illinois at Chicago.

What are the potential risks and discomforts?

To the best of our knowledge, the things you will be doing have no more risk of harm than you would experience in everyday life. A risk of this research is a loss of privacy (revealing to others that you are taking part in this study) or confidentiality (revealing information about you to others to whom you have not given permission to see this information). Your information will remain confidential, and is not linked to your performance with the University of Illinois at Chicago. An additional risk associated with viewing 3D stereoscopic content involves the potential for discomfort, eyestrain, headaches and/or dizziness. There are no long lasting effects, and the above symptoms only occur in a small subset of the general population. The risk of experiencing any of the above symptoms is no greater than when attending a 3D movie in a theater, or watching a 3D movie on a 3D TV. If you ever feel any of the above symptoms during the study, please remove the 3D glasses and look away from the display.

Are there benefits to taking part in the research?

You will not directly benefit from participation in this study. By participating in this research, however, you will also be contributing to the improvement of visualization software for Hybrid Reality Environments.

Will I be reimbursed for any of my expenses or paid for my participation in this research?

You will not be reimbursed or compensated for your participation in this research.

What are the costs for participating in this research?

There is not cost to you for participating in this study.

What about privacy and confidentiality?

The only people who will know that you are a research subject are members of the research team and fellow participants. No information about you, or provided by you during the research, will be disclosed to others without your written permission, except:

APPENDIX A (continued)

- When necessary to protect your rights or welfare (for example, if you are injured and need emergency care or when the UIC Institutional Review Board monitors the research or consent process); or
- When required by law.

When the results of the research are published or discussed in conferences, no information will be included that would reveal your identity. If photographs, videos, or audiotape recordings of you will be used for educational purposes, your identity will be protected or disguised. Any information that is obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission or as required by law.

- All identifying data, including images, recordings, and questionairs will be kept under lock and key at the Electronic Visualization Laboratory for the duration of the study. Access to this data will be restricted to the prinicipal investigator, *Alessandro Febretti*, and his co-investigators.
- All other information that might identify you, such as response times, will be labeled with a numerical identifier to maintain your anonymity. The index of study participant names and numbers will be kept under lock in the Elecronic Visualization Laboratory.
- We will request additional consent from you if we desire to use identifying images or recordings of you in a publication or for public presentation.
- All identifying images, recordings, and questionnaire results will be destroyed once the data has been fully analyzed or within one year, whichever comes first.

Can I withdraw or be removed from the study?

You can choose whether to be in this study or not. If you volunteer to be in this study, you may withdraw at any time without consequences of any kind. You may also refuse to answer any questions you don't want to answer and still remain in the study. The investigator may withdraw you from this research if circumstances arise which warrant doing so.

Who should I contact if I have questions?

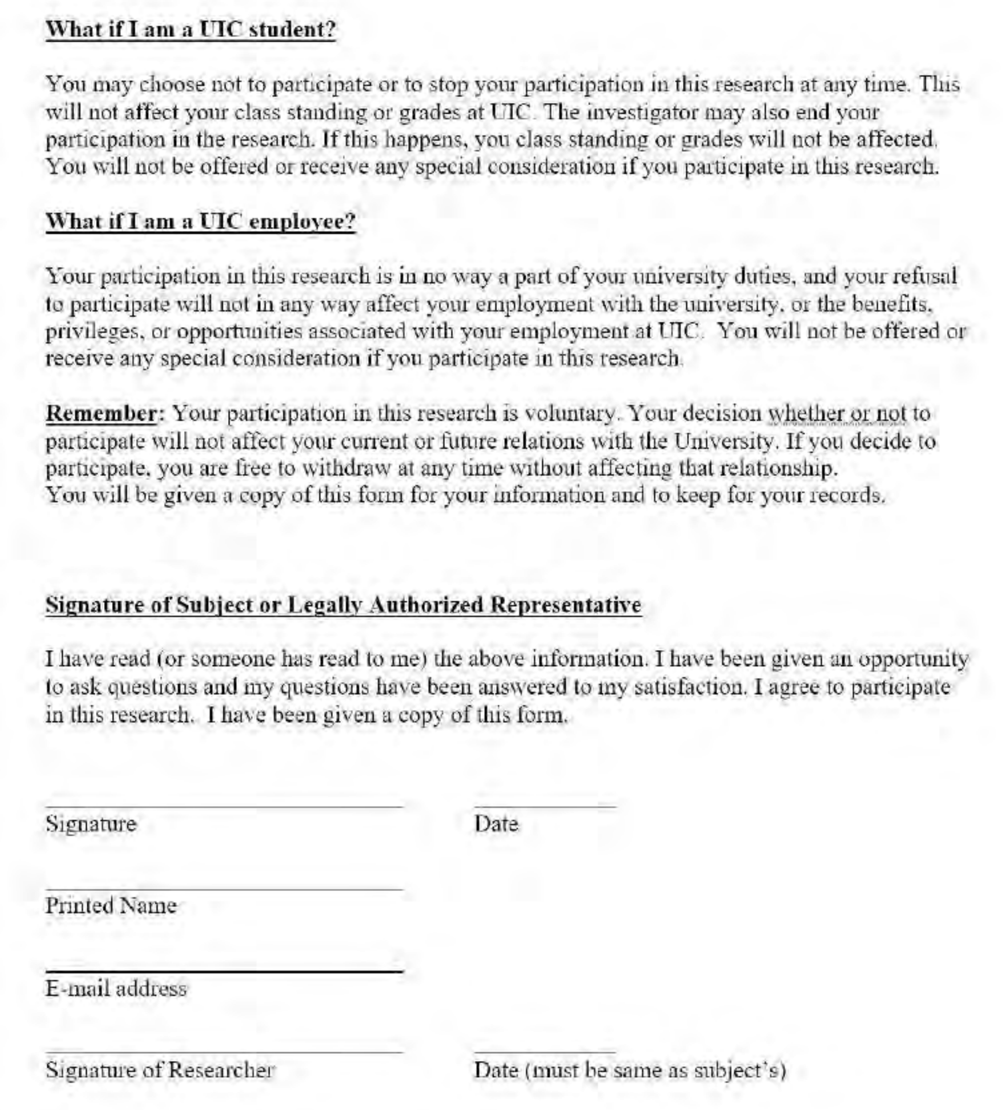
The researcher conducting this study is *Alessandro Febretti*. You may ask any questions you have now. If you have questions later, you may contact him at:
Phone: 773-426-2146 Email: afebre2@uic.edu

The faculty sponsor of this research is Associate Professor *Andrew E. Johnson*. You may contact him at:
Phone: 312-996-3002 Email: ajohnson@uic.edu

What are my rights as a research subject?

If you have any questions about your rights as a research subject, you may call the Office for Protection of Research Subjects at 312-996-1711.

APPENDIX A (continued)



What if I am a UIC student?

You may choose not to participate or to stop your participation in this research at any time. This will not affect your class standing or grades at UIC. The investigator may also end your participation in the research. If this happens, you class standing or grades will not be affected. You will not be offered or receive any special consideration if you participate in this research.

What if I am a UIC employee?

Your participation in this research is in no way a part of your university duties, and your refusal to participate will not in any way affect your employment with the university, or the benefits, privileges, or opportunities associated with your employment at UIC. You will not be offered or receive any special consideration if you participate in this research.

Remember: Your participation in this research is voluntary. Your decision whether or not to participate will not affect your current or future relations with the University. If you decide to participate, you are free to withdraw at any time without affecting that relationship. You will be given a copy of this form for your information and to keep for your records.

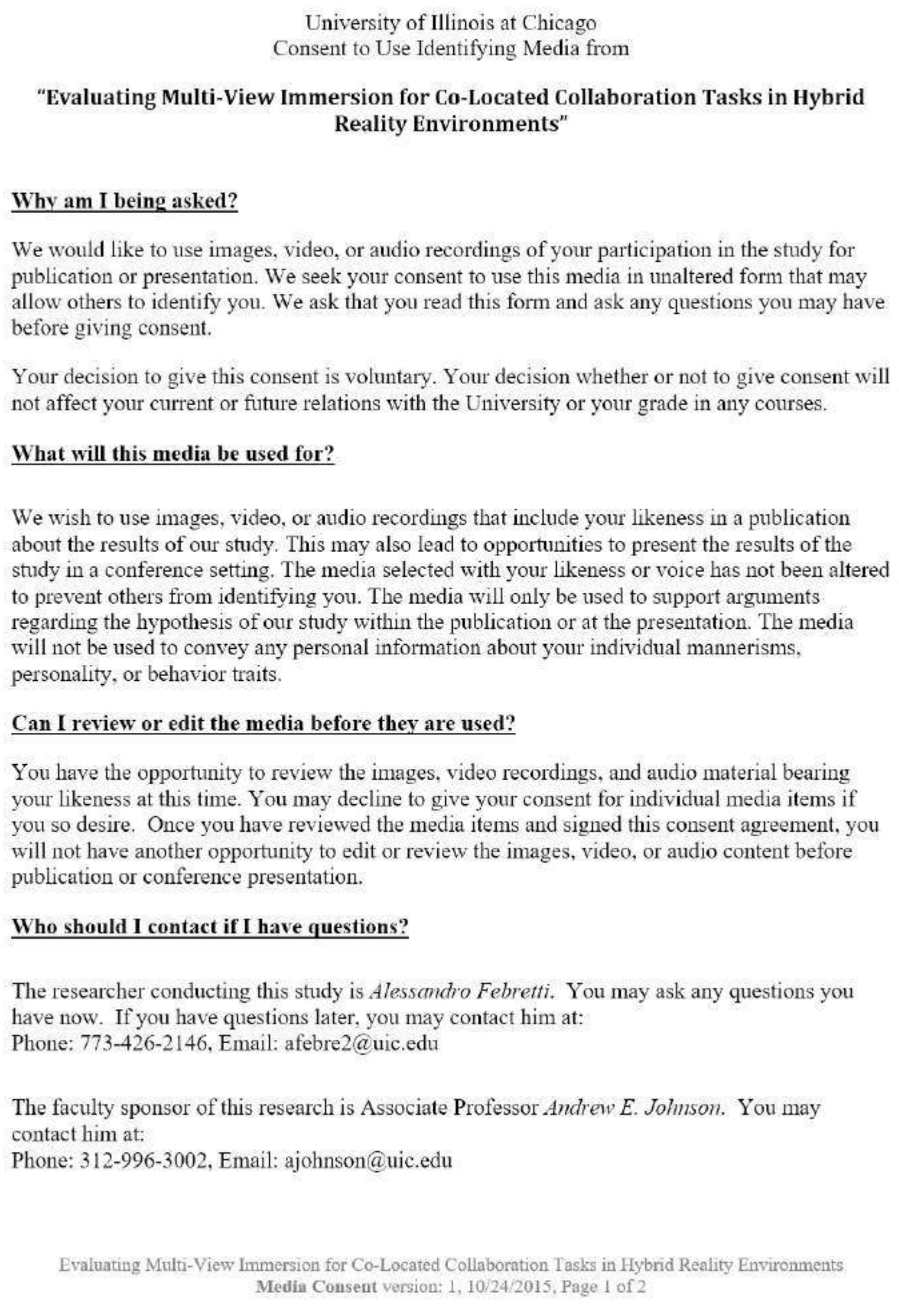
Signature of Subject or Legally Authorized Representative

I have read (or someone has read to me) the above information. I have been given an opportunity to ask questions and my questions have been answered to my satisfaction. I agree to participate in this research. I have been given a copy of this form.

Signature	Date
Printed Name	
E-mail address	
Signature of Researcher	Date (must be same as subject's)

APPENDIX A (continued)

Media Release Form



University of Illinois at Chicago
Consent to Use Identifying Media from

"Evaluating Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments"

Why am I being asked?

We would like to use images, video, or audio recordings of your participation in the study for publication or presentation. We seek your consent to use this media in unaltered form that may allow others to identify you. We ask that you read this form and ask any questions you may have before giving consent.

Your decision to give this consent is voluntary. Your decision whether or not to give consent will not affect your current or future relations with the University or your grade in any courses.

What will this media be used for?

We wish to use images, video, or audio recordings that include your likeness in a publication about the results of our study. This may also lead to opportunities to present the results of the study in a conference setting. The media selected with your likeness or voice has not been altered to prevent others from identifying you. The media will only be used to support arguments regarding the hypothesis of our study within the publication or at the presentation. The media will not be used to convey any personal information about your individual mannerisms, personality, or behavior traits.

Can I review or edit the media before they are used?

You have the opportunity to review the images, video recordings, and audio material bearing your likeness at this time. You may decline to give your consent for individual media items if you so desire. Once you have reviewed the media items and signed this consent agreement, you will not have another opportunity to edit or review the images, video, or audio content before publication or conference presentation.

