Stony Brook University

Interacting with Gigapixel Displays

A Dissertation Presented

By

Charilaos Papadopoulos

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2015

Stony Brook University

The Graduate School

Charilaos Papadopoulos

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Arie E. Kaufman - Advisor
Distinguished Professor, Dept. of Computer Science, Stony Brook University

Dimitris Samaras - Chairperson of Defense
Associate Professor, Dept. of Computer Science, Stony Brook University

Klaus Mueller
Professor, Dept. of Computer Science, Stony Brook University

Amitabh Varshney
Professor, Dept. of Computer Science, University of Maryland

This dissertation is accepted by the Graduate School.

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Interacting with Gigapixel Displays

by

Charilaos Papadopoulos

Doctor of Philosophy

in

Computer Science

Stony Brook University

2015

Large, high-resolution displays (LHiRDs) are a powerful visualization and data exploration tool. These facilities, with resolutions in the hundreds of millions of pixels, have proliferated in industry and research laboratories, enabling scientists, engineers or physicians to better understand the problems that they face. Recently, the Reality Deck pushed LHiRDs past the gigapixel resolution barrier, offering 1.5 gigapixels and providing a 360° horizontal field of view, within a large workspace of $33' \times 19'$.

Room-sized facilities such as the Reality Deck simultaneously promote and demand physical navigation on behalf of the user. Consequently, static user interfaces (e.g., keyboard and mouse) do not translate themselves well to such systems. Additionally, the sheer size and resolution of the Reality Deck can trigger new and interesting usage patterns in how users navigate within the visualization space. These patterns are worthy of investigation and can also be exploited in order to improve the system performance.

The goal of this dissertation is to evaluate, leverage and further enable the physical navigation aspects of room-sized gigapixel resolution displays such as the Reality Deck. This is accomplished via four pillars of research work. The first pillar is the introduction of interfaces for unencumbered, device-less and hand-driven interaction with such systems. The second pillar utilizes the perceptual characteristics of LHiRDs and the human visual system in order to improve performance when displaying gigapixel-resolution data. The

third pillar focuses on the evaluation of user performance within LHiRDs, while performing core visualization tasks through physical navigation. The fourth pillar is the introduction of *VEEVVIE*, the Visual Explorer for Empirical Visualization, VR and Interaction Experiments. *VEEVVIE* is a visual analytics tool that enables the visual exploration of data that stems from visualization, virtual reality and interaction experiments, such as those conducted in LHiRDs, allowing researchers to validate and generate insights and hypotheses in an interactive way.

To my family

Contents

List of Figures

List of Tables

Acknowledgements

Personal Acknowledgments

The pursuit of a PhD is a long, sometimes arduous yet ultimately rewarding journey. My academic travels would not have been successful without the mentoring, guidance and personal support of a number of people.

Most important of all, I am immensely grateful to my parents, George Papadopoulos and Margarita Rapti Papadopoulos for enabling me to pursue my goals and supporting me along the way. Had they not been there for me, this dissertation would not exist. I would also like to give a shoot-out to my brothers, Thymios and Panagiotis!

Another person of extreme importance to my academic pursuits is my advisor, Prof. Arie E. Kaufman. His leap of faith, when offering me a position at Stony Brook University, is what enabled the research pursuits detailed in this document. Through his continued advice and personal support, Prof. Kaufman taught me what it means to be an academic and guided me through six years worth of studies. For the chance to be his student and ultimate academic offspring, I am extremely grateful. I would be remiss if I did not acknowledge the other three members of my dissertation committee. Professors Dimitris Samaras and Klaus Mueller of Stony Brook University allowed me to grow academically through their excellent courses and, along with Professor Amitabh Varshney of the University of Maryland, have made this a better dissertation via their feedback and support. Finally, I would like to thank Prof. Georgios Papaioannou at the Athens University of Economics and Business for advising me through my undergraduate degree and inspiring me to pursue a PhD.

I would also like to thank several members of the Center of Visual Computing for their personal and professional guidance and work, which made this dissertation possible. Firstly, many thanks are due to Dr. Kaloian Petkov, a dear academic sibling and repeat co-author, for guiding me as I fumbled through my first few years at Stony Brook. My gratitude goes out to Ken Gladky for his impeccable technical support at the Center and to Brad Nelson, the amazing engineer who helped put together the numerous virtual reality facilities I got to tinker with. I would also like to thank my labmates (and sometimes co-authors), Koosha Mirhosseini, Ievgeniia Gutenko, Qi Sun, Ji Hwan Park, Krishna Chaitanya, Joseph Marino and anyone else I might have missed.

Finally, I would like to offer my sincere gratitude to a number of folks who supported me personally at different periods while at Stony Brook University. Listed in alphabetical order: Mohammadali Bankehsaz, Caspian Blobber, Ritwik Bose, Samira Darvishi, Ievgeniia Gutenko, Saj Hossain, Lena Lazareva, Heraldo Memelli, Carlos Oreggo, Jill Purrington, Jen Sidorova, Juhi Tyagi, Spyros Triantafillakis, Bill Woerner, Tomas Yago. Thank you all ladies and gentlemen!

Data Sources

A substantial number of the experimental results and benchmarks, presented in figures throughout this dissertation, have been conducted using data sources (e.g. geometry models, medical datasets, map tilesets, gigapixel imagery) which have been graciously made available to the public by a number of individuals, institutions and corporations. These parties include: Mapquest, Gigapan, the University of Arizona High Resolution Imaging Science Experiment (HiRISE), Jacco Bikker, Frank Meinl, Crytek, the RCSB Protein Data Bank, the National Institute of Health (NIH), Pelican Mapping, the United States Geological Service (USGS), the North American Space Administration (NASA), the Stony Brook University School of Marine and Atmospheric Sciences (SoMAS) and the New York Resiliency Institute for Storms and Emergencies (NYSRISE). The author of this dissertation makes no copyright claims regarding those data sources.

Funding Sources

The research work presented in this dissertation has been generously funded by a number of institutions, including the United States National Science Foundation (NSF Grants CNS-0959979, IIP-1069147 and CNS-1302246), the Stony Brook University Center of Excellence in Wireless and Information Technology (CEWIT) and the New York Resiliency Institute for Storms and Emergencies (NYSRISE).

List of Publications

JOURNAL PUBLICATIONS:

C. Papadopoulos, K. Petkov, A. E. Kaufman and K. Mueller. "The Reality Deck - Immersive Gigapixel Display". *IEEE Computer Graphics and Applications*. 35(1), pp 33-45, 2015. Refer to Chapter 3.

C. Papadopoulos and A. E. Kaufman. "Acuity-Driven Gigapixel Visualization". *IEEE Transactions on Visualization and Computer Graphics*, 19(12), pp 2886-2895, 2013. Refer to Chapter 7.

K. Petkov, **C. Papadopoulos**, M. Zhang, A. E. Kaufman and X. Gu. "Interactive Visibility Retargeting in VR Using Conformal Visualization". *IEEE Transactions on Visualization and Computer Graphics*, 18(7), pp 1027-1040, 2012.

G. Papaioannou, M. L. Menexi and **C. Papadopoulos**. "Real-time Volume-based Ambient Occlusion". *IEEE Transactions on Visualization and Computer Graphics*, 16(5), pp 752-762, 2010.

REFEREED CONFERENCE PUBLICATIONS:

C. Papadopoulos, I. Gutenko and A. E. Kaufman. "*VEEVVIE* - Visual Explorer for Empirical Visualization, VR and Interaction Experiments". *Conditionally accepted to IEEE Visual Analytics Science and Technology*, 2015. Refer to Chapter 9.

C. Papadopoulos, S. Mirhosseini, I. Gutenko, K. Petkov, A. E. Kaufman and B. Laha. "Scalability limits of large immersive high-resolution displays". *IEEE Virtual Reality*, pp 11-19, March 2015. Refer to Chapter 8.

C. Papadopoulos, H. Choi, J. Sinha, K. Yun, A. E. Kaufman, D. Samaras and B. Laha.

"Practical Chirocentric 3DUI Platform for Immersive Environments". *IEEE 3D User Interfaces Symposium*, pp 31-34, March 2015. Refer to Chapter 6.

C. Papadopoulos, B. Laha and A. E. Kaufman. "Interacting with mixed-reality systems". *Death of the Desktop Workshop* (hosted by IEEE VisWeek), November 2014. Refer to Chapter 6.

I. Gutenko, K. Petkov, **C. Papadopoulos**, X. Zhao, J. H. Park, A. E. Kaufman and R. Cha. "Remote volume rendering pipeline for mHealth applications". *SPIE Medical Imaging*, March 2014.

K. Petkov, **C. Papadopoulos** and A. E. Kaufman. "Visual Exploration of the Infinite Canvas". *IEEE Virtual Reality*, pp 11-14, March 2013. Refer to Chapter 3.

K. Petkov, **C. Papadopoulos**, M. Zhang, A. E. Kaufman and X. Gu. "Conformal Visualization for Partially Immersive Platforms". *IEEE Virtual Reality*, pp 143-150, March 2011.

WORKSHOP PUBLICATIONS AND POSTERS:

Q. Sun, S. Mirhosseini, I. Gutenko, J. H. Park, **C. Papadopoulos**, B. Laha, and A. E. Kaufman; "Buyer's Satisfaction in A Virtual Fitting Room Scenario Based on Realism of Avatar". *IEEE 3D User Interfaces Symposium*, pp 183-184, March 2015.

C. Papadopoulos, S. Mirhosseini and A. E. Kaufman. "Immersive Visualization of Storm-surge Simulations". *CEWIT International Conference & Expo on Emerging Technologies for a Smarter World*, October 2014. Refer to Chapter 3 and Chapter 4.

C. Papadopoulos, H. J. Choi, J. Sinha, K. Yun, D. Samaras and A. E. Kaufman. "Gestural Interfaces for the Reality Deck". *Center for Dynamic Data Analytics (CDDA) Workshop*, September 2013. Refer to Chapter 6.

C. Papadopoulos, K. Petkov, A. E. Kaufman, A. Pinkas-Sarafova, C. Chipev, M. Simon. "Immersive Training for Work under cGMP". *NYSTEM Collaboration & Renewal*, May 2013.

C. Papadopoulos, K. Petkov and A. E. Kaufman. "Building the Reality Deck". *POWERWALL International Workshop on Interactive, Ultra-High-Resolution Displays* (hosted by ACM CHI), April 2013. Refer to Chapter 3.

A. Kaufman, **C. Papadopoulos** and K. Petkov. "1,500,000,000 pixels on a budget - Building the RealityDeck". *CEWIT International Conference & Expo on Emerging Technologies for a Smarter World*, November 2012. Refer to Chapter 3.

C. Papadopoulos, D. Sugarman and A. E. Kaufman. "NuNav3D: A Touch-less, Body-driven interface for 3D Navigation". *IEEE Virtual Reality*, March 2012. Refer to Chapter 5.

C. Papadopoulos, D. Sugarman and A. E. Kaufman. "Body-driven Navigation for 3D Visualization using NuNav3D". *IEEE Pacific Vis*, February 2012. Refer to Chapter 5.

A. E. Kaufman, K. Mueller, D. Samaras, H. Qin, A. Varshney, **C. Papadopoulos** and K. Petkov. "The RealityDeck - Immersive Giga-Pixel Display". *CEWIT International Conference & Expo on Emerging Technologies for a Smarter World*, November 2011. Refer to Chapter 3.

C. Papadopoulos and G. Papaioannou. "Realistic Real-time Underwater Caustics and Godrays". *Graphicon*, October 2009.

TALKS:

C. Papadopoulos. "Immersive Visualization of Storm-surge Simulations". *CEWIT International Conference & Expo on Emerging Technologies for a Smarter World*, October 2014. Refer to Chapter 3 and Chapter 4.

C. Papadopoulos. "Immersive Display and Exploration of Gigapixel Images". *IEEE Virtual Reality 2014 Doctoral Consortium*, March 2014. Refer to Chapter 3 and Chapter 6.

C. Papadopoulos and A. E. Kaufman. "Acuity-Driven Gigapixel Visualization". *IEEE Visualization*, October 2013. Refer to Chapter 7.

C. Papadopoulos and A. E. Kaufman. "Gestural Interfaces for the Reality Deck". *CEWIT International Conference & Expo on Emerging Technologies for a Smarter World*, October 2013. Refer to Chapter 6.

A. E. Kaufman, **C. Papadopoulos** and K. Petkov. "Immersive Visualization for Big Data". *Center for Dynamic Data Analytics (CDDA) Workshop*, September 2013.

C. Papadopoulos. "Looking at Bits & Bytes - A History of Virtual Reality". *Stony Brook University Provost's Graduate Lecture Series*, May 2013.

Chapter 1

Introduction

1.1 Why build Large High-Resolution Displays?

Most domain experts, like the vast majority of computer users, interact with visualizations on a single screen via a pointer based interface. This setup can manifest itself in a number of ways. A desktop computer with a mouse and keyboard, a laptop with a touchpad and, more recently, a tablet with a touch-based interface. In all of these cases the visualization is delivered via a screen of fixed resolution, between two and four megapixels (or millions of pixels) in a modern setup. Moreover, pointer-based visualization interfaces expose some sort of panning and zooming functionality, allowing users to reposition their virtual viewport into the visual representation of the data only focusing on a segment of it.

Such an approach to visualization is efficient as long as the data "fits" within the available screen resolution. If the data is substantially larger than the available screen real-estate, then the user has to resort to extensive virtual navigation motions (panning and zooming) in order to inspect areas of interest. From this abundance of virtual navigation, two issues arise. First, the user spends substantial amounts of time doing "grunt work" while repositioning the virtual camera, increasing fatigue and overall taking longer to complete visualization tasks. Secondly, when zoomed into a sub-section of the data, the user is no longer able to appreciate the surrounding context (which is critical for decision making, as outlined earlier).

Increasing the size and resolution of the display is an obvious solution to this problem.

In fact, multi-display desktop computer setups are becoming more commonplace. They allow for increased productivity as users can partition multiple data views or applications into separate monitors. Consequently, important information can remain visible all the time. Furthermore, the user has to spend less time context-switching and wrestling with the window manager in order to recover the relevant application from a multitude of minimized or backgrounded windows. When taken to the extreme, this multi-display approach can be scaled up to wall-sized display setups. Such systems are often comprised of tens or hundreds of displays (connected to multiple computers), which are abstracted into a single desktop via distributed rendering software. Multiple monickers exist for these facilities. *Tiled display walls*, *Large High-Resolution Displays* or *Powerwalls* are a few terms found in the literature used to describe these systems. In this dissertation, the term Large, High Resolution Displays and the acronym LHiRD are used to refer this category of visualization environments. Generally speaking, these facilities are differentiated from multi-display desktop setups in the following ways. First, LHiRDs are significantly larger than a multi-display desktop, in terms of screen count and, more importantly, aggregate screen size/resolution. Second, LHiRDs enable and encourage physical navigation of the data, rather than forcing users to virtually manipulate the visualization. Effectively, users can zoom by walking up to the displays in order to resolve high-frequency details and pan simply by looking at a different section of the LHiRD. Physical navigation in the context of LHiRD visualization can result in substantially increased performance for numerous visualization tasks (this has been well documented in the human-computer interaction literature, as overviewed in Chapter 2). As a result, even though constructing an LHiRD can be an expensive proposition, such systems have proliferated throughout academic, research and industrial settings throughout the years.

1.2 Evolution of LHiRDs

The progenitor of LHiRDs was the University of Minnesota Powerwall (constructed in the early 90s), providing a 3200 by 2400 viewport using 4 projectors driven by SGI Onyx computers. Since then, advances in hardware (particularly the proliferation of 3D graphics accelerators and high-resolution LCD panels) permitted the scalability of the LHiRD into systems that offer hundreds of megapixels in resolution and hundreds of square feet of display space. However, up until 2012 it appeared that the size of such systems had

reached a cap. In fact, at that time, the largest LHiRD in the world was the Stallion Powerwall at the Texas Advanced Computing Center (with a few similar systems being installed at other research facilities), offering approximately 300 megapixels in resolution. This resolution cap was not imposed by hardware or software limitations and felt rather arbitrary. Rather, it appeared feasible that a significantly more capable LHiRD could be constructed, offering substantially higher resolution and an enlarged workspace. Additionally, such a system could further enable physical navigation by providing a panoramic experience to the user (whereas prior tiled displays had mostly been planar). A group of visualization researchers at Stony Brook University conceived such a system and proceeded to implement it in 2012. The author of this dissertation had the good fortune to be part of this group of scientists. The resulting facility, termed the *Reality Deck* [PPKM15], is a gigapixel resolution immersive display. It is comprised of four walls made out of a total of 416 modified LCD monitors. It offers a total resolution of over 1.5 billion pixels in a horizontally immersive setting and a workspace of approximately $33'$ by $19'$ (and $11'$ high). The Reality Deck's design and implementation is described in detail in Chapter 3 and the software and visualization applications in Chapter 4. This visualization environment served as the motivation and implementation platform for the majority of the research work described in this dissertation.

1.3 Overarching Motivation

The recurring theme of this dissertation is the notion of physical navigation. Physical navigation is the most "natural" way in which users interact with LHiRDs. Thus, being an interaction modality in and of itself, physical navigation should be enabled, exploited and evaluated. Other interaction methods should be designed with physical navigation in mind. Rendering algorithms and techniques can be enhanced to take advantage of physical navigation in order to improve performance. And physical navigation within a LHiRD, along with its implications in the conduct of various tasks, needs to be measured. Finally, the conduct and analysis of such usability experiments can be enhanced through interactive visual exploration tools. These four main research directions grew organically from the author's interaction with LHiRDs through the years. They are briefly discussed in the following paragraphs and, in more detail, in later chapters of this dissertation.

1.4 Enabling Physical Navigation

The large workspace of the Reality Deck promotes physical navigation, even more so than previous tiled display designs. A simple pan of the head can provide an overview of 1.5 gigapixels of visualized information. But data exploration is not constrained to "looking". Users must be provided with appropriate interaction modalities. Most LHiRDs, including the Reality Deck, support traditional interaction techniques (e.g., mouse/keyboard). However, the implementation of the LHiRD should not require that the user return to a fixed location every time she wishes to interact with the visualization. Indeed, research has shown that such *tethering* effects can affect the ergonomics of user interactions with LHiRDs [BN08]. Thus, it is important to provide users with interfaces that are fully untethered. Optimally, such an interface should be deviceless, controllable by the user's hands, with gestures and motions mapping to different manipulations of the data. The first contribution of this dissertation examines a hand-based natural user interface that provides 4 degrees of freedom (DoFs) of camera control without the need for external devices. This interface, termed NuNav3D [PSK12a] and described in Chapter 5, is implemented using a Microsoft Kinect sensor and evaluated in two different data exploration settings.

The evaluation of NuNav3D revealed a shortcoming of mapping hand-motions to camera manipulations. Specifically, users state their intent to begin and end navigation by assuming a predetermined navigation pose. When entering or exiting that pose, unintended navigations could occur which could affect a carefully positioned viewport into the data. It quickly became obvious that a more "explicit" way of declaring navigation intent would be required. Additionally, NuNav3D was strictly focused on navigation and did not provide a modality for triggering other actions within the visualization. Both of these shortcomings are addressed in our chirocentric user interface platform for immersive virtual environments, described in Chapter 6.

1.5 Leveraging Physical Navigation

Since physical navigation is a de-facto characteristic of LHiRDs, it can be leveraged to augment the performance of the software and hardware driving the system. When streaming

high resolution data, positional tracking can be used to guide the level-of-detail (LoD) selection across the display, delivering high-resolution data near the user, but lower fidelity imagery farther away. By ensuring that the LoD calculation takes into account the user's visual acuity and the physical specifications of the display, this manipulation of the rendering can be unnoticeable. This technique, termed *Acuity-Driven Gigapixel Visualization* [PK13b] is the third major contribution of this dissertation and is described in Chapter 7.

1.6 Evaluating the limits of LHiRDs

Prior research into the ergonomics of LHiRDs has thoroughly quantified the benefits they can offer to user performance in various types of data exploration tasks. However, a closer look at the literature yields an important fact. The vast majority of user studies have been conducted on LHiRDs of relatively small size compared to the state of the art (even prior to the construction of the Reality Deck). In fact, the largest LHiRD utilized in a peer review study offered an aggregate resolution of approximately 131 megapixels, substantially smaller than the 300 megapixel systems that predate the Reality Deck. Consequently, the quantifiable benefits of LHiRDs can only be claimed for displays of similar sizes. Conceivably there exists a point of diminishing returns, past which the total pixel count or surface area of the display is too large for further performance improvements to be obtained. The fourth major contribution of this dissertation is the report on a user study aimed at quantifying the scalability limits of LHiRDs. By thoroughly examining the literature, we identified several core visualization tasks and designed a user study, with the display resolution being the main variable, targeted at quantifying user performance. We report on this study in Chapter 8.

1.7 Visual Analytics for Visualization Experiments

Empirical experimentation and hypothesis-driven evaluation have been at the core of research centered around human-factors for computing systems (including the study presented in Chapter 8). Visualization often seems to serve as a supplementary tool to this type of evaluation, helping researchers generate new insights and hypotheses. In fact, in our study, it was through the visualization of user movement, gaze and presence that we unearthed interesting patterns for how users tackle different tasks within a large immersive gigapixel display. The final contribution of this thesis aims to take the application of visualization in empirical experimentation to the next level. In Chapter 9, we describe *VEEVVIE*, the Visual Explorer for Empirical Visualization, VR and Interaction Experiments. Driven by an ontology which is informed by prior visualization and interaction experiments, *VEEVVIE* allows scientists to visually analyze experimental data and enables them to gain insight that would not be obvious through statistical analysis.

1.8 Summary

The overall goal of this dissertation is to enable, leverage and evaluate physical user interactions with gigapixel-resolution LHiRDs. This task is accomplished by offering insights on natural user interface design, providing techniques for improving system performance by taking advantage of the perceptual affordances such systems, quantifying their scalability limits and providing tools that empower researchers to better understand these limits. Through these works, the author hopes to document the design of the current state of the art LHiRD, enable new forms of user interactions with such systems, allow engineers to squeeze every drop of performance out of their facilities, and finally guide future LHiRD designs.

Chapter 2

Related Work

The contributions presented in this dissertation are based on foundational work in the broad fields of visualization, virtual environments engineering and user interaction. The goal of this chapter is to provide a thorough overview of related work in numerous areas:

1. Central concepts of visualization system design
2. Existing immersive visualization environment installations
3. Software frameworks that ease the development of distributed visualization applications
4. Gesture-based user interfaces and 3D navigation techniques
5. Techniques for rendering and exploring gigapixel imagery
6. User studies on large high resolution displays or similar systems

2.1 Fundamentals of Visualization System Design

During the dawn of Virtual Reality (VR) research in 1965, Ivan E. Sutherland introduced the concept of the "Ultimate Display" [Sut65], a room within which the computer can control the existence of matter. While such a machine does not currently exist, Sutherland's vision triggered an explosion of different approaches that strived to achieve the

objective of the "Ultimate Display". These approaches, aimed to fulfill two main requirements, saturating the observer's field of view and providing sufficient visual acuity for the displayed imagery to be considered realistic.

The field-of-view (FoV) requirement can be further analyzed as follows. The FoV provided by a single display is a single angle equal to $2tan^{-1}(\frac{W}{2D})$ [CNSD+92], where W is the width of the display and D is the distance of the user from the display. For example, a single 19-inch diagonal display provides the user with a 45° FoV when the user's eyes are 18" away. Modern Head Mounted Displays (HMDs) produce a FoV in the 100° range. When examining FoV for multi-display installations, the FoV for each individual display will vary depending on the user's position inside the environment but the entirety of the system will generate a full 360° FoV (for a completely immersive setup such as a 6-wall CAVE). The impact of the provided FoV on immersion can be exemplified by commercial entertainment applications such as IMAX movie theaters, in which the position of a very large screen in relation to the viewer facilitates the suspension of disbelief and creates a more immersive experience.

The quality of a rendered image on a computer graphics display is often measured by its resolution - the number of pixels the display uses to generate the image. However, the perceived detail of a computer generated image by a human observer is a factor of both the resolution of the display and the position of the observer in relation to the display. This measurement is termed the visual acuity of the display. Formally, visual acuity is defined as the portion of a pixel taken from the center of the display that spans one minute of the field of view. For a display of H pixels of horizontal resolution with a width of W inches, the pixel pitch of that display is equal to $P = \frac{H}{W}$ pixels per inch. If the user is standing D inches away from the center of the display, then the angular imprint of a single pixel on the user's retina is roughly $tan^{-1}(\frac{P}{D})$ (measured in minutes) [CNSD+92]. The inverse of this metric is the visual acuity of the display. Another popular metric of visual acuity is the Snellen fraction. For a viewer with visual acuity of $\frac{20}{X}$, according to the Snellen fraction, that viewer is able to perceive at $20'$ what a person of average eyesight perceives at $100'$. The straightforward implication is that $\frac{20}{20}$ is the visual acuity metric for an average person (for whom a visual angle of one minute is perceivable), $\frac{20}{10}$ being above average, $\frac{20}{40}$ being the minimum for driving at night-time and $\frac{20}{200}$ being the definition for legally blind. As an example, we will apply the definition of visual acuity to a 1280×1024 display with a 19" diagonal and the user standing at a distance of $0.45m$

away from it. This provides us with a $\frac{20}{45}$ acuity metric, good enough to drive at night but not enough to saturate the average human visual system.

Apart from the two main considerations of FoV and visual acuity, the *intrusion* of the VR instrument into the user experience must be taken into consideration. Such intrusion is a result of the isolation of the viewer's sensory input. A necessity stemming from this shortcoming of certain VR systems, is the need for a visual representation of the user's body and the entirety of the physical environment the user will be operating in. This demand can be complicated even further when there exists a need for collaborative interaction inside the virtual environment. In that case, multiple users have to be tracked inside the physical space, then subsequently modeled and displayed inside the virtual environment in order to reduce the disconnection between the physical and virtual world. Certain types of IVEs (such as CAVEs) overcome this shortcoming, as the representations of the users and the physical constraints of the space are implicit, a result of the fact that the users visual system is not obstructed by a head-mounted display or other device.

Designing and constructing a large visualization system is an act of balancing the above three criteria. In the following section, we examine different implementations of such systems and discuss the user experience they provide to users.

2.2 Large-scale Immersive Visualization Environments

Large-scale visualization systems that strive to satisfy the above factors are frequently and interchangeably termed "Immersive Visualization Environments" or "Immersive Virtual Environments". Through this dissertation we utilize the acronym IVE when referring to these facilities. Such systems are generally split into two broad categories. On one hand are facilities that aim at maximizing immersion, creating a VR experience without the presence and ergonomics issues of head-mounted displays. On the other side exist facilities whose goal is the optimization of visual acuity, often going hand-in-hand with a high aggregate resolution. Until relatively recently those two categories had been relatively distinct. In this section we present several IVE examples from each category as well as a few "hybrid systems".

2.2.1 CAVE Automatic Visual Environment and Variations

The concept of the CAVE Automatic Visual Environment as a virtual reality system was introduced in 1992 by Cruz-Neira et al. [CNSD+92]. The CAVE was proposed as an alternative to VR systems of the time (HMDs, BOOMs and basic CRT stereo displays), effectively addressing individual issues with each system. When compared to HMDs, the CAVE offered a similarly immersive FoV without any sensory intrusion and superior visual acuity (due to the limitations of head-mounted displays of the time).

From an engineering perspective, the original CAVE consisted of 3 walls and a floor surface. The wall images were displayed using back-projection, whereas the floor was rendered via front projection from above, using a highly reflective mirror to reduce space requirements. The original CAVE was driven by SGI workstations and utilized an active shutterglass system to achieve stereoscopic rendering. A magnetic tracking system was used to track the viewer's head position in a single user scenario and offer correct perspective projections. The display surfaces were framelocked among each other so as to refresh simultaneously and eliminate screen tearing along CAVE edges. Refresh for the entire CAVE was also synchronized to an external IR beacon system that drove the shutterglasses. The engineering considerations mentioned above remain highly relevant in modern CAVE systems.

More recent CAVE designs such as the 5-wall Immersive Cabin (IC) [QZP+08] have increased the available immersion while reducing costs. The IC utilizes low-cost commodity workstations and a cheap IR based-tracking system to achieve an immersive visualization with 4 back-projected wall surfaces and a front-projected floor. In a similar fashion, 6-wall CAVEs [1] can also be constructed, in which the floor and ceiling surfaces are also back projected using mirrors to reduce the spatial requirements while accommodating for the throw distance of the projectors. However, in the case of a back-projected floor surface, there exists a significant consideration in that the floor must also be able to bear a load equal to the maximum number of simultaneous users of the environment. This sometimes demands the utilization of at least one support beam running underneath the floor surface, the footprint of which must be taken into account when designing the projection subsystem (a potential solution would be to split the floor surface into two subsurfaces, with each projection effectively grazing across the support structure to create a seamless

[1] http://kvl.kaust.edu.sa/Pages/CORNEA.aspx

image). In any case, 6-wall CAVEs present several engineering challenges that increase cost significantly when compared to 4 or 5 surface solutions while the benefit in immersion is relatively small (since the horizontal FoV remains the same whereas the vertical FoV is already almost saturated in 4-5 wall systems). In terms of resolution, the original CAVE offered a visual acuity of $\frac{20}{140}$ on the Snellen scale, which was state-of-the-art for the time but still nowhere near the required amount for visualization of intricate datasets. This lack of detail was a result of projecting a 1 megapixel image on a $3m^2$ screen. Modern projectors resolutions can scale up to 8 megapixel and multiple projectors can be tiled to further increase resolution on a single surface. However, the cost of such projectors (such as the Christie 4k[2]) can be prohibitively high for most CAVE setups.

In original CAVE paper, Cruz-Neira et al. [CNSD+92] touched upon several issues that are still of significant importance when designing IVEs. It is well known that to achieve a correct stereo effect, the rendering system must be aware of the viewer's position inside the CAVE volume (in order for proper off-axis projections to be calculated). However, a second important consideration is the orientation of the viewer's head. If one assumes a static vertical head orientation, then as the viewer turns his head while observing the visualization, the stereo effect may be inverted or lost (this is particularly noticeable in the floor and ceiling surfaces of CAVEs). Another important matter related to CAVEs is the inherit clash between collaborative usage and head-tracking. At the moment, there exist few methods that allow multiple people to get individual perspective correct visualizations out of a single IVE. A simplistic method for accommodating such multi-user projections in a CAVE would be to multiplex the projections in the time-domain by using active shutter glasses (the Responsive Workbench [ABM+97] is such a system that supports two simultaneous users). However, this imposes certain requirements in terms of the refresh rate of the display and shutter-glass system since, in order to achieve a natural and non-disorienting effect, a refresh rate of $60Hz$ per user per eye must be attained. Usually when multiple users are present, a static head position is assumed for the entirety of the visualization. Kulik et al. [KKB+11] have described a solution for providing unique stereo views for up to six users via a multiplexing system. However their approach multiplies the number of projectors required for each display surface. Modern IVEs tend to either generate the collaborative visuals assuming a fixed viewer position or use compromise solutions such as "panoptic" stereo [FNM+14].

[2] http://www.christiedigital.com/AMEN/Products/christieMirageS4K.htm

2.2.2 Large High-Resolution Displays

While CAVEs offer a near-total immersive feeling (depending on the configuration), they do not offer enough resolution to saturate the human visual system (even for state of the art systems). The lack of large-scale high-resolution visualization systems triggered the development of Large, High-Resolution Displays (LHiRDs) or "Powerwalls". These visualization systems are comprised of a lattice of tiled display devices, traditionally, either LCDs or Projectors (with more expensive display system solutions such as the Christie Microtile[3] being also available).

Early work on Powerwalls began in the early 1990s in places like University of Minnesota[4] , University of North Carolina and Princeton University. Princeton University pioneered the field with an eight-tile display wall using back-projection which was later scaled to a 24 tile setup for a total resolution of 18 megapixels. In their series of papers [CWG+02] [LCC+00] the researchers from Princeton have tackled various issues regarding the usage of projectors to drive tiled display arrays as well as design principles for clustered applications. One of the main shortcomings with projector-driven Powerwalls is the necessity for calibration to compensate for projector misalignment, color and brightness mismatching and distortions introduced by peculiarities in the projector lenses [WCL03] [YCF+00]. Additionally, projectors require regular maintenance (mainly replacement of the internal light bulbs) and tend to have high power requirements. They also produce large amounts of heat, leading to the need for improved HVAC support.

An alternative to projector-driven Powerwalls, is using a lattice of LCD displays. The University of Texas Stallion Powerwall[5] , the highest resolution single-wall display in the world, utilizes 75 4-megapixel monitors for a combined resolution of over 300 megapixels. Multiple other facilities with comparable offered resolutions exist[6] . Using LCD displays instead of projectors largely deals with the mechanical alignment and lens distortion issues that plague projector based setups. A major disadvantage of LCD-based systems is the existence of bezels, the material that surrounds a single monitor and often provides structural support for the internal components. A number of techniques exist targeted at ameliorating the existence of bezels. For example, Ebert et al. [ETO+10] have overlaid

[3] http://www.christiedigital.com/en-us/microtiles/pages/digital-display.aspx
[4] http://www.lcse.umn.edu/research/powerwall/powerwall.html
[5] http://www.tacc.utexas.edu/resources/visualization/
[6] http://calit2.net/research/areas/materials/project?id=34

visuals on top of the bezel surface that are generated by a projector. Another approach is to utilize positional user tracking to provide perspective shift for the visualization, allowing users to peek past the bezels via simple head movements [APPC12]. On the hardware side, there exist several "ultra-narrow bezel" products, some of which have been utilized in LHiRD construction [FNT+13]. Those products however tend to be targeted at the promotional industry, offering large diagonals at relatively low resolutions. Consequently, they reduce the overall pixel density and visual acuity of the LHiRD.

2.2.3 Hybrid Systems

While the original CAVE offered immersion and, for the time, relatively high visual acuity, it still placed users in the "legally blind" territory with regard to the amount of perceived detail (approximately $\frac{20}{140}$ on the Snellen metric). As mentioned earlier, high resolution CAVEs exist (one such CAVE is the Cornea at King Abdulah's University of Science and Technology [7]), but they require prohibitively expensive projector arrangements to achieve the required resolution. Striving for complete immersion (including floor and ceiling surfaces) makes the overall cost of the IVE even greater. A number of research institutions have constructed hybrid visualization systems, that blur the lines between CAVEs and LHiRDs in terms of display technology, arrangement and provided immersion and visual acuity.