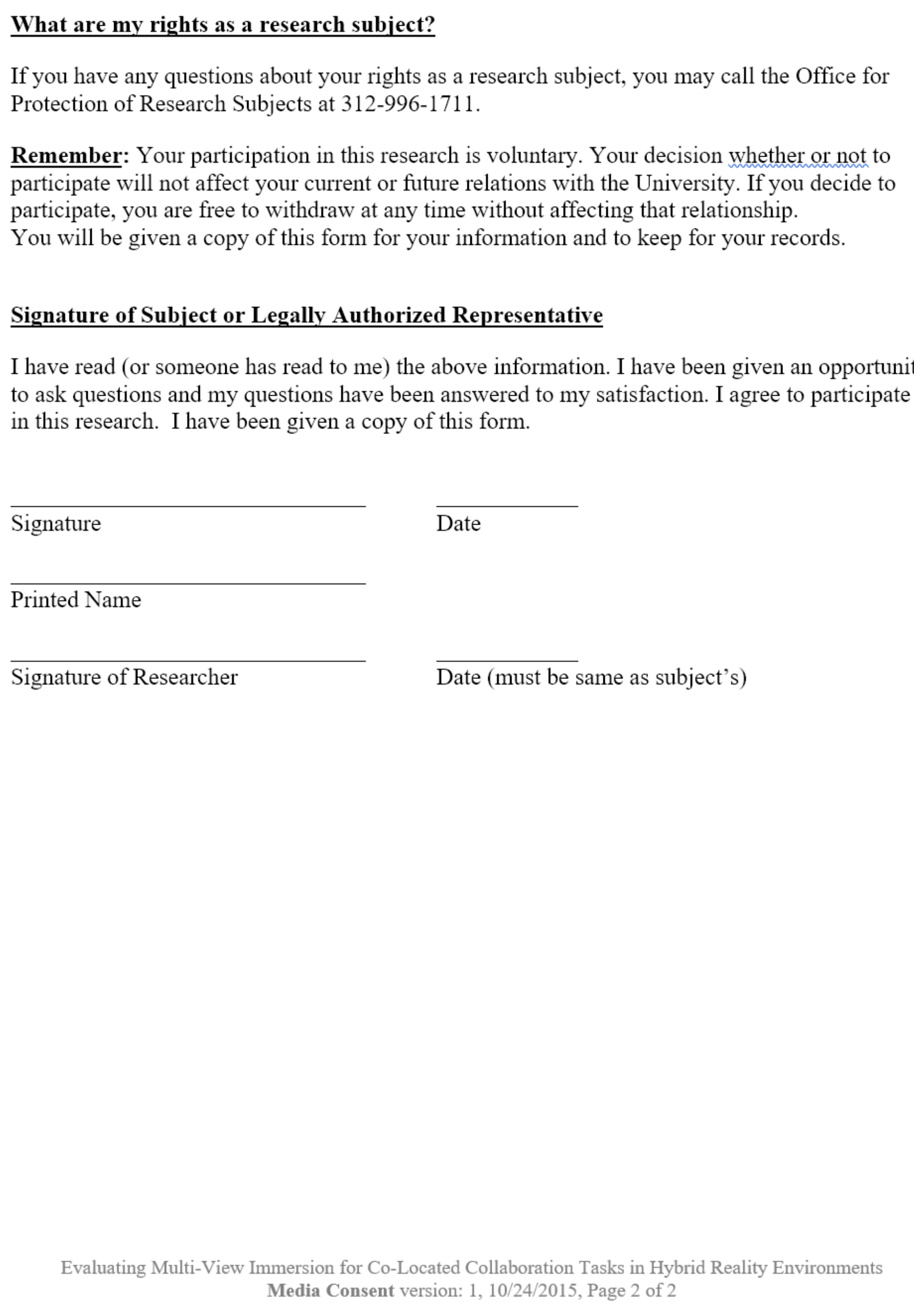
Who should I contact if I have questions?

The researcher conducting this study is *Alessandro Febretti*. You may ask any questions you have now. If you have questions later, you may contact him at:
Phone: 773-426-2146, Email: afebre2@uic.edu

The faculty sponsor of this research is Associate Professor *Andrew E. Johnson*. You may contact him at:
Phone: 312-996-3002, Email: ajohnson@uic.edu

Evaluating Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments
Media Consent version: 1, 10/24/2015, Page 1 of 2

APPENDIX A (continued)



What are my rights as a research subject?

If you have any questions about your rights as a research subject, you may call the Office for Protection of Research Subjects at 312-996-1711.

Remember: Your participation in this research is voluntary. Your decision whether or not to participate will not affect your current or future relations with the University. If you decide to participate, you are free to withdraw at any time without affecting that relationship.
You will be given a copy of this form for your information and to keep for your records.

Signature of Subject or Legally Authorized Representative

I have read (or someone has read to me) the above information. I have been given an opportunity to ask questions and my questions have been answered to my satisfaction. I agree to participate in this research. I have been given a copy of this form.

Signature	Date
Printed Name	
Signature of Researcher	Date (must be same as subject's)

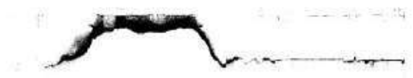
Evaluating Multi-View Immersion for Co-Located Collaboration Tasks in Hybrid Reality Environments
Media Consent version: 1, 10/24/2015, Page 2 of 2

APPENDIX A (continued)

Dive Sets

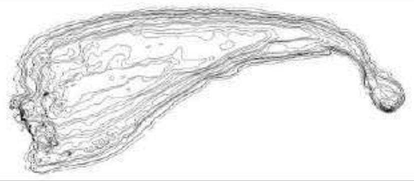
DS1 Group ID: __________ COND ID: __________

DIVE 09-23



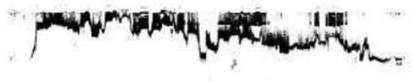
Notes

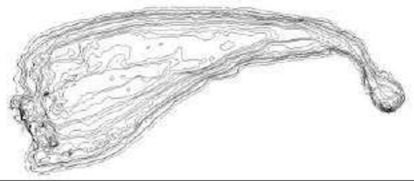
DIVE 08-17



Notes

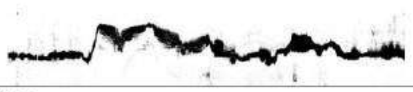
DIVE 09-17

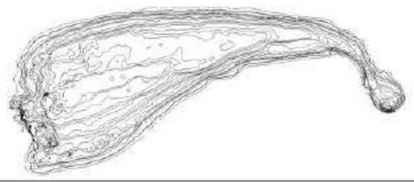




Notes

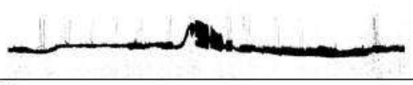
DIVE 09-27

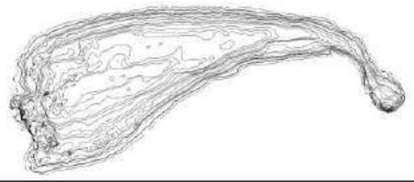




Notes

DIVE 08-12





Notes

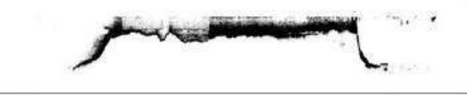
Sort dives from best to worst based on **1: noise level 2: coverage**

best				**worst**

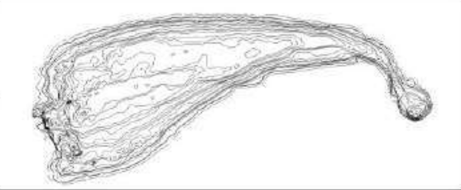
APPENDIX A (continued)

DS2 Group ID: __________ COND ID: __________

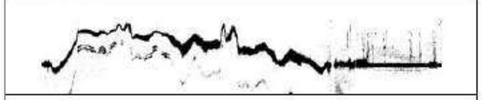
DIVE 09-22



Notes

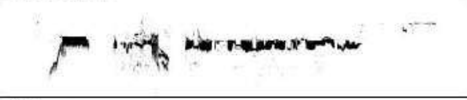


DIVE 09-25

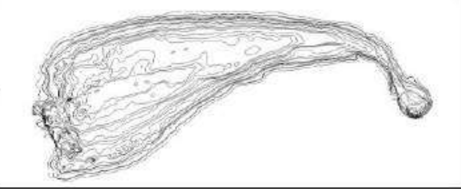


Notes

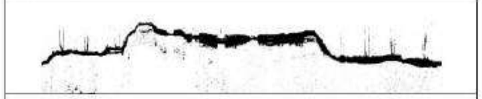
DIVE 09-20



Notes

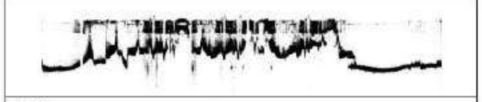


DIVE 08-09

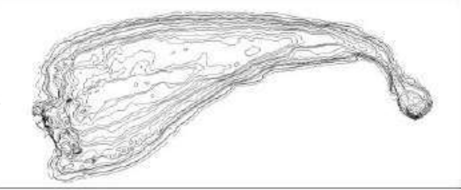


Notes

DIVE 08-16



Notes



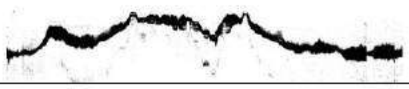
Sort dives from best to worst based on **1: noise level 2: coverage**

best				**worst**

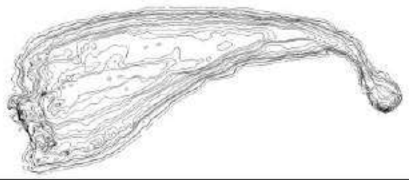
APPENDIX A (continued)

DS3 Group ID: __________ COND ID: __________

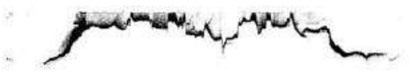
DIVE 09-26



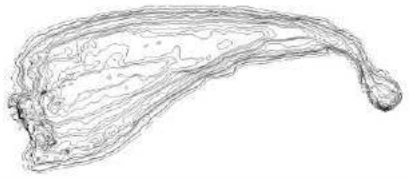
Notes



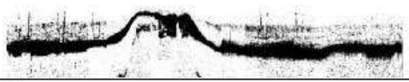
DIVE 09-13



Notes

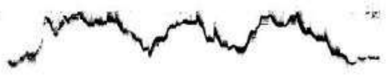


DIVE 08-05



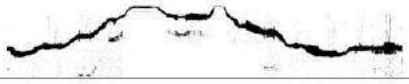
Notes

DIVE 09-11



Notes

DIVE 08-10



Notes

Sort dives from best to worst based on **1: noise level 2: coverage**				
best				worst

APPENDIX A (continued)

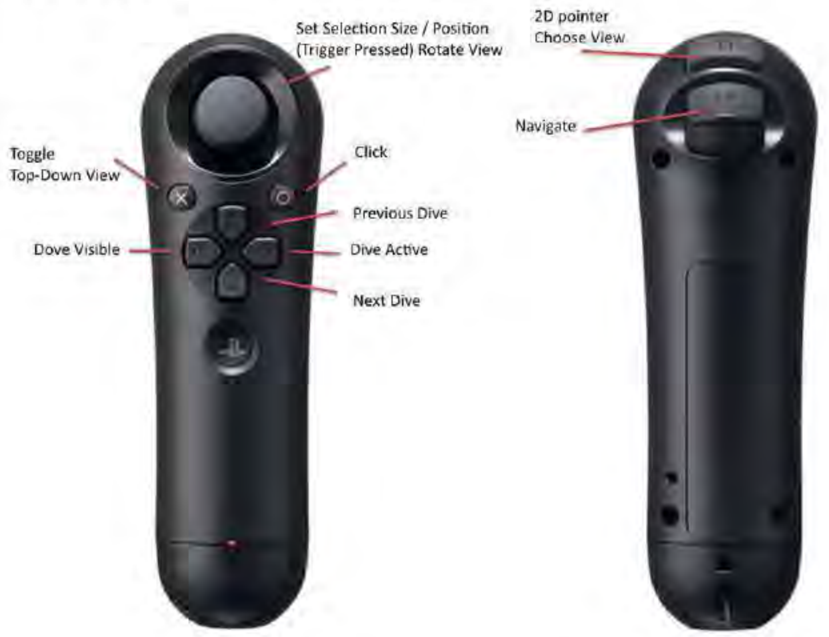
View Guide

- Pink markers: high-quality depth markers
- Gray surface: lower quality depth model
- Valid sonar data is within pink markers and grey surface. Points outsize these bounds are noise.

Quality Assessment Guide

- Order dives based on noise (less noise/errors = better)
- For dives with similar noise, order based on how big the area they cover is (larger covered area = better)
- If you are undecided on whether a feature of the sonar data is noise or a valid scan, ask the domain expert.
- You have 20 minutes to complete the task.

Controller Guide



APPENDIX B

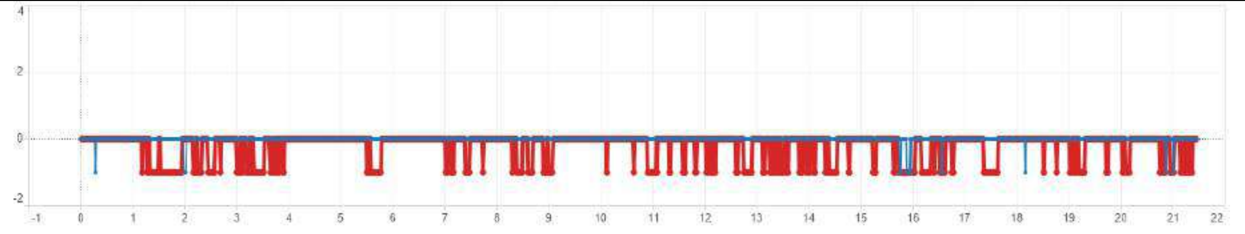
User Study Data

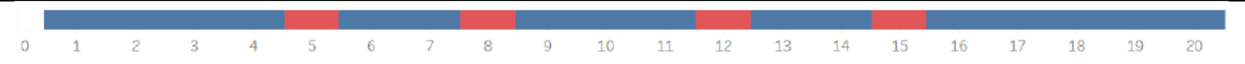
This appendix contains a summary of collected data for each two-participant group in the user study. Group 1 was the pilot group and is not included in this summary. The tables include the task time, raw error score, view usage log and collaboration log for each condition. These data presented here is discussed in chapter 7.

APPENDIX B (continued)

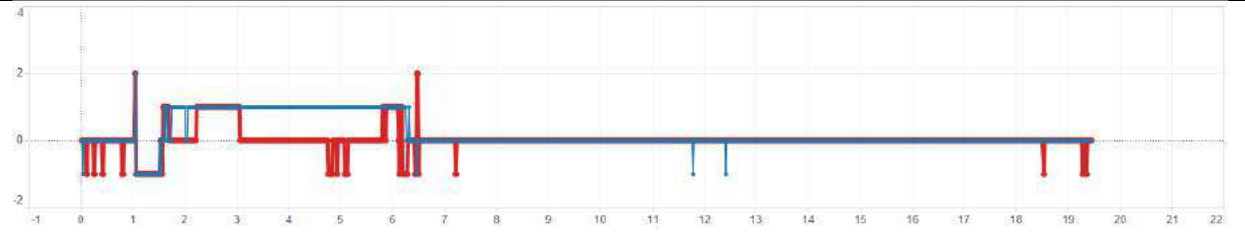
Group 2

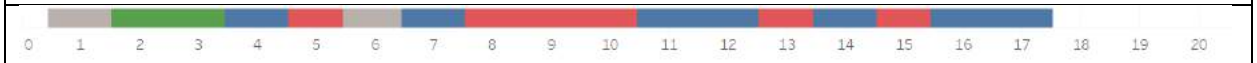
Condition	Single View Immersive	**Dive Set**	3
Time	20	**Error Score**	4



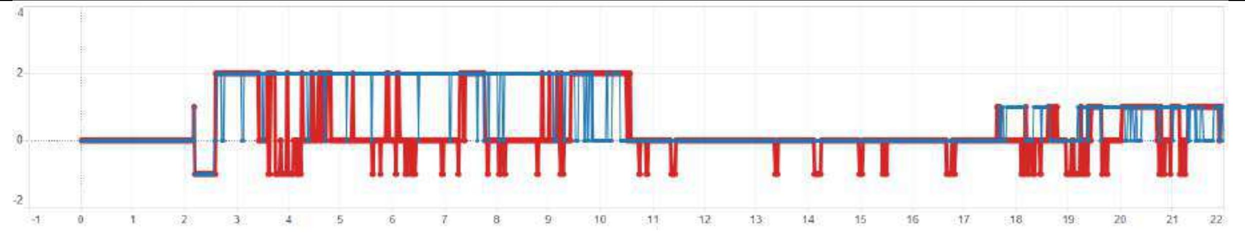


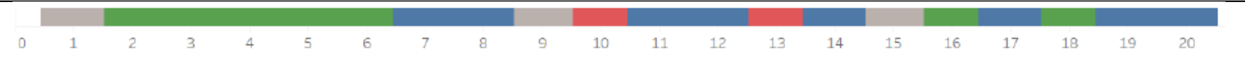
Condition	Multiview 2D	**Dive Set**	1
Time	19	**Error Score**	6





Condition	Multiview Immersive	**Dive Set**	2
Time	20	**Error Score**	2



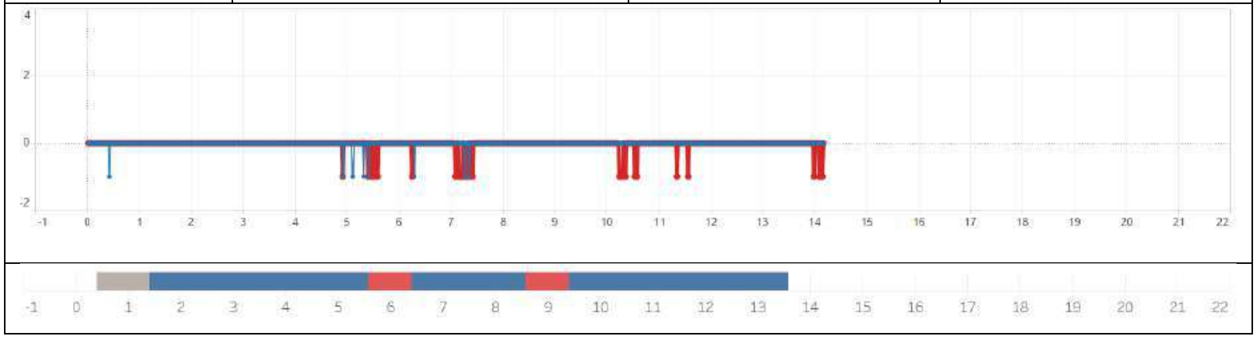


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

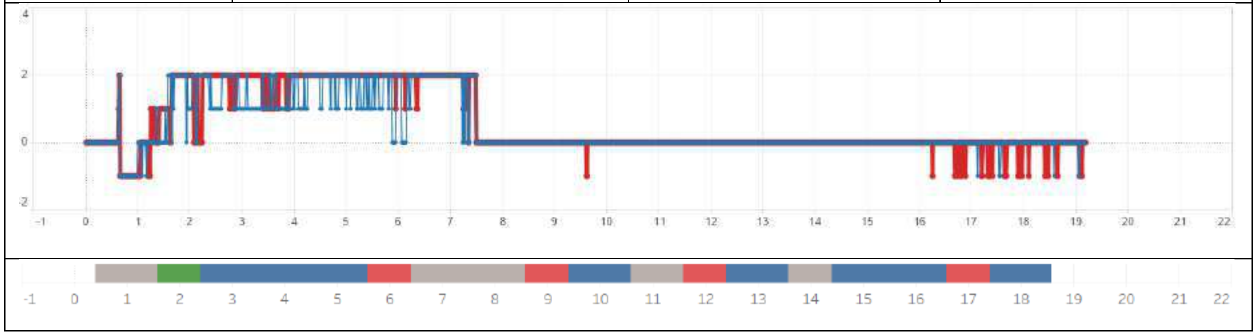
APPENDIX B (continued)

<u>Group 3</u>

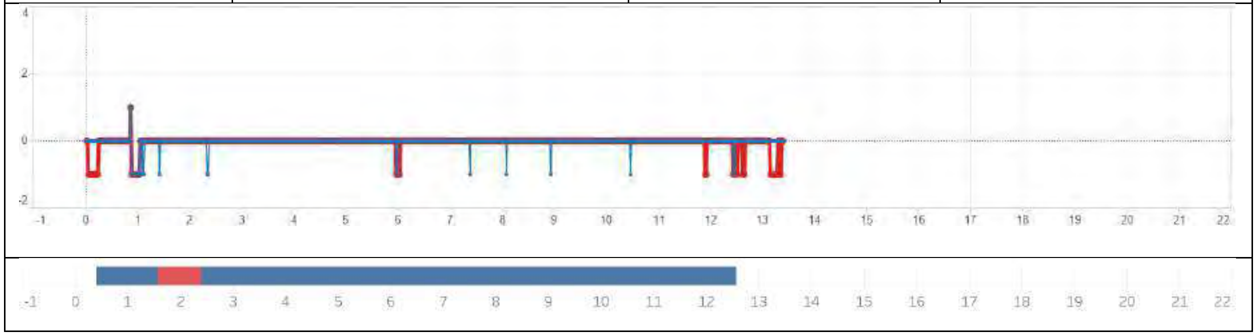
Condition	Single View Immersive	**Dive Set**	3
Time	14	**Error Score**	4



Condition	Multiview 2D	**Dive Set**	1
Time	19	**Error Score**	4



Condition	Multiview Immersive	**Dive Set**	2
Time	14	**Error Score**	4

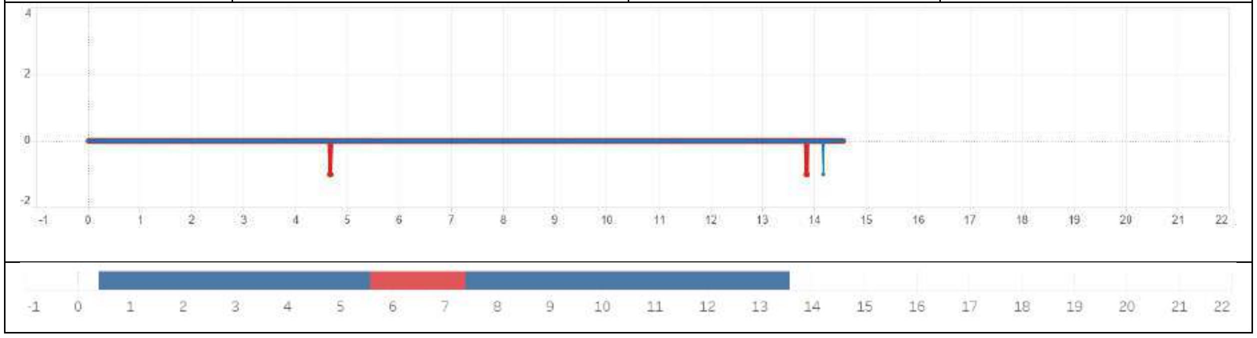


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

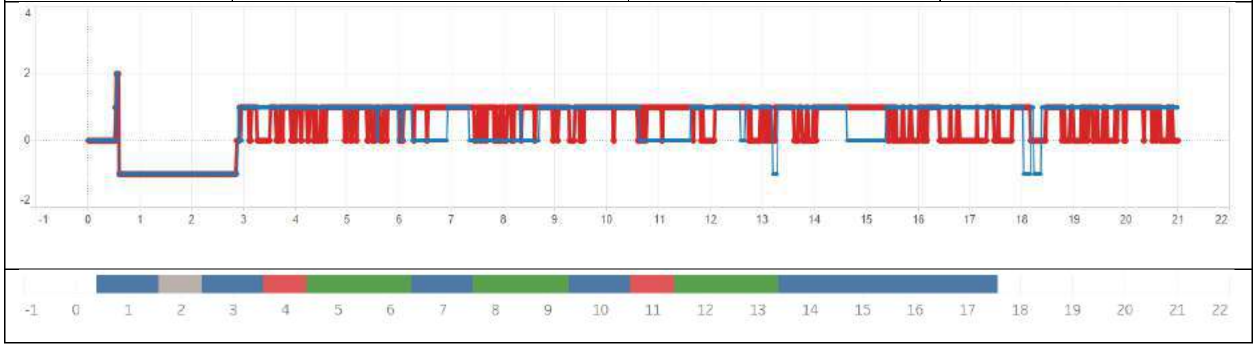
APPENDIX B (continued)

Group 4

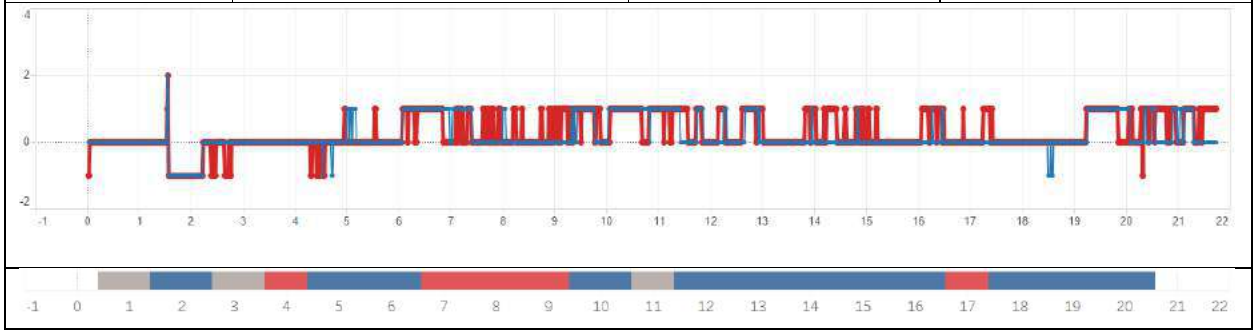
Condition	Single View Immersive	**Dive Set**	3
Time	15	**Error Score**	6



Condition	Multiview 2D	**Dive Set**	2
Time	20	**Error Score**	4



Condition	Multiview Immersive	**Dive Set**	1
Time	20	**Error Score**	0

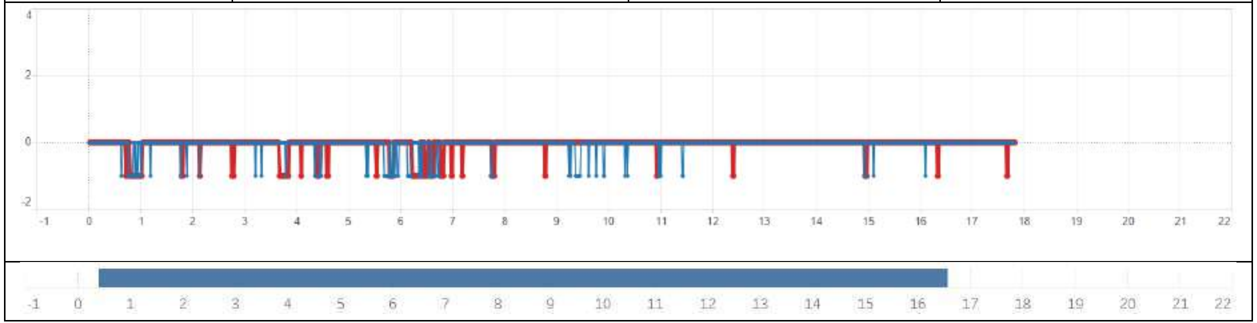


🟦 Joint Work / Discussion 🟫 Organization 🟩 Parallel Work 🟧 Single User Work

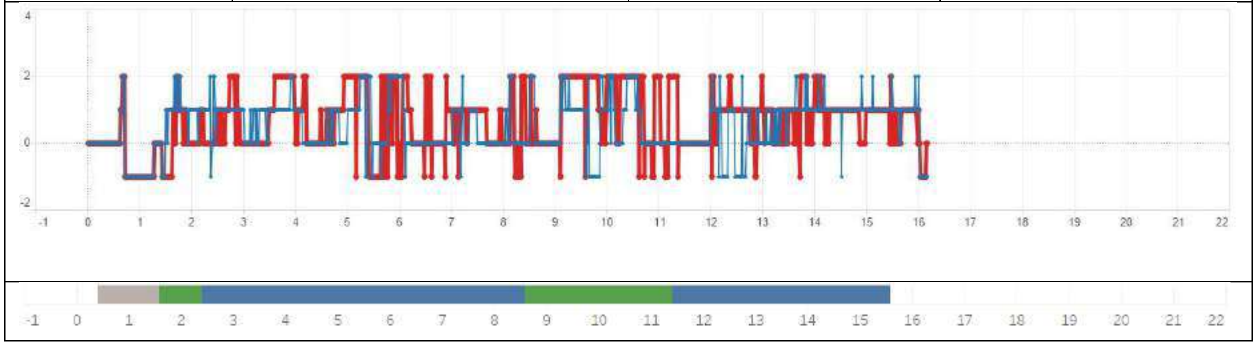
APPENDIX B (continued)