The pursuit for a relatively affordable fully immersive IVE with a high degree of visual acuity led to the development of the StarCAVE in 2007 at the California Institute of Telecommunication and Technology [DDS+09]. While the StarCAVE draws inspiration off the original CAVE designs, it augments them a variety of ways. It uses a pentagonal wall arrangement instead of the conventional 3 or 4 wall system. Each wall is divided into 3 segments. The middle segment is perpendicular to the viewer's eye-sight at head level, while the top and bottom segments are slanted inwards. This surface arrangement provides several benefits. It reduces reflections between surfaces since no surface is perfectly parallel to another. The user experiences less oblique views since the typical viewing angle is less off-axis that in a square CAVE. The slanted top segments of the CAVE significantly increase immersion on the vertical axis as the lack of a ceiling surface becomes harder to notice. Finally, the slanting of each wall segment at the top and bottom (and

[7] http://kvl.kaust.edu.sa/Pages/CORNEA.aspx

by extension, the offset projector placement) results in a brighter perceived image for the viewer. It is important to note that the StarCAVE implements stereoscopic rendering using passive light polarization, which had significant implications in the design (a result of the creators striving to maximize the brightness of the resulting visualization). Details pertaining to their approach of this problem can be found in the paper by DeFanti et al. [DDS+09]. Other challenges of the design include the mounting superstructure for supporting the screens and projectors and the need for one of the walls to be movable in order to serve as an entrance to the environment. Overall, the StarCAVE utilizes 34 2 megapixel projectors. It offers an experimental $\frac{20}{40}$ visual acuity metric at 3 meters or a more realistic $\frac{20}{60}$ at 1.5 meters (which would be the average distance from a display for a user).

A more recent hybrid system is the CAVE2, constructed at the University of Illinois Chicago [FNT+13]. The CAVE2 is comprised of 72 LCD displays, arranged in a near-complete cylinder (with a horizontal FoV of approximately 320°, accommodating for an entry point to the system's interior). These displays support stereoscopic 3D via polarized stereo and have been modified to reduce the polarization-related ghosting which is often present at off-axis view angles. With each display having a resolution of 1366×736 pixels, the facility provides an aggregate resolution of 72 megapixels in 2D mode or half that number when using stereoscopic 3D. In fact, the CAVE2 is described as a "hybrid" reality visualization facility, with the ability to intermix 2D and 3D applications as appropriate. The facility does offer a visual acuity of $\frac{20}{20}$, but only at its very center, as is usually the case for most CAVE-like systems.

From surveying the landscape of existing IVEs, it became apparent that there existed no system that offered $\frac{20}{20}$ acuity for the majority of the visualization space while also providing a substantial degree of immersion. Both of these characteristics promote physical navigation, enable collaboration and can potentially improve user performance. This "gap" in IVE design served as a primary motivator for the construction of the Reality Deck, which is outlined later in this thesis.

2.3 Distributed Visualization Software

If one excludes single user systems such as HMDs or relatively relativel small tiled displays, a pattern arises among all IVEs. Due to the necessity to push a large amount of pixels to a significant number of monitors, there exists a requirement for multiple Image Generators (IG) to work together, in parallel in order to create the final visualization. For example, in the case of the StarCAVE, a total of 16 high-end workstations with dual Graphics Processing Units (GPUs) have to work in tandem. Similarly the CAVE2 utilizes 36 IGs, each driving two displays.

This distributed rendering process has to be carefully synchronized among all nodes. It is imperative that at every point in time, the visualized image among all display surfaces of the IVE has the same global timestamp so that the user does not observe any image tearing (or stereo inconsistencies). The rendering of a single frame is a factor of a variety of parameters such as the user's head position, timing information, virtual camera position based on user interaction, the location of tracked interaction devices within the physical space of the IVE, etc. All this information must be robustly distributed to all IGs in a synchronized fashion. Furthermore, modern visualization often demands the rendering of very large data-sets, reaching multiple gigabytes or terabytes in size. Since this data cannot live in main or graphics memory, it must be streamed to and distributed among IGs depending on the current timestamp and the part of the virtual scene that each IG is rendering. Additionally, this distribution of rendering responsibilities between a number of machines should optimally result in performance that is at least equal to that of a feature-equivalent application running on a single workstation.

The above requirements define an outline of features that a distributed rendering framework should expose. Generally, current research and development of such frameworks falls into two broad categories, stream replication frameworks and distributed rendering APIs. The two categories offer a trade-off between ease of application porting (from single-node to a distributed environment) and variety of features and flexibility. Following is an overview of the most popular frameworks that fall under either of these categories. Additionally, we briefly touch on frameworks that allow multiple applications to be executed simultaneously on a clustered visualization system. Chung et al. [CAN14] offer a very thorough overview of the distributed rendering framework landscape.

2.3.1 Graphics API Stream Replication

Early work into distributed rendering frameworks was motivated by the need to drive large tiled displays. The WireGL framework [HBEH00], developed at Stanford University, effectively unified a large number of rendering nodes into a single "framebuffer" (not in the strict sense, as separate framebuffers obviously existed at the IGs but in terms of what was exposed to the application via the API), via an OpenGL-like interface to facilitate development. It also introduced a compositing framework that provided additional flexibility in terms of combining the framebuffers into a final image. Thus different configurations of displays were supported (for example, a multi-display lattice could be driven by a cluster of machines or the same cluster could render a single-display complex visualization, with each machine handling a different part and the compositor aggregating the resulting images). WireGL utilized sort-first processing of the OpenGL streams, forwarding them to the rendering clients. This indiscriminate sort-first process failed to effectively utilize GPU resources on the rendering nodes. As a result, while performance was not hurt in comparison to a single-node application, no performance benefit would often be observed either.

The Chromium Stream Processing Framework described in the paper by Humphreys et al. [HHN+02], has expanded on WireGL in a variety of ways. It introduced the notion of Stream Processing Units (SPUs) that can accept and output one or multiple OpenGL streams, performing transformations to them, with an effect on rendering. Chromium provided much more flexibility than WireGL, allowing multiple OpenGL applications to run in a distributed fashion in the same cluster and permitting rendering effects that WireGL could not perform by modifying the streams. For instance, a Chromium server could modify a default OpenGL stream by replacing instances of the OpenGL calls that define the rendering mode (fill, line or framework). Thus, two Chromium servers could work in tandem to produce a fused wireframe+solid rendering from a single unmodified OpenGL application, by simply intercepting the relevant rendering commands and replacing them appropriately, then forwarding the results to the rendering server. However, Chromium still suffered, to a large extend, from the sort-first bottleneck that plagued WireGL. It also required that the whole OpenGL specification be replicated by the Chromium library.

More recent work at CALIT2, inspired by WireGL and Chromium, resulted in the introduction of the Cross-platform Cluster Graphics Library (CGLX) [DK10]. CGLX takes a step back from the complete reimplementation of OpenGL by only intercepting a few select OpenGL calls (that pertain mostly to modelview and projection matrix control as well as viewport specification) and reimplementing them to compensate for the different physical locations of display surfaces in the cluster. All other calls run natively on the IGs with full 3D acceleration using whatever implementation is provided by the graphics driver. A side-benefit of CGLX is that all nodes effectively run the same application binary.

The benefit of stream-replication-centric libraries is the ability to relatively easily port a desktop application to a distributed rendering system. Often an application recompilation is all that is required (or a binary-compatible version of the OpenGL interface can conceivably be loaded at runtime). However, this approach imposes some restrictions on application design. In particular, most stream-replication APIs intercept and manipulate calls to OpenGL functions that set the projection and modelview matrices for rendering. In early OpenGL versions (up to OpenGL 2.0) these matrices are passed through functions with well defined semantics (by specifying the active OpenGL matrix stack, e.g., GL_MODELVIEW, and using a glLoadMatrix call). However, this functionality is deprecated in more recent versions of OpenGL and, in fact, its usage is strongly discouraged or disallowed on certain implementations. Thus, stream-replication approaches tend to not function with modern OpenGL applications. On the other hand, intrusive distributed rendering libraries can work around this problem, at the expense of additional development work.

2.3.2 Distributed Rendering Libraries

On the opposite side of the distributed rendering spectrum are frameworks that provide a "runtime" for the development of clustered visual applications. These frameworks often define an application model (often with a clear separation between the "state" and "rendering" components) and abstract several OS-level functionalities. For example, a distributed rendering framework often takes care of window and OpenGL context creation, provides a main application loop and aggregates input from a variety of devices

(and even across the nodes of the distributed rendering cluster). Such software development libraries permit the creation of truly "cluster-aware" applications but porting an existing application to them can be time consuming.

A recent framework that follows the above paradigm is Equalizer [EMP09]. Equalizer is a collection of libraries that expose different types of functionality relevant to distributed rendering (such as a complete cluster configuration system, a distributed object system, a networked event system, etc). It also exposes a robust compositing framework allowing for complex configurations of clusters, with some machines arbitrarily performing the rendering and others handling the display of images. Other functionality offered includes different rendering modes (such as sort-last rendering, variable frame rendering, etc), scene-graph decomposition and distribution, etc. However this additional functionality comes at the expense of application development time. While, in principle, the main rendering loop of an application can be maintained relatively intact when porting to Equalizer, taking advantage of more advanced functionality increases development time, as the application state needs to be distributed from the main controller node to the IGs. An older, Java-centric, framework that offers similar features to Equalizer is VRJuggler [BJH+01]. The visualization applications that were created during the development of this dissertation are based on the Equalizer parallel rendering framework.

2.3.3 Multi-application Clustered Rendering Frameworks

In a collaborative scenario, there exists a demand for multiple applications to be displayed simulatenously on a clustered IVE. These applications can potentially be running locally on the cluster or remotely on a computer controlled by a collaborator. The two categories of frameworks described above cannot accommodate for such a usage scenario as they are targetted at the distribution of a single 3D application. The extremely high bandwidth demand of stream replication makes it unsuitable for transfer over the Internet, whereas libraries such as Equalizer impose restrictions that third-party developers might not conform to.

The Scalable Adaptive Graphics Environment library, developed at the Electronic Visualization Laboratory (EVL) by Jeong et al. [BJ05] [JJR+05] has aimed to answer this challenge. SAGE is comprised of three main components. The Freespace Manager that acts as a window manager for the entirety of the IVE and manages different application

windows, the SAGE Application Interface Library, an API that allows an application to transmit its framebuffer and communicate with Freespace and the SAGE Receiver that runs on the IGs of a tiled display and receives generated images and Freespace commands in order to compose the final image on the display. SAGE also includes a networking framework that handles pixel distribution and synchronization among the cluster. An interesting aspect of SAGE is the ability of users to run applications on their personal computer and stream the resulting visuals onto the IVE's displays.

A more recent multi-application framework is OmegaLib, developed again at EVL by Febretti et al. [FNM+14]. OmegaLib is a high-level application development framework for clustered visualization systems. It utilizes Equalizer for low-level window creation and input handling. On top of that foundation, OmegaLib provides an easy to use 3D scenegraph interface that supports various types of visualization toolkits and user interface widgets. Additionally, OmegaLib implements an input system aggregator and a cross-application communication system. Finally, the majority of OmegaLib is exposed over Python bindings, enabling rapid application development. In contrast to SAGE, OmegaLib applications can be executed locally on the IGs, taking advantage of the rendering prowness of the rendering cluster. Framebuffer streaming is also supported. The main drawback of OmegaLib is the need to pre-define the maximum rendering extends of locally-executing applications prior to launch. Additionally, OmegaLib (at least as of the time of writing of this dissertation) does not support compositing and overlapping application windows.

2.4 Gesture-based User Interfaces

In this section, we overview previous work on various areas that are relevant to this dissertation's contributions on gesture-driven hand-centric user interfaces. Specifically, we cover 3D navigation interfaces, hand pose recognition, activity recognition (which drives our gesture detection system) and chirocentric user interfaces in general.

2.4.1 Navigation in Immersive Virtual Environments

To fully control a virtual camera inside a 3D visualization, a navigation scheme must expose 6-DOF. Metaphors for directly controlling all 6-DOF exist [WO90], but devices that implement such metaphors (e.g., the 3DConnexion Space Navigator[8]) are hard to use for non-experts and not ubiquitous. For desktop applications, virtual camera manipulation work has focused on abstracting 6-DOF behind a 2-DOF device such as a mouse. Depending on the navigation task, different methods are available. For the task of orbiting around an object, the prevalent abstraction is that of the ARCBALL [Sho92] (and other virtual trackball techniques [BRP05]) that allows for rotational control around an object as well as zooming. However, this abstraction is not suitable for other navigation tasks that have a changing point of interest such as flying through a scene. Commercial applications usually abstract camera manipulation behind widgets that toggle different modalities like virtual trackballs or simple 1 and 2-DOF schemes. A more advanced widget-driven interface is the iBar [SGS04] that exposes most extrinsic and intrinsic camera parameters through hotspots on a single widget. However, the iBar still "locks" the user into manipulating only a few DOF at a single time. Another interesting widget-based abstraction is the Navidget [HDKG08a], that simultaneously handles point-of-interest (POI) based navigation and camera positioning around the POI in a fluid fashion. It has also been adapted for usage inside Immersive Virtual Environments [HDKG08b]. The Navidget however does not deal with the task of free flight through a 3D scene.

Traditionally navigation in IVEs is handled via some sort of tracked navigation device or prop. World-in-Miniature [SCP95] modalities facilitate navigation but must usually be projected either inside the virtual environment or projected on a surface and manipulated via props or touch (such as in [BH06] [SD10] or, to an extend, the Responsive Workbench [KBF+95]). LaViola et al. [LFKZ01] have introduced the concept of foot gestures for navigation inside an IVE but demands the usage of sensor-equipped "slippers" and the presence of a floor-projected WIM visualization thus being mostly limited for use in CAVE-like environments. Arch-Explore [BSH09] is a navigation paradigm for "walking" exploration of indoor environments that utilizes path bending to map a larger virtual space onto a constrained physical domain. Zielinski et al. [ZMB11] have proposed a low-cost walk-in-place implementation for 6-wall CAVEs. This approach however requires

[8] http://www.3dconnexion.com

that the visualization be explorable in a "walkable" fashion. It also demands a fully immersive environment to surround the user, who also must be extensively motion tracked. Other recent approaches in locomotion interfaces utilize the entirety of the user's body. Marchal et al. [MPL11] introduce the concept of a human-scale joystick, tracked using a mechanical apparatus. Kapri et al. [KRF11] have presented a novel navigation scheme in which the projections of the user's hands and head on the floor of an IVE are used to define a direction of travel as well as modulate the translation speed. However both these interfaces allow for navigation on a 2D plane, making them unsuitable when a free-flight approach is required. The goal of the NuNav3D interface (described in Chapter 5) is to expose multiple degrees of translation and rotation (technically up to 6 DoFs are supported) in a generic fashion, enabling multiple interaction paradigms.

2.4.2 Hand pose recognition

The term "hand pose" can refer to either the orientation and position of a subject's palm in relation to some coordinate system or, the configuration of the hand's finger joints, given some input data. We are interested in the second definition, as it allows us to correlate different types of manipulations to particular poses. Wang et al. [WP09] utilize an image of a textured glove with a known pattern, as input to a nearest neighbor search of a database of synthesized image-pose pairs, achieving real-time estimation. More recently, Hackenberg et al. [HMB11] have proposed a recognition pipeline for palm and finger tracking that utilizes a depth image as input and does not require any prior knowledge. However, their method assumes that the user is facing the depth sensor head-on and does not successfully deal with situations when the user's hand is perpendicular to the sensor's image plane. Work by Keskin et al. [KKKA13] also utilizes depth images for hand pose estimation, along with a large database of synthetic images as training data for Random Decision Forests, similar to the approach of Shotton et al. [SSK+13] for body pose estimation. Compared to most previous work that aims at estimating the actual finger joint configuration for some input data, the hand-pose recognition algorithm utilized in our chirocentric user interface platform (described in Chapter 6) classifies the user's current hand pose under a set of predetermined pose labels that are correlated with specific interactions. By reducing the problem scope in this way, we have developed an algorithm that performs adequately well for an interactive application, given the sparse

marker cloud data provided by the IR tracking system.

2.4.3 Activity recognition

Human activity (or action) recognition is an important field for applications such as surveillance, human-computer interaction, content-based video retrieval, etc. [AR11, Pop10]. Depending on the way of feature extraction, activity recognition techniques are roughly classified into two types: sensor-based (e.g., motion capture data), and vision-based approaches [MMY11].

In sensor-based approaches, sensors are usually placed in the environment or attached to the human body to capture its motion. Action recognition is done using the joint angles, point trajectories or acceleration data from an articulated body skeleton [MBS09, KG04, MRC05, YGVG12, ZDlTH13]. However, they suffer from widely known shortcomings such as: errors due to intrusive setup, problems with occlusion, necessary post- processing, high cost, etc.. Early attempts at vision-based human action recognition used the tracks of a person's body parts as input features [GD95, YB98]. However, most recent research [LMSR08, DRCB05, NWFF08, SLC04] moves from the high-level representation of the human body (e.g., skeleton) to the collection of low-level features (e.g., local features) since full-body tracking from videos is still a challenging problem.

Recently, the proliferation of depth sensors (e.g., Microsoft Kinect) enabled low-cost full-body tracking with relatively robust accuracy [SSK+13]. Such technologies improved the accuracy and reach of skeleton based features for activity recognition. Masood et al. [MEN+11] have used skeleton joints as a feature for real-time activity recognition and actions are detected by logistic regression. However, action categories are chosen from gestures for playing video games, and can easily be discriminated from each other using a single pose. Sung et al. [SPSS11] have used color, depth and skeleton joints as features for classifying daily activities by a hierarchical maximum entropy Markov model (MEMM). However, various classes detected by their system do not exhibit significant motion and their chosen skeleton features are highly dependent on the input data. Recently, Yun et al. [YHC+12] have explored several different types of body-pose features for two-person interaction detection. They demonstrated that geometric relational body-pose features based on distance between all pairs of joint outperformed other features on noisy skeleton

data. Moreover, these features are more generally adaptable to various applications for both single person [MRC05, YGVG12] and multiple people activity recognition [YHC+12]. The gesture-recognition algorithm utilized in our chirocentric user interface contribution (exposed in Chapter 6) is based on this work.

2.4.4 Chirocentric User Interfaces

Unencumbered, hand-driven (or chirocentric) user interfaces are, in some ways, the holy grail of UI research. In the early 1980s, Bolt incorporated gestural input in his *"put that there"* experiment [Bol80]. Later, Baudel and Beaudouin-Lafon [BBL93] have described a prototype system that utilized a wired DataGlove in order to expose a set of gestural commands to the user for controlling a presentation. Importantly, they also outlined one of the earlier models for defining gestural commands in chirocentric interfaces. More recently, Grossman et al. [GWB04] have described a simple chirocentric interface for gestural interactions for a volumetric display, which utilized an optical tracking system. Hackenberg et al. [HMB11], in addition to describing a finger tracking pipeline from depth sensor data, also utilized their system as a backbone for a direct-manipulative 3D user interface. However, their approach is not really suitable to large immersive virtual environments, due to the small range of commercial depth sensors and the assumption that the sensor is looking at the user head-on. In the commercial realm, Oblong Industries[9] has developed a chirocentric UI termed *"g-speak"*, which also utilizes a high end IR tracking system. However, to our knowledge there exists no published work detailing the inner workings of the system, such as the various recognition algorithms. Based on the g-speak system, Zigelbaum et al. [ZBL+10] have implemented a user interface for the exploration of a data set of animated videos. Banerjee et al. [BBGV11] designed the *WaveForm* interface, aimed at Video Jockey-ing. Their system is similar to ours, in that it uses an IR tracking system in conjunction with passively tracked marker gloves, however they offer little exposition in terms of the algorithm used for hand pose detection. The goal of our chirocentric user interface contribution is to describe a practical end-to-end system that provides this sort of functionality and demonstrate its use within a large IVE.

[9] www.oblong.com

2.5 Gigapixel Visualization

Gigapixel resolution image data is becoming increasingly commonplace as related sensor technologies (e.g. gigapixel cameras, telescopes, microscopes) improve. One of the main contributions of this dissertation is an "acuity-driven gigapixel visualization" framework (described in Chapter 7), enabling performant exploration of such visuals via physical navigation within the confines of a LHiRD. In this section, we review previous work relevant to this contribution, including techniques for rendering gigapixel imagery in an out-of-core fashion, various systems for improving the exploration of these vast data sets and technologies that levelerage the physiology of the human visual system to improve the delivery of such content to the display.

2.5.1 Rendering and Exploring Gigapixel Images

Gigapixel resolution images present unique challenges due to their size. Even though modern PCs provide a large amount of memory to applications (both on the CPU and GPU side), data-set resolutions often exceed available memory capacity and demand that out-of-core schemes be utilized. In fact, most gigapixel content is generally broken up into constant resolution tiles for more performant management and indexing prior to visualization.

Fitting very large images to a limited amount of graphics memory has been a topic of constant interest in the rendering field. Effectively it amounts to mapping a large $M \times N$ texture into a much smaller $M_{Phys} \times N_{Phys}$ GPU-resident *physical* texture. Early work by Tanner et al. at SGI introduced the *Clipmap* concept [TMJ98], the notion of only using the visible part of a mipmap [Wil83] pyramid while rendering. More recently and motivated by the increasing texture requirements of video games, the concept of *virtual textures* has been pursued in the rendering industry [Mit08]. In virtual texture implementations, a determination of the visible parts of the original texture is made (usually in a preliminary rendering pass at lower resolution, which is read from the GPU to the host and processed on the CPU) and the necessary tiles are paged into the physical texture in the background (without blocking the rendering process in order to allow for smooth interaction). At this stage, the pipeline also makes a determination of the appropriate LoD for each pixel (we cover this aspect in more detail in the next section). Graphics hardware is now shipping

with support for *Sparse Virtual Textures*[10] , effectively implementing virtual texturing at the GPU/driver level.

The visualization community has also been investigating the rendering and manipulation of gigapixel resolution data. Powell et al. [PRS10] have presented a distributed framework for image operations on high resolution data from planetary imaging with on-demand paging of tiles. The Giga-stack framework, by Ponto et al. [PDK10], suggests a rendering scheme in which tile state is determined using a "texture table" and tiles are loaded to the GPU memory on demand in a fashion similar to Virtual Texturing. This technique has been extended include large image collections [YDK11]. Kopf et al. [KUDC07] have presented a system for capturing panoramic gigapixel images using an automated panning mount, preprocessing and rendering with a custom projection and tone mapping scheme to improve the visualization quality. Summa et al. [SSJ+11] have introduced an efficient framework for interactive editing of gigapixel imagery. By using a progressive Poisson solver and intelligent data accessing, it allows for interactively applying complex operators on data spanning hundreds of gigapixels.

Exploring gigapixel resolution images adds an additional layer of complexity on top of the existing challenges of 2D visualization. When visualizing on a limited resolution display, salient features of the data may not even create a screen-space impact. Ip and Varshney [IV11] have presented a framework for saliency assisted navigation of gigapixel images that combines multi-scale computer vision techniques with user feedback to detect and visualize important regions in the data. Luan et al. [LDK+08] have presented a system for inserting and intelligently rendering annotations to large images. Appert et al. [ACP10] have tackled the issue of *quantization* in F+C lens manipulation on high resolution data, meaning the disparity between visual precision and user interface accuracy in the magnification region. Han and Hoppe [HH10] have introduced a system for the generation of a mipmap pyramid that ensures smooth transitions between coarse and fine imagery with different visual characteristics.

Our gigapixel rendering framework utilizes virtual texturing for out-of-core rendering. This design decision is motivated by the fact that this method makes no assumptions about the underlying geometry as the visibility and LoD determination is made through

[10] https://www.opengl.org/registry/specs/ARB/sparse_texture.txt

a rasterization-based preprocessing pass (rather than analytically). As a result, manipulations to the gigapixel image that translate to operations on the geometry can be transparently handled with correct texturing. Out-of-core rendering (and more specifically virtual texturing) is an active research topic but to our knowledge there is no significant work on its acuity-driven integration within a large visualization environment that affords significant physical navigation.

2.5.2 Focus and Context Techniques

Traditionally, large 2D datasets are explored using a series of panning and zooming motions. However, when zooming in to examine a ROI, the context (which is very often important in the decision making process) is lost. Various techniques can be combined with traditional exploration controls to enhance the user's ability to make sense of data. A classical approach, the overview-detail model [PCS95] would display the context region in a separate viewport of the visualization, however this has the side effect of disconnecting the focal region from the overall context. F+C techniques aim to merge the focus and context regions in the same visualization by providing in-place magnification of the ROI while applying a transformation/deformation to the context (that may or may not preserve some of its properties, e.g. shape, size, etc.)

F+C techniques have been investigated for a large number of visualization modalities such as tree structures [MGT+03], graphs [GKN05], networks [TS99] and volume visualization [CBP08]. Here, we focus on techniques that are applicable to 2D image datasets and in particular F+C techniques for *in-situ* magnification that require the deformation of the transition area to maintain the size of the context. Other F+C techniques, such as the DragMag [WL95] will present the ROI in an offset inset and show its connectivity with the context region but will not actually illustrate the transition between focus and context.

Spence and Apperley [SA82] have proposed the concept of a bifocal representation, effectively viewing the entirety of a data set while part of it is presented to the user in full detail. This concept sparked vast amounts of research in F+C such as the generalized fish-eye view concept [Fur86]. Early work by Biet et al. [BSP+93] introduced a see-through interface for interacting with different types of lenses (not limited to magnification). Pietriga et al. [PA08] have introduced the *Sigma Lens* concept that incorporates transitions in the

time domain and formalizes the F+C lens as a composition of several different functions. Additionally, they proposed a framework for representation-independent magnification based on the Sigma Lens [PBA10]. In their work, a lens is implemented as a two-pass rendering method (with the first pass rendering the context, while the second pass, at a higher resolution, dealing with the Region of Interest (ROI) and the transition region). In the second pass, the resulting pixels are redirected based on a displacement and compositing function. The pixel redirection component of a Sigma-lens can be implemented in a fragment shader. The *focus* rendering pass of a sigma lens assumes that the ROI and transition region information is readily available for rendering. Since most gigapixel resolution schemes work in an out-of-core fashion on the GPU (as we detail later on in this section) this may not always be the case, thus making the Sigma Lens somewhat unsuitable for our target application.

Another lens formulation is the *Elastic Presentation Framework* (EPF) of Carpendale et al. [CM01]. EPF places the information to be visualized on a 2D plane and renders it using a 3D perspective camera model, defining different presentation variations by displacing the points on the plane. When very large magnification factors are used, occlusion becomes a problem as the enlarged focus area blocks visibility towards the distorted context. Further work by the same group [CLP04] deals with this problem. EPF-based F+C lenses can be implemented simply on the GPU as vertex shaders displacing texture mapped geometry based on an F+C lens function. The final image quality of the F+C lens application is then directly dependent on the tessellation of this geometry and as a result, the method is not directly applicable to gigapixel images. However, the simplicity of EPF and its direct application to the GPU led us to adopt it as the F+C formulation for our acuity-driven gigapixel visualization framework.

2.5.3 Perceptually Optimized Level-of-Detail

The spatial resolution of the human retina decreases as distance from the retina increases. This property has been been utilized by a number of researchers in order to improve rendering performance. For example, Guenter et al. [GFD+12] have achieved a 5-fold acceleration in rendering speed by combining 3 image layers rendered at decreasing spatial resolutions with an efficient antialiasing algorithm. Previous approaches focused on specific rendering modalities. Levoy and Whitaker [LW90] have introduced gaze-directed volume

rendering. Most of the work on foveal rendering utilizes eye-tracking and is targeted at single displays and/or head-mounted systems. Watson et al. [WWH97] have combined head-tracking and eye-tracking to investigate the effect of reduced peripheral acuity in large displays. Some large display systems, such as that by Minakawa et al. [MMYT01] are based on this principle. However, we are not aware of any work that drives LoD based on the formulation of visual acuity and its variation during the visual exploration process within an immersive system.

2.6 Experimental Evaluation in Visualization, VR and HCI

2.6.1 Experimental Design

Even before Lind's work on the treatment of scurvy [Lin72], controlled experimentation has been an essential tool of the scientific process. Critical to the value of a controlled experiment is its design, the set of parameters that define how it will be conducted. Experimental design has been covered in countless books and scientific articles, from Fisher's fundamental work [Fis92] to modern textbooks [Mon08][CW11]. Such designs involve several important decisions, like the selection of meaningful factors at which to perform measurements, the assignment strategy of participants to factor levels, the types of measurements that are important to a productive statistical analysis, etc. In order to define the semantics that drive the visual analytics framework that is described in Chapter 9 of this dissertation, we are interested in the aspects of an experiment's design that can be used to characterize the resulting information. Specifically, we are interested in the following:

- Independent Variables - The aspects of the experiment that are controlled and explicitly varied by the experimenters. Independent variables have *levels* or predetermined values. For example, in an HCI experiment, the *interaction device* could be an independent variable, with *mouse*, *touch* and *gamepad* being its levels.
- Dependent Variables - Values or measurements that result or are *dependent* upon the experiment. In the case of an HCI experiment, dependent variables could include

accuracy, *speed*, etc.

- Participants - A sample of the population being evaluated that takes part in the experiment. In an HCI experiment conducted at a research university, the participants may include students taking a class for credit, laypeople who respond to a recruitment flyer, etc. Participants often are characterized based on demographical information (such as age or sex), various aptitude questionnaires (e.g., spatial cognition [EFH79], simulator sickness [KLBL93]), etc.

2.6.2 User Studies

A number of user studies have been conducted that evaluate different aspects of user performance when examining data on high resolution displays of various sizes and pixel counts. Tan et al. [TGSP06] have showed that the larger field of view offered by bigger displays has a tangible effect on user performance. Ball and North [BN05] have evaluated different navigation tasks when using a traditional pan and zoom interface on three display grid configurations (single monitor, 2×2 and 3×3). Later work by these authors [BN07] evaluated user performance in core visualization tasks (search, navigation, pattern finding and insight gathering) on a 100 megapixel data set, while scaling up the size to the display. The maximum display resolution of that experiment was 32 megapixels and in that condition, the display extended to $2.74m$ by $1.06m$ in size. Through this work, Ball and North demonstrated very tangible performance increases for large displays (up to $10\times$ when the information space remains constant). Additionally, they measured a substantial user preference towards physical navigation in certain tasks. In a later experiment [BN08], the same authors quantified the effects of peripheral vision and physical navigation on a similar GIS-centric visualization, performed on a 100 megapixel wall with an area of $4.4m \times 1.7m$. They demonstrated that users are more inclined to physically navigate and also perform better when using LHiRDs. Shupp et al. [SADK+09] have evaluated user performance in a study with display resolution and display curvature as the independent variables. They discovered that a curved display decreases performance timings and allows for less strenuous physical navigation for certain tasks. Still, the largest display condition used in their study was approximately 32 megapixels and spanned roughly $2.74m$ by $1.06m$ when arranged planarly (this appears to be the same experimental setup as in [BN07]).

The above studies include tasks designed around geospatial visualization scenario (e.g., prices of houses for sale within an area). Other studies evaluate the suitability of different visualization designs for display on LHiRDs. Yost and North [YN06] have evaluated the scalability of space-centric and attribute-centric visualizations by varying the information space along with the display shape, size and resolution.They demonstrated that increasing the information space 20 times results in a 3-fold increase to task completion. A follow-up study by Yost, Haciahmetoglu and North [YHN07] evaluates the scalability of these visualizations past the visual acuity threshold. Here, the authors showed that a display that exceeds visual acuity does not offer benefits for detail-centric tasks but users did realize some performance gains for certain overview tasks.

Endert et al. [EALN11] have investigated the impact of viewing distance from the LHiRD on aggregating visual information. Later work by Endert and his collaborators [EBZ+12] has given insights on the design of high-resolution display workspaces. Ruddle et al. [RFT+13] evaluated LHiRDs as a platform for exploring genomics visualizations. In parallel, Ruddle et al. [RTR+15] have also investigated the effect of resolution on the users' ability to locate targets in large imagery. Recently, Liu et al. [LCBL+14] have reported on the scalability of an abstract classification task from a desktop to a wall sized display.

While the above studies have demonstrated the benefits of LHiRDs in a number of scenarios, the apparatuses have been limited in terms of resolution and size. To our knowledge, the largest facility in terms of resolution to have been used in an evaluation focused on user performance, is the apparatus of Ball et al. [BN08] while the overall-largest display (131 megapixels) was used by Bezerianos et al. [BI12] in their perception-centric experiment. Consequently, the scalability of such facilities to extreme resolutions and sizes has not been quantified. The user study outlined in Chapter 8 provides guidance on this question. Specifically, we evaluate user performance across several display size conditions, starting at 100 megapixels (with an area of $4.41m \times 1.47m$) and scaling up to 1 gigapixel (planarly spanning $30.24m$ by $2.22m$).

Additionally we can identify several unifying characteristics in the experimental design of the above works:

- Small numbers of independent (e.g. interaction modality, visualization design, task, etc) and dependent variables (elapsed time, accuracy, etc.).

- Pre- and post- experiment questionnaires that are administered per-subject.
- Supplementary per-trial data, such as motion tracking which is leveraged to either produce additional dependent variables (e.g. travel distance) or as material for discussion.

These observations inform the design of the ontology that drives the visual analytics framework that is described later in this thesis.

2.6.3 Visual Analytics for Experiments

Visual analytics tools have been proposed for several domain-specific applications, some of which involve or are centered around hypothesis-testing via empirical experiments. For example, recent work by Palmas et al. [PBO+14] has described the *MovExp* tool which fused biomechanical simulations with performance data from interaction tasks in order to assist HCI developers. Their tool is similar to *VEEVVIE*, the visual analytics system described in Chapter 9, in that it allows the linked visualization of generic performance data together with interface-specific visualizations and domain-relevant views. However, it is does not seem to expose data related to hypothesis testing (such as statistically significant effects). Papers by Andrienko et al. [AABW12] and Kurzhals and Weiskopf [KH14] have introduced several techniques for the exploration of data stemming from eye-tracking studies. Tools proposed by Steenwijk et al. [SMvB+10] and, more recently, Klemm et al. [KOJL+14] have focused on the exploration of cohort studies in the medical domain. Their motivation is similar to ours (e.g., hypothesis validation and generation). Overall, while visualization is often a part of several experimental evaluations, the tools that enable such types of analyses do not readily exist.

Chapter 3

The Reality Deck - Immersive Gigapixel Display

3.1 Introduction

A vast number of data sources, such as supercomputers and sensors, have become "fire hoses", generating information at a far greater rate than it can be digested. For example, the AWARE-2 camera system can capture a 1.47 gigapixel photograph [MST+11]. In cosmology, the Large Synoptic Survey Telescope[1] , will feature a 3.2 gigapixel sensor and capture approximately 200,000 images annually.

One approach towards visualization is to fit vast quantities of data on a single display. Various summarization, abstraction and focus + context techniques aim at accomplishing that, while providing users with the overall patterns and structure of the data.