Group 5

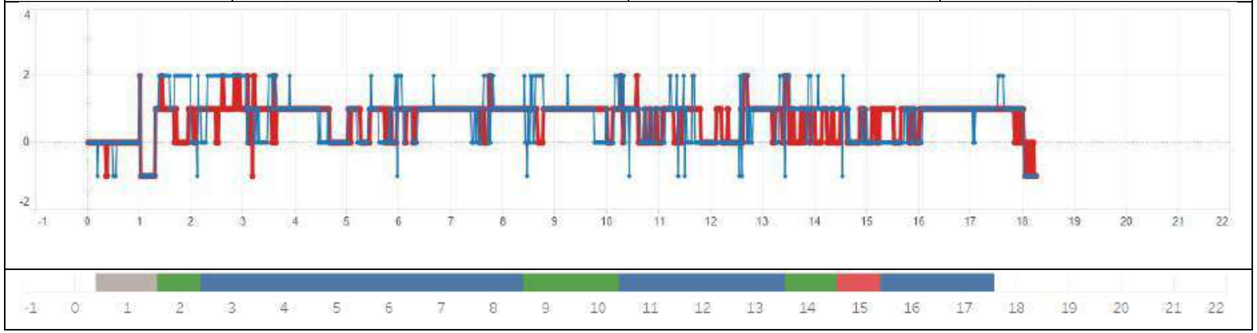
Condition	Single View Immersive	**Dive Set**	1
Time	18	**Error Score**	4



Condition	Multiview 2D	**Dive Set**	2
Time	16	**Error Score**	6



Condition	Multiview Immersive	**Dive Set**	3
Time	18	**Error Score**	2

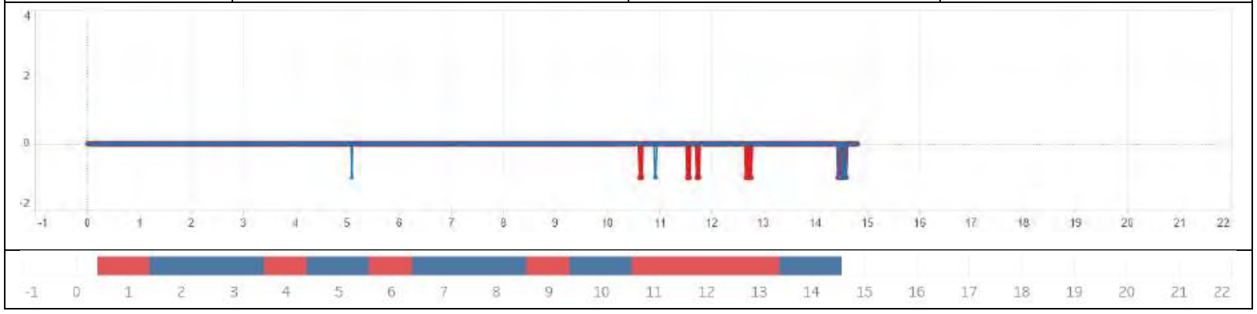


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

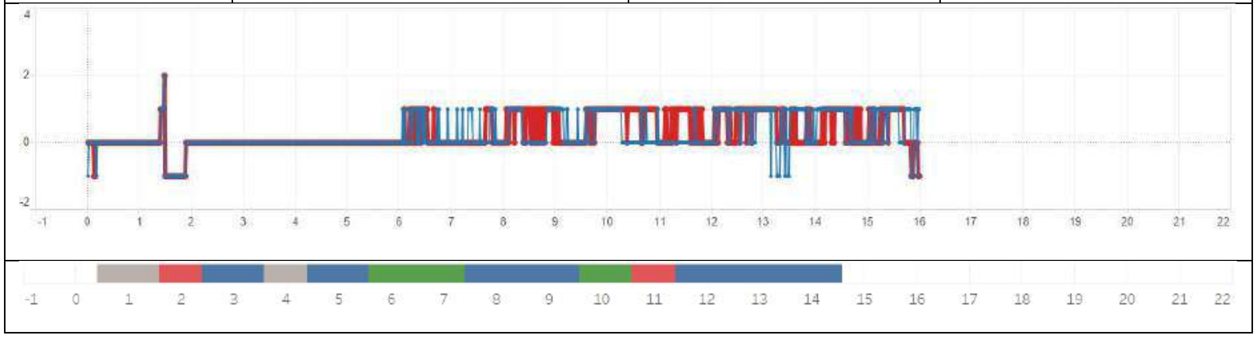
APPENDIX B (continued)

Group 6

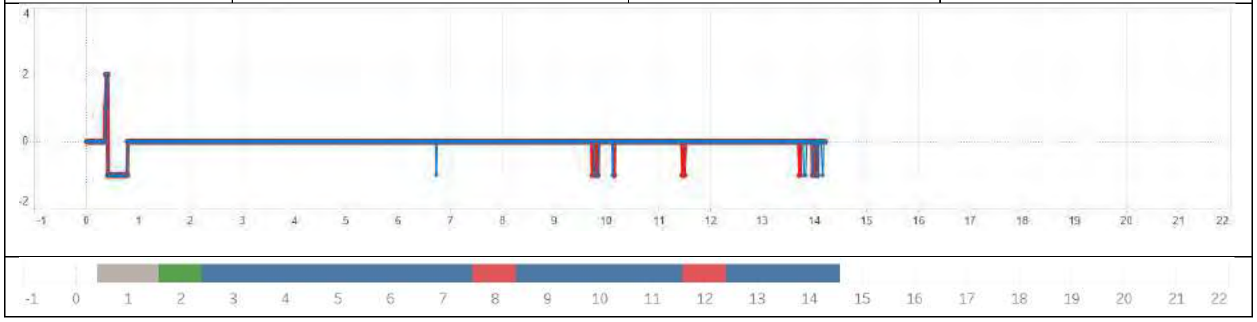
Condition	Single View Immersive	Dive Set	3
Time	15	Error Score	0



Condition	Multiview 2D	Dive Set	1
Time	16	Error Score	0



Condition	Multiview Immersive	Dive Set	2
Time	14	Error Score	2

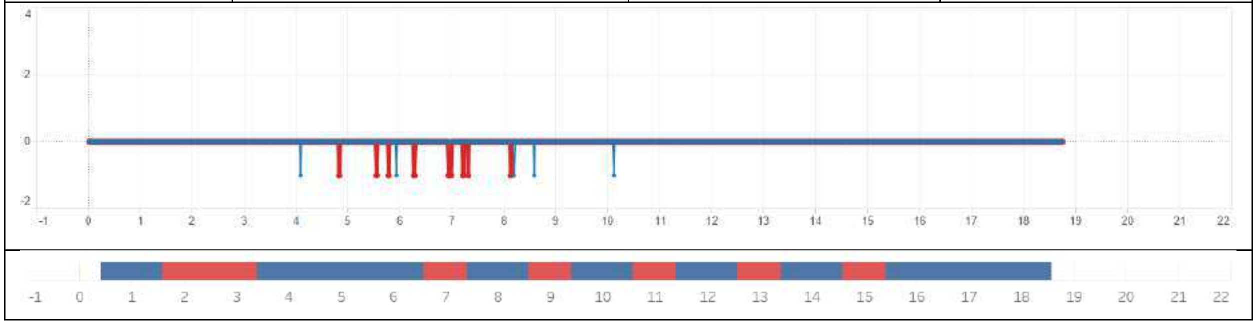


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

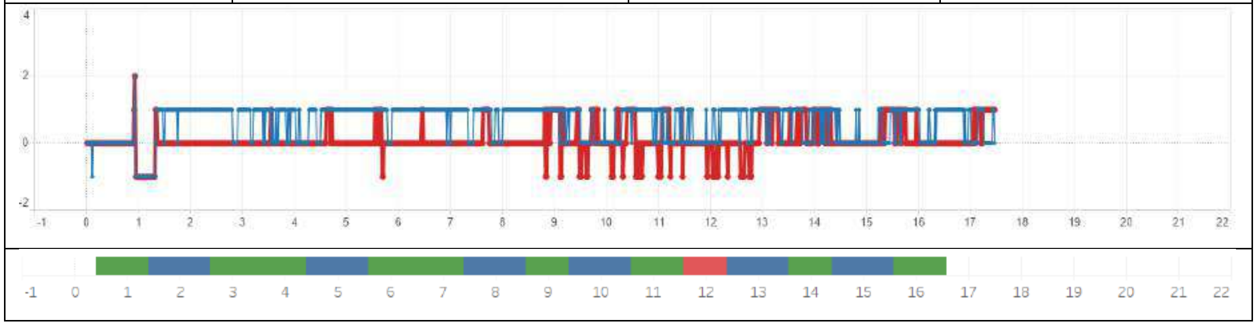
APPENDIX B (continued)

Group 7

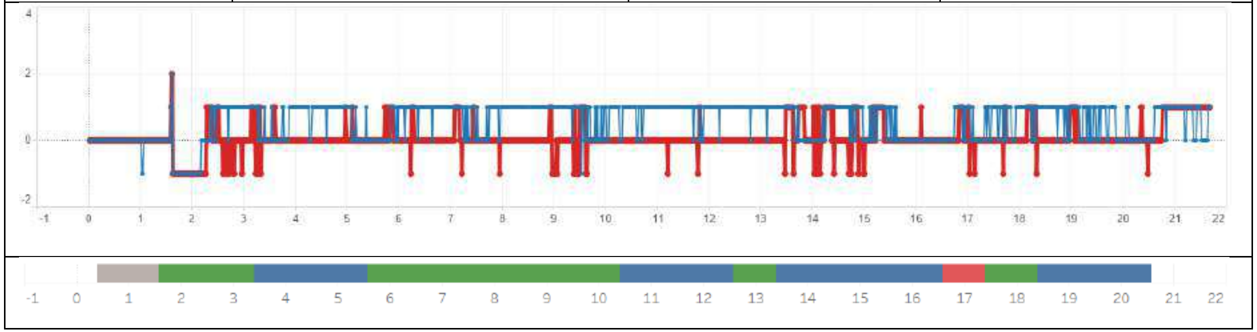
Condition	Single View Immersive	**Dive Set**	3
Time	19	**Error Score**	0



Condition	Multiview 2D	**Dive Set**	2
Time	18	**Error Score**	2



Condition	Multiview Immersive	**Dive Set**	1
Time	20	**Error Score**	2

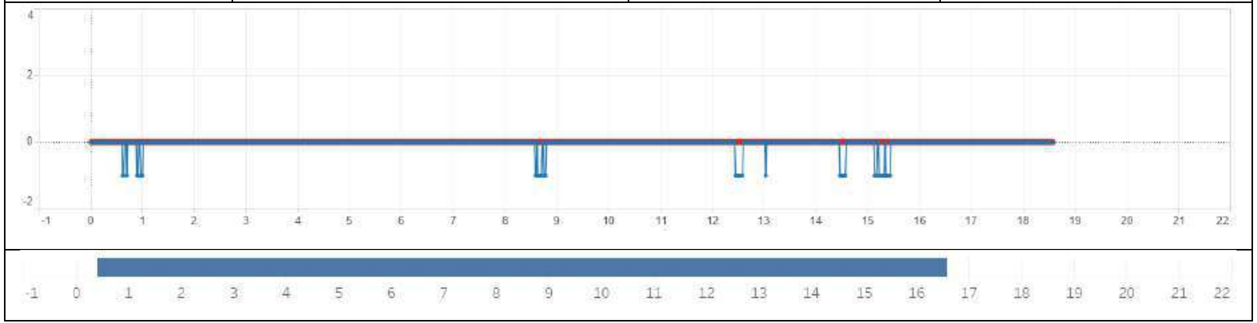


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

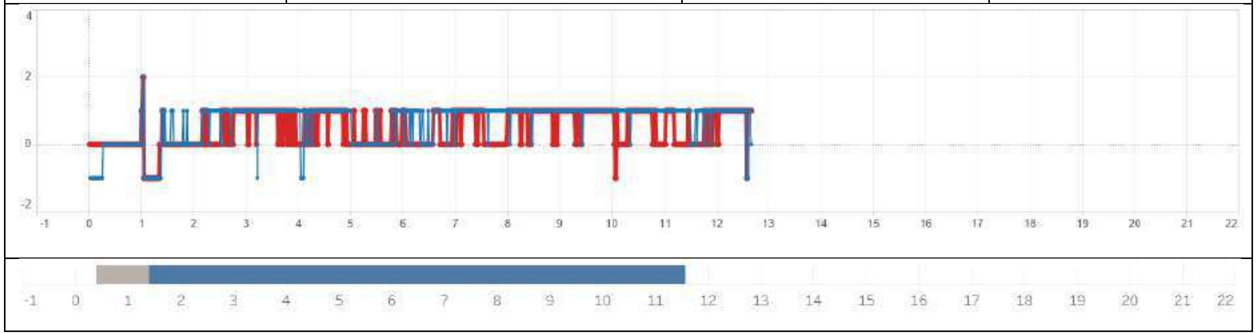
APPENDIX B (continued)

Group 8

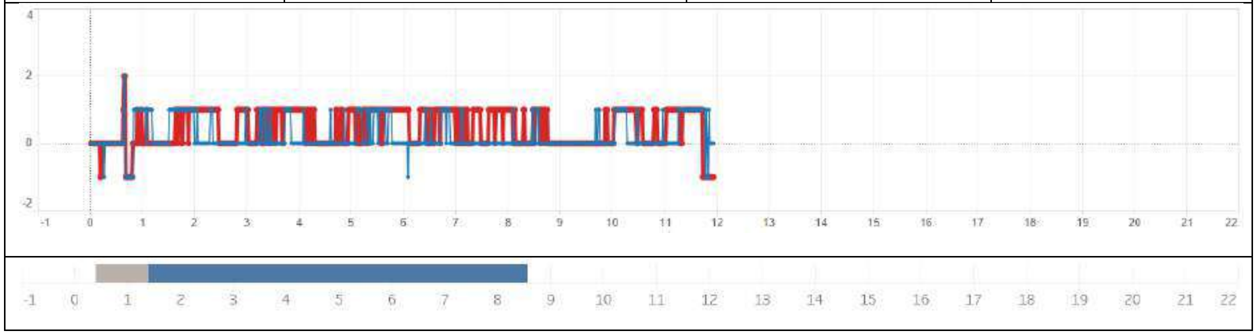
Condition	Single View Immersive	**Dive Set**	1
Time	19	**Error Score**	4



Condition	Multiview 2D	**Dive Set**	2
Time	13	**Error Score**	4



Condition	Multiview Immersive	**Dive Set**	3
Time	12	**Error Score**	6

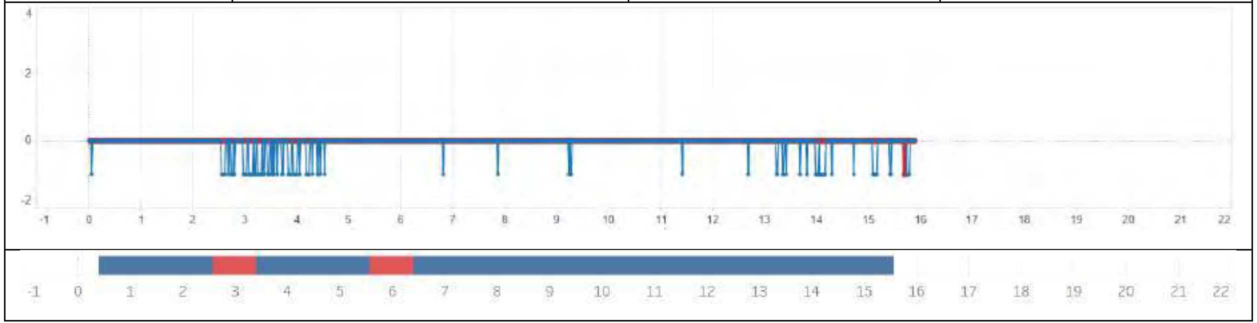


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

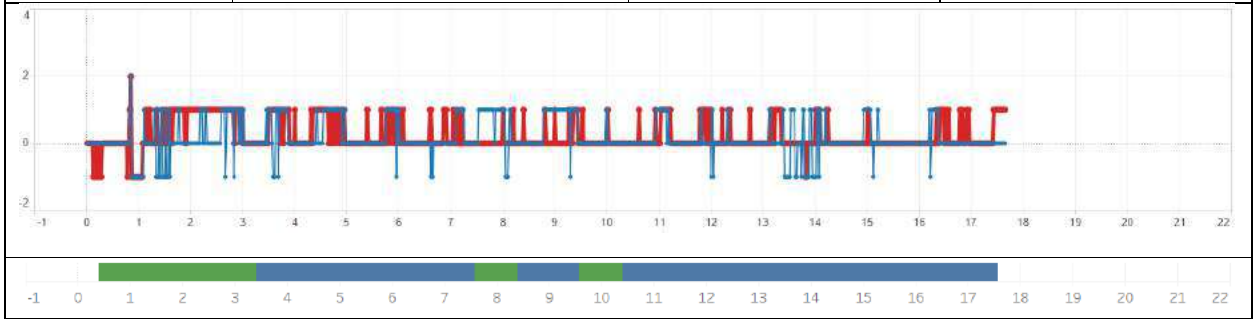
APPENDIX B (continued)

Group 9

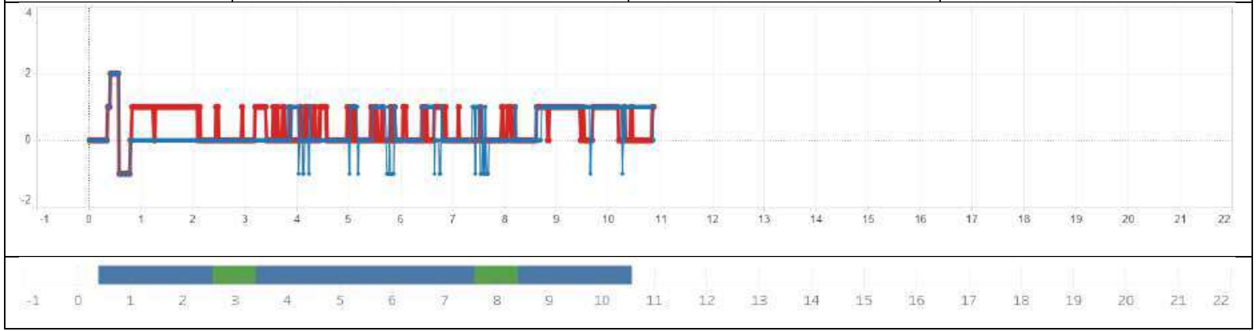
Condition	Single View Immersive	**Dive Set**	1
Time	16	**Error Score**	8



Condition	Multiview 2D	**Dive Set**	2
Time	18	**Error Score**	4



Condition	Multiview Immersive	**Dive Set**	3
Time	11	**Error Score**	7

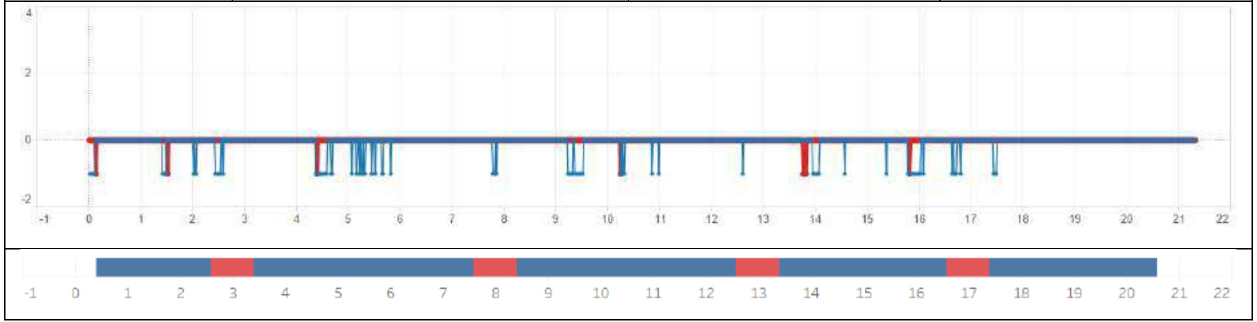


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

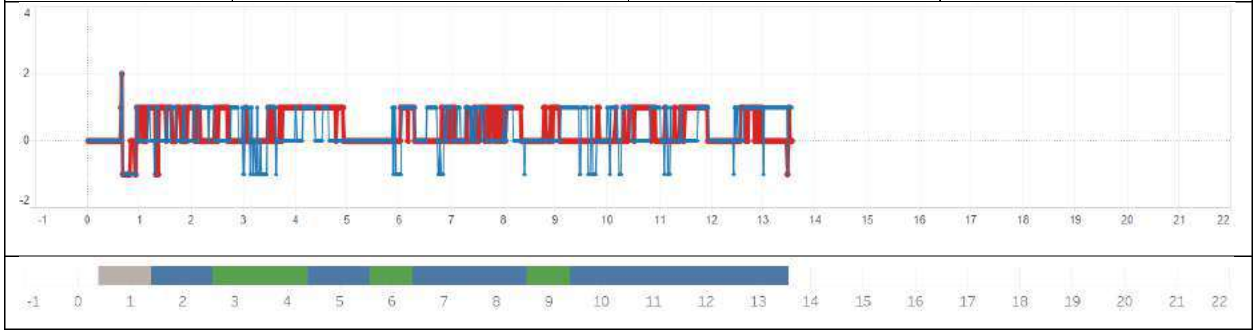
APPENDIX B (continued)

Group 10

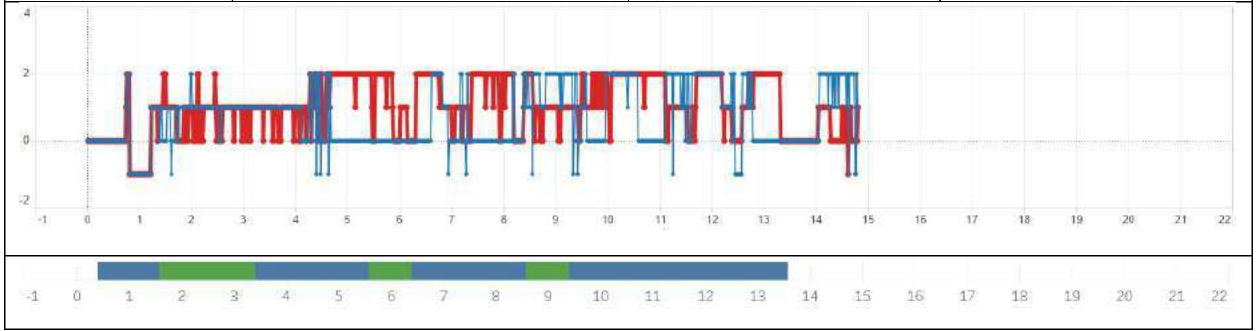
Condition	Single View Immersive	**Dive Set**	1
Time	20	**Error Score**	4



Condition	Multiview 2D	**Dive Set**	2
Time	14	**Error Score**	4



Condition	Multiview Immersive	**Dive Set**	3
Time	15	**Error Score**	0

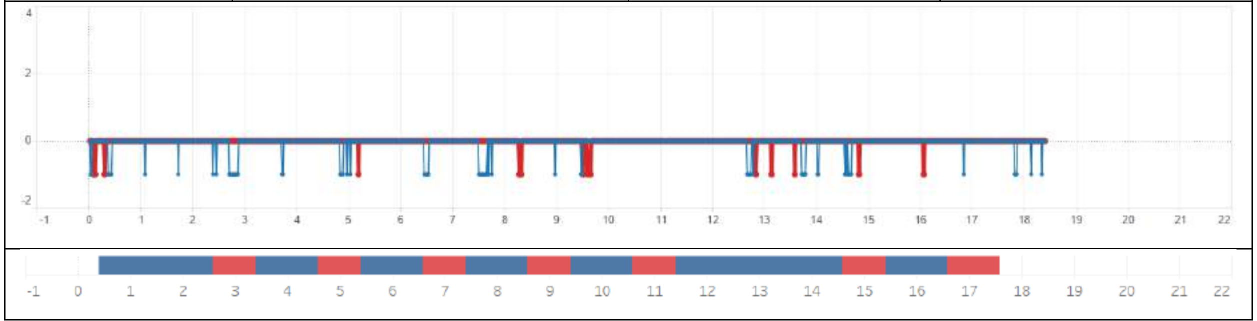


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

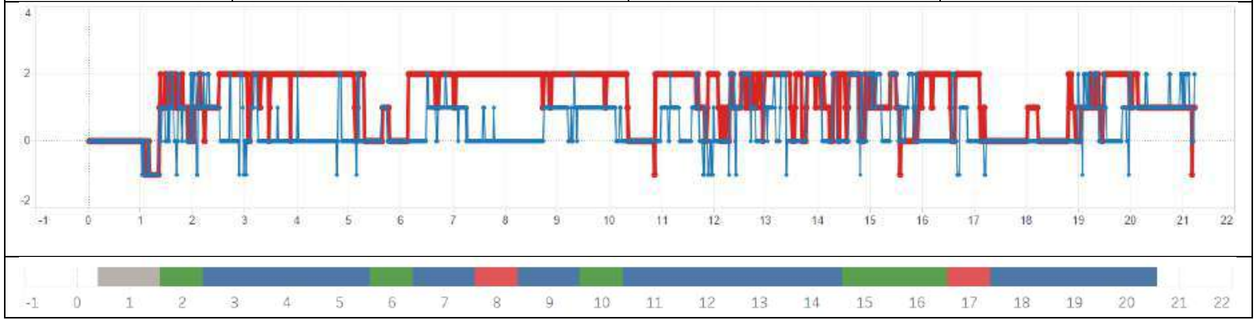
APPENDIX B (continued)

Group 11

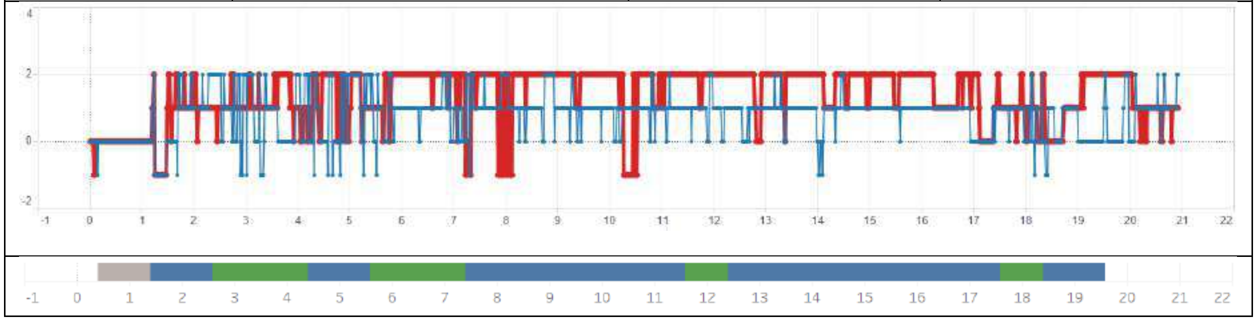
Condition	Single View Immersive	**Dive Set**	3
Time	19	**Error Score**	2