Maximizing available screen real estate demonstrably affects the visualization process [NBC06]. Tiled display arrays (or powerwalls) are a realization of this concept, offering a large, high-resolution, collaborative workspace. For applications that benefit from *physical immersion* (e.g., surrounding the user with visuals), CAVEs [CNSD+92] are a suitable visualization setting, potentially offering a fully immersive field of view (FoV) and stereoscopic 3D. Immersive Virtual Environments (IVEs), however, present a fundamental dichotomy. When large-scale datasets are being visualized, total screen real estate

[1] www.lsst.org

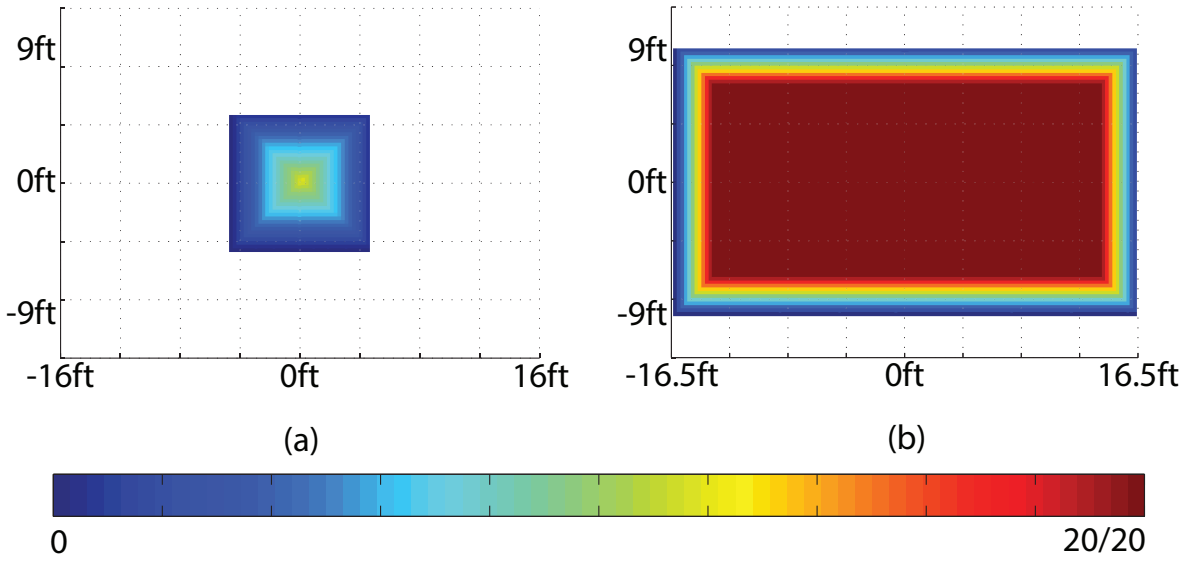


Figure 1: Visual acuity heatmaps for the CORNEA and the Reality Deck. The CORNEA (a) facility offers $\frac{20}{34}$ visual acuity in a $10' \times 10'$ workspace. However, the maximum visual quality is achievable only at the very center of the facility. In contrast, the Reality Deck (b) offers a large workspace of $33' \times 19'$. Given its monitor configuration, $\frac{20}{20}$ visual acuity can be achieved approximately $31''$ from the displays. This translates to a total workspace area with $\frac{20}{20}$ visual acuity of about $384'^2$.

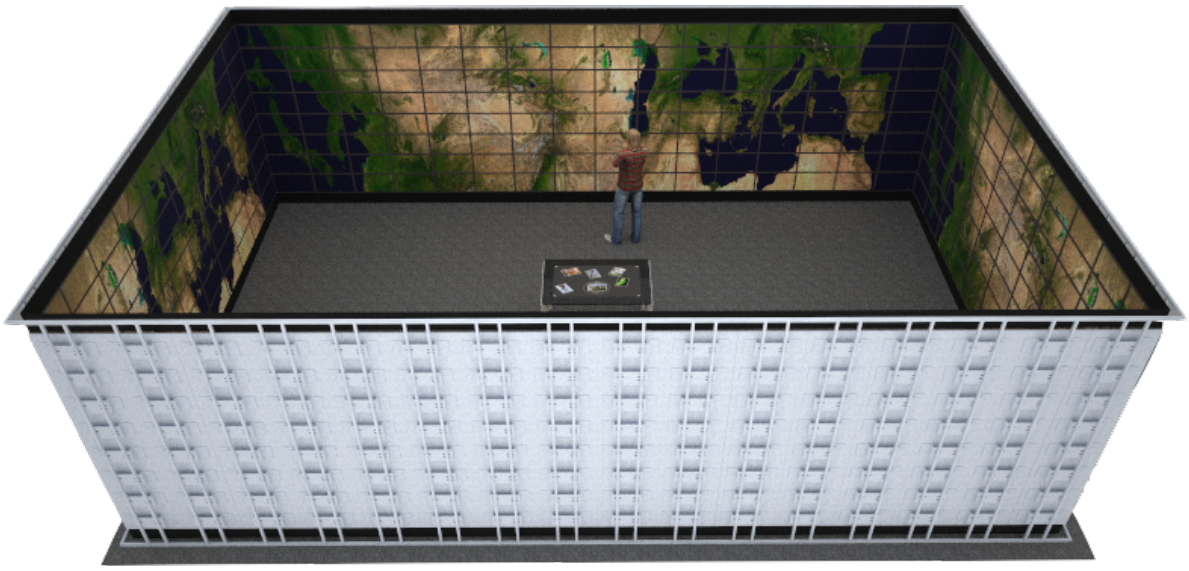


Figure 2: Synthetic view of the Reality Deck with the door closed. This render shows the Reality Deck facility. The human figure is rendered to scale in relation to the size of the system. Also visible is the position of a Microsoft Pixelsense touch table which is used for controlling the system.

and resolution must be maximized, making powerwalls more appropriate. However, powerwalls lack immersion and their resolution can be limited by physical constraints (e.g., a $100'$ planar powerwall is unwieldy to construct and utilize). Conversely, CAVEs offer immersion but their maximum resolution is roughly 100 megapixels per eye for state-of-the-art systems. As a result, the visual acuity offered by CAVEs tends to be limited and maximized at a sweet spot in the center of the facility (see Fig. 1) limiting physical navigation. Furthermore, their total workspace can be somewhat constrained (approximately $9' \times 9'$ for typical setups).

Motivated by the current landscape, we designed and built the Reality Deck [PPKM15, PK13a, PPK12, KMS+11], the world's first gigapixel resolution display in a fully enclosed setting. A synthetic, to-scale, rendering of the facility is depicted in Fig. 2. The Reality Deck offers more than 1.5 gigapixels worth of resolution, a 360° horizontal FoV and a workspace of approximately $33' \times 19' \times 11'$, allowing multiple users to naturally explore data at different scales by approaching or distancing themselves from the displays while maintaining the panoramic context. This display real estate can be abstracted in planar or immersive configurations. In this chapter, we examine the motivation and design process

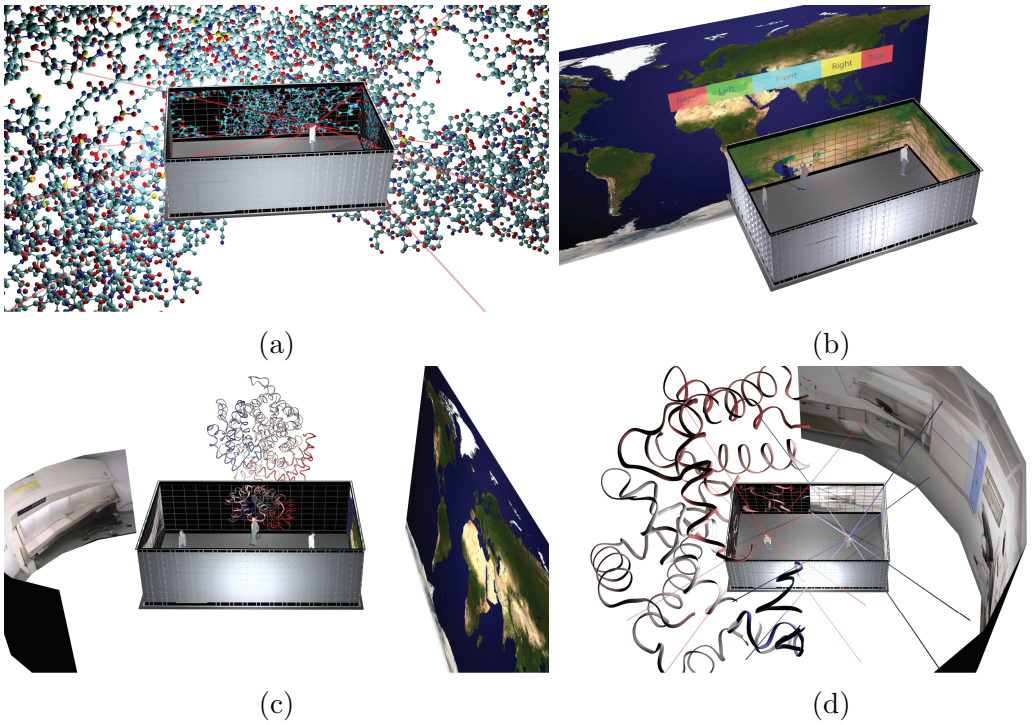


Figure 3: Four usage scenarios for the Reality Deck. (a) Four-wall gigapixel CAVE, visualizing a large proteomics data set. (b) Single, "virtually planar", gigapixel display, offering an immersive view into 2D GIS data. (c) Four independent high resolution tiled displays, providing independent views into a panoramic gigapixel photograph, a 3D data set and a GIS application (the data corresponding to the rear wall is not shown). (d) Dual-CAVE configuration, each with about 0.76 gigapixels worth of resolution. A gigapixel panorama and a 3D proteomics model are immerisvely and independently visualized.

behind the Reality Deck. We expose findings and benchmarks that arose during the engineering process. Finally, we present a number of novel techniques that evolved while utilizing a gigapixel resolution display.

3.2 "Immersifying" a Tiled Display Wall

As a higher-level goal, we felt that the Reality Deck should provide the high pixel density of tiled displays but also the full FoV of CAVEs. As a next generation facility, it should offer a significantly leap in aggregate resolution (with the gigapixel milestone being an obvious choice). Additionally, $\frac{20}{20}$ visual acuity should be available for the majority of a large visualization space, to promote physical navigation. Two further constraints imposed that

the facility fit within the available $40'$ by $30'$ lab space and the budget of $\$1,000,000$.

The Reality Deck defines an enclosed space, surrounded by high pixel density displays. The arrangement of the displays presented an open design problem. After considering different placements of display surfaces, we opted for a rectangular arrangement with four walls, spanning approximately $33' \times 19'$. This layout enables interesting usage scenarios, depending on the nature of the data and collaborative situation. It is different to most CAVE systems that use a cube-like arrangement of displays. Our rectangular layout allows for more flexibility in operation and also maximizes usage of available lab space.

The four walls of the Reality Deck serve as configurable viewports into visualizations. The straightforward mapping allows the four walls of the system to serve as a 4-viewport configuration into the virtual world, akin to a CAVE. Alternatively, the display can be interpreted as a single continuous planar viewport. This configuration is useful for large scale two-dimensional data (e.g., GIS, parallel coordinates). A visual discontinuity exists at the point where the two extremes of the logical frustum meet on the physical display surface (typically in the middle of the rear wall). The *Infinite Canvas* technology [PPK13] can be used to naturally ameliorate this shortcoming - this "continuous viewport" display mapping addresses scalability problems of traditional, planar, powerwalls.

In multi-user scenarios, the facility display space can be subdivided in various ways. First, it can be split into 4 planar tiled-displays, one per wall. Here, each of the two long walls offers approximately 471 megapixels of resolution while each narrow side wall has 295 megapixels. For additional immersion, we can create two CAVE-like systems, with three walls each, operating independently. Each one of these CAVEs offers roughly 0.76 gigapixels worth of resolution, depending on the "border" space that separates the viewports of the two CAVEs. This pixel count is several times larger than that offered by state of the art CAVE systems, at the expense of bezels and anaglyph-only stereo (due to the selected panel technology). These four viewport configurations are illustrated in Fig. 3.

3.3 Building an Immersive Gigapixel Display

3.3.1 Display Selection and Customization

Arguably the most critical component of a visualization environment is the display subsystem. CAVEs are usually based on projectors while tiled display arrays are constructed using both projectors and LCD monitors.

The main benefit of projectors is that they can create a nearly seamless image. On the other hand, projectors require regular maintenance. Based on our experience with our 5-sided Immersive Cabin [QZP+08], maintenance must always be followed by a manual recalibration of the system, which can be time consuming for the 10 projector CAVE-like Immersive Cabin and unmanageable for a system that utilizes hundreds of projectors. Additionally, projectors produce significant amounts of heat and noise and the space requirements are affected by the need to accommodate the throw distance (as much as $1.5'$ for short-throw lenses). Finally, projectors are generally much more expensive than LCDs to acquire and maintain. Due to these drawbacks, projectors were eliminated early in our design process.

We then considered different types of LCD monitors based on the following criteria:

Resolution target: Based on the available space and super-gigapixel resolution target, the monitors should provide approximately 100 PPI.

Bezel size: Ideally smaller than 5mm, however the size should not exceed 8mm for a 23" display and 15mm for a 30" display. These metrics were based on bezel dimensions of commercially available monitors with potential structural modifications.

Display size: Larger monitors are preferable as long as they can deliver the required pixel density.

Image quality: The monitors should use high quality panels with good contrast, backlight uniformity and viewing angles.

Stereo support: Stereo is a very desirable feature, but not at the cost of image quality or significantly reduced pixel density.

Based on these characteristics, we evaluated a number of displays with panel sizes from 23"

to 60", IPS, PVA and PLS panel technologies and various bezel sizes. We also considered secondary factors, such as power consumption and weight, which affect the requirements for the mounting, power and cooling infrastructure. The different tiled display designs were simulated in our Immersive Cabin in order to analyze the perceptual effects of the bezels on different visualization tasks, including medical and architectural visualization. The results of an informal user study were then used as a contributing factor in the display selection process.

We considered several offerings from a variety of vendors, including ultra-narrow bezel LCDs and monitors with stereoscopic 3D support. At the time of construction no commercially available display satisfied all five of these criteria, however the Samsung S27A850D provided a good balance. It is a professional 27" PLS panel with 2560×1440 resolution and excellent contrast, color saturation and viewing angles. In contrast to CCFL monitors, the S27A850D uses LED backlighting, which significantly reduces the weight and power requirements (46W for the S27A850D versus 134W for a Dell U2711). Finally, while the original bezel is relatively large, the monitors were easily modified with a custom mount that reduces the bezel to 14mm.

Given the available physical space, we arranged the monitors in four orthogonal surfaces. The front and back walls are 16 displays wide while the left and right span 10 displays. All four walls are 8 displays tall for a total of 416 tiled monitors.

The mass-produced nature of commercial monitors entails certain variation in image quality, even for products from the same batch. We evaluated every monitor before modification, looking primarily at image uniformity when displaying a full white and a full black signal, as well as identifying issues with color reproduction. Three photographs were taken of each monitor from a fixed camera position and with a standardized set of camera and display settings. Approximately 30% of tested monitors had to be replaced due to inconsistencies in backlight uniformity. After testing the second batch, we selected the best 416 displays, as well as a set of spares, for modification and use in the Reality Deck.

Using lightweight monitors allowed us to design custom mounting brackets and a simple aluminum frame so that individual monitors can be aligned with sub-millimeter accuracy (confirmed via laser leveling) and can also be replaced by a single person. The plastic cover of the S27A850D houses the circuit board, user controls and power supply. These

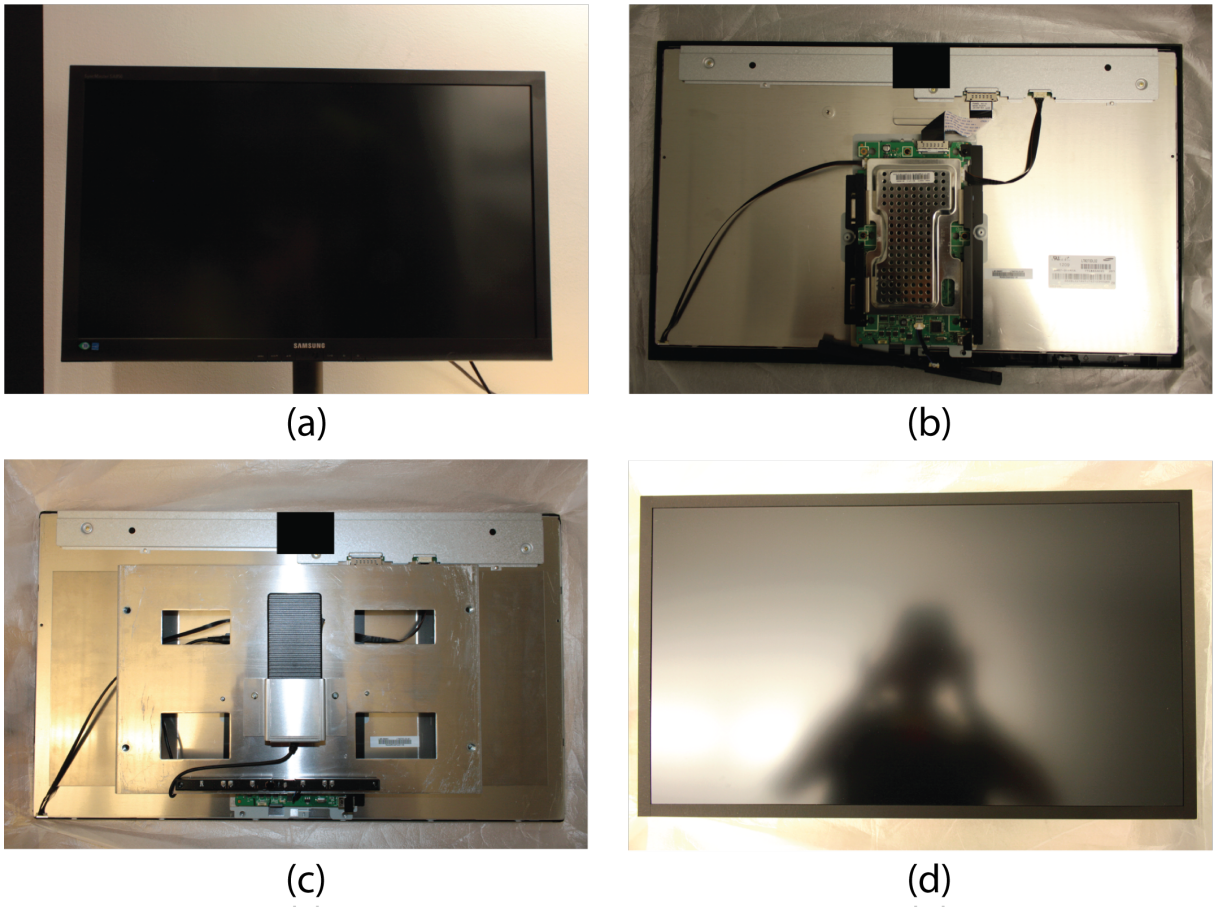


Figure 4: Reality Deck Monitor Customization. (a) The Samsung S27A850D fully assembled. (b): The monitor with its backing plate removed and the electronics exposed. (c) The monitor with our custom mounting bracket installed and the PCB/PSU mounted. (d) The front of the monitor post-customization.

(a) Door Open (b) Door Closed

Figure 5: The Reality Deck door assembly. A section of 3×5 displays are attached to the Reality Deck door, shown open in (a) and closed in (b). They are mounted on an aluminum subframe that attaches with spring-loaded hinges to the main frame and rests on a pivot wheel base. The mechanism can be remotely operated. When closed, the door blends into the rest of the structure creating a continuous display surface (in this case, showing a GIS visualization of New York City).

components have been moved to the rear bracket, resulting in a uniform black frame around the display with no visual distractions. The modification process is illustrated in Fig. 4.

The door to the facility is a section of the frame that is mounted on a hinge and holds a 3×5 grid of monitors. It is power operated but can also be opened manually in case of an emergency. When closed, the door rests completely flush with the rest of the wall and it is visually indistinguishable from the other displays (Fig. 5). The displays are offset from the floor of the facility by approximately 7" to allow the installation of tracking cameras and sound speakers.

The facility provides a visualization space of approximately $33' \times 19' \times 11'$ ($W \times D \times H$). Fig. 1 shows a heatmap of the visual acuity within our facility, illustrating the 384 sq. ft. space in which the system achieves $\frac{20}{20}$ or better visual acuity.

3.3.2 Visualization Cluster and Peripherals

The vast number of displays presented a challenge when designing a cost-efficient high-performance visualization cluster. We evaluated a number of different configurations, at various GPU and display per node densities. Our final setup consists of 18 ExxactCorp nodes, with dual hexcore Intel Xeon E5645 CPUs. Each node contains four AMD FirePro V9800 GPUs. The head node is a similarly configured machine with a single GPU. The majority of the cluster nodes drive 24 displays, six per GPU, in a 3×2 monitor grid. The displays of two render nodes in the front-right and back-left corners of our facility operate in groups of 1×4 to ensure that no display group "straddles" the corners of the facility, which would necessitate two rendering passes when the facility operates in "immersive" mode. Each display group is abstracted as a single framebuffer using AMD Eyefinity functionality, simplifying software development and improving performance. The cluster is located in a machine room adjacent to the facility and connects to the displays using Gefen DisplayPort fiber optic extenders. All nodes are interconnected via Ethernet and 40 Gbps InfiniBand networks. A rough total of seven miles of cables were utilized in the facility.

The Reality Deck is also equipped with a 24-camera tracking system from OptiTrack, based on the S250e IR camera. A number of research techniques, described later in the paper, utilize this tracking system for both user interaction and performance optimization. Additionally, we have deployed a 24.4 surround sound system with Genelec 6010A speakers and JBL LSR4312SP subwoofers. The total material cost of the facility, including spare monitors and a hot-swappable spare visualization node was less than $\$1,000,000$.

Chapter 4

Visualization Software and Applications

4.1 Visualization Software Architecture

For the purposes of this dissertation, we developed a multi-platform distributed visualization application framework that enables the display of a variety of data types within the Reality Deck. Our framework is built upon several open-source libraries. Window creation, basic input handling, data distribution and application lifetime are managed by the Equalizer library [EMP09]. Equalizer allows our software to scale from a traditional, single computer setup to a full clustered rendering environment simply by changing human-readable configuration file. An additional benefit of Equalizer is the modularity of its various components. For example, the distributed object fabric ("Collage") can be utilized on its own, separate of the clustered rendering framework. This permits us to develop applications that communicate with the clustered visualization but do not operate as part of the main rendering loop. For instance, we have developed a complementary "Control Application" that allows users to interface with the visualization from external devices (such as portable x86 tables). Since this software does not utilize Equalizer's distributed rendering functionalities, it can run on low power devices and can be more easily ported across platforms. In fact, our visualization software compiles and runs on 32- and 64-bit platforms and supports several versions of Microsoft Windows and Apple Mac OS X.

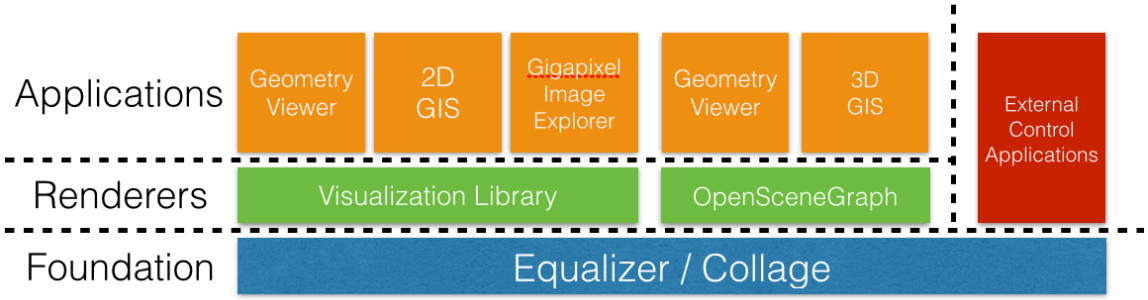


Figure 6: Diagram of the high-level architecture of our visualization software, with renderer modules leveraging the Equalizer framework for distributed rendering. On top of the renderer modules, we develop several visualization applications.

Figure 7: In this view, our Reality Deck software pipeline is rendering the Crytek Sponza Atrium Scene[2]via the Visualization Library renderer module.

Equalizer provides the basic application model within which graphical visualization applications can be developed. In our software, we have developed several *renderer* modules, that permit the visualization of different types of data via two rendering frameworks. Our software includes support for two OpenGL-based rendering libraries. The first rendering module utilizes the Visualization Library[1] (shortened to VL throughout this dissertation) to render numerous common geometric file formats. Additionally it provides support for a binary VL-specific file format which enables faster loading. A sample view from the VL rendering module can be seen in Fig. 7.

The second core rendering module is based on OpenSceneGraph[3] . OpenSceneGraph (OSG) is an extremely popular, open-source rendering library with support for OpenGL rendering backends of different versions. It supports numerous popular file formats and

[1] http://www.visualizationlibrary.org
[2] Available at www.crytek.com/cryengine/cryengine3/downloads
[3] http://www.openscenegraph.org

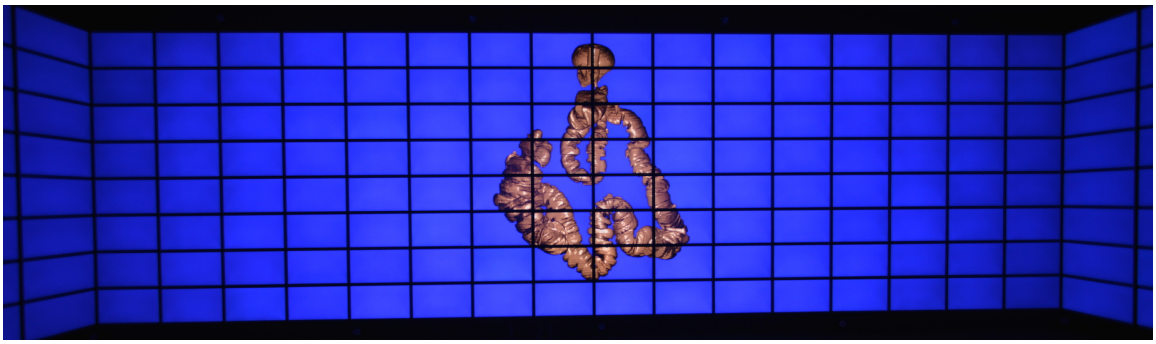


Figure 8: A geometric dataset of the human colon is being rendered within the Reality Deck using the OpenSceneGraph-based renderer of our visualization application.

also offers a robust and easily extensible asset loading system. Additionally, OSG is geared towards performance and scalability, offering multi-threaded scenegraph traversal. While at first glance the results of the OSG-based rendering module are not different from the VL-centric version, the additional flexibility of OSG was very useful in the development of the GIS-centric applications described later in this dissertation. An example visualization generated with the OSG rendering module can be seen in Fig. 8.

This visualization software provides support for several input modalities. Traditional mouse and keyboard controls are exposed by Equalizer. Gamepad and joystick input is handled by platform-specific libraries. Under Microsoft Windows, we provide support for OptiTrack tracking systems from NaturalPoint via the NatNet library[4] . Natural user interface via the Leap Motion controller[5] is supported across all platforms. All inputs are aggregated within the main Equalizer application frame loop and then forwarded to application-specifc interaction handlers. This approach decouples interaction handling from general input event processing (which is important since a particular controller input can result in substantially different interaction calculations and state manipulations for different applications). Interaction handlers are generally tied to a relevant renderer module and expose varying levels of interactivity depending on the application. For instance, the VL-specific interaction handler is relatively simple, translating inputs into manipulations of the virtual camera's position and orientation. Conversely, the OSG-specific interaction handler actually maintains a copy of the OSG scenegraph and runs a simplified, "head-less" rendering loop internally, allowing for more complicated interactivity. The supported interactions are covered in more detail in the next section.

[4] http://www.optitrack.com/products/natnet-sdk/
[5] https://www.leapmotion.com

Following the input processing, each interaction handler manipulates the master copy of a distributed "FrameData" object. This object is mapped on application startup at the render nodes. While rendering, each node utilizes the relevant information from this distributed object (e.g., camera position and orientation, overloaded camera modelview matrix, shader uniform variables, etc) in order for a consistent visualization to be generated across the cluster. As mentioned earlier, external applications can also map and affect this object, enabling some visualization control from computers that are not a part of the Equalizer application. The overall architecture of our distributed visualization software can be seen in Fig. 6.

4.2 Visualization Applications

During the development of the contributions outlines in this dissertation, we focused on two main application areas that are a natural fit for a gigapixel resolution display like the Reality Deck. The first application is the display and exploration of gigapixel resolution imagery which presents unique challenges due to the size of the underlying data. The second application area is the exploration of large geospatial data sets. The focus on this second application area stems from two motivations. GIS visualizations can scale in terms of complexity and simple versions can be easily understood by layment. Consequently GIS applications have been used frequently in various user studies on LHiRDs, as we discussed in Chapter 2. Thus, the development of a flexible GIS scenario renderer was imperative for some of the contributions outlined within this text. Additionally, GIS applications are extremely relevant in several scientific fields. In particular, we describe the development of a visualization system for the display of the effects of storm-surge due to extreme weather phenomena, as derived by ADCIRC (or ADvanced CIRCulation) simulations.

4.2.1 Gigapixel Image Exploration

An obvious application for a gigapixel resolution display such as the Reality Deck is the visualization and exploration of gigapixel resolution imagery. It is important to note that the term imagery should not be constrained to traditional photography as several scientific sensors often yield similarly large datasets that can be treated as gigapixel images and

visualized with similar techniques (e.g., high resolution microscopy, data from telescopes, etc). The resolution of this data is often so vast that it can not be contained within the available memory on the visualization device.

Virtual Texturing When a traditional image is rendered via GPU acceleration, the entirety of the pixel data is uploaded onto a buffer in GPU memory. Then, a piece of geometry will be submitted for rendering, with the image buffer provided as a texture, which is to be mapped on to it. This geometry will be instrumented with texture coordinates, providing a correspondence between itself and the imagery. In modern programmable GPUs, the geometry is rasterized, producing "fragments" on which a fragment shader is executed. Within the fragment shader, the interpolated texture coordinates are used to sample the provided texture in order to resolve the appropriate color. Additionally, GPUs offer various texture filtering functions, often implemented in hardware, which enhance the resulting image quality at very low (or no) computational cost.

However, when the imagery does not fit into GPU memory, an "out-of-core" rendering scheme must be utilized. We provide an overview of several such schemes in Chapter 2. For the visualization application described herein, we utilize the "sparse virtual texturing" technique, described by Mittring [Mit08]. Virtual texturing obviates the need for the entirety of the image data to be immediately accessible during rendering on the GPU through a memory management scheme somewhat similar to "virtual memory" implemented by most modern operating systems. Effectively, virtual texturing allows access to a very large texture space (the virtual texture) when only a substantially smaller physical texture memory space is available.

Virtual texturing works by decoupling the rendering process of large textures into two steps. During the first step, the visible sections of the abstracted texture are determined. This determination can happen in a number of ways. For simple geometries (such as a plane which is perpendicular to the camera), the determination can happen analytically and on the CPU side. However, most virtual texturing schemes target arbitrary geometries, making an analytical determination non-viable. Thus, visibility determination is usually carried out via a secondary rendering step that occurs before the rendering of the full visualization. In this rendering step, which happens off-screen in a dedicated buffer, the renderer rasterizes the virtual texture coordinates of all objects and any auxiliary information (e.g., the identifier of a particular virtual texture). This step often happens

Figure 9: Our Visualization Library renderer module is displaying a gigapixel resolution panoramic view of Dubai[6].

at a reduced resolution in order to increase performance. Following this rendering, the resulting buffer is streamed on the CPU and then processed in order to determine which sections of the virtual texture are visible during the current frame. Virtual texture data is usually divided into fixed-resolution chunks (often called tiles or pages). By keeping track of the tiles that already reside in GPU memory, the virtual texture system then generates page-faults for the sections of the data that need to be brought into memory for correct texturing. Upon uploading the pages to the GPU (something that frequently happens in a separate thread to improve interactivity) the virtual texture system also updates a secondary indirection texture (described later). It is worth noting that there exist techniques that can accelerate this visibility determination step by compacting the texture coordinate buffer and calculating the page faults on the GPU [HPLVdW10].

After this visibility determination step, the geometry is then rendered to the screen. However, instead of using traditional texture mapping functionality exposed by the graphics API, a custom fragment shader needs to be utilized in order to properly sample the physical texture that contains the virtual texture pages. This shader translates the virtual texture coordinates of the geometry into the physical texture space by sampling an indirection texture which contains information for the position of each page within the physical texture. If a page is not available, then the indirection texture can potentially point to a mipmapped page which will contain a lower-resolution version of the required image data. Upon determining the appropriate physical texture coordinates, regular texture-sampling APIs are used to determine the color values from the physical texture.

[6]Sourced from http://www.gigapan.org.

Our gigapixel image exploration application handles the rendering of large images by implementing this above virtual texturing functionality. We utilized the openly-available libVT[7] , which was extended to integrate within our distributed rendering framework. Specifically, we integrated libVT with our Visualization Library rendering module. We have also developed a preprocessing framework which can transform imagery from common sources into the libVT-specific texture atlas format used by our application. An example of the gigapixel image rendering functionality provided by our system can be seen in Fig. 9. The system supports virtual textures up to approximately four gigapixels in size with real-time interactivity. It can be trivially expanded to larger virtual texture sizes.

F+C Lenses Focus and Context (F+C) techniques are an important tool for the exploration of large datasets, particularly when the display space is limited. Even in situations when the display space is not a problem, F+C techniques can be useful by allowing the in-place magnification of sections of the data, which can reduce the need for physical navigation as users do not have to approach the screens more closely in order to zoom in.

The F+C technique domain is quite vast (several techniques are covered in Chapter 2). For our software, we implemented the *Elastic Presentation Framework* (EPF) F+C lens technique, described by Carpendale et al. [CM01]. Our choice of EPF is motivated by its simplicity and direct applicability to GPU-accelerated rendering pipelines. Specifically, EPF lenses are implemented in our software via vertex shaders, which distort the image geometry based on a displacement function. The displacement function can be analytically defined, or precomputed and provided to the vertex shader via a texture. The distorted geometry is then rendered via perspective projection. The magnified areas are brought closer to the virtual camera and thus appear larger. Utilizing EPF for F+C lenses results in customizable lenses that can be applied to high resolution imagery at negligible computational cost. Additionally, this approach to F+C integrates directly with the virtual texturing system described earlier, allowing the application of lenses onto gigapixel resolution imagery. Some examples of gigapixel image exploration via F+C lenses can be seen in Fig. 10.

[7] http://www.sourceforge.net/projects/libvt

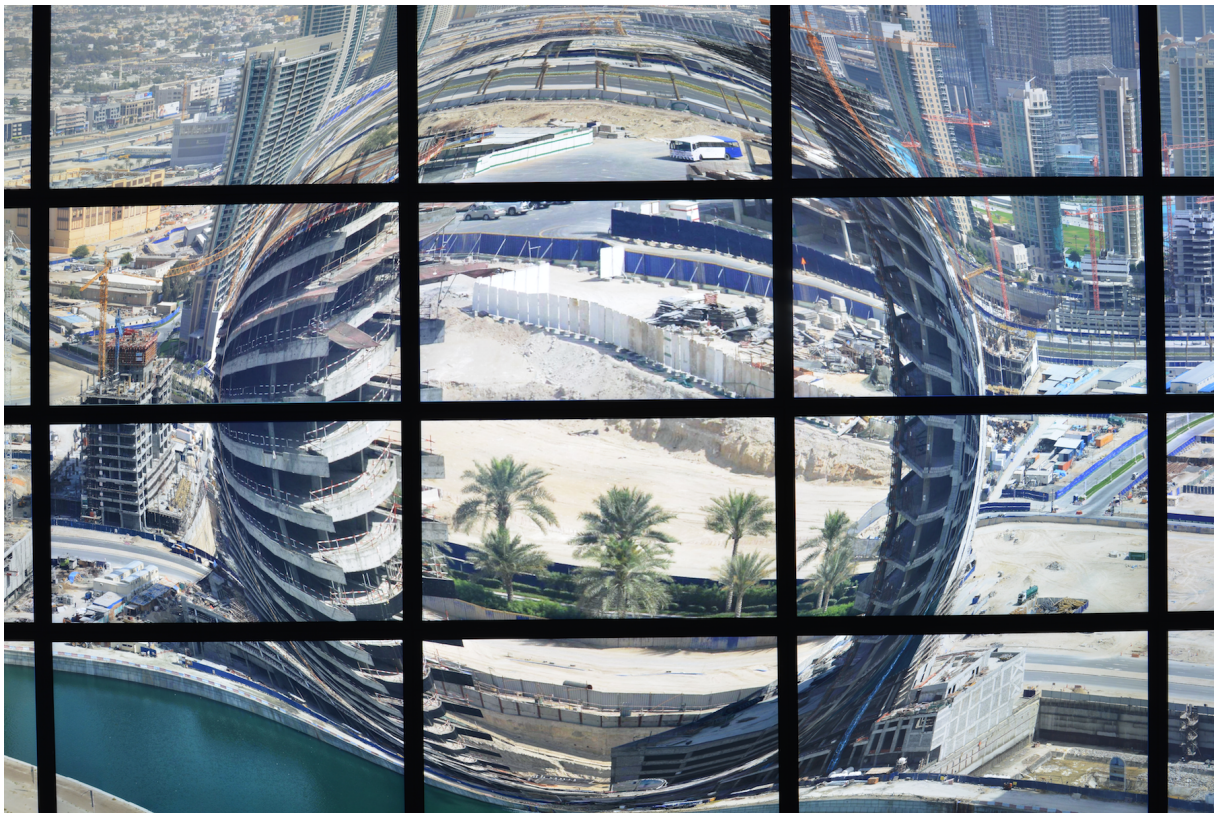


Figure 10: A gaussian F+C lens is applied via the Elastic Presentation Framework formulation on top of a gigapixel image. The lens acts as a displacement function (implemented in a vertex shader) bringing sections of the image closer to the camera under perspective projection and resulting in magnification.

4.2.2 Geospatial Data Visualization

The second application area that this dissertation focuses on is the visualization of world-wide geospatial data. On this front, we have developed two Geographic Information System modules (shortened to GIS throughout this text). The first module is a two dimensional GIS renderer, providing support for tile-based maps, georeferenced image overlays, information glyphs and ADCIRC-related visualizations. This GIS module integrated with the Visualization Library renderer. The second module utilizes the osgEarth SDK[8] to provide support for complex GIS visualization scenarios. We also extend osgEarth to allow the visualization of storm-surge effects determined by simulations. As with all visualization applications described in this thesis, this software can scale from running on a single computer all the way up to clustered rendering setups, such as the Reality Deck.

4.2.3 2D GIS Visualization

The first GIS visualization application provides support for complex two-dimensional geospatial scenarios, viewed under Mercator projection. To this front, we have developed a map layer hierarchy that can be described by users in simple human-readable scenario files. A scenario file describes various map layers that are to be visualized. Several types of map layers are supported:

1. Tiled-based map services such as Mapquest Open[9] .
2. Georeferenced images in various file formats.
3. Glyph-based visualizations from a custom-developed spatial datasource.
4. Color-mapped overlay visualizations of ADCIRC data.

Layers are described in scenario files, one layer per line. The first token in each line states the layer type while following tokens provide relevant layer parameters. Additional visualization settings, such as the focal point of the map, zoom scale and a bounding box within which rendering is restricted can also be communicated in this way. Finally, scenario files include descriptions of the datasources used to drive various glyph and

[8] http://www.osgearth.org
[9] http://developer.mapquest.com/web/products/open/map

overlay layers (the datasource architecture is described below). As an example, consider the following scenario file:

```
CENTER|-121.85548|37.35197
SCALE|0.16763
BBOX|-121.91673|37.34776|-121.79423|37.3561
DATASOURCE|FILE|DATA0|sample-DataSource.dat
LAYER|TILE|256|18|http://otile1.mqcdn.com/tiles/1.0.0/sat/
LAYER|GLYPH|DATA0|0|1|2|3
```

This scenario file describes a visualization in which the map is centered at approximately 121.85° West and 37.35° North, with a scale of 0.167 (a scale of 1.0 displays the entire globe). The **BBOX** line specifies a restrictive bounding box. Only visual information within this bounding box will be displayed during rendering. The **BBOX** directive is especially useful during user studies when we wish to limit the visible visual information. The **DATASOURCE** line describes a source of data that is used to drive the **GLYPH** layer described at the end of the file. This datasource (with identifier **DATA0**) will load information from the provided file. For debug purposes our software also supports datasources with random and constant values (these can be obtained by changing the **FILE** token in the relevant line). The first **LAYER** line described a tile-based imagery layer. Following tokens provide parameters such as the tile resolution (in this case 256 and used for level-of-detail calculations), maximum available zoom levels (18) and a URL to the content distribution network that serves the tiles. The final layer definition requests a glyph-based visualization of the geospatial information provided by **DATA0**. Each data source can expose several different types of information. The trailing tokens in the glyph layer specify the data identifiers that should be visualized by this particular layer. A sample of the resulting visualization can be seen in Fig. 11.

After the visualization application parses the scenario file, it generates the layer hierarchy which is maintained within the map renderer module. Before rendering each frame, the renderer utilizes the distributed camera state in order to perform a load operation on each layer. Specifically, utilizing the camera's position, current zoom scale and frustum information that is provided by Equalizer, each layer queries the underlying data source for information.

Figure 11: Example view of a 2D GIS scenario visualization. A scenario file was used to describe a visualization which includes a tiled based map streaming from Mapquest Open and a glyph data overlay driven from an ADCIRC simulation.

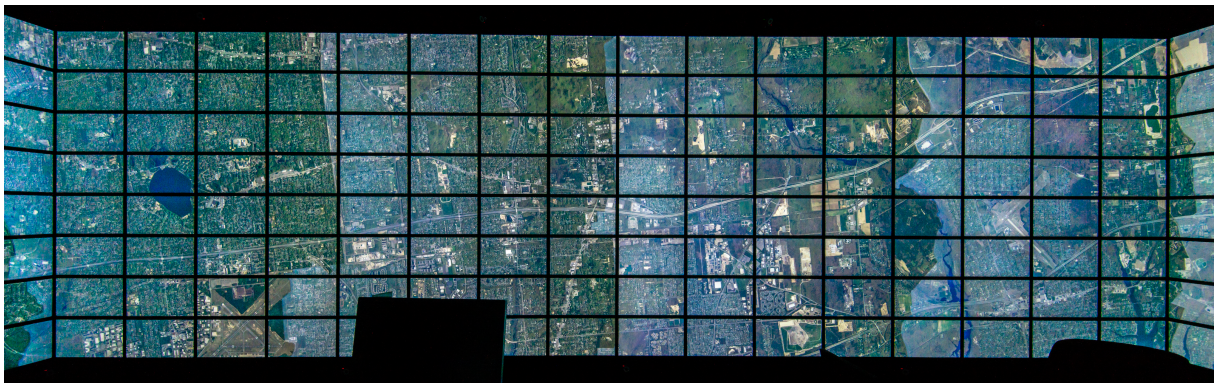


Figure 12: A two-dimensional tiled map is being streamed from an online service and visualized within the Reality Deck.

Tiled Map Layers For tile-based layers, this results in a zoom-level calculation, determining the appropriate tile LoD that needs to be fetch from the server. This calculation factors in the screen resolution and also the resolution of the tiled map service in order to provide imagery that saturates the pixel density of the display. Once the zoom level has been determined, the software also calculates the appropriate tile identifiers that need to be fetched from the map service. Most popular web-based mapping services expose raster imagery via an $X/Y/Z$ tiling scheme, with Z being the zoom level and X and Y being quad tree node identifiers. The tile identifier calculations, converting from a coordinate bounding box to the appropriate $X/Y/Z$ tuples can differ from one service to another[10] . Once the tile identifiers are resolved then the software proceeds to download the tiles asynchronously on multiple threads. Image tiles are cached locally on the visualization cluster using a Least-Recently-Used scheme in order to improve performance. During rendering, the tiles are textured mapped onto geo-referenced geometry which is injected into the Visualization Library scenegraph and rendered using the relevant layer parameters (such as Z-index and opacity). An example of a tiled map rendering from our application can be seen in Fig. 12.

Geospatial Data Sources - ADCIRC Data Handling For semantic data that can be visualized via the glyph and overlay layers, we have developed a spatial datasource that allows for low-latency bounding box queries. Our datasource can store geo-referenced points and polygons. Each point (and polygon vertex) can carry an arbitrary number of

[10] A good resource on the topic can be found at http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/

datums (or pieces of information) and datums can also be time-varying. This approach provides a significant amount of flexibility, allowing us to store complex, multi-modal simulations in a single data source and query them at run-time. For example, the results of an ADCIRC simulation[11] can include information such as the *residual surge* across multiple simulation timesteps (which is a single scalar value), wind conditions across multiple timesteps (which can potentially be a vector) and several minimum and maximum values over the simulation domain. This data is different for each point in the simulation grid and a grid can have hundreds of thousands of points. Moreover, the output of an ADCIRC simulation is a set of ASCII files, which do not lend themselves to real-time processing. We have developed a preprocessing pipeline that generates a quad-tree based spatial data source from an input ADCIRC model. The user can specify an ADCIRC grid file along with several input information files. The resulting data source (which is comprised of a human-readable descriptor file and several binary data files) is generated promptly (approximately $1-2$ minutes on a consumer grade laptop for a simulation with roughly $200,000$ grid points) and from that point on it can be reused. Our datasource implementation also offers runtime introspection functionality, allowing applications to query for the provided information at each grid point (including its name, time resolution and format -e.g., floating point, double precision, vector, etc.). Finally, we have developed a datasource generator application which can be used to create semi-randomized geospatial datasources that can be used for debugging and user study purposes. A screenshot of our datasource generator interface can be seen in Fig. 13.

Glyph Layers The glyph visualization layer allows the display of arbitrary numerical information that is provided by a geospatial datasource. When a user defines a glyph layer in the scenario file she provides a series of numerical identifiers (in the example above the identifiers are 0, 1, 2 and 3) which correspond to available information sources in the data source. During runtime, at the layer load step, the renderer module queries the datasource for available points within the visible bounding box. The returned points are then processed in a background thread. For each point, the required information is extracted based on the requested identifiers. Each data points position along with the required information are then forwarded to a geometry generation system. The geometry

[11] http://www.adcirc.org

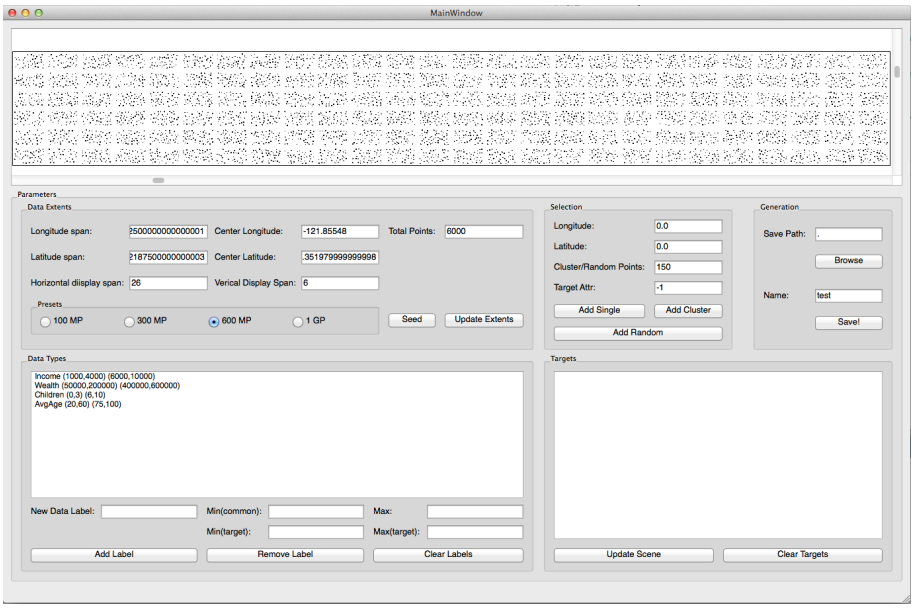
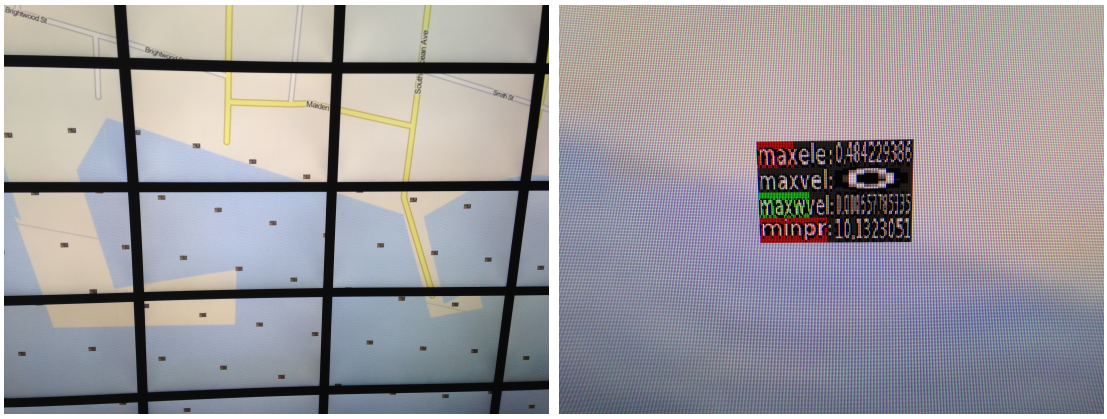


Figure 13: Sample view of our spatial datasource generator tool, used to create semi-randomized datasources for debugging and user study purposes. Users can control data point density, properties and alignment with respect to the bezels of the Reality Deck.

generator returns a generic "georenderable" object which can be injected into the scene-graph and drawn with at a fixed screen size. This approach allows for new visualizations to be created by developing a new geometry generator rather than a completely new layer. It is also worth noting that the generation of geometry also happens in a separate thread, improving interactivity for the application. In our current implementation the glyphs provide a fused text + barchart display, similar to the visualizations used by Ball et al. [BN08] in their user studies. A sample glyph visualization can be seen in Fig. 14.

Overlay Layers Our 2D map renderer also supports color-mapped overlay layers driven by a spatial datasource. In part these layers operate similarly to the glyph layers described above. However, once the datasource has been queried and the relevant points are returned, rather than generating individual glyph per point, this layer proceeds to triangulate all visible points into a single mesh. This triangulation, powered by the CGAL library[12] also happens in the background in order to preserve application performance. The resulting mesh is then uploaded to the GPU and injected into the scenegraph for

[12] http://www.cgal.org



(a) (b)

Figure 14: Example of 2D GIS glyph visualization. (a) Section of the Reality deck showing glyphs visualized using our two-dimensional GIS renderer. (b) Closeup of a glyph illustrating the dual text+barchart visualization.

rendering. In the layer declaration, the user provides the identifier of the piece of information to be visualized. During the generation of the overlay geometry, the system will query the datasource for the value range of this particular datum and use that to provide a color-mapped visualization of the values by normalizing each value and then mapping it to the HSV color-space. An example of this overlay visualization driven by ADCIRC simulation data can be seen in Fig. 15.

4.2.4 3D GIS Immersive Visualization

While two-dimensional maps can provide a familiar visualization modality for certain situations, some scenarios can benefit from a more immersive graphical experience. For example, when visualizing ADCIRC simulations in order to predict impact areas during a storm, the user can benefit from seeing the results of the simulation overlaid on top of a three dimensional terrain model. Additionally, providing additional context (by adding visual cues such as buildings) can be beneficial to the data exploration process.

In order to enable these complex, immersive visualizations we developed a second map renderer, this time based on the OpenSceneGraph renderer module of our software. The terrain engine which drives map rendering is implemented by osgEarth. The osgEarth SDK is an open-source plugin for OpenSceneGraph. From a bird's eye view, it enables OpenSceneGraph to parse and load .earth descriptor files (which correspond to layer

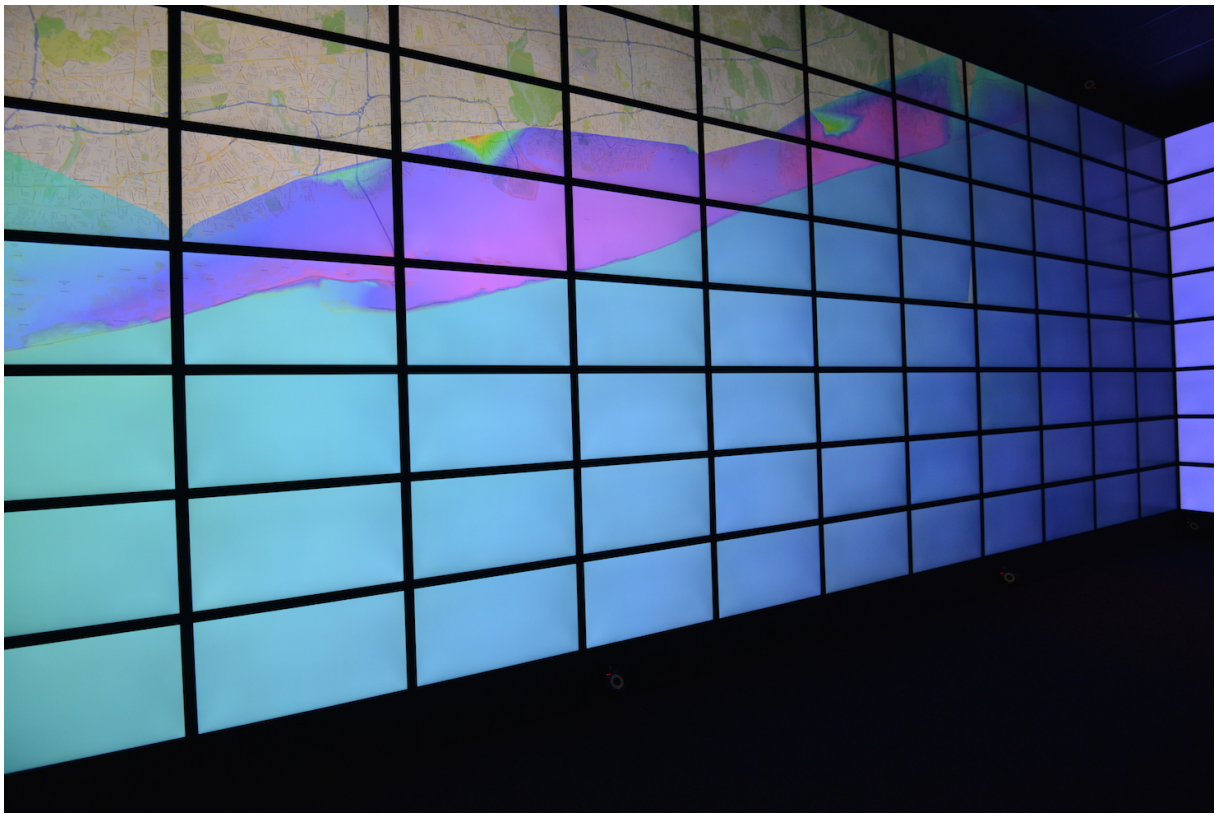


Figure 15: Sample view of 2D GIS overlay visualization. Residual surge values stemming from the ADCIRC simulation are color-mapped to the HSV color-space.

hierarchies), generating appropriate geometry for rendering. However, osgEarth is in fact very complex, providing a separation between map data and rendering, enabling the utilization of a vast range of geospatial data formats, providing support for scriptable rendering and finally offering a foundation on which custom GIS-related applications can be constructed.

Integrating osgEarth within our OpenSceneGraph renderer was not without challenges. Simply loading .earth files proved to be extremely straightforward (all that is needed is the presence of the appropriate plugin libraries within OpenSceneGraph's plugin folder), actually interacting with the visualization was more complex. The reason is that the geometry generated by osgEarth spans over extreme value ranges (which is to be expected since it can represent an entire planet). This leads to interaction issues. Interactions are affected since the utilized modality needs to provide large enough manipulations when far away from the planet's surface yet offer high fidelity when zoomed-in. Additionally, users expect that applications which visualize an entire planet offer an "ARCBALL"-style navigation interface (similar to Google Earth for example). Consequently, the approach for interactions within our VL-based renderer module (manipulating the distributed camera state directly based on user inputs) did not apply itself in this case. Thankfully, the osgEarth SDK contained an interaction module, termed "EarthManipulator". Manipulators are a component of the OpenSceneGraph architecture that can be installed on an OpenSceneGraph "Viewer" object. They consume input events from the window that the viewer is drawing to and use them to manipulate the virtual camera a viewer is rendering with. This EarthManipulator implements an ArcBall style interface along with additional functionalities (such as double tap to zoom, camera pitch, etc) while being cognizant of osgEarth's geometry. The osgEarth manipulator is integrated within our renderer application in the following way. Within the application interaction handler (which was described earlier in this chapter) we integrate a "headless" OpenSceneGraph Viewer. This Viewer performs most steps of a normal viewer's frame loop (e.g. cull and update traversals) but does not perform any rendering. Overall, it has been modified to ensure that no OpenGL calls whatsoever occur during its simulation loop. We then install an EarthManipulator on our custom headless viewer. When the interaction handler receives input events, it "translates" them to OpenSceneGraph-specific input events and forwards them to the Viewer's proxy window object. The Viewer processes these events during its simulation loop and manipulates its internal camera. We then capture the camera state

Figure 16: Sample view generated via our osgEarth-based 3D GIS renderer. This image shows a view of downtown Boston with synthetic building geometry derived from building outlines. The base image layer and building facades were sourced from the osgEarth distribution.

(e.g. its modelview matrix) and distribute that to the visualization nodes for rendering. The visualization nodes utilize this distributed modelview matrix and also view frustum information provided by Equalizer to draw the appropriate view. A sample view of our osgEarth-based renderer can be seen in Fig. 16.

By leveraging the flexibility of osgEarth we can generate powerful immersive geospatial visualizations. This is achieved by manipulating .earth files which define the visualizations.

Consider the following example:

```
<map name="sample map" type="geocentric" version="2">
  <elevation name = "elevation layer" driver = "gdal">
      <url>./dem/</url>
    <extensions>tif</extensions>
    </elevation>
    <image name="image layer" driver="xyz">
        <url>http://oatile[1234].mqcdn.com/tiles/1.0.0/sat/{z
           }/{x}/{y}.jpg</url>
        <profile>spherical-mercator</profile>
    </image>
</map>
```

This very basic .earth file defines a visualization with one elevation layer and one image layer. The elevation layer will source its data using the "gdal" driver. The driver infrastructure is a core component of osgEarth, allowing developers to provide support for new sources of data while integrating within the existing layer architecture. We utilize the driver architecture for our ADCIRC-related extensions to osgEarth, described below. In this example, the gdal driver will interface with the Geospatial Data Abstraction Library[13] in order to load any .tif (GeoTIFF) files within the provided folder and utilize them as elevation data for the planet model. The image layer on the other hand will utilize the "xyz" driver (a source for tile-based imagery from online services) to obtain color imagery from Mapquest Open (as is obvious by the url property). It is worth noting that multiple elevation and image layer definitions can exist within the same .earth file, with different priorities and geospatial coverages. Additional immersive visualizations with osgEarth in the Reality Deck can be seen in Fig. 17 and Fig. 18.

ADCIRC-related extensions to osgEarth In order to support the visualization of ADCIRC simulations within our osgEarth-based renderer we utilize the "model" layer architecture. Model layers are the simplest of all osgEarth layers but also provide the greatest degree of flexibility. Out of the box, osgEarth provides two drivers for model layers. The first is a "feature" driver (with two implementations) which can be used to

[13] http://www.gdal.org

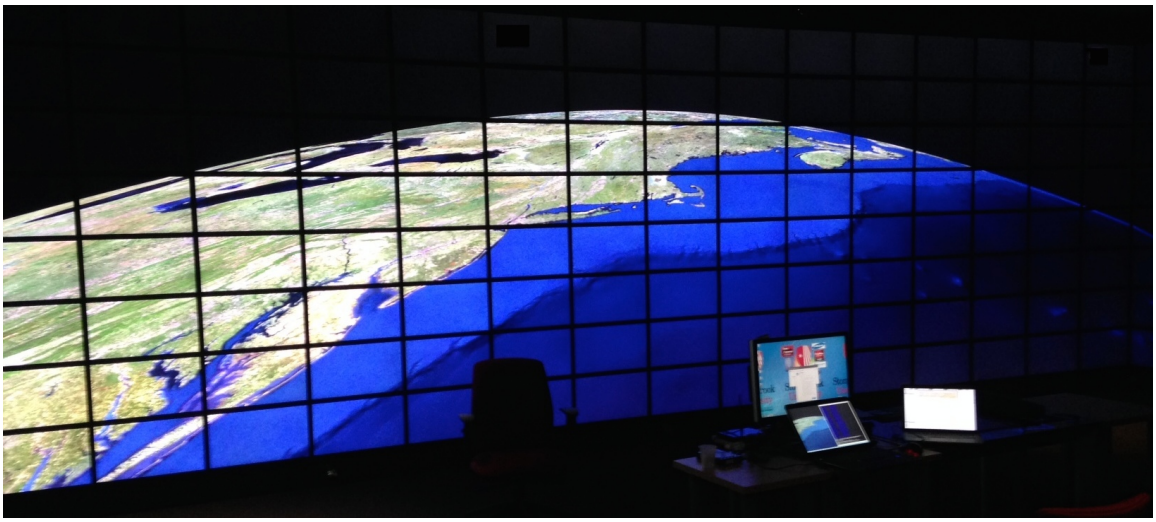


Figure 17: Zoomed-out view of the earth from our 3D GIS renderer. The base map layer was streamed from Mapquest Open.

Figure 18: State-level view of Massachusetts in the Reality Deck. A high-resolution image layer (stored offline) is overlaid on top of a steamed base map. This GIS scenario is part of the osgEarth distribution.

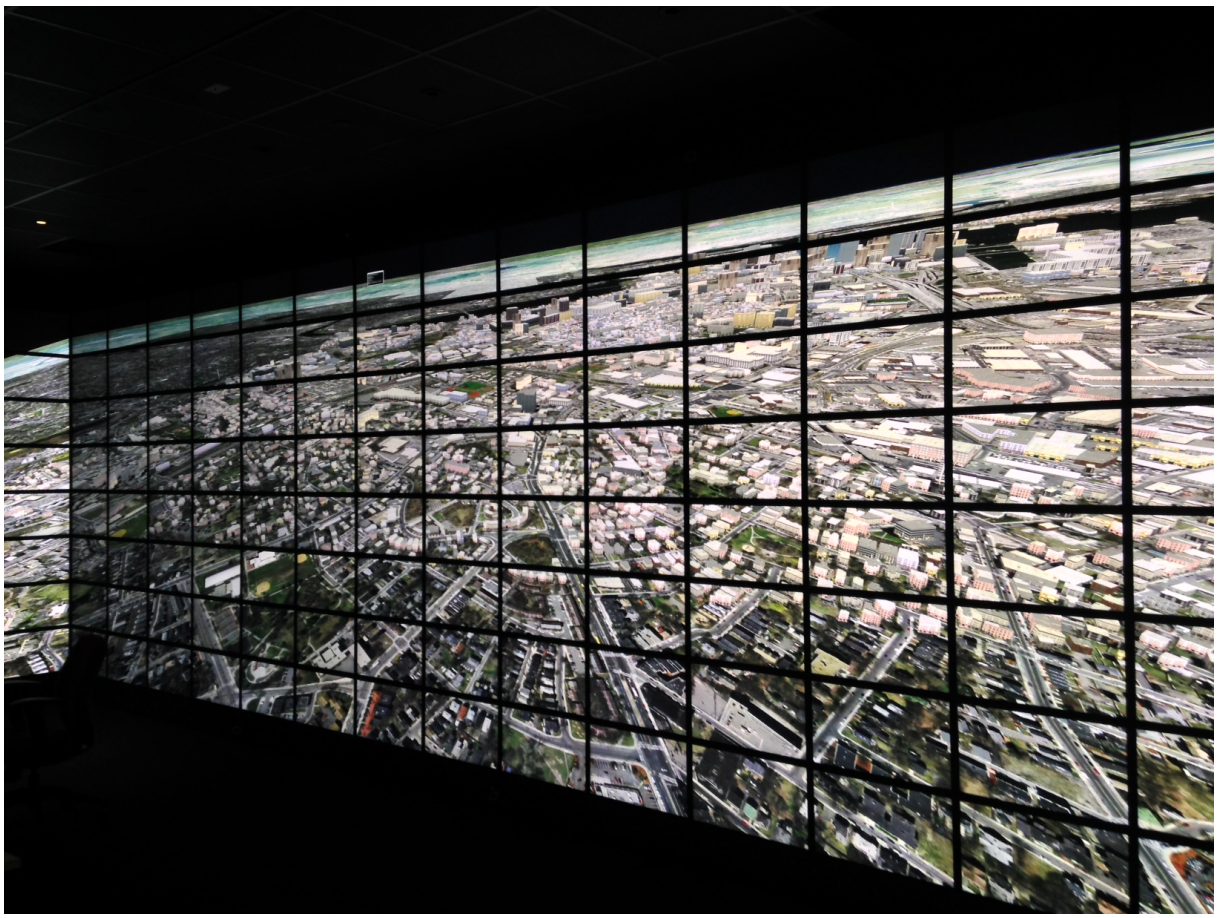


Figure 19: 3D immersive GIS visualization of synthetic building geometry. This geometry is generated by osgEarth through processing of building outlines provided as a shapefile. The facade textures are selected automatically depending on building properties.

render vector-based data. In our applications we utilize the built-in feature driver in order to generate procedural 3D building geometry in the areas which are included in our visualization, as seen in Fig. 19. The second model driver is termed "Simple Model" allowing the display of general 3D models, in formats supported by OpenSceneGraph, on top of the map. Generally, all osgEarth model drives are expected to return an OpenSceneGraph node which is inserted into the scenegraph. Additional operations (such as paging and level of detail) can be implemented within this node.

To support the visualization of ADCIRC data, we have developed an "adcirc_overlay" driver for osgEarth model layers. The driver leverages the spatial datasource that we described earlier in this chapter in order to visualize ADCIRC simulations in real-time. Since the size of the ADCIRC simulation domain can be substantial our driver uses a deferred paging scheme and plugs into the OpenSceneGraph pager system. Specifically, when the model layer is initialized, our driver is queried. It proceeds to load the metadata of the ADCIRC simulation (e.g. simulation extents, total timesteps, etc). Then, the simulation extents are subdivided based on a user-provided parameter. For each subdivision of the coordinate extent, the driver then generates a single OpenSceneGraph paged-loading node. The totality of the nodes are then aggregated under an OpenSceneGraph group node and returned to the system. During runtime, the OSG paged loading system comes into effect. When one of the paged-loading nodes is close enough to the camera, an asynchronous load request is placed to the OSG asset loading system using a custom URI, specific to our application. Our driver is registered as supporting this specific URI and responds by querying the spatial datasource for the relevant data and synthesizing the geometry. The returned geometry is then inserted into the scenegraph and rendered.

Our system [PMK14] supports multiple visualization modes and provides a number of user configurable parameters. Consider the following ADCIRC model layer definition:

```
<model name="ADCIRCOverlay" driver="adcirc_overlay">
    <url>./data_folder/</url>
    <datasetname>sampleDataSource.dat</datasetname>
    <subdivisionfactor>50.0</subdivisionfactor>
    <renderingmode>flat</renderingmode>
</model>
```

The **URL** and **DATASETNAME** parameters specify the location and name of the

Figure 20: Immersive 3D ADCIRC simulation visualization.
Illustration of simulated flooding in down-town Manhattan, generated via our extensions to osgEearth which permit interfacing with ADCIRC simulations.

spatial datasource that will be used to generate the ADCIRC grid geometry. The **SUBDIVISIONFACTOR** parameter specifies how many times each side of the coordinate extent bounding box will be decimated (and consequently the number of paged-loading nodes that will be inserted into the scenegraph). Finally the **RENDERINGMODE** parameter allows the user to specify the shading and geometry generation mode for the overlay. Our system currently supports two modalities:

- Flat-shaded overlay with DEM interaction In this modality the results of the ADCIRC simulation are interpreted as a water surface which is offset from the base elevation provided by the vertical datum of the visualization. This allows the overlay to "interact" with the digital elevation model, covering parts of it if the water level is adequately high. Effectively this modality allows the visualization of flood impact areas, granted that an appropriate digital elevation model is available. For a sample visualization using this rendering mode, please see Fig. 20.
- Color-mapped overlay This second modality is similar to the overlay layer provided

Figure 21: Immersive 3D ADCIRC simulation visualization.
In this example, the ADCIRC simulation mesh is visualized directly, and the residual surge is color-mapped to a blue-green hue range.

by our two-dimensional map renderer. The key difference is that rather than triangulating the visualization mesh on the fly, we instead preserve and visualize the same mesh topology that was used for simulation purposes. The overlay is rendered with depth-testing disabled in order to ensure that the map's digital elevation model does not cause occlusions. Some examples of this visualization modality can be seen in Fig. 21.

Chapter 5

NuNav3D - Natural User Interface for 3D Navigation

5.1 Introduction

3D navigation is perhaps the most important component of interaction with virtual reality environments. Traditional solutions include gamepad controllers and tracked wands. Touch-based interaction is also used under certain situations (large tiled displays). Locomotion techniques such as Walking-In-Place or real walking (perhaps in combination with path bending) also tackle the issue of navigation inside a VR visualization. We presented an overview of traditional and more novel navigation methods in Chapter 2. The common denominator of most of these methods is that they require the use of some sort of navigation device, prop and/or the utilization of a tracking system. However under certain situations, these conditions may be unattainable, i.e:

- Using a controller for extended periods of time may cause fatigue.
- Public displays or VR systems, for which providing a device for every user may be infeasible.
- Large touch-driven system require the user to repeatedly walk up to the display for interaction and then walk away for observation.

Bowman et al. [BCF+08] have described some of these scenarios as promising for 3D user interface research.