Condition	Multiview 2D	**Dive Set**	2
Time	20	**Error Score**	4



Condition	Multiview Immersive	**Dive Set**	1
Time	20	**Error Score**	4

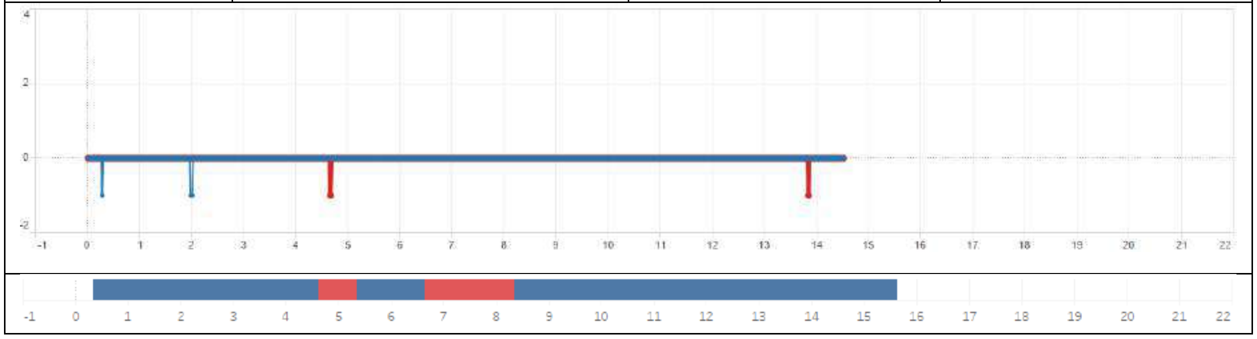


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

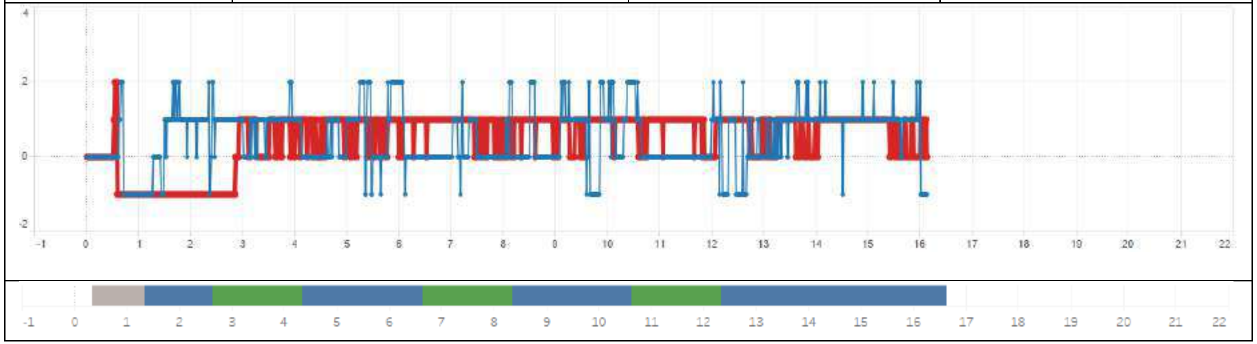
APPENDIX B (continued)

Group 12

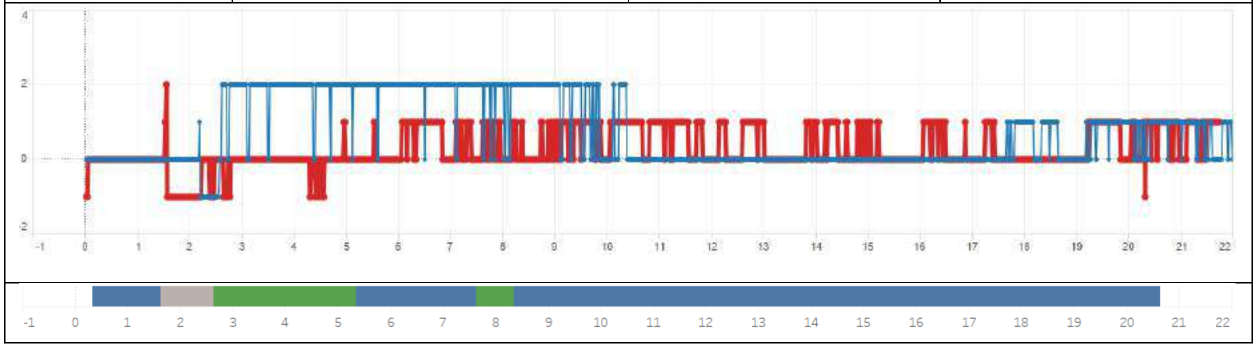
Condition	Single View Immersive	**Dive Set**	1
Time	15	**Error Score**	2



Condition	Multiview 2D	**Dive Set**	2
Time	16	**Error Score**	0



Condition	Multiview Immersive	**Dive Set**	3
Time	20	**Error Score**	0

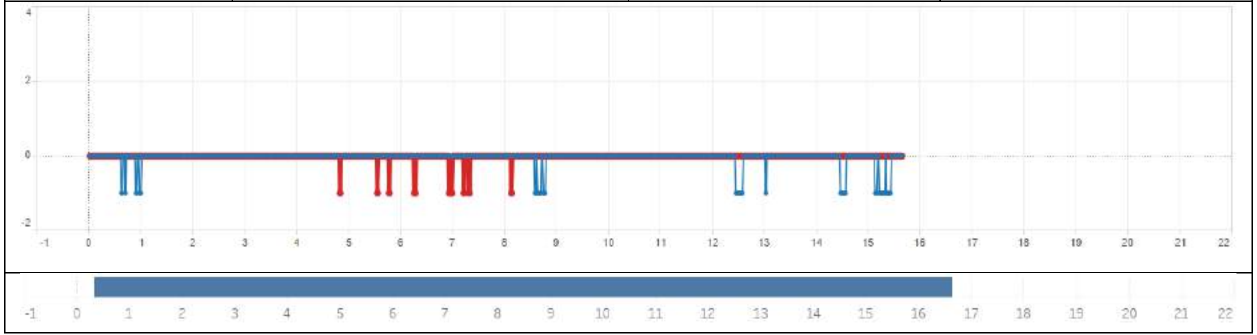


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

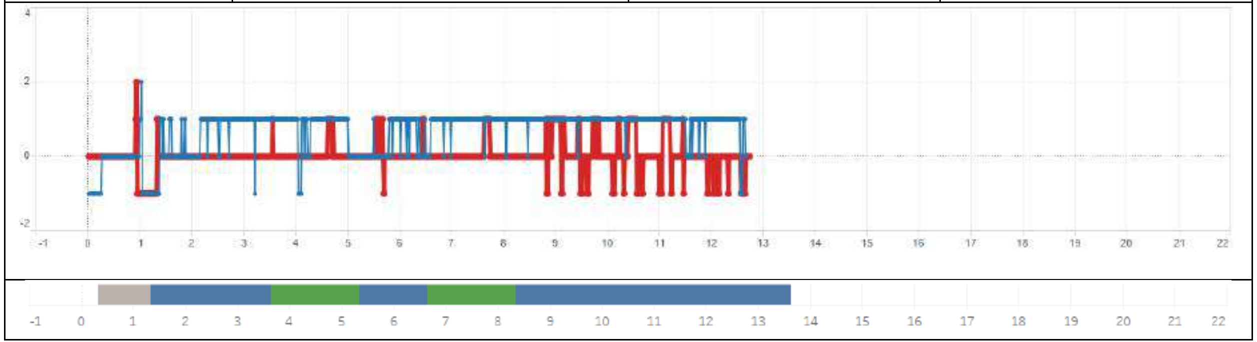
APPENDIX B (continued)

Group 13

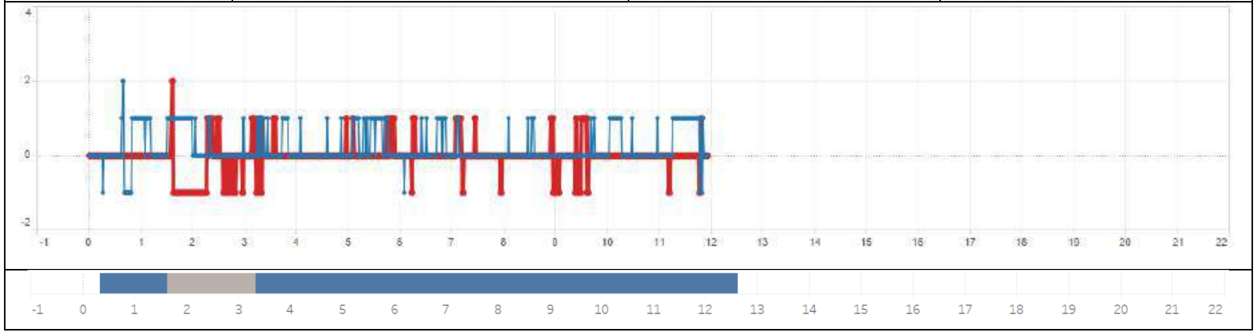
Condition	Single View Immersive	**Dive Set**	3
Time	16	**Error Score**	4



Condition	Multiview 2D	**Dive Set**	1
Time	13	**Error Score**	6



Condition	Multiview Immersive	**Dive Set**	2
Time	12	**Error Score**	2

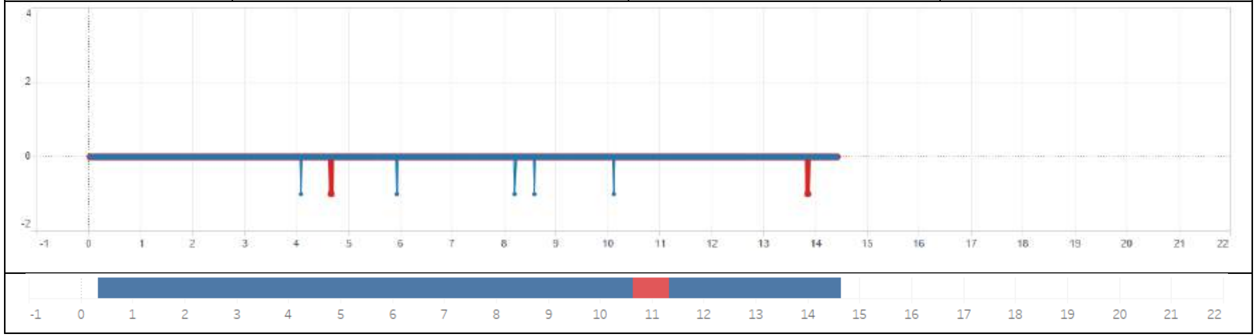


🟦 Joint Work / Discussion 🟫 Organization 🟩 Parallel Work 🟧 Single User Work

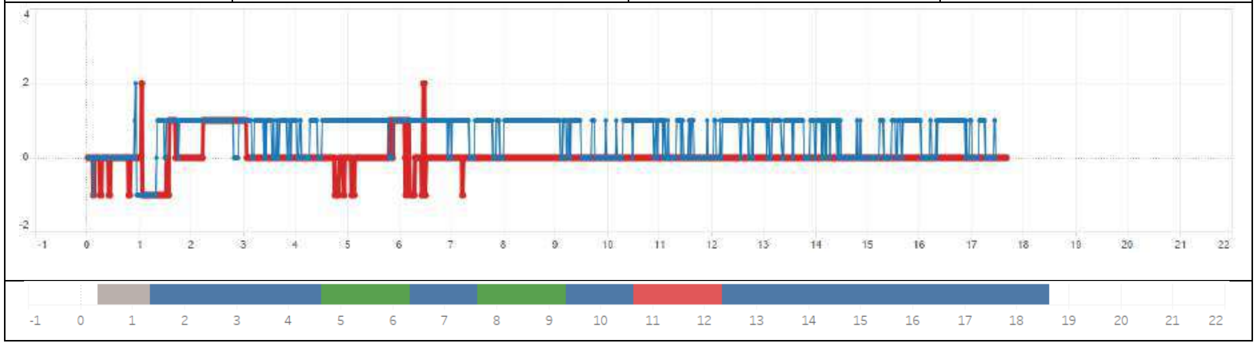
APPENDIX B (continued)

Group 14

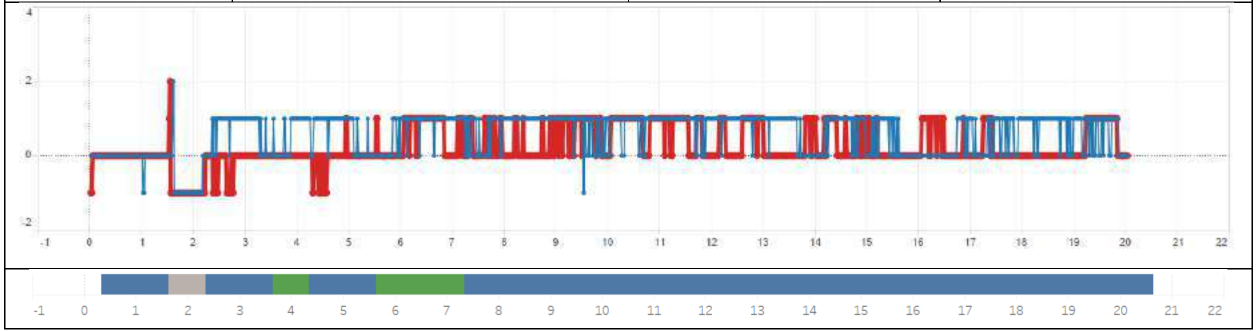
Condition	Single View Immersive	Dive Set	1
Time		Error Score	4



Condition	Multiview 2D	Dive Set	3
Time		Error Score	4



Condition	Multiview Immersive	Dive Set	2
Time		Error Score	4

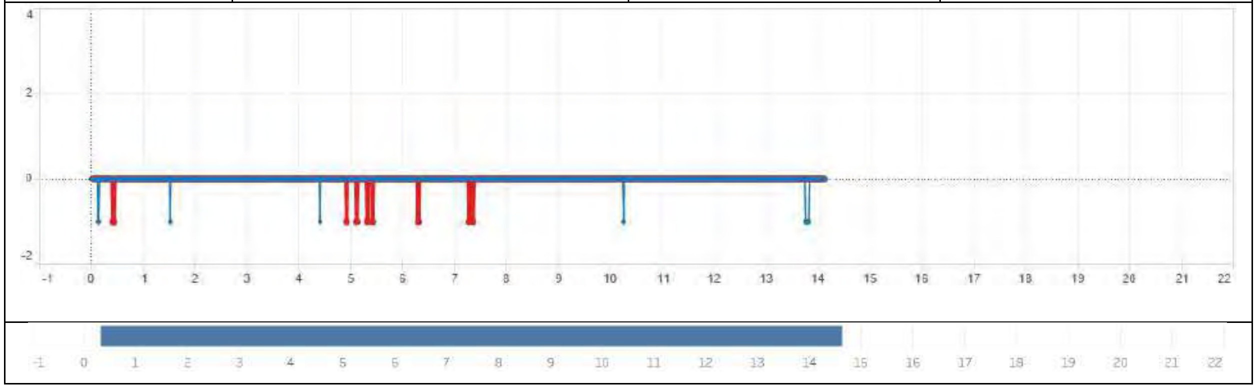


🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

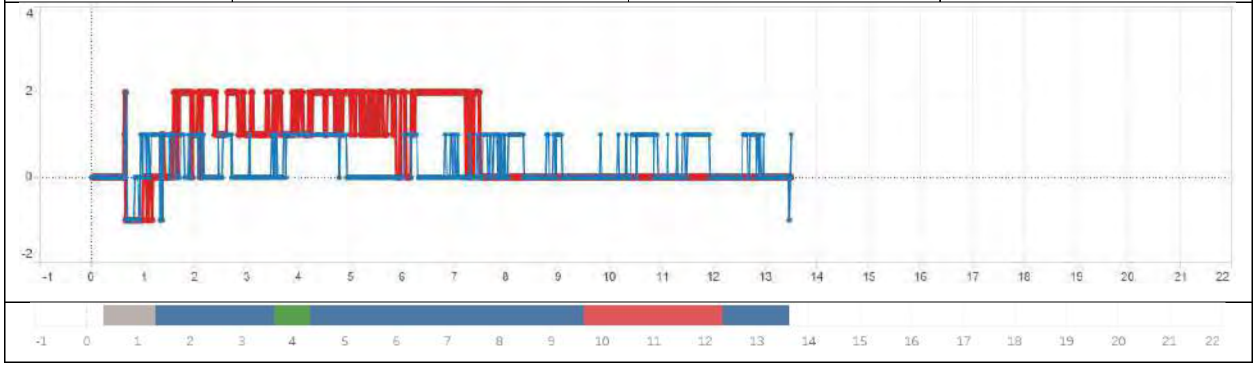
APPENDIX B (continued)