NuNav3D [PSK12a, PSK12b] tackles the 3D navigation task based on one main principle. That the user shall require no devices, props or markers to navigate through the VR environment. NuNav3D directly exposes 4 DOFs of navigational control to the user, mapping rotational and transitional controls for a virtual camera to hand movements in a manner similar to analogue sticks on a game controller. Additional DOFs can also be exposed (up to a total of 6 maximum) by direct mapping to hand movements or through the usage of gestural modifiers.

It is worth noting that development of NuNav3D occured prior to the construction of the Reality Deck so the implementation and experiments were conducted on a smaller-scale visualization setup. Specifically, we have implemented NuNav3D using a low-cost Time of Flight (ToF) sensor (Microsoft Kinect[1]) and compatible skeletal mapping software (Primesense NITE[2]). We have developed a simple pose recognition framework that allows the definition and detection of full body poses that trigger the navigation modality or affect visualization parameters. The usage of a ToF sensor makes NuNav3D an essentially transparent interface as it does not require any additional controllers or tracking markers to be mounted on the user before interaction. NuNav3D is evaluated under two scenarios:

- A path-following task within tight geometrical constraints (inspired by Virtual Colonoscopy [KLKB05]).
- A general exploration task of an open scene.

In both cases, it is compared to a traditional joypad controller.

In this chapter, we begin by presenting our NuNav3D framework and defining the base navigation scheme. We describe our pose recognition algorithm and its usage in transitioning to and from the navigation modality. Afterwards, we introduce the current implementation of NuNav3D and proceed to describe our experimental protocol. We conclude by presenting findings that guide the design of the user interface described in Chapter 6.

[1] http://www.microsoft.com/en-us/kinectforwindows/

[2] Since the development of NuNav3D, Primesense was purchased by Apple Inc. and no longer maintains an online presence.

5.2 NuNav3D: A Navigation NUI for 3D Visualization

Before introducing the NuNav3D scheme for 3D navigation, we must define some basic concepts used throughout this chapter. NuNav3D operates on poses, skeletal data provided by an external system. We define a *pose* P as a collection of n joint positions $p_1, \ldots, p_n$ and orientations $R_1, \ldots, R_n$ in 3D space:

$$P = \{p_1, \ldots, p_n, R_1, \ldots, R_n\}$$

The positions are defined as 3-component vectors in 3D space and the rotations as 3×3 matrices. We use the notations p_{LH} and p_{RH} to refer to the positions of the joints corresponding to the user's left and right hands respectively, as they are important to our navigation scheme. We define a *pose history* $P(1, \cdots, m)$ as a sequence of consecutive poses:

$$P(1, \ldots, m) = \{P_1, \ldots, P_m\}$$

5.2.1 Pose Recognition Framework

We have developed a simple, threshold-based pose recognition framework. To make our system invariant to changes in user physique, we transform each pose P_i from 3D world-space coordinates as provided by the skeletal mapping framework into a *uniform* coordinate space. We term this process *pose uniformization* and describe it below:

1. We translate the skeleton to the center of the coordinate system. This is achieved by expressing each joint position as $p_i^{'} = p_i - p_{Torso}$ (p_{Torso} being the position of the sternum as returned by the skeletal tracking system).

2. We compensate for any rotation of the user with respect to the tracking sensor by rotating each joint position by the inverse of the torso's rotation $p_i^{''} = R_{Torso}^{-1} p_i^{'}$.

3. Each joint position can be expressed as an addition of vectors starting from the torso. For instance, $p_{LH} = p_{Torso} + (p_{LeftShoulder} - p_{Torso}) + (p_{LeftElbow} - p_{LeftShoulder}) +$

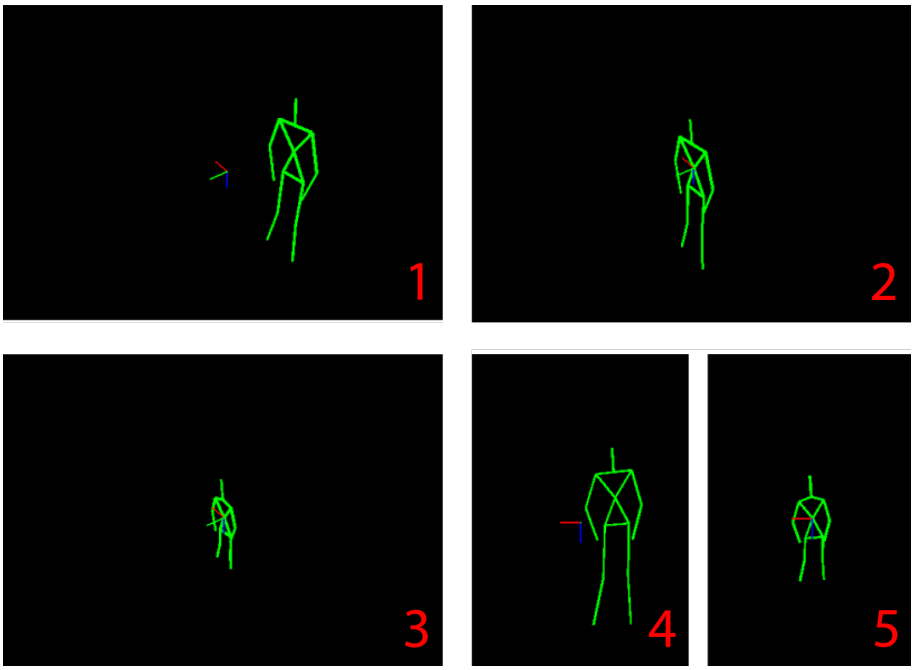


Figure 22: NuNav3D pose uniformization process. All figures have been extracted from a tracked user who was standing with his arms to his sides. (1) Original skeletal mapping. (2) Skeletal mapping after centering. (3) Skeletal mapping after rotation compensation and size normalization. (4) Frontal view of the skeleton pre-uniformization. (5) Frontal view of the skeleton post-uniformization.

$(p_{LH} - p_{LeftElbow})$. Similar linear combinations can be straightforwardly defined for all other joint positions. We normalize these vectors to unit length and then add them to calculate the joint positions in uniform space. For the above example, we have $p_{LH}^{unif} = (0,0,0) + norm(p_{LeftShoulder} - p_{Torso}) + norm(p_{LeftElbow} - p_{LeftShoulder}) + norm(p_{LH} - p_{LeftElbow})$ since we have already translated the joint positions in Step 1, thus the torso is now aligned with the center of the coordinate system.

A visual view of the uniformization process can be seen in Figure 22.

We use the symbol P_i^{unif} to denote a pose P_i after the uniformization process. In the uniform space, we can define a scalar pose variation metric s that expresses the difference between two poses P_A and P_B:

$$s_{AB} = \frac{1}{n} \sum_{i=0}^{n} \{ |P_A^{unif}(i) - P_B^{unif}(i)| \}$$

where $P_A^{unif}(i)$ and $P_B^{unif}(i)$ are functions that return the positions of the i-th joint of P_A^{unif} and P_B^{unif}.

Experimentation has demonstrated that the uniformization process makes the metric s_{AB} relatively invariant against differences in user physique (height, length of appendages, etc).

We define a collection of uniformized tracked poses $T(1,\ldots,l) = \{P_1^{unif},\ldots,P_l^{unif}\}$. Our pose recognition algorithm scans through a pose history $P(1,\ldots,m)$ and computes the metric $s_{P(m),T(l)}$ for each tracked pose. If the metric is lower than some threshold value s_{thres}, then a vote is cast in favour of pose T_l. In order for a pose action to be triggered, we require that a pose receives at least 30 votes throughout a pose history. This vote threshold ensures that actions are not triggered by involuntary user movements such as conversational hand gestures. Additionally, a pose history P will usually contain more than a few seconds worth of pose data (5 seconds in our implementation). In this case, the pose with the highest number of votes is triggered. Upon recognition of a pose, the pose history is cleared to prevent deprecate poses from triggering recognition events.

5.2.2 Definition of the Navigation Scheme

In order to formalize the navigation scheme, we first need a well-defined navigation pose. The user assumes this pose when he wishes to trigger the 3D navigation interaction modality. While technically any pose can be used, for comfort and usability reasons the navigation pose is set as the user standing or sitting, with the arms parallel to the chest and the forearms extended perpendicularly to them. This pose is termed P_{Nav} and can be seen in Fig. 23.

5.2.3 Transition to/from Navigation Mode

Upon startup, our system is running in the *pose tracking state*. At any point the user may assume P_{Nav}. Once the navigation pose is detected (as described above), the system transitions to the *navigation state*. Once this transition happens, the system takes a snapshot of the user's current pose P_{Nav}^{Base}. Furthermore, it relaxes the threshold value s_{thres} by a factor λ. To transition back to the pose tracking state the user must assume

Figure 23: The NuNav3D navigation pose. (1) The image map captured from the Kinect while the user is in the navigation pose. (2) The equivalent skeletal map pre-uniformization.

another pose P_i that has $s_{P_i P_{Nav}^{Base}} > s_{thres} \cdot \lambda$ (as opposed to just $s_{P_i P_{Nav}^{Base}} > s_{thres}$ which would be required for a pose to no longer be recognized by the system under the pose tracking state). This relaxation is necessary in order to afford the user with a greater range of motion for his hands inside navigation mode. During our testing, users achieved transitioning out of the navigation state by dropping both their hands and standing naturally. A schematic representation of the pose recognition and navigation pipeline can be seen in Fig. 24.

5.2.4 Hand Motions to Navigation

We describe the process of mapping the left hand motion to a normalized navigation vector. The same process can be applied for the right hand. We define an offset vector $v_{offset} = P_i(LeftHand) - P_{Nav}^{Base}(LeftHand)$. We wish to scale this 3D vector to a $[0, 1]$ range, with $|v_{offset}| = 1$ being the maximum hand displacement the user can achieve without transitioning out of the navigation state. Assuming that the user maintains perfectly still, this would allow a maximum displacement of $\frac{s_{thres} \cdot \lambda}{2}$ per hand. Thus scaling by this factor yields a roughly normalized navigation vector. Practically the scale factor is slightly larger, to accommodate for natural shifts in body position during navigation.

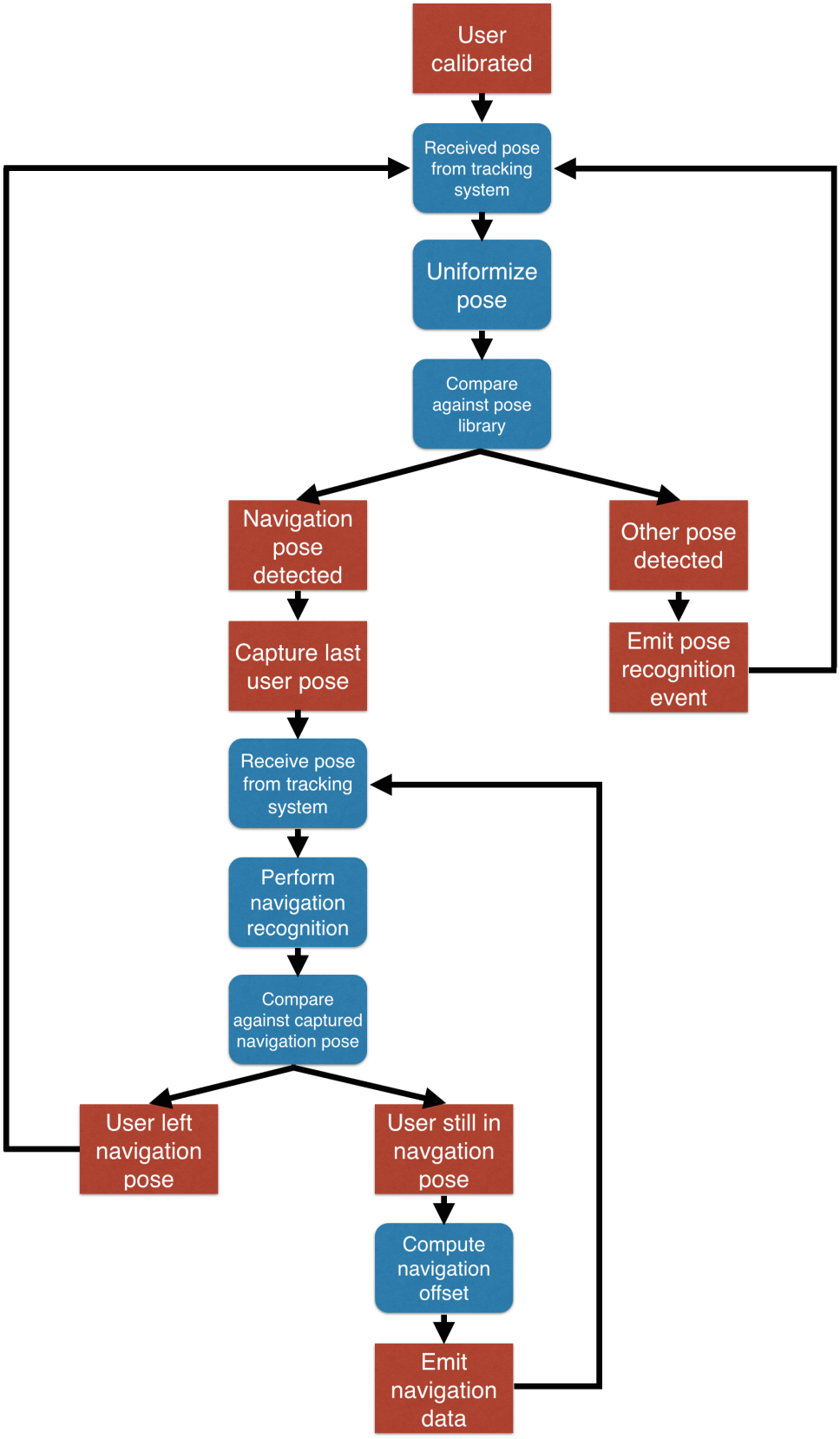


Figure 24: High level overview of the NuNav3D pipeline. This flow diagram illustrates the data flow and interaction trigger process for our system.

When the user maintains one hand close to the original position of P_{Nav}^{Base} then the other hand can achieve a maximum $|v_{offset}| \approx 2$. This case can be either handled by clamping the vector length to 1 or allowing it to extend to provide the user with a wider range of motion. The two resulting vectors (one per hand) from this process can then be interpreted by a virtual camera manipulator.

5.3 Implementation

We have created a C++ based implementation of NuNav3D. The current version utilizes a Microsoft Kinect sensor. The Kinect is a Time-Of-Flight sensor, providing a scene depth map that is generated from an infra-red pattern projection. It also offers visible-light images via a second camera. However, our implementation is not strictly limited to the Microsoft Kinect for sensory data, as it is built upon the OpenNI middleware. OpenNI abstracts depth and skeletal data generators behind generic nodes that can be easily reconfigured using an XML file. Thus alternate versions of NuNav3D can use other types of depth sensors or even full motion tracking systems, as long as they are abstracted by OpenNI. The current version uses the PrimeSense NITE framework for extracting skeletal data from depth images. NITE offers 15 joint positional and rotational data (although rotational data for the arms and feet is not robust and should not be used). Our implementation also uses the Qt framework for threading and graphical user interface purposes. It exposes the navigation data through the Qt event system and thus is very easy to integrate with other applications.

The implementation of the pose recognition framework closely mirrors the theoretical description of the above section. In practice, we have opted to ignore the head and leg positional data for the pose classification algorithm since users often adjust their leg positions during interaction with the system. Additionally, this relaxation allows usage of the system when the user is sitting down (however the tracking robustness of the NITE framework for sitting users is reduced). The values for s_{thres} and λ used in our implementation were 0.12 and 1.4 respectively and were determined experimentally.

The resulting vectors from the process are interpreted by the virtual camera manipulator of our visualization software. The vector lengths are clamped to $[0, 1]$ length then their

components are individually interpreted as translations or rotations. The y and x coordinates of the left hand offset vector v_{LH} are used for rotating along the x and y axis of the virtual camera respectively. Similarly the x and y coordinates of the right hand offset vector v_{RH} are interpreted as translations along the x and z axis (in camera coordinates, y is the up vector, z is the view direction vector and $y \times z = x$). In both cases, we apply thresholding to the offset values and then raise them to an experimentally defined exponent to ensure smoother controls. The same manipulator is applied to navigational data derived from the joypad controller against which we compare in our user study.

5.3.1 4-DOF versus 6-DOF

The above mapping effectively ignores the z components of each offset vector (displacement towards and away from the camera sensor). We initially utilized them to add support for rolling the virtual camera and translating along its y axis, but the additional degrees of freedom were hard to control for non expert users (particularly in the case of rolling the camera where there is no intuitive mapping between moving one's hand closer or farther from the depth sensor and having the movement result in a rotational change along the z axis of the camera). We can expose these 2 remaining DOF via modifier poses that allow the user to explicitly manipulate them with one hand.

5.4 Evaluation

We have performed experimental evaluation of NuNav3D via a user study.

5.4.1 Hypothesis and Metrics

Our experimental hypothesis is that, given a group of test subjects with limited experience under 3D navigation scenarios, NuNav3D will perform comparatively to a conventional dual-analogue stick controller under two separate 3D navigation tasks. The two tasks are *path following* and *exploration*. We also hypothesize that users will find the NuNav3D experience to be more natural and less intrusive than the one provided by the game controller.

Path following involves test subjects controlling a 3D visualization and being asked to follow a predetermined path through a scene. Exploration involves the users being placed inside a 3D scene with a number of out-of-sight objects. The users are asked to explore the scene and find these objects. In both cases, the quantitative performance metric is the amount of time each user takes to complete the task. We also performed a qualitative evaluation of the user experience via questionnaire after the study trials were completed.

5.4.2 Apparatus

The experimental apparatus used in our user study is as follows. A Dell Precision T7500 workstation with a quad-core Intel Xeon CPU and 16 Gigabytes of memory drove our visualization framework. The workstation used an AMD FirePro V9800 graphics processing unit to drive a lattice of 6 23-inch Samsung MD230 displays. The displays were arranged in a 2 row by 3 column geometry for an effective screen resolution of 5760-by-2160. A Microsoft XBOX 360 Joypad Controller was connected to the workstation via a wireless RF dongle. A Microsoft Kinect sensor was positioned under the display array and connected via USB. While in pose-detection mode, the visualization software displayed the user's current skeleton as well as the skeleton of the target navigation pose. When the user entered navigation mode, the two navigation offset vectors were visualized instead. A photograph of our experimental apparatus can be seen in Fig. 25.

5.4.3 Trial Data Sets

For the path-following task we created two synthetic virtual colonoscopy data sets. These datasets are based on actual colon centerlines extracted from real patient data but provide a smoother tubular structure. An example of such a data set can be seen in Figure 26. The initial virtual camera placement was in the location of the rectum. The data set for the exploration task is comprised of a collection of 9 simple buildings[3] on a plane. Each building had up to three levels. The trial area was well-defined by a surrounding wall structure. Four teapot objects were scattered throughout the scene, either in the buildings or out-of-sight. Two different building and teapot location arrangements were

[3]Building geometry was sourced from http://igad.nhtv.nl/~bikker/.

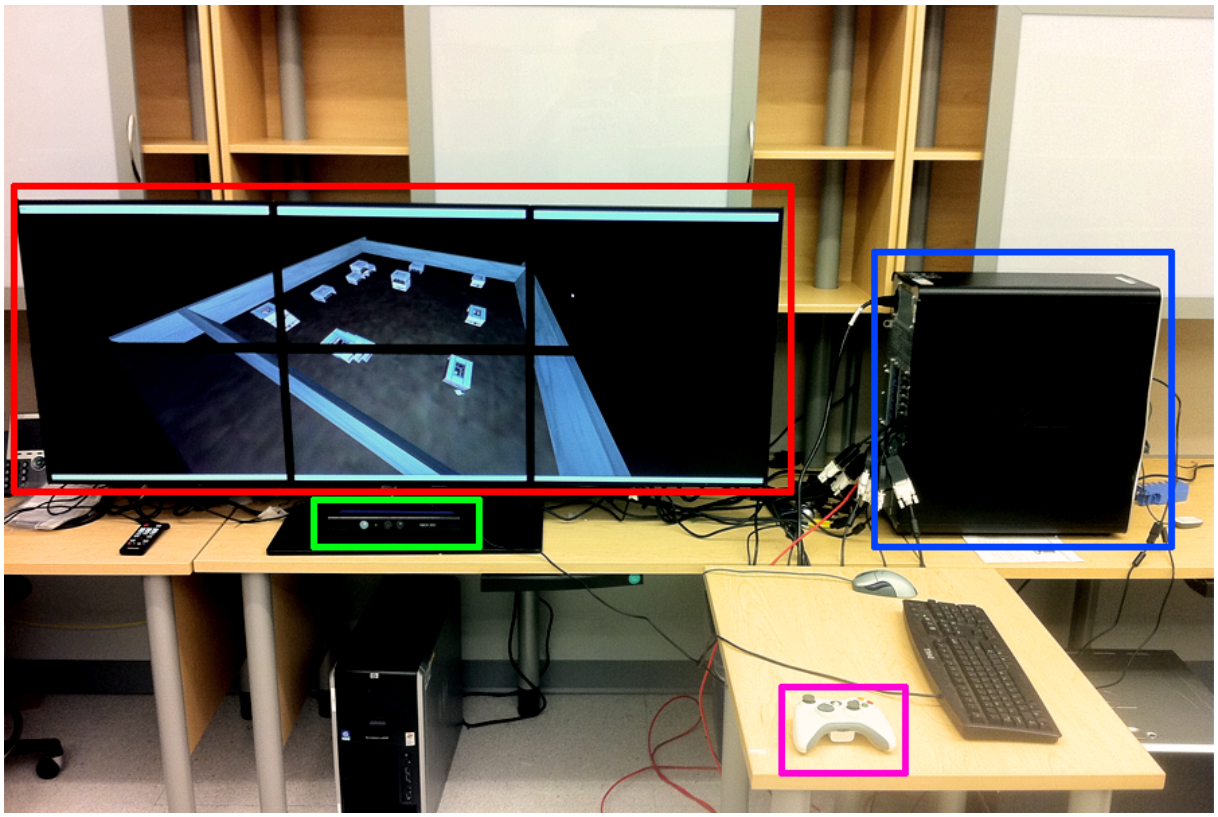


Figure 25: NuNav3D user study experimental apparatus. Red frame: Samsung MD230 display lattice. Green frame: Microsoft Kinect sensor. Blue frame: Dell Precision T7500 workstation. Purple frame: Microsoft XBOX 360 Wireless Controller.

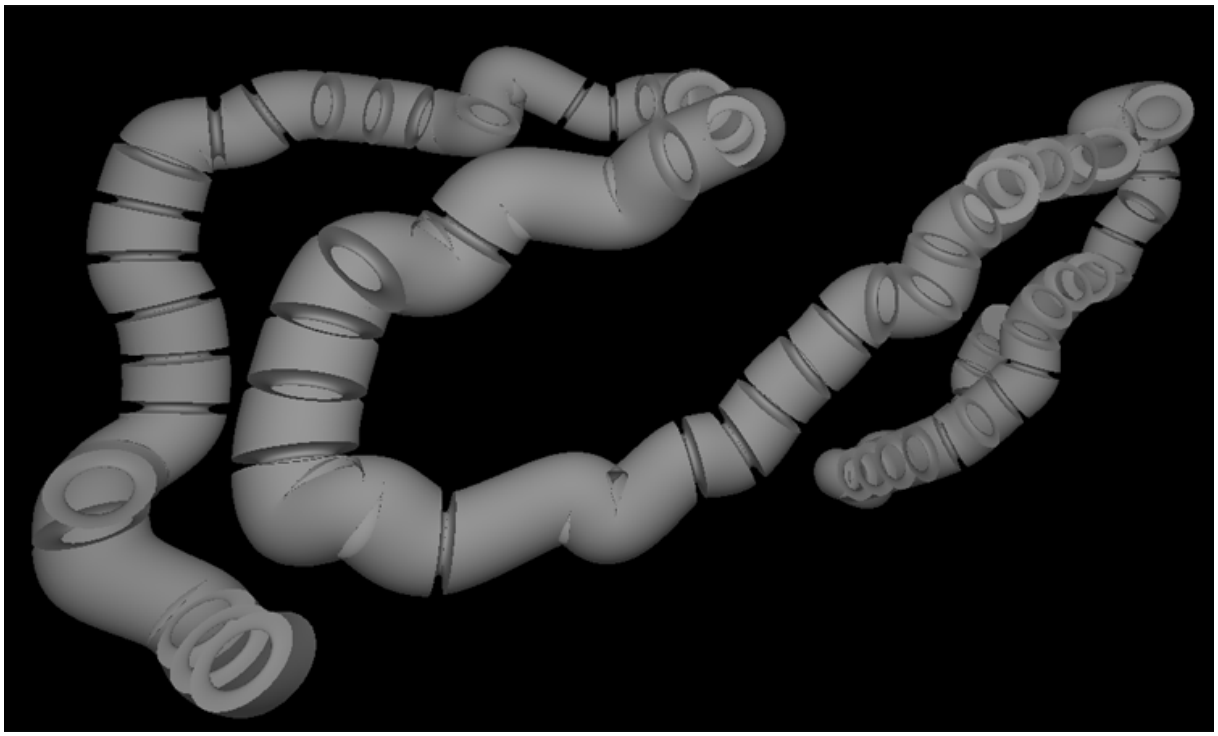


Figure 26: Sample colonoscopy dataset used for NuNav3D path-following task. The geometry normals are pointing towards the interior of the colon, making it transparent from the outside.

developed, one for each part of the trial. In both cases the camera was placed in the center of the scene, with a consistent initial direction. A sample screenshot of this data set can be seen in Fig. 27.

5.4.4 Trial Procedure

Each participant was introduced to the concept of NuNav3D by an instructor. Both the joypad scheme and NuNav3D were briefly (for approximately 5 minutes) demonstrated to the participant by the instructor. Then the participant was asked to step in front of the sensor for calibration (necessary for the NITE skeletal mapping implementation). Following, the instructor loaded a basic scene and asked the participant to familiarize herself with both navigation schemes (for a period no longer than 10 minutes). Each participant was then asked to complete the path following task using the game controller. Then the alternate colon data set was loaded and the participant was asked to repeat the process using NuNav3D. The order of the first interface used was alternated between

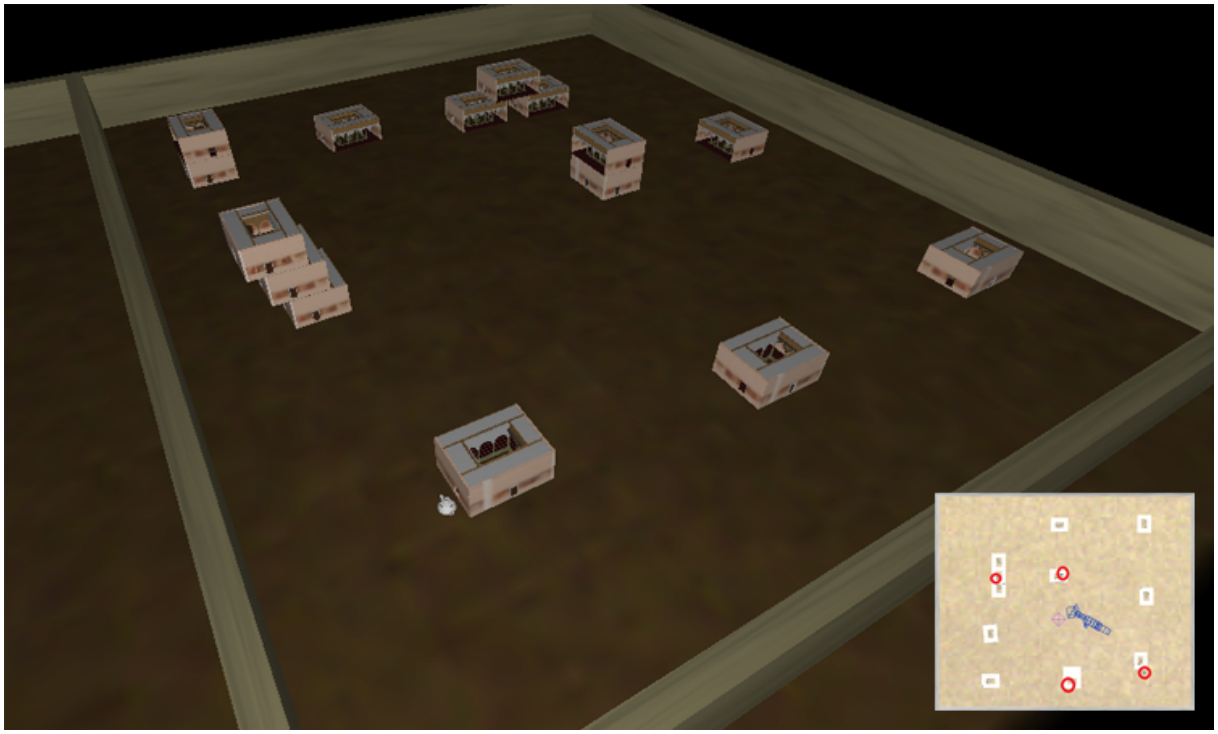


Figure 27: Sample scene used for the NuNav3D exploration task. The inlet demonstrates a top view of the scene with the starting camera position and direction visible. The hidden object positions have been marked with red circles.

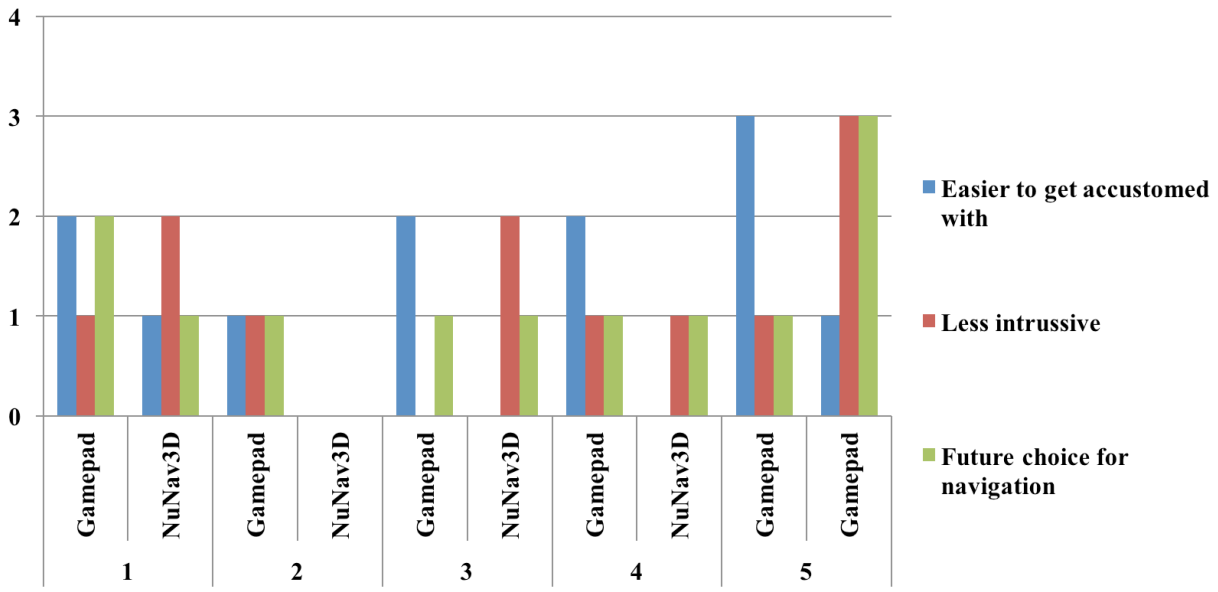


Figure 28: NuNav3D qualitative evaluation results. These timings are broken down based on the participants' 3D navigation experience level.

participants. A similar protocol was followed for the exploration task. In both cases the participants were timed. Collision detection was off during the trials to allow for maximum mobility in navigation. After trial completion, the participants were asked to fill a questionnaire regarding their user experience.

5.4.5 Results

The study was conducted on a group of 12 participants (10 male, 2 female). Average age was 24.16 and standard deviation was 1.89. On a scale from 1-5, the users were asked to rate their familiarity with 3D navigation (whether in an entertainment or professional context), with 1 being no experience and 5 being daily experience. The mean experience rating was 3.25 (SD 1.65).

In the path-following task, average time to completion was $2m26s$ when using the joypad controller and $3m50s$ using NuNav3D, failing to verify our original hypothesis. However, all participants completed the trial with our proposed interface. We observed that participants, when using the game controller, opted to move in discrete steps by flicking the translation stick as opposed to smooth controlled movements. On the other hand, NuNav3D exposed a wider range of motion and allowed for smoother translations and

rotations. However, on several occasions participants would overtranslate and exit the colon interior (which required corrective manoeuvres that in turn increased trial completion time). Some participants also complained about the ability to accidentally translate along the x axis of the camera while flying (strafing).

For the exploration task our comparative performance hypothesis was verified. The mean completion time between the two systems was very similar. Mean time for NuNav3D was $2m\ 20s$ whereas for the game controller it was $2m\ 3s$. Our observations with regards to smoothness of movement were consistent with those of the first trial. Participants commented that the open-area nature of the trial scene did not impose the rigid positional restrictions of the Virtual Colonoscopy dataset and allowed for a more pleasant and intuitive navigation experience.

The qualitative results of the evaluation are as follows. On the question "Which user interface did you find easier to get accustomed with?" 83.3% of participants preferred the game controller as opposed to NuNav3D. Users with high levels of 3D navigation experience particularly favoured the game controller. For the question "Which user interface did you find less intrusive during your experience?", 66.6% of participants answered NuNav3D. Finally, we asked users to express their navigation interface of choice for the future. 50% of users preferred NuNav3D overall. Users that expressed a preference towards the game controller stated that they would opt to use NuNav3D if the performance for navigation in constrained spaces was improved. These results are summarized in Fig. 28. To further clarify Fig. 28, no participants of experience level 2 indicated NuNav3D as an answer to the questions of the qualitative evaluation questionnaire.

5.5 Conclusion

Our evaluation of NuNav3D yields some interesting conclusions. Even users that have not had significant experience with game controllers found that system easy to pick up and use. This finding may be a result of the pervasiveness mainstream interaction entertainment applications. Contrarily the concept of using one's hands to navigate in 3D was considered more foreign for participants in the study. Nevertheless, by the end of the trial, participants were able to navigate effectively in 3D using NuNav3D, especially when the scenario at hand did not require following a very strictly defined path or remaining within

tight geometrical constraints. Another observation we made was that the assumption of a single, well-defined, navigation pose did not feel comfortable for all users. Some users felt that the pose was unnatural, at least initially. Furthermore using a threshold-based approach to enter and exit the navigation modality resulted in users being forced to recenter themselves to the navigation pose mid-motion, interrupting the smoothness of the experience. Users with more experience in NuNav3D did not encounter this issue and were able to complete the entirety of the trials without dropping out of the navigation modality. Still, in the interest of user experience, a less naive method of detecting the user's action intent may be necessary. This observation led to the development of our chirocentric user interface which utilizes hand-pose recognition in order to identify the user's interaction intent. Additionally NuNav3D did not expose a method for triggering actions within the visualization - rather it was strictly a navigation interface. The second contribution of this dissertation also provides this functionality via gesture detection.

Chapter 6

Practical Chirocentric 3DUI Platform for Immersive Environments

6.1 Introduction

Interaction with large immersive systems (CAVEs, HMDs, etc) is usually conducted via dedicated controller devices, such as tracked wands. These devices provide tactile buttons that can be used to either trigger specific actions (e.g., manipulating a visualization parameter) or enter and exit various manipulation states (e.g., dragging an object, translating the virtual camera, etc). In the majority of recent systems, tracking of the interaction props is provided via an infra-red (IR) tracking system, and generally through a commercial rigid-body solver that translates fixed arrangements of IR markers to positions and orientations within the tracking space. Deviceless approaches such as NuNav3D suffer from ambiguity in user intent which can result in unwanted camera movements.

These dedicated devices can instead be replaced with hand gestures and hand poses, creating effectively a *chirocentric* user interaction experience. In this chapter, we introduce two algorithms that enable the implementation of such a user interface, within the constraints imposed by a commercial IR tracking system. The first algorithm is targeted at the recognition of unimanual or bimanual hand gestures, which can then be used to

affect the virtual environment in specific ways (such as cycling through Points of Interest or switching rendering modes). This algorithm is inspired by work in the field of human activity recognition and driven by a gesture data set collected over 9 subjects. Our second algorithm tackles the problem of hand pose recognition from sparse point clouds provided by the IR tracking system. By utilizing a pair of low-cost gloves, with attached retroreflective markers, we can accurately determine the user's hand poses for each hand and use them to trigger various directly manipulative interaction modalities. This algorithm is evaluated over a 5 subject data set.

Utilizing the platform defined by these two techniques [PCS+15, PCS+13], we have developed a prototype chirocentric user interface for the exploration of 2D and 3D data within immersive environments. Our system exposes unimanual and bimanual manipulative interactions. We discuss the implementation details of our system and present various insights gained through its development and deployment within a large tiled display. We conclude with an outline of future plans for utilizing our platform for the formal evaluation of bimanual chirocentric 3DUIs under the Framework for Interaction Analysis (FIFA) [McM11].

6.2 Algorithmic Framework

Our chirocentric UI platform is driven by two algorithms that utilize prior knowledge for the recognition of hand poses and gestures based on input data from a tracking system. For hand poses, we utilize a feature vector of pair-wise distances between markers that are assigned to each of the user's hands, which is then used to predict a hand pose label via an Support Vector Machine. For gesture recognition, we utilize a more complex feature that incorporates both the distances between different joints of interest, as well as their motion. In this section, we expose in detail the specifics of our recognition algorithms. Both algorithms are targeted at commercial IR tracking systems and operate on two types of data:

- Rigid Body Positions - Denoted $\mathbf{P_i}$ for a single body, these rigid body positions correspond to the positions of various body parts of the user within the tracking space. Our gesture recognition generalizes to an arbitrary number of body parts, although for practical reasons we utilize the hands and head positions.

- Marker Clouds - Annotated as $\mathbf{M_i}$ for the i-th marker of the cloud, these positions correspond to the raw markers that are reconstructed by the tracking system. It is worth noting that, for a single physical marker at frames t and $t+1$, there is no guarantee that $\mathbf{M_i^t}$ and $\mathbf{M_i^{t+1}}$ will report its position. Effectively, our algorithm can not expect per-frame consistency in the ordering of reported markers from the tracking system.

6.2.1 Hand Pose Recognition

For our hand pose recognition algorithm, we assume that the tracking system provides us with $\mathbf{P_H}$ which is the position of the hand (obtained by a rigid body mounted on the back of a simple glove). Additionally, we are provided with a marker cloud $Markers = \{\mathbf{M_0} \cdots \mathbf{M_n}\}$, which contains all markers reconstructed by the tracking system. This cloud includes markers that are mounted on the tips of the thumb, middle and index figure of the user. Our low-cost tracked glove is shown in Fig. 29. In total, we are interested in the markers that correspond to the hand's rigid body (3 in our case), and the 3 finger tip markers. We construct a subset of $Markers$ termed $FilteredMarkers$ as follows:

Algorithm 1 FilterMarkers($Markers, P_H, DistThres$)

```
ClusteredMarkers = {}
for Mi ∈ Markers do
  if |Mi − PH| ≤ DistThres then
    ClusteredMarkers.append(Mi)
  end if
end for
while ClusteredMarkers.size() < 6 do
  ClusteredMarkers.append(PH)
end while
SortedMarkers = DistanceSort(ClusteredMarkers, PH)
FilteredMarkers = SortedMarkers.subarray(0, 6)
return FilteredMarkers
```

In this algorithm, $DistanceSort(ClusteredMarkers, \mathbf{P_H})$ sorts the $ClusteredMarkers$ vector based on the distance of each marker from the hand rigid body. Additionally, if

Figure 29: Low-cost, passively tracked gloves used by chirocentric user interface. It is constructed by attaching a tracking system rigid body to the back of a soft glove, using wires running through the fabric to preserve its elasticity and ensure a deformation-resistant connection. Retroreflective markers are attached on the tips of the thumb, index and middle fingers. The total cost of materials for one glove is under $25.

the original $Markers$ set does not contain enough markers (due to tracking occlusion), we insert placeholder markers at the position of the rigid body itself. Given $FilteredMarkers$ we can then proceed to the feature calculation for a particular hand pose.

Feature Calculation For a particular hand pose we construct feature vector $\mathbf{F^h}$ from markers $\mathbf{M_i} \in FilteredMarkers$:

$$\mathbf{F^h}(\mathbf{i}, \mathbf{j}) = \|\mathbf{M_i} - \mathbf{M_j}\|, \forall \mathbf{M_i}, \mathbf{M_j} \in FilteredMarkers$$

Effectively, our feature vector is defined as the pairwise Euclidean distance between all markers. However, as mentioned earlier, the ordering of the markers is not guaranteed to

be consistent between each frame delivered by the tracking system. To ameliorate this, we sort $\mathbf{F^h}$ in descending fashion, resulting in $\mathbf{F^h_{sorted}}$, ensuring consistency in its ordering. The dimensionality of the feature vector is 36 for our experiments (assuming 3 rigid body markers and 3 finger tip markers).

Training and Classification With the feature calculation defined, training and classification of hand poses are quite straightforward. We utilize a Support Vector Machine (implemented via the libSVM library [CL11]) using a Radial Basis Function kernel. We determined SVM parameters $C = 2$ and $\gamma = 1$ via grid search on a subset of the labelled training data. We report additional details on the training data in the experiments section below. Finally, our classification algorithm runs in real time (approximately 5 to 10 milliseconds per incoming frame of tracking data).

6.2.2 Gesture Recognition

Contrary to the potentially unreliable marker cloud data (which maybe require the insertion of placeholders and does not guarantee between-frame consistency between marker identifiers), rigid body data is significantly more robust. We assume that of $\mathbf{P_i}$ is the position of the i-th rigid body in 3D space as reported by the tracking system (practically, we utilize 3 rigid bodies, for the head, left and right hands of the user but our feature calculation generalizes to an arbitrary number). In the rare occasion that a rigid body is not tracked during a particular frame, we replace it with a placeholder at the center of the coordinate system. Since these rigid bodies correspond to joints of a very basic human skeleton, we refer to their positions as *joint positions* for the remainder of this chapter.

Feature Calculation Our feature is a combination of the distance between joints and their motion, aggregated over a window of time. In contrast to our hand pose feature, which identifies static poses, without a progression component, this feature calculation allows us to capture the dynamics of a particular gesture as it advances through time.

We augment the earlier notation by letting $\mathbf{P_{i,t}} \in \Re^3$ be the 3D location and of joint i of the subject at time t. Let T be the set of all the frames within the size of a frame window,

W. The feature of each such frame window is a single vector, defined as the concatenation of all computed features $\mathbf{F}(\cdot;\mathbf{t})$, where $t \in T$. In particular, we compute two sub features, one based on the pair-wise distance of joints for the current frame and the second based on the pair-wise distance of all pairs of joints in consecutive frames.

Joint distance The *joint distance* feature $\mathbf{F^{jd}}$ is defined as the pairwise Euclidean distance between all the joints of a persons at time t. It is defined as:

$$\mathbf{F^{jd}}(\mathbf{i},\mathbf{j};\mathbf{t}) = \|\mathbf{P_{i,t}} - \mathbf{P_{j,t}}\|, \tag{1}$$

where i and j are any joints of the user and $t \in T$

Joint motion The *joint motion* feature $\mathbf{F^{jm}}$ is defined as the Euclidean distance between all pairs of joints of a person at time t_1 and at time t_2. It captures dynamic motions between joints and mathematically formulated as:

$$\mathbf{F^{jm}}(\mathbf{i},\mathbf{j};\mathbf{t_1},\mathbf{t_2}) = \|\mathbf{P_{i,t_1}} - \mathbf{P_{j,t_2}}\|, \tag{2}$$

where i and j are any joints of the user, $t_1, t_2 \in T$, $t_1 \neq t_2$.

Training and Classification The window W spans a total of 13 frames (we report on the performance impact of experimentally determined constant later in this chapter). In order to ensure that the between-timestep differences are substantial enough (since our tracking system delivers data at $120hz$), we sample every 3rd frame of this 13 frame window. This value was chosen to balance the algorithm's performance, response time and the dimensionality of the feature vector. For each of the 5 sampled frames, we calculate the aforementioned joint distance feature, resulting in a total of 3 distances per frame (or 15 for the entire frame window). Additionally, for every combination of the 5 frames sampled from the window, we utilize 10 pairs of frames (5 choose 2) as sources for our joint motion feature calculation. For every pair, we determined the euclidean distance between joint i of the 1^{st} element in the pair and all the 3 joints of the second element of the pair and hence we obtain a 9 dimensional vector for each pair and in totality we have a 90-dimensional joint motion vector extracted from a window of 13 frames. The dimensionality of the combined feature vector is $15 + 90 = 105$.

We utilize a SVM driven by an RBF kernel for training and classification. The parameters $C = 1.0$ and $\gamma = 1.0$ were determined via grid search on our test data.

6.3 Experiments

Prior to the development of a chirocentric user interface based on our two algorithms, we evaluated their efficacy on two data sets, collected using an IR tracking system. Specifically, we utilized an Optitrack system, with 24 S250e cameras. Our cameras are arranged to cover the volume defined by a large immersive environment, which spans approximately $33' \times 19' \times 11'$. Practically, approximately 10 cameras closest to the user contributed to the rigid body tracking and marker cloud reconstruction (this determination was made using the visualization tools of Optitrack's Motive software).

6.3.1 Data Sets

Our first data set is targeted at evaluating the performance of the gesture detection algorithm outlined earlier. Specifically, we selected a set of 7 gestures that were relevant to various interactive applications within an immersive virtual environment. These are as follows (the parenthesized bold-face shorthand notations are used throughout the rest of the chapter):

- Pointing (**POINT**) - Pointing movement towards a display located in front of the user.
- Left Swipe (**SWIPEL**) - Swipe to the left.
- Right Swipe (**SWIPER**) - Swipe to the right.
- Zoom in (**ZOOMIN**) - The user's hands start extended in front of them and then diverge horizontally, mimicking a zoom-in gesture on a multi-touch display.
- Zoom out (**ZOOMOUT**) - The opposite of **ZOOMIN**.
- Expand (**EXPAND**) - The user's hand start extended in front of them and then expand vertically, in a gesture similar to expanding a scroll.
- Contract (**CONTRACT**) - The opposite of **EXPAND**.

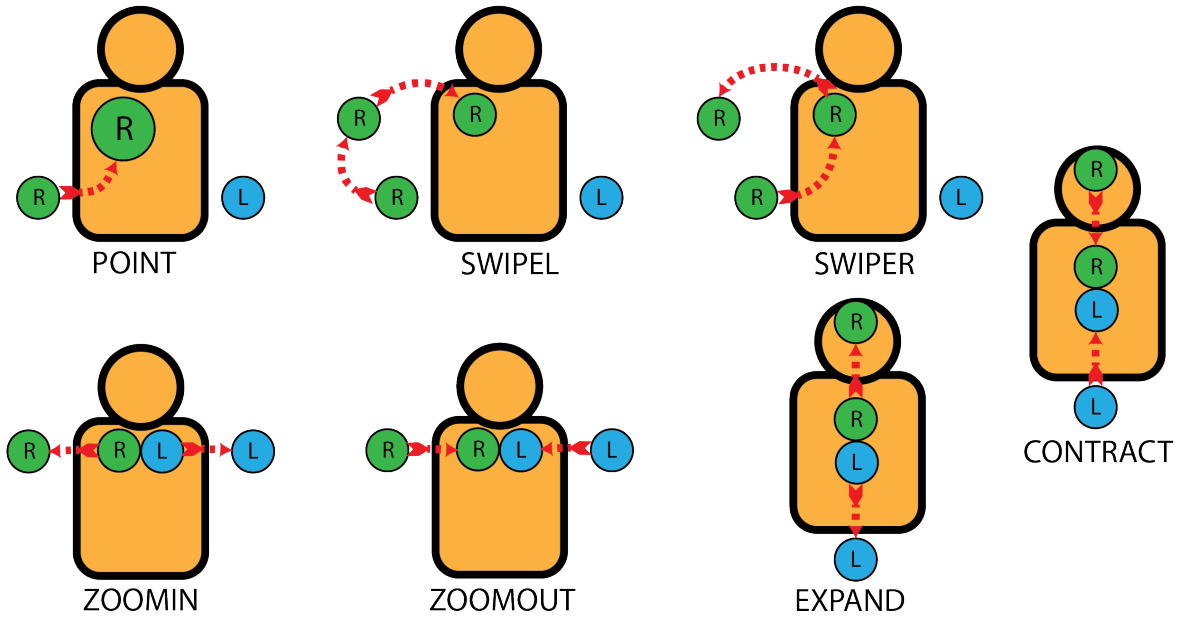


Figure 30: Gestures supported by our chirocentric user interface prototype. Abstract frontal view of the user, with his right and left hands represented by the green and blue spheres. The red arrows illustrate the hands' trajectories during the gesture progression.

These gestures are schematically visualized in Fig. 30. We also captured several seconds of each user idling in a resting state (termed **NEUTRAL**). It is worth noting that, depending on the user's natural preference, some of these gestures can be performed with either the left or right hand taking preference. For example, **CONTRACT** can be performed with the user's right hand being on top as both hands converge, but the same gesture can be performed with the left hand being on top instead. During our capture session, we specially directed subjects to perform the gestures in both ways. The resulting data was then merged under a single label during training/classification. In total, we captured data from 9 subjects, which each subject repeating the gesture 5 times. The data was segmented manually to remove the downtime between each gesture repetition.

Additionally, we gathered a second dataset of hand poses, for the purpose of evaluating our respective recognition algorithm. This is comprised of marker clouds and rigid body positions, as delivered by the tracking system, of 5 users holding their hands in the requested poses over approximately 5 seconds. We captured data for both the right and left hands, as the rigid body marker arrangements differ for each tracked glove. The captured poses are:

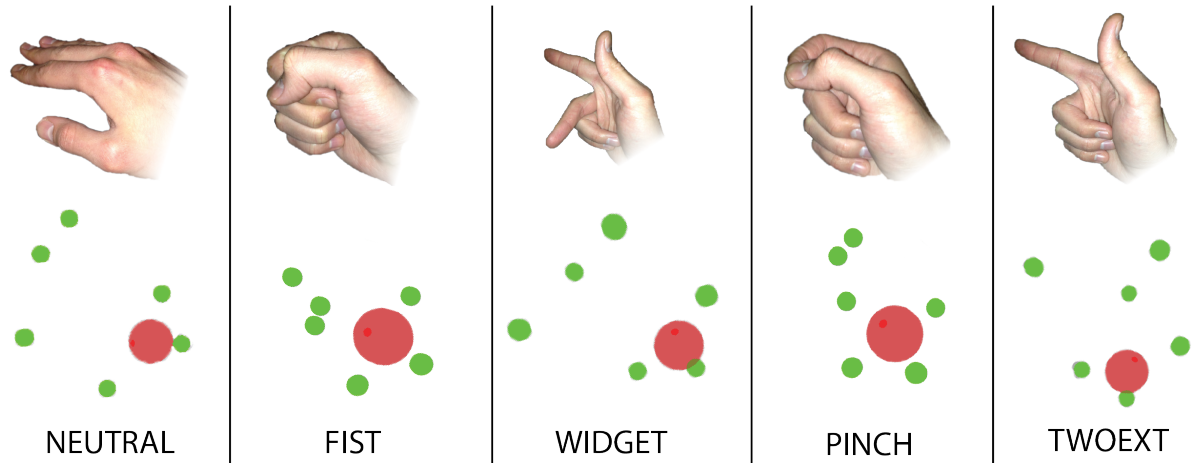


Figure 31: Hand poses supported by our chirocentric user interface prototype. Top Row: Photographs of the poses performed by a user (without wearing the tracked glove). Middle row: Equivalent tracking system data. The green spheres are markers and the single red sphere reports the rigid body position.

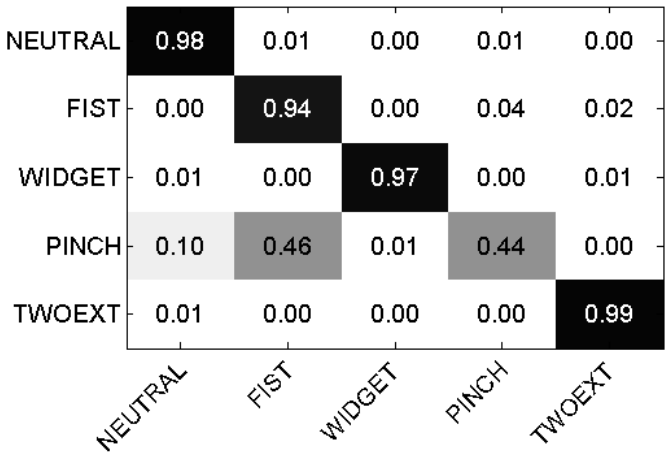
- Neutral (**NEUTRAL**) - The user's hand is resting naturally.
- Fist (**FIST**) - The subject's hand is clenched into a fist.
- Widget (**WIDGET**) - The thumb, index and middle fingers are arranged perpendicularly to each other, resembling a translation widget from 3D modeling software.
- Pinch (**PINCH**) - The subject pinches with his thumb and index finger.
- Two fingers extended (**TWOEXT**) - The subject's index and thumb fingers are extended while her middle finger is clenched.

Samples of the captured hand poses are visualized in Fig. 31.

6.3.2 Algorithm Performance

Hand Pose Recognition We evaluated our hand pose recognition algorithm via leave-one-out cross validation on our data set (training on data from 4 subjects and testing on 1). Our training and test sets are comprised of frames of frames of marker cloud and rigid body data, which are labeled for each particular gesture. In each iteration, the training set is filtered to only incorporate "valid" poses that did not require the insertion of placeholders during the execution of the $FILTERMARKERS$ step of our algorithm.

Figure 32: Average confusion matrix for our hand pose recognition technique. The misclassification of **PINCH** is the only outlier and we attribute it to unreliable training data from a single subject. Excluding that subject from the cross-validation process boosts the accuracy to 89.9%.



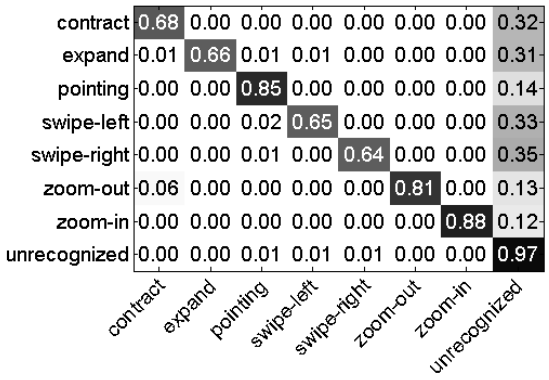
The test set however is used directly as provided by the tracking system. We report the average recognition performance and standard deviations for our algorithm under this test setting. Specifically, our technique achieved an average precision of 94.9% (±7%) and an average recall of 92.2% (±17%). By inspecting the average confusion matrix (see Fig. 32) we can observe that the algorithm can clearly differentiate between the most hand poses with an accuracy of 95% or more. The only exception is **PINCH** which gets misclassified as **NEUTRAL** or **FIST** approximately 24% of the time. By more closely examining the cross-validation results, we determined that the confusion with **NEUTRAL** was localized to a particular iteration (with **PINCH** getting classified as **NEUTRAL** approximately 80% of the time) which led us to believe that the underlying data for that subject session was unreliable. Indeed, excluding the subject in question from the cross validation resulted in an improved recognition accuracy for **PINCH** of 89.9%. As we will discuss below, even the full data set provided sufficient accuracy for the development of a chirocentric user interface.

Gesture Recognition An important parameter of our hand gesture recognition algorithm is the size of window W over which the feature calculation occurs. We experimentally determined the window size by evaluating a number of candidate values and

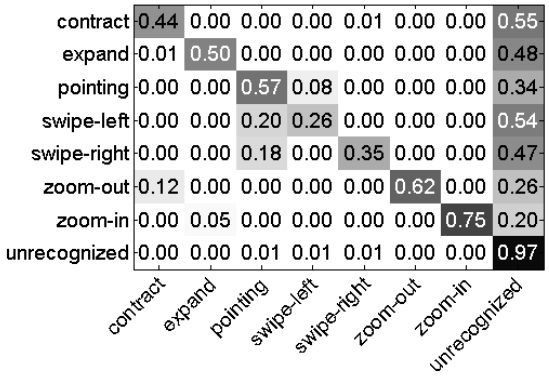
inspecting precision and recall values. This evaluation occurred over a 9-fold cross validation over our gesture data set and the results are reported in Table 1. The joint motion feature was competitive in terms of recall with the combined feature for $W = 10$ and $W = 13$. The combined feature demonstrated smaller variability in the reported results and was thus used in our implementation. Small window sizes reduced the efficacy of the joint motion feature, as they do not allow it to capture the underlying dynamics of the gesture. However, increasing the window size past 20 frames results in potentially multiple gestures being captured within it, reducing recognition accuracy. Based on our benchmarks, we selected a window size of 13 for our system.

Additionally, we evaluated the effect of the joint feature vector on the recognition process. We performed the cross-validation process by utilizing the joint distance and joint motion features separately (in addition to the combined feature). This evaluation exposed a number of interesting findings. Using solely joint distance feature resulted in low performance for smaller window sizes ($W = 5$ and $W = 10$). Joint motion exhibited similar results for $W = 5$ but was competitive with the combined feature in terms of precision for $W = 10$ and $W = 13$. The combined feature vector offered better or equivalent precision to the singular approaches at all window sizes, provided a higher recall % compared to joint motion for $W = 13$ and demonstrated smaller variability between the cross-validation iterations. These results are summarized in Table 1. Additional motivation can be found in the confusion matrices generated by our evaluation. Specifically, we notice that by using the joint distance feature on its own, we observe significant confusion between **SWIPEL** and **POINTING** and a similar result between **SWIPER** and **POINTING**. This is due to the fact that the progression of distances between the user's static hand and head and his gesturing hand is very similar in these two gestures. Utilizing the joint distance feature resolves this ambiguity and the combined feature provides an increase in accuracy. These results are summarized in Fig. 33. Since the overhead of computing the combined feature is not substantial when compared to only computing the joint-motion feature, we opted to use the higher dimensionality vector for our system.

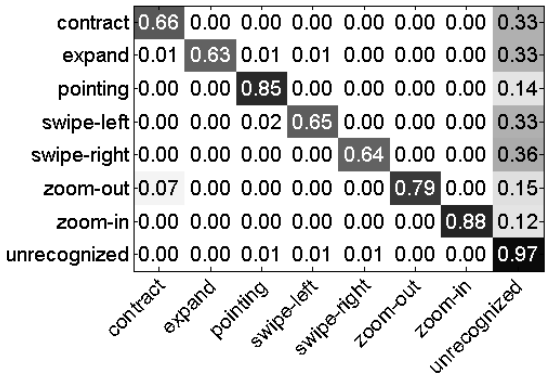
Figure 33: Confusion matrices for our gesture recognition technique. Results are presented for all feature combinations, with W=13. Using only the joint distance feature results in significant confusion between **SWIPEL/SWIPER** and **POINT**. The joint motion feature captures the underlying difference between the gestures and resolves the confusion. Using the combined feature yields an additional increase in accuracy with minor computational overhead.



(a) All features



(b) Joint distance



(c) Joint motion

Table 1: Summary of the performance of our gesture recognition algorithm. The results are presented for different frame window sizes (column marked W). Utilizing only the joint distance feature results in low performance for windows of 5 and 10 frames.

Features	W	Average precision (%)	Average recall (%)
Joint distance	5	43.49 ± 13.40	30.64 ± 10.44
Joint motion	5	39.09 ± 15.21	29.24 ± 10.39
All features	5	57.68 ± 14.76	46.00 ± 12.83
Joint distance	10	56.45 ± 14.41	43.10 ± 14.75
Joint motion	10	77.42 ± 7.39	69.80 ± 14.80
All features	10	77.99 ± 6.60	70.79 ± 14.31
Joint distance	13	79.14 ± 7.52	72.57 ± 13.17
Joint motion	13	80.59 ± 7.74	65.97 ± 14.34
All features	13	79.62 ± 6.81	73.60 ± 12.97

6.4 A Practical Chirocentric User Interface

We demonstrate our chirocentric user interface platform by implementing a bimanual 3DUI for control of visualizations on the Reality Deck. For more information on the Reality Deck and the underlying software framework we refer the reader to Chapter 3 and Chapter 4. Fig. 34 shows a user utilizing our prototype for interaction with a GIS application within our facility.

6.4.1 Visualization platform and applications

Arguably, the most important aspect of interacting with a visualization is the ability to manipulate the virtual camera, allowing for new views into the data. In the case of a first person view into 3D data, this implies the ability to precisely translate the camera, and also rotate on one or more axes (with control over the camera's yaw being the most important). If an arcball style camera manipulator is used, then the ability to independently control rotation and zoom variables is a useful modality. Generally, a 3D navigation interface should expose the highest number of Degrees of Freedom (DoFs) possible, allowing powerful manipulations, without overwhelming the user.

For two dimensional data, the range of DoFs that must be exposed decreases, as the possible transformations are limited to translations along the 2D plane, zooming and a single degree of rotational freedom. This frees up a number of interaction triggers, than

can be used to expose more delicate modes of camera manipulation. Additionally, 2D data (particularly maps) is frequently consumed by users on tablet devices and explored via multitouch Rotate-Scale-Translate gestures. Thus, a chirocentric UI targeted at this application domain should expose these features.

Such 2D and 3D camera manipulations are traditionally mapped to the translation of a physical input device (such as a mouse), with buttons acting as modifiers and affecting the active axes of manipulation. In our system, these physical modifiers are instead replaced by the user's hand poses, which place the system into a particular interaction state. Within that state, the user's hand motions are either directly applied to the virtual camera, or act as relative transforms that continuously affect the visualization until the interaction modality is terminated by the hands returning to their resting state.

Obviously, interactivity with visualizations is not limited to camera manipulations. For example, in a 3D visualization the ability to manipulate rendering parameters (e.g., toggling from solid surface rendering to a wireframe representation) is important and should be readily exposed. Similarly, in the case of 2D GIS data, the ability to rapidly cycle between locations of interest or traverse map zoom levels is very useful. These interactions are traditionally triggered via a selection on a graphical user interface (either on top of the visualization or on a second screen). Alternatively, they can be mapped to buttons on a controller device. In our system, such actions are activated directly through our chirocentric UI, with a gesture serving as the trigger.

6.4.2 Supported Interactions

Our prototype supports a number of chirocentric camera manipulations. In describing these interactions, we use the notation $\mathbf{P_R^t}$ and $\mathbf{P_L^t}$ for the positions of the user's right and left hand at time t as reported by our tracking system. Some manipulations assign certain actions to particular hand. This assignment can be changed at runtime based on user preference. In the descriptions below, we assume that the right hand is set as the "primary" hand. Additionally, to simplify the descriptions, we assume that the user is always oriented towards the front wall of the immersive system. Thus, the physical coordinate system in which positional data is reported by the tracking system matches the virtual camera's coordinate space. In the general case, a change of basis would have to occur, based on the user's orientation within the visualization space.

For 3D data, the primary forms of interaction are:

Unimanual 3 DoF Continuous Translation The system enters this mode when it detects the **WIDGET** pose on the primary hand. It sets $\mathbf{P_{Center}} = \mathbf{P_R^{t_0}}$. In subsequent frames $t^{'}$, it calculates $\tilde{\mathbf{v}} = \mathbf{P_R^{t^{'}}} - \mathbf{P_{Center}}$. $\tilde{\mathbf{v}}$ is then applied as a translation vector to the virtual camera, translating it in 3D space. For example, if the hand is offset upwards from $\mathbf{P_{Center}}$, the camera is continuously translated along its vertical axis. A small amount of tresholding is applied (approximately $2cm$) to ensure that natural motion while the user is at rest does not trigger an unintended interaction. This interaction mode continues until a pose other than **WIDGET** is detected on the primary hand.

Bimanual 4 DoF Continuous Translation and Rotation The system enters this mode when it detects the **WIDGET** pose on both hands. A continuous translation is mapped to the right hand, in the same way as described above. Additionally, the system sets $\mathbf{P_{Center}^L} = \mathbf{P_L^{t_0}}$ upon entering the state. For following frames, the system calculates $\tilde{\mathbf{v}} = \mathbf{P_L^{t^{'}}} - \mathbf{P_{Center}^L}$. The x (horizontal) component of $\tilde{\mathbf{v}}$ is used to determine a rotation, which is continuously applied to the camera. This allows control of the camera's yaw, using the secondary hand. The system remains in this state until a pose other than **WIDGET** is detected on either hand. If **WIDGET** is maintained on the right hand while the left hand assumes **NEUTRAL**, then the system reverts to the above modality without resetting $\mathbf{P_{Center}^R}$.

Unimanual Continuous Flythrough This mode is triggered once **TWOEXT** is detected on the user's right hand. From then on, at every frame t, $\mathbf{P_R^t}$ is used along with the hand's orientation information to define a pointing direction $\tilde{\mathbf{p}}_{\mathbf{physical}}$ within the virtual environment (this is one of a variety of ways to determine a user's intended pointing direction within the visualization space). $\tilde{\mathbf{p}}_{\mathbf{physical}}$ is then transformed to the 3D scene's coordinate system, yielding $\tilde{\mathbf{p}}_{\mathbf{virtual}}$ which is then applied as a per-frame translation to the virtual camera's position.

For our 2D GIS application, we expose the following functionality:

Unimanual Directly Manipulative Translation When the user's right hand is in the **FIST** pose, this mode is entered. Upon entry at time t, the system stores $\mathbf{P_R^{prev}} = \mathbf{P_R^t}$. At each subsequent frame $t^{'}$, the system calculates $\tilde{\mathbf{v}} = \mathbf{P_R^{t^{'}}} - \mathbf{P_R^{prev}}$. The x and y components of $\tilde{\mathbf{v}}$ are then applied as a translation to the virtual camera, manipulating its position on the 2D plane (the z component is unused). Following the manipulation, the system updates $\mathbf{P_R^{prev}} = \mathbf{P_R^{t^{'}}}$. This process continues for as long as **FIST** is maintained.

Bimanual Rotate-Scale-Translate This mode (which has also been referred to as "air multitouch" by some of our users) in triggered when both hands are in the **FIST** pose. At each incoming frame t, we calculate $\tilde{\mathbf{diff}}^{\mathbf{t}} = \mathbf{P_R^t} - \mathbf{P_L^t}$, $\mathbf{M^t} = \tilde{\mathbf{diff}}^{\mathbf{t}}/2$ and the between-hand distance $d^t = \|\tilde{\mathbf{diff}}^{\mathbf{t}}\|$. Based on these values and the previous frame's data, we can then define a translation vector $\tilde{\mathbf{t}} = \mathbf{M^t} - \mathbf{M^{t-1}}$ which is used to translate the virtual camera. Additionally, we calculate a scale factor $z = d^t/d^{t-1}$ which is applied to the current zoom factor. Finally, a rotation value ϕ is applied to the camera based on the angle between $\tilde{\mathbf{diff}}^{\mathbf{t}}$ and $\tilde{\mathbf{diff}}^{\mathbf{t-1}}$. Effectively, our system mirrors traditional multitouch functionality. This manipulation continues until either hand exits the **FIST** pose. If the primary hand remains in **FIST**, the system transitions to the unimanual directly manipulative translation mode instead.

Unimanual Continuous Translation and Zoom This interaction mode is similar to the Unimanual 3 DoF Continuous Translation for 3D scenes that we described earlier. However, instead of directly applying $\tilde{\mathbf{v}}$ to translate the camera position, only its x and y components are used to translate the camera along the 2D plane, while the z component is scaled and applied as an offset to the current zoom factor. Effectively, forwards/backwards offsets of the user's hand result in zooming in and out respectively.

Additionally, we correlate gestures to certain application-specific actions. For example, in our GIS viewer, the **SWIPEL** and **SWIPER** gestures are used to sequentially cycle between a list of predetermined points of interest. **ZOOMIN** and **ZOOMOUT** allow the user to instantly increment or decrement the current zoom level. **EXPAND** minimizes the zoom setting, providing the user for a view of the entire world, while **CONTRACT** zooms to the highest available level for the particular region (while maintaining the camera's longitude and latitude coordinates).

6.4.3 Implementation

Our chirocentric user prototype is implemented in C++ and integrated with the visualization stack described in Chapter 4. We utilize the NatNet API for the streaming of rigid body and marker cloud data from the OptiTrack system controller over standard TCP/IP networking. We utilize libSVM [CL11] for quering our trained models. For hand pose recognition, we aggregate the predicted labels over a 10 frame pose window and utilize a voting scheme to determine the current active pose. This approach improves interaction reliability (as singular outlier labels do not disturb an ongoing manipulation) although it does introduce a small amount of input lag before a hand pose is detected (approximately 0.04 seconds, or enough time for 5 or more slots of the pose window to be occupied by the hand pose label at $120Hz$). The gesture detection algorithm is only active while both of the user's hands are in the **NEUTRAL** pose, in order to avoid intended actions due to hand motions that naturally occur during other manipulations. Finally, we impose a 2 second cool down period when a gesture is detected and before another gesture can be recognized by the system. A view from within the Reality Deck, illustrating our chirocentric user interface in operation can be seen in Fig. 34.