Group 15

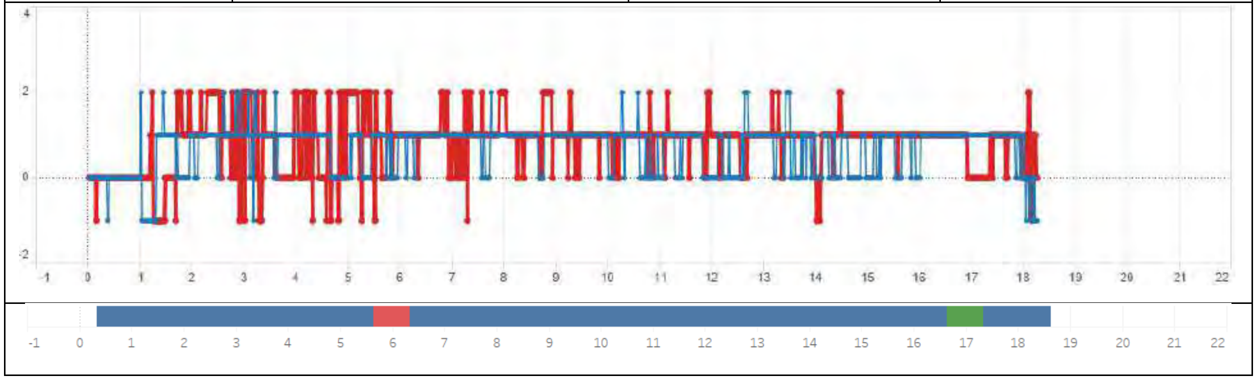
Condition	Single View Immersive	Dive Set	1
Time		Error Score	6



Condition	Multiview 2D	Dive Set	2
Time		Error Score	4



Condition	Multiview Immersive	Dive Set	3
Time		Error Score	0



🟦 Joint Work / Discussion ⬜ Organization 🟩 Parallel Work 🟧 Single User Work

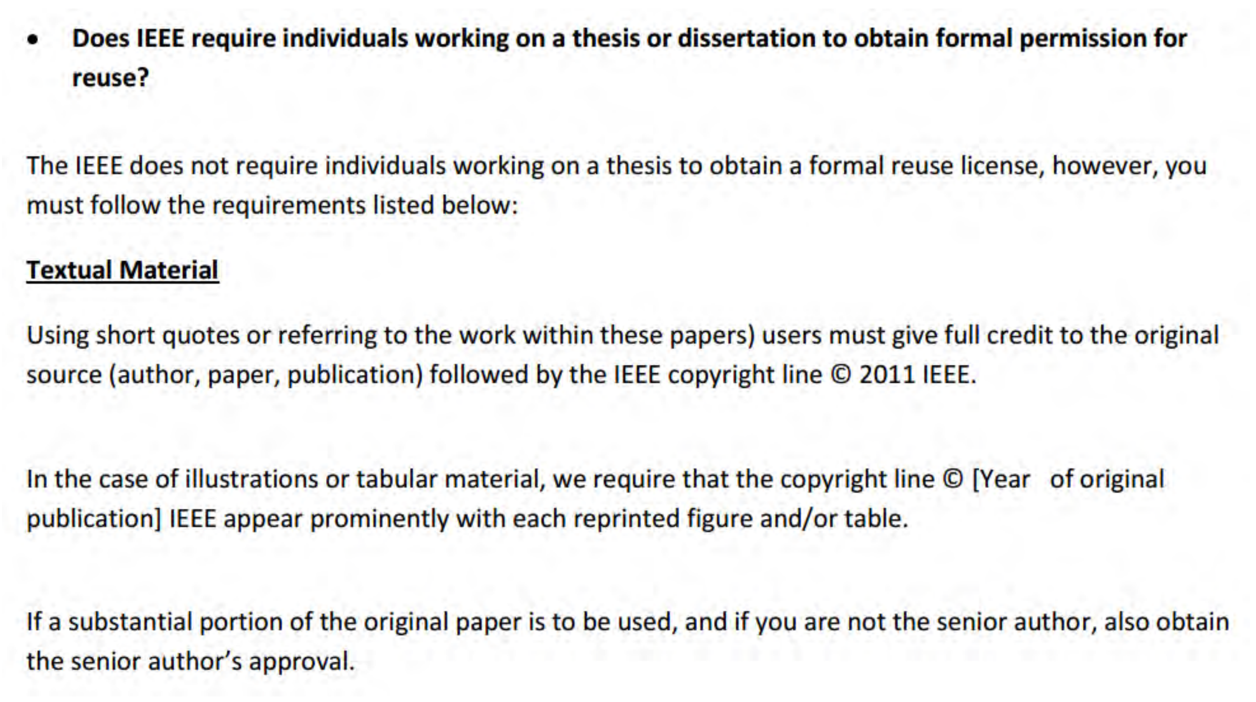
APPENDIX C

Permissions for reuse

The following are the reuse rules for dissertations relevant to previously published material included in this work. Each entry indicates the issuing organization and reference to the relative publication (as listed in *References* and in the *contributions by the Author* section).

IEEE

https://www.ieee.org/publications_standards/publications/rights/permissions_faq.pdf



- **Does IEEE require individuals working on a thesis or dissertation to obtain formal permission for reuse?**

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you must follow the requirements listed below:

Textual Material

Using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

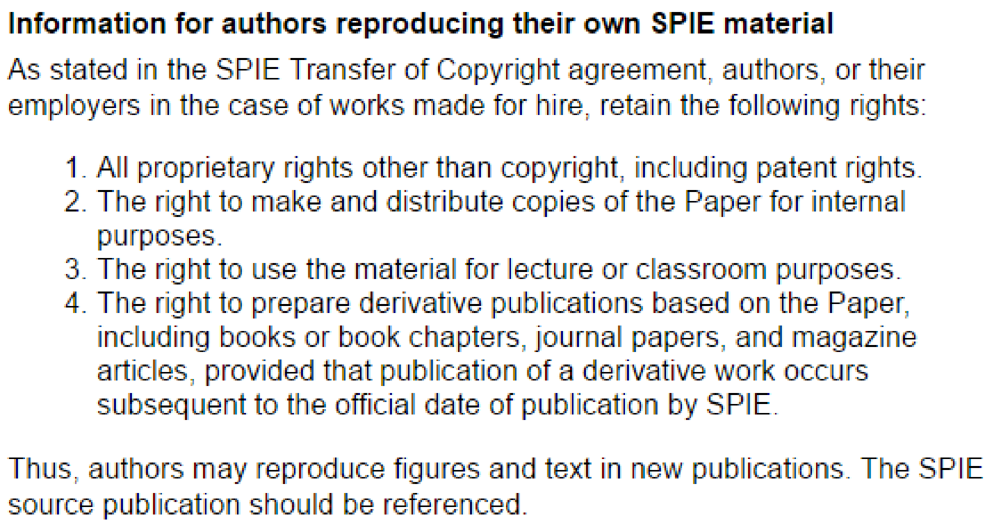
Febretti, A., Nishimoto, A., Mateevitsi, V., Renambot, L., Johnson, A., & Leigh, J. (2014, March). Omegalib: A multi-view application framework for hybrid reality display environments. In *Virtual Reality (VR), 2014 iEEE* (pp. 9-14). IEEE. © IEEE

Reda, K., Febretti, A., Knoll, A., Aurisano, J., Leigh, J., Johnson, A., Papka, M.E. and Hereld, M., 2013. Visualizing large, heterogeneous data in hybrid-reality environments. *IEEE Computer Graphics and Applications*, *33*(4), pp.38-48. © IEEE

APPENDIX C (continued)

SPIE

https://spie.org/publications/contact-spie-publications/reprint-permission



Information for authors reproducing their own SPIE material

As stated in the SPIE Transfer of Copyright agreement, authors, or their employers in the case of works made for hire, retain the following rights:

1. All proprietary rights other than copyright, including patent rights.
2. The right to make and distribute copies of the Paper for internal purposes.
3. The right to use the material for lecture or classroom purposes.
4. The right to prepare derivative publications based on the Paper, including books or book chapters, journal papers, and magazine articles, provided that publication of a derivative work occurs subsequent to the official date of publication by SPIE.

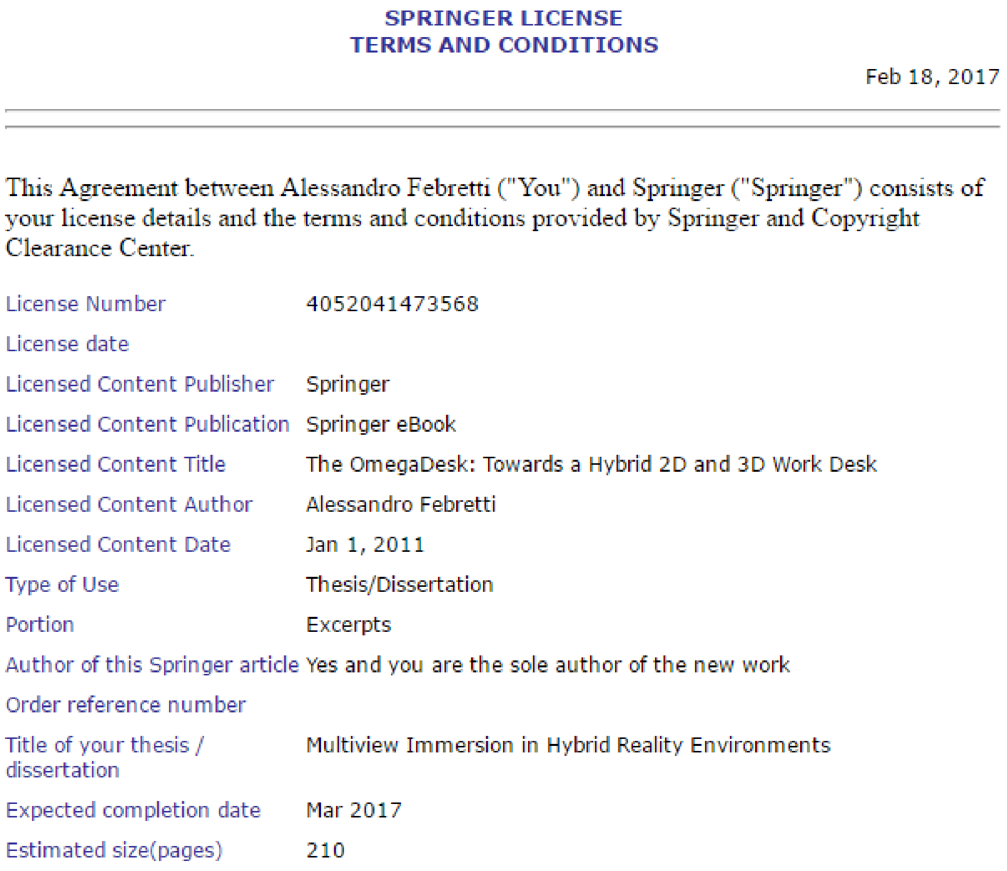
Thus, authors may reproduce figures and text in new publications. The SPIE source publication should be referenced.

Febretti, A., Nishimoto, A., Thigpen, T., Talandis, J., Long, L., Pirtle, J.D., Peterka, T., Verlo, A., Brown, M., Plepys, D. and Sandin, D., 2013, March. CAVE2: a hybrid reality environment for immersive simulation and information analysis. In *Is&t/spie electronic imaging* (pp. 864903-864903). International Society for Optics and Photonics.

APPENDIX C (continued)

Springer

https://www.springer.com/gp/rights-permissions/obtaining-permissions/882



SPRINGER LICENSE
TERMS AND CONDITIONS

Feb 18, 2017

This Agreement between Alessandro Febretti ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	4052041473568
License date	
Licensed Content Publisher	Springer
Licensed Content Publication	Springer eBook
Licensed Content Title	The OmegaDesk: Towards a Hybrid 2D and 3D Work Desk
Licensed Content Author	Alessandro Febretti
Licensed Content Date	Jan 1, 2011
Type of Use	Thesis/Dissertation
Portion	Excerpts
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	
Title of your thesis / dissertation	Multiview Immersion in Hybrid Reality Environments
Expected completion date	Mar 2017
Estimated size(pages)	210

Febretti, A., Mateevitsi, V.A., Chau, D., Nishimoto, A., McGinnis, B., Misterka, J., Johnson, A. and Leigh, J., 2011, September. The OmegaDesk: towards a hybrid 2D and 3D work desk. In *International Symposium on Visual Computing* (pp. 13-23). Springer Berlin Heidelberg.