6.4.4 Observations from Deployment

We report a number of anecdotal findings that arose from the utilization of our UI prototype internally, as well as a by small number of external users. In 2D exploration scenarios, users were able to accurately navigate, using both the unimanual and bimanual directly manipulative modalities. In a way, these modalities are direct mappings of traditional single and multitouch interactions on modern tablets, making users more likely to be familiar with their operation. However, we received commentary that, for long exploration sessions (or when the traversal of a large amount of virtual space is required), these two modalities can impose additional user fatigue, as they demand multiple repeating arm motions. We reached the same conclusion early in the design process, which was one of the drivers for the addition of the unimanual continuous translation and zoom modality. Here users can just determine the direction and speed of translation by offsetting their hand, and the camera manipulation continues until they return to the **NEUTRAL** pose. Effectively, there exists a precision-versus-comfort tradeoff between these two modes of manipulation. Arguably, the comfort level for the **FIST** based manipulations can also be

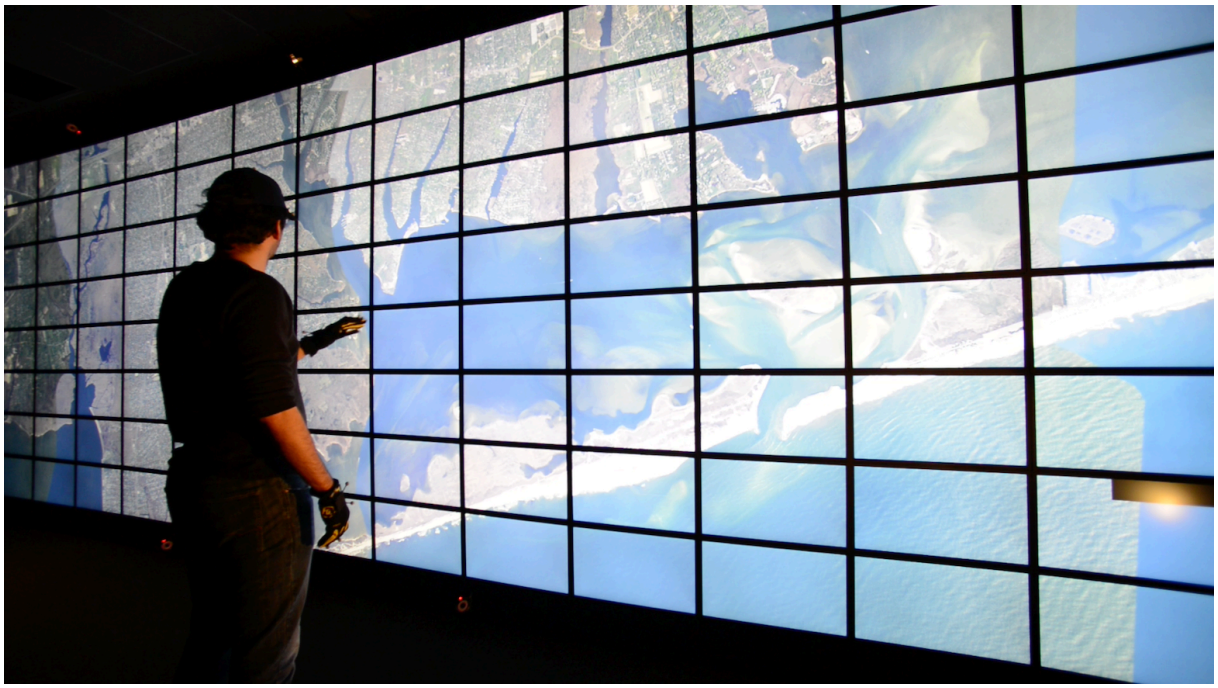


Figure 34: Photograph of a user leveraging our chirocentric user interface. The user is exploring a 2D GIS dataset in the Reality Deck. The rendering is generated with our VL-based renderer module.

improved by implementing support for inertial camera manipulations in our visualization system.

For 3D navigation, our system exposes a powerful tool in the form of the bimanual translation and rotation feature. Effectively, it provides a total of 4 navigational DoFs, without a dedicated controller device. More experienced users were able to perform complex maneuvers within and around 3D structures with ease. For non-experts, this type of manipulation proved somewhat unwieldy, but we hypothesize that this may be related to a general lack of familiarity with 3D navigation in general. Originally, we attempted exposing additional degrees of rotation (camera pitch and roll) through this modality, but they proved to be overwhelming for almost all users. The continuous flythrough modality was found very intuitive to use, but it is naturally somewhat constrained in its functionality (particularly if the virtual environment is not fully immersive).

A point of contention is the selection of support gestures and hand-poses that can be recognized by the system. In our current implementation, various interactions were assigned to hand-poses and gestures somewhat arbitrarily. While some of these assignments make

sense (for example the **FIST** pose, similar to a "grabbing" movement, triggering a direct manipulation) others may not (a vertical "zoom-out" gesture minimizing the zoom scale). Nielsen et al. [NSMG04] and several other scholars can provide guidance on this front when developing further UI prototypes. Additionally, the notion of frames of reference is extremely important, particularly in an immersive setting. Our existing implementation assumes that the user is aligned to the front wall of the facility. Consequently, interactions along the axes of the virtual camera map to hand motions along the width, depth and height of the physical space. In a practical setting, this assumption may not hold, since users can physically navigate and interact with the display from any point and with any body and head orientation. Consequently, the mapping between the physical interaction space, the visual feedback space and any manipulations is not well-defined for some modalities.

Our initial observations place the gesture recognition quality on-par with our reported cross validation figures, meaning that users occasionally had to repeat a gesture before it would recognized by the system. However, these gestures proved to be a useful tool for rapidly accessing functionality from anywhere within the virtual environment, that would otherwise require that the user either carry a controller device or physically move to a controller computer. Finally, early usage of our system underlined the need for some sort of feedback to the user upon entering and exiting a particular interaction mode. This is due to the fact that rare hand pose misclassifications can lead to occasional unintended interactions and disturb the user's view into the data. Since our gloves are passively tracked and contain no electronics, tactile feedback is not an option. In future versions of the system, we plan on incorporating some visual feedback, informing the user that their intended mode of interaction has been detected by the system. Overall however this chirocentric approach to interfacing with a visualization works much more robustly than NuNav3D.

6.5 Conclusion

This section described the development of a practical chirocentric UI platform, to be used for the development of bimanual 3DUIs. Our platform is based on two algorithms for the recognition of hand poses and hand gestures, the two main pillars of a chirocentric

user experience. They are targeted at data provided by commercial IR tracking systems, which are usually found in immersive environments such as CAVEs and tiled displays. Using this platform, we developed a prototype 3DUI that exposes a number of uni- and bimanual modalities for the navigation of 2D and 3D data and a set of hand gestures as triggers for various effects on the visualization.

Chapter 7

Acuity-driven Gigapixel Visualization

7.1 Introduction

Recent advances in data acquisition, display and rendering technology have brought to the masses the ability to interact with extremely high resolution data. For instance, the Gigapan project[1] allows any person with a web browser to explore images that span hundreds of gigapixels. While these vistas are composed over a period of time using offline stiching, research projects are yielding devices (such as parallel arrays of micro-cameras [BGS+12, CMN11, MST+11]) that can capture gigapixel images with a wide Field-of-View (FoV). Simultaneously, devices for visualizing such high resolution data are also expanding in size and fidelity. Large-format displays or Powerwalls are a traditional visualization platform for scientific and industrial applications. While the original Powerwall only spanned a few megapixels in size, modern systems of this kind, that maintain a planar form-factor, can expand into the hundreds of megapixels. In 2012, the Reality Deck [PPKM15] broke the gigapixel resolution barrier for large-format displays by utilizing 416 LCD panels in an immersive setting, providing a 360° horizontal FoV. Visually exploring data within such systems has a number of advantages. Such facilities have been shown to greatly improve performance in basic visualization tasks ([BN05, YHN07] and many others). These benefits largely stem from the ability of users

[1] www.gigapan.org

to naturally navigate through the data by physically moving within the space of the system. It has been shown that, when given an option between virtual and physical navigation, users will generally prefer to move to different areas of the display rather than manipulate the virtual camera [BN07].

An obvious side effect of the physical navigation in front (or within) a high resolution immersive system is the fact that at any point in time, a user may be at an optimal distance from some display surfaces but at a suboptimal distance from others. This means that the user may not be able to perceive the full amount of detail delivered by all displays. For gigapixel data this implies a waste in terms of computational resources and network bandwidth as the visualization system has loaded and rendered a version of the data that can not be perceived by the user due to her physical location. When integrated over the area of a large immersive system and over time, this waste is significant and can actually hamper certain use-cases (for example dynamically updating data or video that would result in local caches being invalidated at every frame). We can also consider the case where an F+C lens is applied on the visualization. If the rendering system is oblivious to the user's position within the visualization space, it will strive for maximum visual fidelity in the lens application, when a lower quality approximation would have yielded the same effect to the user and provided performance gains.

In this chapter we propose optimizing the visualization process of gigapixel data by smoothly degrading the quality of the visuals based on the user's physical position in relation to the display surfaces. We apply this concept to two different aspects of the visualization pipeline:

- Rendering of gigapixel images in an out-of-core fashion using virtual texturing.
- Application of displacement-based F+C zoom lenses (based on Carpendale's Elastic Presentation Framework [CM01]) with enhanced visual quality through GPU-based tessellation in an effort to ameliorate visual artifacts such as those seen in Fig. 35.

We term our approach *acuity-driven gigapixel visualization* [PK13b] since the rendering optimizations are guided by the analytical formulation for *visual acuity*. We posit that, due to the physiological basis of this formulation (based on the spacing of photoreceptors on a person's retina), the visual quality degradation will not be noticeable to the majority users of the visualization system and will not affect performance in various visualization tasks. We test our assumption via a user study, carried out by deploying our visualization

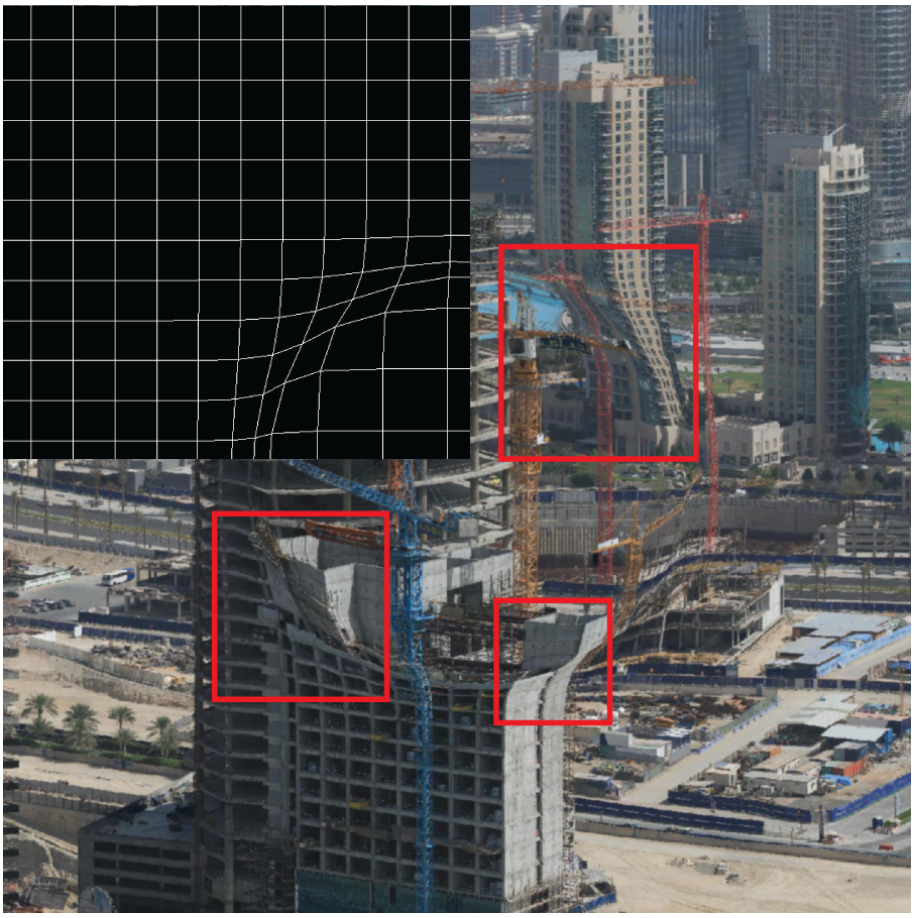


Figure 35: Motivation for acuity-driven F+C lens tessellation. Without adaptive tessellation, the geometry that an image is mapped on may not be able to capture the smoothness of the F+C lens, particularly in the transition region. The red boxes annotate some of the visible rendering artifacts. Top left inset: Wireframe of the underlying geometry after lens application. The gigapixel photograph of Dubai used in this figure is available online[2].

framework in the Reality Deck. Additionally, utilizing the tracking data gathered from our study (positions of users over time) we create a number of synthetic usage scenarios, based on which we conduct a performance evaluation of our method. Finally, we discuss insights gained from our user study and potential visualization scenarios enabled by our acuity-driven framework.

[2] http://www.gigapan.com/gigapans/75554

7.2 Acuity-driven LoD Selection

A virtual texture pipeline maintains several sub-sampled levels of the original texture data. When texture mapping a particular pixel, the system must determine which LoD is appropriate given the screen-space impact of the geometry. On a GPU virtual texture implementation, this determination is performed within a shader using local data, in a way that is similar to traditional texture mipmapping [Wil83]. Briefly, the virtual texture pipeline makes the determination based on the image-plane spatial derivatives of the texture coordinates $[u, v]$:

$$[\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}] \text{ and } [\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}]$$

These spatial derivatives can be used to estimate the size d of the pixel projection into texture space, based on Heckbert's heuristic [Hec83]:

$$A = \sqrt{max([\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}] \cdot [\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}])}$$

By estimating the subdivision level that reduces this texture space projection to approximately 1 texel, one can determine the appropriate MIP level (with 0 being the original, non-downsampled texture data):

$$m_{MIP} = log_2(A)$$

This MIP level m is a factor of the display resolution (as lower resolution implies larger texture coordinate spatial derivatives) and the orientation of the geometry with respect to the virtual camera. It does not take into account the visual information delivered to the user, based on her spatial relationship to the display. For example, a $2048px \times 2048px$ texture is mapped on a quadrilateral that is aligned precisely with the boundaries of an imaginary $2048px \times 2048px$ resolution display of 20" diagonal. For this display, $D_{opt} = 23.74$". Based on the standard virtual texture LoD determination, for this configuration $m = 0$ and the highest resolution is accessed for each pixel. For a user standing at a distance $\leq D_{opt}$ the entirety of this detail is perceivable. However, as the user moves away from the display, the visual angle between individual pixels crosses the $\frac{1}{60}^\circ$ threshold

and they (and their underlying texture detail) become increasingly indiscernible.

Our acuity-driven gigapixel visualization framework is based on the assumption that, as the visual angle of the pixel on the user's retina gets smaller, so does its effective projection within the texture space. More concretely, for a certain display configuration (with dot pitch P) we can define a visual angle $V_{opt} = \frac{P}{D_{opt}}$. Now assume that the user is at distance D' from the display, then the visual angle of a pixel is $V' = \frac{P}{D'}$ or $V' = \frac{D_{opt}}{D'}V$. We scale the texture-space projection A of each pixel $A' = \frac{D'}{D_{opt}}A$. The intuition behind this is that as the user moves away from the screen and neighboring pixels begin to overlap on his retina, texture coverage from both pixels should be considered, thus the texture-space projection of the pixel should be expanded. Having already determined m_{MIP}, we calculate a secondary mipmap level m_{acuity} as follows:

$$\frac{1}{2^{m_{acuity}}} = \frac{D_{opt}}{D'} \Rightarrow 2^{m_{acuity}} = \frac{D'}{D_{opt}} \Rightarrow m_{acuity} = log_2(\frac{D'}{D_{opt}})$$

It is worth noting that for $D' \leq D_{opt}$, m_{MIP} should be selected (as there is no point in actually going "lower" in the LoD pyramid if the display/geometry arrangement can not deliver the detail). Thus,

$$m_{acuity} = max(0, log_2(\frac{D'}{D_{opt}}))$$

Additionally, this calculation can be biased towards quality by ensuring that the finest acuity-driven LoD is selected at each time by setting:

$$m_{acuity} = \lfloor max(0, log_2(\frac{D'}{D_{opt}}) \rfloor$$

Thus at distance D' the final LoD level m' is:

$$m' = m_{MIP} + m_{acuity}$$

7.3 Acuity-driven Tessellation for F+C Lenses

Applying an F+C lens onto an image under the EPF framework is tantamount to displacing the proxy geometry of the image along its normal based on the lens function. Under a rasterization environment, this implies that between individual vertices of the underlying geometry, the lens function is linearly approximated rather than accurately sampled. By naively increasing the granularity of the geometry, the lens function can be captured more accurately but at a fixed performance overhead and without taking into account the user's position in relation to the visualization.

Instead, we propose that the geometry can be adaptively tessellated by considering the following two conditions:

- The curvature of the F+C lens. For example, if the lens has a flat focal region, then tessellating its geometry would not yield a visible effect. On the other hand, it makes sense to more finely tessellate the transition region of the lens in an effort to avoid the visual artifacts seen in Fig. 35.
- The distance of the viewer from the physical display. Similarly motivated to the acuity-driven LoD selection, the tessellation can be set to a maximum available upper limit (user or hardware imposed) when the viewer is standing at a distance $\leq D_{opt}$ and reduced as she distances herself from the display.

In our acuity-driven visualization framework we calculate an adaptive parametrization for the gigapixel image geometry based on the above conditions. This parametrization is comprised of two factors: a lens-based tessellation metric and a view-based tessellation metric, which we describe below.

7.3.1 Lens-based Tessellation Metric

The goal of this method is to provide an adaptive tessellation metric that captures the curvature of the F+C lens function $\mathrm{F}_{lens}(x, y)$, parametrized over the image-plane x, y. Effectively, the tessellation should be maximized in areas that demonstrate high curvature variation (e.g. the transition regions of the F+C lens) but in the flat regions it should remain minimal (maintaining the complexity of the base mesh). Additionally, since the tessellation metric is calculated on the GPU and inside a tessellation shader (that only has

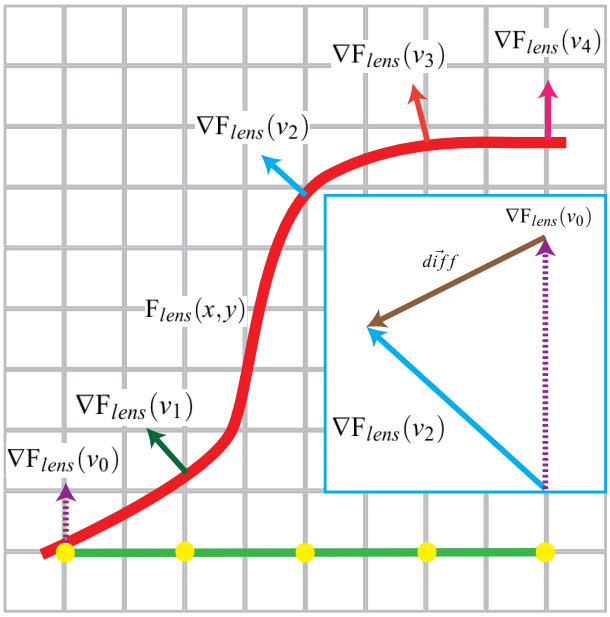


Figure 36: Schematic representation of lens-based tessellation factor calculation. Starting at vertex v_1 the algorithm calculates the difference between v_0 and other vertices along the edge at regular intervals. The inlet shows the resulting $\vec{diff}$ for this invocation of the algorithm. In this example, $\vec{diff}$ is defined using the the sampled F+C function gradients that show maximum divergence (in this case v_0 and v_2).

local knowledge of the geometry, meaning that it can only access the vertex information of the original, coarse, primitive), the metric calculation should be feasible without global knowledge of the structure of $F_{lens}(x, y)$.

Each edge e_i of the underlying geometry is sampled uniformly $k_{PerEdge}$ times at regular intervals. At each sampling point v_{cur}, we calculate the difference $\vec{diff}$ of the gradient of $F_{lens}(x, y)$ between v_{cur} and v_0 (the first vertex of the edge), effectively capturing the curvature variation along the edge. The length of the difference is normalized by the distance between the two vertices $|v_0^{Displ} - v_{cur}^{Displ}|$. The resulting tessellation factor is the maximum of this metric across e_i. The algorithmic process is outlined below:

Algorithm 2 For edge $e_i = [v_0, v_1]$:
$s_{lens} = \text{edgeLensTessellationFactor}(v_0, v_1, k_{SamplesPerEdge})$

$v_0^{Displ} := v_0 + \vec{n}_{v_0} * \mathrm{F}_{lens}(v_0)$
$result := 0$
$\vec{dir} = \|v_1 - v_0\|$
for $i = 1, i \rightarrow k_{SamplesPerEdge}$ **do**
  $v_{cur} := v_0 + i * step * \vec{dir}$
  $v_{cur}^{Displ} := v_{cur} + n_{v_{cur}} * \mathrm{F}_{lens}(v_{cur})$
  $dist := |v_0^{Displ} - v_{cur}^{Displ}|$
  $\vec{diff} := \nabla \mathrm{F}_{lens}(v_{cur}) - \nabla \mathrm{F}_{lens}(v_0)$
  if $\frac{|\vec{diff}|}{dist} > max$ **then**
    $result := \frac{|\vec{diff}|}{dist}$
  end if
end for
return $result$

In Algorithm 2, n_{v_i} is the normal vector for vertex v_i of the image proxy mesh (in our case a simple, 4 vertex quadrilateral). Effectively, $v_i^{Displ} := v_i + n_{v_i} * \mathrm{F}_{lens}(v_i)$ is the geometric displacement process that results in magnification under EPF. Algorithm 2 is schematically illustrated in Fig. 36.

7.3.2 View-based Tessellation

Practically, render-time tessellation of geometric primitives can be defined as the choice between a base mesh with subdivision s_0 and a maximum available tessellation s_{max}. An additional constraint is imposed by the size of the projection of the geometry on to the screen. Intuitively, edges at a distance from the virtual camera position need only be tessellated densely enough so that the resulting sub-edges have a desirable screen-space footprint. We term this amount of tessellation s_{view}. Of course, $s_{view} \leq s_{max}$.

In our acuity-driven gigapixel visualization framework, s_{view} represents the upper tessellation parametrization that is utilized when the user is at distance $\leq D_{opt}$ from the display. We scale this tessellation factor based on the user's actual distance D' from the display in order to define s_{acuity}:

$$s_{acuity} = \frac{D_{opt}}{max(D_{opt}, D')} * (s_{view} - s_0)$$

The impact of the view-dependent tessellation component on the parametric error can be see in Fig. 37.

7.3.3 Combined Metric

We can combine these two parametrization metrics in the following way to determine s_{final}:

$$s_{final} = s_{lens} * s_{acuity} + s_0$$

Effectively, s_{acuity} sets the upper tessellation bound for the parts of $\mathrm{F}_{lens}(x, y)$ that maximize the response of s_{lens}. For example, if the user is standing at D_{opt} from the display (and is thus maximizing s_{acuity}), the focal region of a flat-top F+C lens would still remain coarsely tessellated as s_{lens} would go to zero.

7.4 Implementation

We integrated our acuity-driven LoD selection with our gigapixel image renderer described in Chapter 3. Our system supports gigapixel data in tile form (usually $256px \times 256px$). Each image tile at full resolution is mapped onto a single quadrilateral. With this approach (rather than utilizing a single piece of proxy geometry for the entire gigapixel image), we can work around hardware-imposed GPU tessellation limits on individual geometric primitives. The polygon count for such an approach still remains low (e.g., a 4 gigapixel images requires approximately $61,000$ quadrilaterals). At the same time, it allows us to implement our tessellation technique entirely on the GPU. The virtual texture visibility determination pass is performed by rendering to an off-screen buffer at 25% the resolution of the display and streamed to the CPU for processing using a Pixel Buffer Object. Pages are loaded in order of "importance" (effectively tied to their screen-space coverage).

Our acuity-driven LoD selection scheme is implemented on the GPU. Since the LoD determination for virtual texturing happens on the per-pixel level, any shaders need to

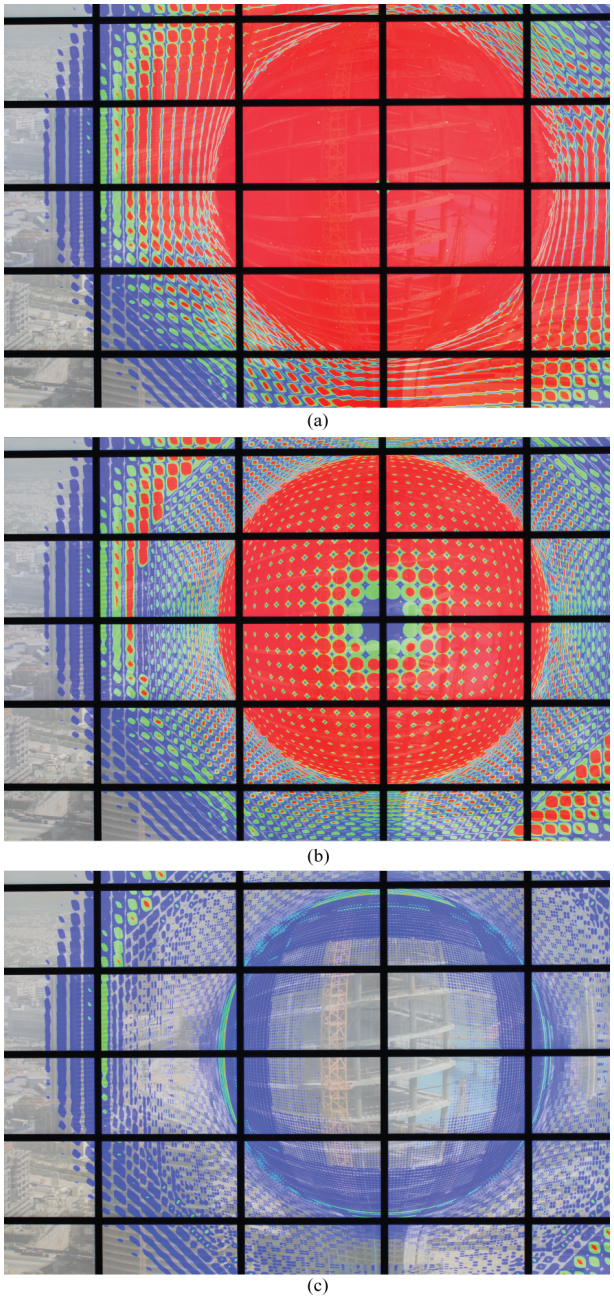


Figure 37: Parametric error for a Gaussian lens under different conditions. The lens is defined analytically allowing exact error metric calculation. Each pixel has a [grey, blue, green, red] hue if the error is $[< 0.1, < 0.5, < 1.0, \geq 1.0]$ pixels respectively. (a) A low-density static mesh. (b) Our acuity-driven tessellation algorithm at a distance of approximately $9'$ from the display. (c) Our proposed method at $D_{opt} = 31$". At D_{opt}, the parametric error for the majority of the lens is reduced to less than 0.5 pixel. The visible Moire pattern is the result of the parametric error varying along the new sub-edges, with it being minimal at the added vertices and slightly increasing along each edge.

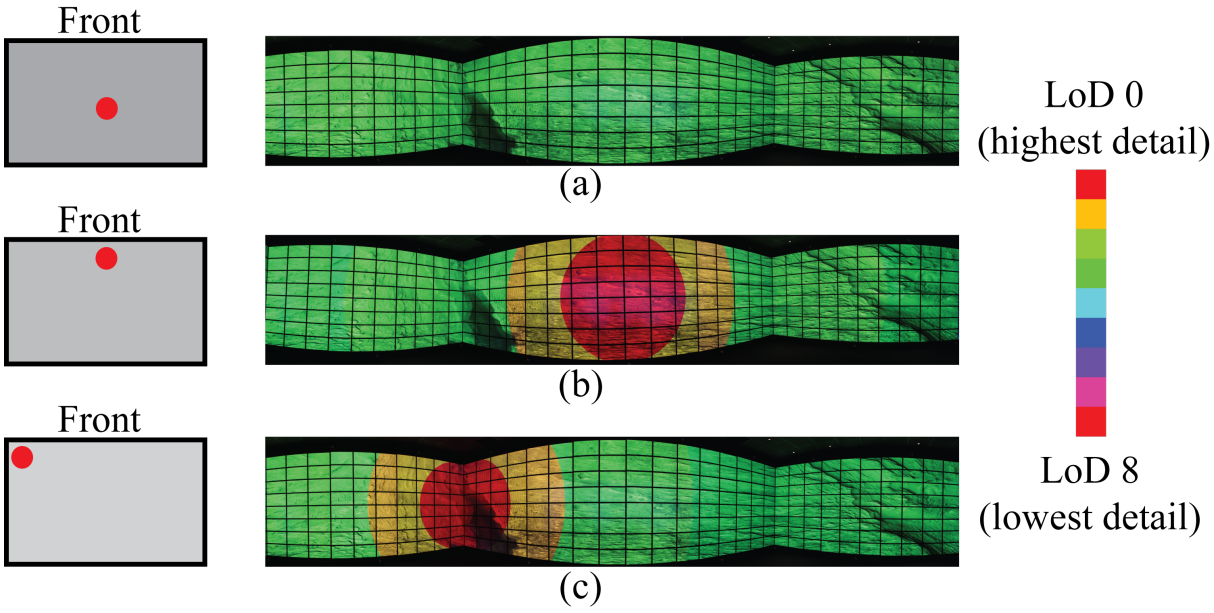


Figure 38: Results of the acuity-driven LoD selection scheme. Without acuity-driven visualization, the entire scene would be texture-mapped at LoD 0. However, using our acuity-driven gigapixel visualization framework, the system adaptively selects the LoD based on the viewer's position within the space. (a) Standing in the middle of the visualization space and based on the D_{opt} value of the display (≈ 31"), the system selects LoD 4 for the visualization. (b) The user comes within D_{opt} of the middle of the front wall, thus the LoD 0 is selected to offer the maximum amount of visual detail for that area. (c) The high detail area tracks the user as she moves to the front-left corner of the facility.

be able to calculate the position of each pixel within the physical space. Assuming that the canonical screen space position p_{css} for each pixel is known, then its physical position p_{phys} is:

$$p_{phys} = p_{screencorner} + (p_{css}.xy * 0.5 + 0.5) * d_{screen}$$

with d_{screen} being a two dimensional vector containing the display width and height and $p_{screencorner}$ being the physical position of the bottom left corner of the screen in the physical space. This calculation assumes that the display lays on the $x - y$ plane of the real world coordinate system and can be easily generalized for multiple screens that are not co-planar (as is the case for the Reality Deck, our testbed facility). The value of p_{phys} is calculated within the GLSL shaders that are involved in the visibility determination or rendering of the virtual texture. The screen position and size information as well as the user's head position are passed to the shaders though uniform variables. Our acuity-driven LoD calculation is combined with the MIP default, as described earlier. It is worth mentioning that in our implementation, we do not interpolate between adjacent LoD levels ($m^{'}$ is clamped to the most detailed level).

Our acuity-driven tessellation scheme for F+C lenses is also entirely implementable on the GPU. Our implementation of the EPL framework uses vertex shaders for displacement of the underlying mesh. In particular, we utilize OpenGL tessellation shaders for performing runtime subdivision of the proxy geometry for our gigapixel image. We calculate s_{acuity} for each edge of every quadrilateral of the base mesh via OpenGL geometry shaders. More specifically, we utilize s_{final} as the per-edge tessellation factor of the *Tessellation Control* shader. $\mathrm{F}_{lens}(x, y)$ can either be defined analytically within the shader program or it can be precomputed and stored in a lookup texture.

While framework can be used on a variety of visualization facilities with minor changes, ranging from a single desktop to a fully immersive setup. For the purposes of this contribution, we deployed our software on the Reality Deck, described earlier in this thesis in Chapter 3.

7.5 Results

In this section, we provide some results of our acuity-driven framework within the Reality Deck facility. First, we illustrate the expected behavior of our acuity-driven LoD scheme. The displays of the Reality Deck have a resolution of $2560px \times 1440px$ with a diagonal of approximately 27" (or $0.68m$") for a D_{opt} of approximately $0.78m$. Given that the Reality Deck is approximately $10.05m$ in width and $5.79m$ in depth, with the viewer in the very middle, we would expect to see a $log_2(\frac{10.05m}{2}/0.78m)$ bias in the LoD for the very middle of the left and right walls. Similarly, we would expect a $log_2(\frac{5.79m}{2}/0.78m)$ bias for the very middle of the front and back walls, which is added to the MIP-based value. Indeed, this effect materializes within the Reality Deck, as Fig. 38 shows.

Our acuity-driven tessellation algorithm experiences a linear behavior between distance from screen and resulting tessellation. Additionally, it accurately captures the underlying structure of the F+C lens that is applied to the gigapixel image, as seen in Fig. 39.

7.6 Evaluation

At its core, our gigapixel visualization framework aims to maximize overall system performance (by minimizing the overhead associated with transferring image data to the GPU and rendering high-fidelity F+C lenses). With regard to the LoD selection, it is crucial that our technique does not hamper the user's ability to visually explore the data efficiently. Consequently, we want to quantitatively compare our acuity-driven LoD scheme against a standard gigapixel visualization pipeline (with no LoD optimizations taking place, other than the usual MIP selection). More formally:

We hypothesize that the operation of our acuity-driven visualization framework will not hinder performance in search tasks on a large format display when compared to naive gigapixel rendering.