APPENDIX D

VITA

ALESSANDRO FEBRETTI

Education

M.S. in Computer Science and Computer Engineering, 2008
University of Illinois at Chicago (Computer Science, GPA: 4.0)
Politecnico di Milano (Computer Engineering, Cum Laude)
Advisors: Dr. Andrew Johnson and Dr. Franca Garzotto
Thesis: *Evaluating the Flash Platform for Web-Based Collaborative Data Visualization*

B.S. in Computer Engineering, 2005
Politecnico di Milano
Advisor: Dr. Marco Domenico Santambrogio
Thesis: *IRDA IP-CORE: Software and Hardware Implementation on Field-Programmable Gate Arrays*

Research and Professional Experience

Senior Interactive Visualization Specialist, Research Computing Group, Northwestern University, 2015-Present
I design and implement tools for scalable interactive visualization in a variety of research domains including Astrophysics, High-Energy Physics, Art History and Chemistry. I lead software development for the 50-Megapixel immersive display wall at the Center for Advanced Molecular Imaging. I organize and teach workshops on web-based visualization, visual analytics and visualization design for students and researchers both at the Evanston and Chicago campuses.

Research Assistant, Electronic Visualization Laboratory, University of Illinois at Chicago, 2009-2015
I concentrated on Visualization Instrument research (hardware and software) and on human-computer interaction in single and multiuser scenarios. In particular, I investigated the use of hybrid immersive systems as scientific visualization platforms for co-located research teams.

I contributed to the creation of a small scale hybrid workspace (the OmegaDesk) and a large scale Hybrid Reality Environment (the CAVE2). I led the development of a software framework to drive these systems, called Omegalib. Compared to existing toolkits, Omegalib is designed to support dynamic reconfigurability of the display environment: areas of the display can be interactively allocated to 2D or 3D workspaces as needed. Omegalib has been used to develop several scientific visualization applications for the

CAVE2 system, and has been used as the project platform in UIC's Visual Analytics course.

As a spin-off of Omegalib, I created the Omicron input management library and protocol. Omicron simplifies the creation of mixed-input applications, providing an event model that supports 2D devices (touch overlays or remote pointers), 3D devices (marker-based or markerless motion trackers) or non-spatial devices such as voice input and brain interfaces. Omicron has been used as the input platform for a variety of projects, both internal and external to EVL.

Contributions to Omegalib and Omicron by other students have generated two M.S. projects (ongoing) and two M.S. theses. I initiated and supervised one of these theses, titled *"Porthole: A decoupled HTML5 Interface Generator for Virtual Environments"* by M.S. graduate Daniele Donghi.

In addition to my research on hybrid systems, I worked on standard, desktop-based visualization tools and user interfaces. I created the data processing and visualization toolset used to analyze the geophysical data collected by the NASA ENDURANCE autonomous underwater robot in Antarctica. My research contribution to this work involved the use of parallel sonar beam tracing and implicit surface methods for 3D bathymetry reconstruction.

Moreover, I contributed to a multi-year research project in collaboration with the UIC College of Nursing, aimed at improving nurse care for end-of-life patients. Main duties involved designing and developing a clinical decision support system prototype that was used in simulated patient care scenarios. I also helped planning the data collection and evaluation protocol used by this ongoing study, which by its end will have involved more than a hundred professional nurse practitioners.

Software Engineer, Milestone Games, 2007-2009
I worked on gameplay, database and multiplayer logic for two worldwide-published games on Xbox360, Playstation 3, Nintendo Wii and PC. Acted as point of contact with an external team for user interface quality assurance. I was responsible for user interface and game state guideline adherence for Microsoft and Sony product certification.

Teaching Experience

Instructor, Information Technology, Northwestern University, 2015-Present
I'm teaching visualization theory & practice for classes, workshops and seminars across campus. Topics taught include:

1. Introduction to Visualization - *Data Science Bootcamp, March 2015;*
2. Interactive Visualization on the Web with D3.js - *Researcher's Toolkit, Fall 2015 and Spring 2016;*
3. Advanced D3.js – *Researcher's Toolkit, Spring 2016;*

APPENDIX D (continued)

4. Introduction to Tableau Desktop - *Computational Skills for Informatics Seminar Series, Feinberg School of Medicine, December 2015;*
5. Data Presentation and Visual Analytics: Best Practices - *Health Sciences PhD Program, April 2016;*
6. Git and Github*, Software Carpentry Bootcamp June 13 2016;*

Teaching Assistant, Dept. of Computer Science, University of Illinois at Chicago, Fall 2011
Class: *Visualization and Visual Analytics*
Introduced students to the Processing language. Provided assistance during project development and was responsible for project grading.

Lecturer, Dept. of Industrial Design, Politecnico di Milano, 2007-2008
Class: *Human-Computer Interaction*
Held a series of lectures on videogame design and development, introducing students to the game industry development process, gameplay design practices and outlining the technical aspects of game creation.

Teaching Assistant, Dept. of Computer Science, 2005
Class: Introduction to Computer Programming
Responsible for the laboratory session of the course. Provided assistance with C programming exercises, held pre-laboratory lectures on basic programming concepts (IO, conditional programming, pointers and pointer arithmetic, basic data structures and iteration)

Publications

Journals

1. Reda, K., **Febretti, A**., Knoll, A., Aurisano, J., Leigh J., Johnson, A., Papka, M., Hereld, M. *"Visualizing Large, Heterogeneous Data in Hybrid Reality Display Environments"*. IEEE Computer Graphics and Applications, Vol. 33.4 (July-August 2013), pp. 38-48;
2. Dunn Lopez, K., Wilkie, D.J., Yao, Y., Sousa, V., **Febretti, A.**, Stifter, J., Johnson, A., & Keenan, G. (In review, January 2015). *“Nurses’ Numeracy and Graphical Literacy: Informing Studies of Clinical Decision Support Interfaces”*. Medical Decision Making

Conference Proceedings

1. **Febretti, A.**, Sousa V., Lopez K. D., Yao, Y. Johnson, A., Keenan, G. M., Wilkie, D. J., *"One Size Doesn't Fit All: The Efficiency of Graphical, Numerical and Textual Clinical Decision Support for Nurses"*, IEEE VIS 2014 Workshop on Electronic Health Record Data Visualization (EHRVis), November 9-10 2014, Paris, France;

APPENDIX D (continued)

2. **Febretti, A**., Lopez, K. D., Stifter, J., Johnson, A., Keenan, G. M., Wilkie, D. J., *"Evaluating a Clinical Decision Support Interface for End-of-Life Nurse Care"* CHI'14 Extended Abstracts on Human Factors in Computing Systems, Toronto, Canada, April 28 - May 01, 2014;
3. **Febretti, A**., Nishimoto, A., Mateevitsi, V., Renambot, L., Johnson, A., Leigh, J., *"Omegalib: a Multi- View Application Framework for Hybrid Reality Environments"*. IEEE Virtual Reality (IEEE VR 2014), Minneapolis, MN, March 29 - April 2, 2014;
4. **Febretti, A**. *"Supporting Multi-View Immersion on Hybrid Reality Environments"*. IEEE Virtual Reality Doctoral Consortium, Minneapolis, MN, March 29 - April 2, 2014;
5. Reda, K., Aurisano, J., **Febretti, A**., Leigh, J. Johnson, A., *"Visualization Design Patterns for Ultra-Resolution Display Environments"*. VISTech Workshop: Visualization Infrastructure and Systems Technology (VISTech 2013), 2013;
6. **Febretti, A**., Lopez, K., Stifter, J., Johnson, A., Keenan, G., Wilkie, D., *"A Component-Based Evaluation Protocol for Clinical Decision Support Interfaces"*. HCI International, 2013;
7. Lopez, K., Stifter, J., **Febretti, A.**, Johnson, A., Wilkie, D., Keenan, G. *"Improving Learnability and Efficiency of Evidence Based Information in Clinical Decision Support"*. 37th Midwest Nursing Research Society Conference, 2013;
8. **Febretti, A**., Nishimoto, A., Thigpen, T., Talandis, J., Long, L., Pirtle, J., Peterka, T., Verlo, A., Brown, M., Plepys, D., Sandin, D., Renambot, L., Johnson, A., Leigh, J. *"CAVE2: A Hybrid Reality Environment for Immersive Simulation and Information Analysis"* IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics, 2013;
9. **Febretti, A**., Richmond, K., Gulati, S., Flesher, C., Hogan, B.P., Johnson, A., Stone, W.C., Priscu, J., Doran, P. *"Poisson reconstruction of extreme submersed environments: The ENDURANCE exploration of an under-ice Antarctic Lake"* 8th International Symposium on Visual Computing (ISVC12), Crete, Greece, Springer-Verlag, Lecture Notes in Computer Science, 07/16/2012 - 07/18/2012;
10. **Febretti, A**., Mateevitsi, V.A., Chau, D., Nishimoto, A., McGinnis, B., Misterka, J., Johnson, A., Leigh, J. *"The OmegaDesk: Towards A Hybrid 2D & 3D Work Desk"* 7th International Symposium on Visual Computing (ISVC11), Las Vegas, Nevada, 09/26/2011 - 09/28/2011;
11. Richmond, K., **Febretti, A**., Gulati, S., Flesher, C., Hogan, B.P., Murarka, A., Kuhlman, G., Sridharan, M, Johnson, A., Stone, W.C., Priscu, J., Doran, P. *"Sub-Ice Exploration of an Antarctic Lake: Results from the ENDURANCE Project"* 17th International Symposium on Unmanned Untethered Submersible Technology (UUST11), Portsmouth, NH, US, 08/21/2011 - 08/24/2011;

APPENDIX D (continued)

12. **Febretti, A**., Garzotto, F. *"Usability, Playability, and Long-Term Engagement in Computer Games"* CHI'09 Extended Abstracts on Human Factors in Computing Systems, pp. 4063-4068, ACM SIGCHI, Boston, MA, 4/4/2009-4/9/2009

Abstracts, Posters and Videos

1. Ye, A., Ajilore O., **Febretti, A.**, Johnson, A., Elkarim, G., Yang, S., Magin, R., Kumar, A., Leow, A., *"Using Dimensionality Reduction to Explore Virtual Reality Lobectomies"*, 23rd Meeting of the International Society for Magnetic Resonance in Medicine, May 30 - June 5, Toronto, Ontario – Canada;
2. Ye, A., Ajilore O., **Febretti, A.**, Johnson, A., Elkarim, G., Yang, S., Magin, R., Kumar, A., Leow, A., *"A Virtual Reality Exploration of Brain Connectivity Metrics using Dimensionality Reduction",* 70th Annual Meeting of the Society of Biological Psychiatry, Toronto, Ontario - Canada;
3. Dunn Lopez, K., Stifter, J., **Febretti, A**., Sousa, V., Johnson, A., Wilkie, D. J., & Keenan, G. M, *"Formative and summative testing of nursing clinical decision support prototypes"*. 10th Biennial Conference of the Association for Common European Nursing Diagnoses, Interventions and Outcomes, e-health and Nursing, Knowledge for Patient Care, Bern Switzerland;
4. **Febretti, A.**, Richmond K., Doran, P., Johnson, A., *"Parallel Processing and Immersive Visualization of Sonar Data"* IEEE Symposium on Large Data Visualization (LDAV), November 9-10 2014, Paris, France;
5. Keenan GM., Stifter J., **Febretti A**., Lodhi M., Dunn-Lopez K., Yao Y., Khokhar A., Johnson A., Ansari R., Wilkie DJ., *"Well tested electronic care planning system produces powerful evidence for nursing care decision."* Advocate Health System Research Day. Chicago, Illinois Masonic, April 9th, 2014;
6. Leigh, J., Johnson, A., Renambot, L., Long, L., Sandin, D., Tolandis, J., **Febretti, A**., Nishimoto, A., *"CAVE2 Documentary"* Video in IEEE VR 2014 and IEEE VR 2014 YouTube channel, Minneapolis, Minnesota, March 29, 2014;
7. Dunn Lopez, K., **Febretti, A**., Yao, Y., Stifter, J, Johnson, A., Wilkie, D., Keenan, G., *"Clinical decision support alerts forms: Nurse preferences and relationships with nurse characteristics."* American Medical Informatics Association, Annual Symposium, Washington, DC., 2013;
8. Dunn-Lopez, K., Stifter, J., **Febretti, A**., Johnson, A., Wilkie, D., Keenan, G., *"Improving Usability of Evidence Based Information in Clinical Decision Support"*, Midwest Nursing Research Society 37th Annual Research Conference, Chicago, IL, March 7-13, 2013;

Non-Refereed Whitepapers and Articles

APPENDIX D (continued)

1. **Febretti, A**. *"Distributed Sum on a 2D mesh using MPI"* ECE566 Parallel Processing, University of Illinois at Chicago, 2013;
2. **Febretti, A**. *"Parallel Gaussian Elimination Using MPI"* ECE566 Parallel Processing, University of Illinois at Chicago, 2013;
3. **Febretti, A**., Frossi, A. *"Template Choice and Template Hierarchy Graph construction during the Partition Phase in reconfigurable environments"* Laboratory of Microarchitectures, Politecnico di Milano, 2006;
4. **Febretti, A**. *"Fast late-bound invocation through Dynamic Method delegates"* Codeproject, 2005;

Invited Talks

1. **Febretti, A.** *“Introduction to D3.js”* 2nd Annual International Conference on Computational Social Science (IC2S2), Kellogg School of Management, Northwestern University, 23-26 June 2016;
2. **Febretti, A.** *“Supporting Co-Located Collaboration in Hybrid Immersive Environments”* MAT Seminar Series, University of California, Santa Barbara, Feb 29 2016;
3. **Febretti, A**. *"Building The Lens - Visualization Instrument Research at the Electronic Visualization Lab"* Talk, Center of Excellence in Wireless and Information Technology, Stony Brook University, 2014;
4. **Febretti, A.** *“CAVE2: Equalizer and Omegalib”*, Talk, Blue Brain Group, École Polytechnique Fédérale de Lausanne, Switzerland, 2014;
5. **Febretti, A.,** Leigh, J. *“The CAVE2 Hybrid Reality Environment”*, UIC Computer Science Graduate Student Association, 2013;
6. **Febretti, A.,** *"...For Science! Building Visualizations, Managing Dynamic Workflows and Coding Without Swearing too Much"* Talk, UIC Computer Science Graduate Student Association, 2012;
7. Doran, P.T., Stone, W., Priscu, J.C., Obryk, M., **Febretti, A.** *"The ENDURANCE Autonomous Underwater Vehicle"* Keynote, 31st Science Week, Scientific Committee on Antarctic Research, 2012;