This hypothesis of equivalency is somewhat unconventional in the domain of visualization research, as most techniques are targeted at improving user performance versus some established baseline or state of the art. In our case, the benefit of our visualization framework lies in the improvements in system performance. The purpose of the qualitative evaluation is to quantify the effect of the acuity-driven LoD selection on visualization

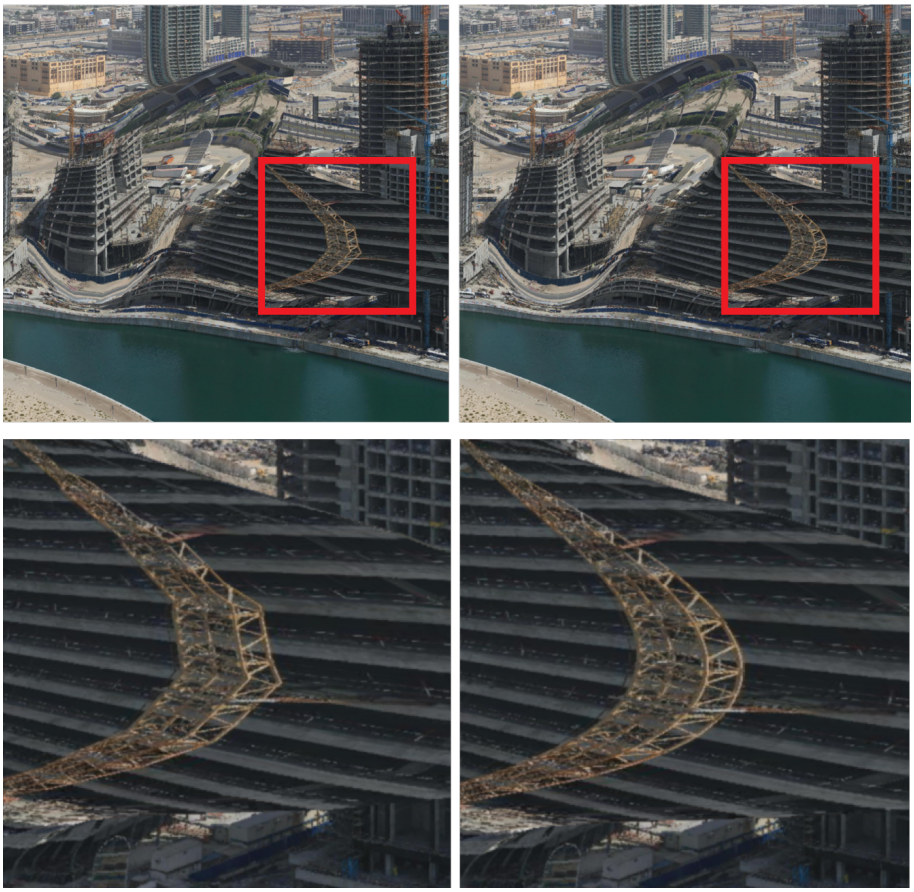


Figure 39: Off-axis rendering of 3 overlapping F+C lenses. Left column: Using a coarsely tessellated mesh. Right column: using our adaptive tessellation technique. Our method accurately captures the underlying lens function and eliminates rendering artifacts. The bottom row shows zoomed views of the areas annotated in red, illustrating the smooth deformation that is achievable with our method and the improvement in visual quality (especially visible along the body of the crane).

search tasks. A negative impact would indicate that utilizing this technique is *unsafe* as it would result in the user requiring additional time to complete visual search tasks or miss the intended targets altogether, both undesirable outcomes. Demonstrating the equivalency of our technique in terms of user performance to a baseline system will allow for its deployment without fear of negative impact on the visual exploration process, while simultaneously realizing the system performance benefits that we discuss later.

In this section we describe the design and results of a user study aimed at testing this hypothesis. Additionally, we discuss some qualitative comments from our users pertaining to the image quality of the adaptive LoD for gigapixel images and the adaptive tessellation for F+C lenses.

7.6.1 Study Design

Apparatus Our user study was conducted within the Reality Deck. Three columns of the rear wall remained inactive during the study. This reduced the effective resolution of the Reality Deck to approximately 1.4 gigapixels. We utilized a baseball cap with mounted IR retroreflective markers (defining a rigid body) for head tracking. The tracking volume of our OptiTrack system is large enough for reliable recognition close to the walls of the facility and past the D_{opt} of the monitors. A joypad controller was utilized for scene manipulation.

Participants We recruited 10 graduate and undergraduate students to participate in our study. Average age was 26 ($\sigma = 2.93$). Seven participants were male and 3 were female. Two had prior experience with the Reality Deck and scientific visualization in general. However, the task (described in the following paragraph) did not require any domain-specific knowledge and all users were given the opportunity to familiarize themselves with the Reality Deck facility prior to commencing the study. Additionally, the exploration of the task data sets during the quantitative portion of the study was conducted purely through physical navigation, eliminating any potential bias due to unfamiliar user interfaces. Consequently, we posit that this discrepancy in user familiarity with the visualization domain did not significantly affect our quantitative evaluation. Finally, all participants reported average $\frac{20}{20}$ vision either naturally or through corrective glasses or lenses.

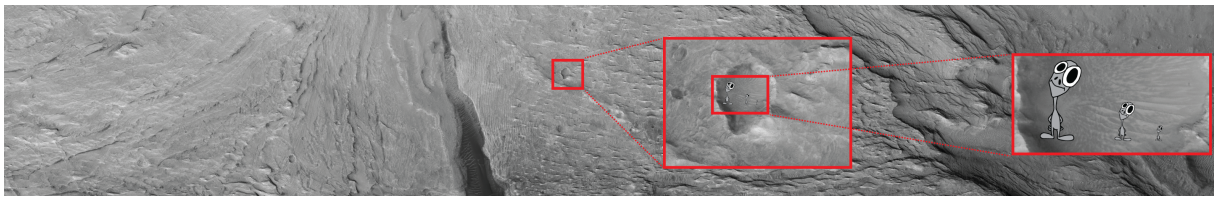


Figure 40: Cross-section of the HiRISE data set used during the user study. This section of the data was visible on the displays of the Reality Deck. The resolution is approximately $85K \times 14K$ pixels. The insets show progressively zoomed-in crops from the main view, illustrating the size of the targets used in our study in relation to the entire data set. The right-most inset shows three targets for the **-E**, **-M** and **-H** values of **DIFF** (respectively from left to right). The smallest target's dimensions are $25px \times 50px$.

Tasks and Procedure We wanted to evaluate the impact of our acuity driven gigapixel visualization (termed **ADGV**) on the data exploration process and compare it against the baseline "standard" gigapixel visualization (**SGV**). This variable is termed technique (**TECH**). We designed a number of search tasks targeted at testing our hypothesis within our gigapixel resolution system. We utilized the High Resolution Imaging Science Experiment[3] data set to generate a gigapixel resolution image of Mars topography. The original data spans approximately 1.8 gigapixels and is texture-mapped to an ellipsoid that surrounds the display volume of the Reality Deck (the resolution of the visible part of the data was approximately 1.2 gigapixels). This vast landscape serves as the background for our search task. Within this backdrop, we hide 3 targets that the study subjects must locate. Our tasks are split into 3 difficulty levels (**DIFF**): easy, medium and hard (denoted by the suffixes **-E**, **-M** and **-H**). The differentiator between the levels is the size of the targets in relation to the data set. In the **-E** level, targets span $256px \times 512px$, in **-M** their size is $128px \times 256px$ and in the **-H** condition they are reduced to $25px \times 50px$. Fig. 40 shows the data set used in the study and illustrates the scale of the targets in relation to the backdrop. The artwork used as a target in our study is available in the public domain. The targets were similar in hue and brightness to the background data set. For this task, we recorded Elapsed Time (**ET**) until discovery of all 3 targets. Furthermore, we captured positional information for the majority of the users for further analysis.

Additionally, we wished to evaluate the image quality provided by our F+C adaptive tessellation scheme. For this purpose, we utilize a second gigapixel photograph, this

[3] http://hirise.lpl.arizona.edu/

time from Dubai[4] . We placed an EPF-based F+C lens, aligned with the center of the view frustum and connected to the translation of the virtual camera. Subjects could translate along the $x-y$ image plane using the left stick of the joypad controller (rotation was fixed) and could alternate between our acuity-guided tessellation (**ADGV**), a coarse version of the image geometry (**NGV**) and a 16-fold pre-tessellated version of the same mesh (**PRE**). They were also free to move within the visualization space. This evaluation was qualitative and subjects were asked to comment on their method of preference in terms of image quality.

Upon arriving at the study location, the users received a brief description of the Reality Deck and its function. The mechanized door was closed and the users were allowed to familiarize themselves with the space. At this stage, the screens of the Reality Deck showed a static splash image. After being queried for their demographic information, the subjects were shown a picture of the survey target and instructed to look for 3 such targets in each task. They were also asked to wear the head tracking prop. This was a single-blind experiment and the subjects were not aware of the choice between **ADGV** or **SGV** (or even the fact that the LoD strategy was the between-subjects variable). Both the order of task difficulty (**-E**, **-M** or**-H**) and the order of method were determined using a Latin Square. For each difficulty, both methods were tested in succession and the user was queried on the quality of the last two visualizations in an effort to determine whether the presence of **ADGV** was noticeable. In summary, the design of the quantitative part study was 10 participants $\times$ 3 tasks $\times$ 2 methods = 60 interactions in total.

7.6.2 Results and Discussion

We began by examining the effect of **TECH** on **ET**. Student's t-test for each **DIFF** ($\alpha = 0.05$) did not show a significant effect (For **-E**, $t(9) = 0.867, p = 0.408$, for **-M**, $t(9) = 0.568, p = 0.584$ and for **-H**, $t(9) = 1.110, p = 0.296$). We performed between-**TECH** Two One Sided T-test (TOST) equivalence analysis for each **DIFF** with a threshold of 5% of mean **ET** but the evidence was not conclusive (for the sake of completeness, relevant t-statistics and p-values can be seen in Table 2). We also evaluated the percentage difference in performance of **ADGV** by considering **SGV** as the baseline performance (in order to normalize the performance metric between subjects). We applied Student's

[4]Available at https://www.gigapan.com/gigapans/75554.

t-test on this percentage metric, with an expected mean of 0. This approach also did not indicate a significant effect of **TECH** on **ET** (for **-E** $t(9) = 1.45, p = 0.089$, for **-M** $t(9) = 0.492, p = 0.317$ and for **-H** $t(9) = 0.016, p = 0.494$). While there is no evidence suggesting an effect of **TECH** on **ET** at any **DIFF**, we can not concretely claim that there is no statistically significant effect at our chosen threshold level.

We conducted post-hoc analysis on the positional tracking data captured during our user study. We focused, in particular, on the **-H** task as it would potentially require the most adjustment of the users' physical location in order to accommodate for the shifts in visual quality during the exploration process. We calculated $d_{closest}$, the distance of the user from the closest screen at any point during the exploration process. A TOST equivalence test of $d_{closest}$ indicated that there is no statistically significant effect of **TECH** on $d_{closest}$ at p-values 0.013 ($t(9) = 2.684$) and 0.012 ($p(9) = 2.726$) at 7% of the mean (approximately 5"). This leads to the conclusion that users did not have to significantly adjust their average distance from the display while exploring the data under **ADGV**.

Out of the 10 study participants, only 1 rated **SGV** as demonstrating superior image quality to **ADGV** during any of the tests (this user described herself as having experience with scientific visualization). Three users commented that under certain **DIFF** values, they actually found the image quality of **ADGV** to be superior (which is obviously impossible since **SGV** presents an upper bound to image quality). Under all other conditions, the subjects claimed to find no perceivable difference in image quality between **ADGV** and **SGV**. We interpret these findings as an indication that the effect of **TECH** on the perceived image quality is generally limited, at least under our test datasets.

Finally, we asked users to evaluate the image quality of F+C lens application. All study participants rated **ADGV** as being of equal visual quality to **PRE**. Interestingly, only 2 users (who identified themselves as having experience in scientific visualization) could easily pin-point the improvement in visual quality between **ADGV** and **NGV** (the smoother approximation of the F+C lens function). The rest of the subjects could detect a visual *difference*, but could not definitively say whether**ADGV** looked better than **NGV**. Some users characterized the **ADGV** visualization as "bulging" or "bowing out", a result of the image geometry following the smooth underlying function more closely. This finding raises some questions about the tangible effect of **ADGV** on the user experience for non-experts.

Table 2: Report of our TOST equivalence analysis for our user study. Our analysis compares between **ADGV** and **SGV** for each **DIFF** at $a = 0.05$. We did not observe an effect of **TECH** on **ET** for any value of **DIFF**.

	DIFF					
	-E		-M		-H	
	$\mu > \theta$	$\mu < -\theta$	$\mu > \theta$	$\mu < -\theta$	$\mu > \theta$	$\mu < -\theta$
$t(9)$	1.083	0.652	0.751	0.385	0.85	1.37
p	0.15	0.27	0.24	0.35	0.21	0.10
θ (s)	5.515		6.755		32.075	

7.6.3 Performance

Utilizing the motion capture data acquired during our user study, we created 3 synthetic use cases under which we evaluated the performance of our framework. We focused on the **-H** difficulty level since it yielded the longest overall user sessions. Based on the tracking data, we created a heatmap of the users' positions within the visualization space of the Reality Deck (at a resolution of 160 × 100 cells). We then drew 1500 random samples from the non-zero heatmap locations and performed k-means clustering on them to yield 6 "hotspots" of user presence. Given this list of hotspots:

1. We randomly select p_{start} from the list.
2. We also randomly select p_{next} from the list.
3. We traverse the path $p_{next} - p_{start}$ at a speed of 2.8mph at 120Hz
4. We remove p_{start} from the list, set $p_{start} = p_{next}$ and repeat from Step 2 until the list is empty.

We opted for this synthetic data set creation instead of utilizing prerecorded data directly from our users in order to reduce the sensitivity of our benchmarks to the search patterns of any one particular study subject. In total, we created 3 data sets (**A**, **B**, **C**), with duration 145, 216 and 175 seconds. For technical reasons, we had to limit our benchmarking to the front, left and right display surfaces of the Reality Deck (a total of 13 nodes and approximately 1.268 gigapixels of display space).

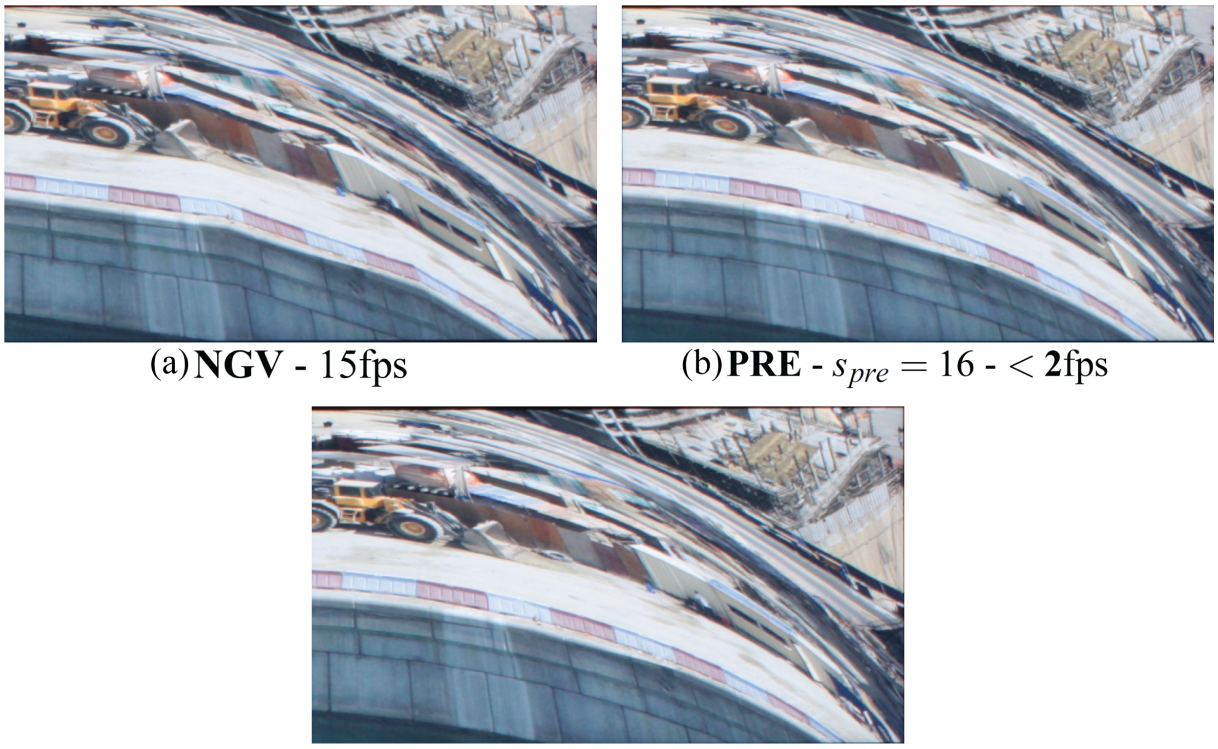
Using these synthetic user sessions, we evaluated two performance metrics, one for each aspect of our technique:

- *Data transfer overhead* - the number of virtual texture pages required to fully texture map the surface of the Reality Deck facility under **ADGV** and **SGV**. Since these pages ultimately need to be transfered from some remote source (long term storage or otherwise) to the render nodes and then to the GPUs, their count serves as a good indicator of system performance under different values of **TECH**.
- *Frame rate* - our acuity-driven tessellation framework is evaluated by measuring the cluster-wide frame rate of the visualization application while displaying an F+C lens under **ADGV** and **PRE** (with an $s_{pre} = 16$). For the sake of comparison, we also report the frame rate under **NGV** (which demonstrates significantly lower image quality).

The only differentiating factor between each benchmark was the visualization technique, allowing us to directly measure any performance gains over the baseline implementation.

7.6.4 Data Transfer Overhead for Acuity-driven LoD Selection

To evaluate the impact of our technique on data transfer overhead, we measured the average number of pages per frame required to completely texture map the display surfaces of the Reality Deck facility. In order to simulate dynamic gigapixel data, we invalidated entirety of the page cache at each node every frame. As expected, under **SGV**, the number of pages required for proper texturing is almost fixed ($\approx 18,000$, with very small variations occurring due to minute differences in the cluster-wide frame rate under each scenario). If one assumes an average tile size of $9kB$ (as is the case with our data), approximately 162 megabytes of data **per frame** are required for complete coverage of the three walls of the Reality Deck. That translates to 4.9 *gigabytes* per second worth of data transfer (with the system running at a hypothetical 30 frames per second). Conversely, when **ADGV** is active, under scenario **A** we observed a **70**% reduction in page transfer overhead. Scenario **B** demonstrated a decrease of **68**% and for scenario **C** the savings were **72**%. To underline the significant reduction in data transfer overhead, under **ADGV**, assuming



(a) **NGV** - 15fps

(b) **PRE** - $s_{pre} = 16$ - $<$ **2**fps

(c) **ADGV** - $s_{max} = 64$ - **7.5**fps

Figure 41: Performance evaluation of our F+C lens tessellation scheme. (a) **NGV** - 15fps. (b) **PRE** with $s_{pre} = 16$ - $<$ 2fps. (c) **ADGV** with $s_{max} = 64$ - 7.5fps. **NGV** exhibits artifacting since the coarse base mesh can not accurately capture the smooth variation of the lens function. This shortcoming manifests as sharp "bends" along straight lines in the image. **ADGV** delivers a much smoother visual result that is indistinguishable to **PRE** while maintaining a substantially higher frame rate.

again a 30fps frame rate, the required bandwidth drops to approximately **1.5** gigabytes per second.

7.6.5 Frame Rates for Acuity-driven Tessellation

Our acuity-driven tessellation scheme (**ADGV**) aims at bettering the rendering quality of F+C lenses while improving system performance. To quantify this gain, we the average cluster-wide average frame-rate under the three synthetic user sessions, while applying **ADGV** on a single F+C lens. This results in a measurement of **7.5**fps with an s_{max} of 64. This is in contrast with the performance of naive pre-tessellation (**PRE**) with a fixed tessellation factor $s_{pre} = 16$. Under this condition, the frame rate was under 2fps while the upper visual quality bound is actually lower when compared to **ADGV** (since $s_{pre} <$

s_{max}). As an additional point of comparison, applying a lens directly on the base mesh (the **NGV** technique) provided a frame rate of 15fps at substantially reduced image quality. Overall, our technique provides image quality that is not distinguishable from naive pre-tessellation (according to the qualitative segment of our user study), at a substantially faster frame rate. These frame rates, along with captures from within the Reality Deck illustrating the improvements in visual quality, are summarized in Fig. 41.

7.7 Conclusion and Lessons Learned

In this chapter, we introduced a framework for acuity-driven gigapixel visualization. Our method utilizes the formulation for visual acuity to guide the LoD selection process for a virtual texturing pipeline. Using the same formulation, we can improve the visual quality of F+C lenses, applied to gigapixel images, through adaptive tessellation. We conducted a user-study in the Reality Deck, which did not show an effect of the acuity driven LoD in search tasks of various difficulties. Also, we determined that users did not have to adjust their distance from the display during the visual exploration process. Qualitatively, our adaptive LoD approach was rated equal to "naive" gigapixel visualization by most users. Our adaptive tessellation for F+C lenses rated equal to a pre-tessellated version in terms of image quality. However, when compared against a coarse mesh, our scheme was rated superior only by experts. While laymen could detect a difference in the rendering, they could not concretely state which of the modalities sported the superior visual quality. Nevertheless, the acuity-driven LoD approach yielded very tangible benefits in terms of data transfer overhead ($\approx$ 70% or 3.4 gigabytes per second at 30fps in our synthetic benchmarks) while the adaptive F+C lens tessellation showed significant frame rate improvements compared to naive pre-tessellation.

Chapter 8

Scalability of Large, Immersive, High-Resolution Displays

Large, high resolution displays (LHiRDs) are regarded as a useful instrument for the exploration of large amounts of information. These devices permit users to digest large amounts of data in a collaborative way. Simultaneously, they enable *physical navigation* of a digital dataset, where, rather than moving a virtual camera within a 2D or 3D coordinate system, a novel view in the data is obtained by the user simply looking at a different section of the display. Several research studies in previous years have demonstrated the benefits of LHiRDs ([BN07, BN08, YN06] and many others).

Recently, the proliferation of inexpensive GPU power and low-cost high resolution displays has resulted in the construction of massive LHiRDs. Driven by ever-increasing data sizes, single wall designs extend past 300 megapixels (e.g., the Stallion powerwall at the Texas Advanced Computing Center) while the Reality Deck has broken the one gigapixel threshold [PPKM15]. Such facilities can define room-sized immersive visualization spaces, with aggregate pixel counts that afford and demand substantially increased amounts of physical navigation. However, past studies that evaluated the ergonomics of LHiRDs have been conducted on relatively small (or some may say "sane") displays, with the largest being in the 100 megapixel range. While these studies serve as evidence to support the exploratory construction of massive LHiRDs, their findings may not generalize to very large systems. Since the effects of resolution and display form-factor are largely unknown for these very large form factors, we feel that a study of those variables is of value to

future builders of LHiRDs and other immersive virtual environments.

In this chapter, we present the results of a user study [PMG+15] targeted at quantifying the performance implications of very large, physically-navigated, high-resolution displays. Our experiment focuses on four basic tasks. We vary the display form factor and task size, starting at 100 megapixels (arranged planarly) and topping out at 1 gigapixel in a horizontally immersive setting. Our study is conducted within the Reality Deck. Following the motivation and description of our experiment, we provide statistical analysis of absolute and normalized user performance metrics, subjective mental effort and physical travel distances. Additionally, we investigate further into the physical navigation data acquired during our study, which yields interesting insights on how users utilize very large tiled displays. Finally, we conclude with a discussion on our experimental findings and the implications of our study on the design of future LHiRDs.

8.1 User Study Design

The development of a robust experimental design requires answers to a number of important questions. We discuss numerous choices relevant to our study design throughout this section.

8.1.1 Selecting the Information Space

A subtle yet pivotal choice in the design of an experiment is the determination of the *information space*. Jakobsen and Hornbaek [JH13] have covered thoroughly the interrelation between display size, scale and information space. Specifically, they found that the choice between a *fixed* and a *variable* information space greatly affects how performance data should be interpreted. Additionally, they investigated the applicability of different user interfaces to the two information space modalities. In a fixed information space experiment, the size of the data remains constant across display size conditions. It needs to be large enough to properly saturate the visualization space at the largest display size, while still remaining manageable at the smaller display conditions. Additionally, a fixed information space necessitates that some sort of interaction device be used so that users can alter the scale ratio of the visualization. For desktop setups, a traditional

pan-and-zoom mouse interface could be used. However, in our case we are investigating very large displays that require physical navigation. Consequently, if we were to opt for a fixed information space design, we would have to use a more elaborate, portable, interaction device (perhaps a tracked wand or a tablet that serves as a multi-touch surface). This would introduce variability to the performance timings, potentially confounding user performance with user interface familiarity. We opted for a variable information space design, meaning that the *size* of the datasets used for a specific task varies across study conditions (e.g., display resolution and size). This choice necessitates that performance timings be examined after normalization against the information space scaling occurs. We will briefly touch on the raw performance results but the brunt of our analysis will focus on normalized timings (similar to [YN06]).

8.1.2 Hypotheses

Our study design is targeted at evaluating the following five hypotheses:

- $H1$ - *Larger display form-factors (and correspondingly scaled information spaces) will negatively affect absolute performance.* This is a relatively straightforward hypothesis, with Yost et al. [YN06] providing guidance on this front.
- $H2$ - *Larger display form-factors (and correspondingly scaled information spaces) will positively affect normalized performance.* This is a critical hypothesis and evidence towards it would indicate that benefits of LHiRDs scale to large display sizes. Similar results have been demonstrated by Yost et al. [YN06].
- $H3$ - *Larger display form-factors will positively affect normalized performance scaling for search tasks. Pattern-finding tasks should not be substantially affected.* Ball et al. [BN07] have observed substantial performance increases for search tasks as the display grew in size and resolution but not for pattern-finding tasks. It is worth noting that their experiment was of the fixed information space variety and also incorporated virtual navigation.
- $H4$ - *Larger display form-factors result in increased amounts of physical navigation.* Numerous studies (e.g., citeBall:2007, [BNB07]) have shown that users will take advantage of additional display space, with the manifestation being an increase in physical navigation, even when presented with the option of also navigating virtually.

- *H5 - Larger display form-factors negatively affect user mental effort.* Yost et al. [YN06] have found that a larger display (and task size) resulted in increased frustration. Jakobsen and Hornbaek [JH13] ohave bserved similar findings on an SMEQ questionaire [ZH93] in their variable information space experiment.

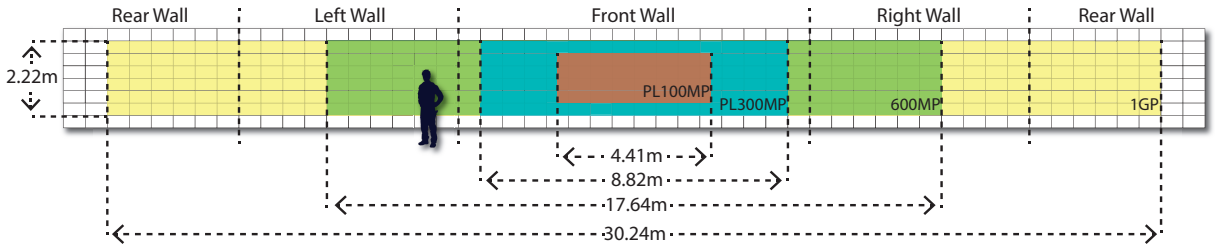


Figure 42: Illustration of the different display conditions in our user study. In this view, the display space of the apparatus has been "flattened" to the 2D plane. The **PL100MP** display condition spanned seven monitors in length and four monitors in height. At **PL300MP**, the horizontal display space grew to 14 monitors while the vertical space encompassed six displays. The display space expanded to 28 monitors at **IM600MP** and finally 48 panels at **IM1GP**. The total "planarized" extents of all display conditions are also visible in the figure. The human figure illustrates a user with a height of $\approx 1.8m$ next to the display surface.

8.1.3 Apparatus, Display Form-factors and Implications

We conducted the user study within the Reality Deck [PPKM15]. For details on the facility, we refer the reader to Chapter 3. The maximum resolution of the facility exceeds 1.5 gigapixels. For this study, however, we excluded the top and bottom rows of the displays from the visualization. We strived to maintain a specific data density in our visualization experiments (as described later). However, at our target density, the visualization glyphs were not clearly legible at the top and bottom rows of the facility unless users approached the walls much closer than the visual acuity distance of $31''$. Rather than rendering two different sizes of glyphs or implementing some sort of semantic zooming (which would confound our study design), we decided to exclude the top and bottom rows of displays from the visualization space. The remainder display space still allowed for meaningfully large display conditions. Additionally, a column of four displays in the middle of the rear wall was not used and served as a natural separator between the two horizontal ends of the visualization.

Based on these constraints, we selected four different display configurations at which to

conduct the user study. The smallest display condition spans roughly 100 megapixels in resolution, over a planar 7×4 display grid (**PL100MP**). We opted for this viewport configuration (rather than one that would span six rows on the vertical axis) in order to maintain a horizontal aspect ratio as is used in other display conditions throughout the study. The second condition extents to approximately 300 megapixels over a planar 14×6 display grid (**PL300MP**). The third condition grows the resolution past 600 megapixels. Here, the display encompasses 16 screens on the front wall of the facility and another 6 screens on the left and right walls (the height of the display lattice remained at six displays). This configuration is termed **IM600MP**. The largest display condition (**IM1GP**) provided approximately one gigapixel in resolution and was 48 displays wide, offering a near-complete horizontal immersion and covering the entire horizontal display surface on the facility with the exception of four columns on the rear wall. Our choices of display form factors were motivated by a number of reasons. The two planar display conditions correspond roughly to the largest display used for a user study in the literature and the largest planar tiled-display facility. The **IM600MP** condition doubles the pixel count of **PL300MP** (allowing for consistent scaling factors for certain tasks as described later). Finally, the **IM1GP** condition reached the gigapixel aggregate resolution milestone. These form-factors, along with corresponding display dimensions are summarized in Figure 42.

There are a number of experimental design implications stemming from the form factor of our apparatus. Firstly, the resolution of each display condition is tightly coupled with the size of the display. Rather than simulating unrealistic LHiRD configurations (such as 100 megapixels at the 1 gigapixel geometry), we opted to scale both in unison, so as to hopefully inform builders of future systems on the usefulness of ultra-high resolution form factors. The second implication is a side effect of the physical layout of our facility. Specifically, at the **IM600MP** condition, the display surface starts to surround the user, almost engulfing her completely at **IM1GP**. Effectively, the geometry of our facility confounds resolution and size with immersion. We posit that the immersive nature of our system should improve performance versus arranging the same amount of displays planarly since travel and view distances between any two points of the visualization space are substantially smaller. A final implication of our apparatus is the fact that we are not able to maintain a consistent display aspect ratio across display conditions.

8.1.4 Data, Visualization and Tasks

Our study was conducted using synthesized data, generated separately for each display size, task and trial. This allowed us to control the data set size and the density of correct answers to task questions.

Overall our visualization scenario was similar to that presented by Ball et al. [BN07, BN08, BNB07], which showed data variables associated with houses for sale that are overlaid on top of a map. Our alteration to the scenario was that the data represented households, each with four attributes (income, wealth, number of children, and average age). We use a similar visual representation to the aforementioned studies, with the data being displayed by textual descriptions. Behind the text, a normalized bar chart illustrates the position of that particular value within the range of values in the dataset that is being visualized. Each glyph was exactly 150 pixels wide and approximately 100 pixels tall, allowing for readability of glyphs across the entire height of the display when a user is standing approximately $0.5m$ away from a screen. An example of our visualization can be seen in Figure 43.

Within this scenario, we defined four tasks that have appeared in numerous past experimental evaluations of LHiRDs [BN08, BNB07, JH13, YN06]:

- Visual Search (**VS**) - Subjects were presented with a base map. Within the map, a single visualization glyph was placed. They were asked to locate this singleton glyph and read out loud one of its attributes. Once the attribute value was verified, the task was completed.
- Attribute Search (**AS**) - Subjects were presented with a base map with several thousand visualization glyphs visible. They were asked to locate three households that satisfied a particular attribute query (e.g., *"Locate three households with more than four children"*). There were multiple correct answers for each trial of **AS**. Once three correct answers were verified, the task was completed.
- Comparisons (**C**) - Subjects were presented with a base map with a small number of visualization glyphs overlaid on top of it. Subjects were asked to locate the household that satisfied a particular attribute query (e.g., *"Find the household with the lowest income"*). These task queries only had a single correct answer for each trial.

- Pattern-finding (**P**) - Subjects were presented with a base map with several thousand visualization glyphs visible. They were asked to locate a cluster of households that satisfied some attribute query (e.g., *"Point out the area of households with the highest incomes"*). These tasks had a single, uniquely correct answer for each trial.

As stated earlier, we used a variable information space design, making the scaling of the task data space critical. For all tasks, the base map grew inside linearly across display conditions, increasing the area that was visible. For **AS** and **P** the map was populated with uniformly scattered synthesized households. The number of visible households started at 1200 for the **PL100MP** condition and grew to 12000 at **IM1GP**. For **AS** the number of data points that provided an answer to the trial query was 150 across all display conditions (distributed uniformly across the display space). Meanwhile, for **P** the size of the clusters that satisfied the trial queries was 200 data points (centered around a manually selected centroid) across all display conditions. For **VS** trials, the unique target glyph was manually placed during data set generation. Finally for **C** data sets, the number of visualization glyphs scaled from two at **PL100MP** to 20 at **IM1GP**. These glyphs were placed manually for each trial in order to span the display space of each condition. The singleton target glyph, that satisfied the trial query, was also manually selected. The data sets were crafted so that no visualization glyphs ever spanned across or were obscured by display bezels. Our study design called for 16 combinations of display form-factors and task types. For each combination three trials were conducted. Overall, we generated a total of $16 \times 3 = 48$ data sets.

The study was implemented using the distributed visualization software stack described in Chapter 4. The base map layer was provided by Mapquest Open[1] .

8.1.5 Participants

Sixteen volunteers (three female), with an average age of 27 ($\sigma = 4.78$) participated in the study. Most participants were science graduate students, recruited by word of mouth. All but one participants reported minimal to no experience with LHiRDs. Also, all participants reported being of average vision, either naturally or via corrective glasses or contact lenses.

[1] developer.mapquest.com

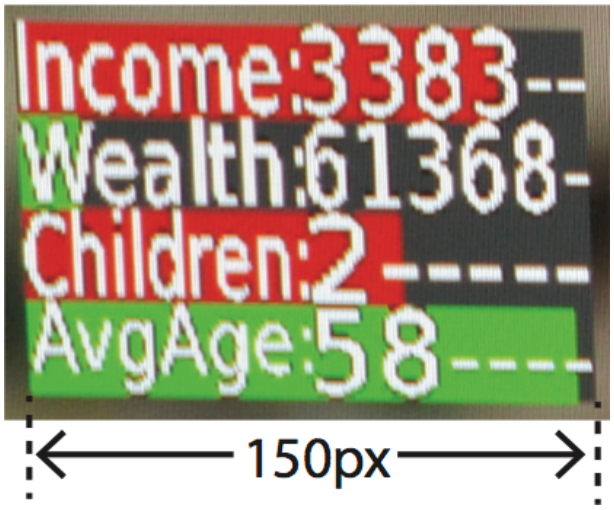


Figure 43: Example of the visualization design of our experiment. Each datum (describing a household) carried four attributes: income, wealth, number of children and average age. The raw value of an attribute is presented to the user next to its label in text form. Additionally, normalized bar charts behind the text showed the position of each particular attribute within the range of values in the data set that was being displayed at each time.

8.1.6 Independent and Dependent Variables

We opted for a within-subjects design, with the independent variables being **DISPLAY** (with values **PL100MP**, **PL300MP**, **IM600MP** and **IM1GP** for the form-factors outlined earlier) and **TASK** (**VS** for visual search, **AS** for attribute search, **C** for comparisons and **P** for pattern finding). The primary dependent variable was elapsed time to task completion (**ET**). However, absolute comparisons of **ET** across different display conditions do not yield very useful information about the impact of different form factors since the size of the data is also changing. With this in mind, we apply a normalization scheme that is strictly based on the size of the data across display conditions. The reasoning behind our scaling strategy is as follows. The data sets for all task types are comprised of *distractors* and *targets*. For the **VS** and **C** tasks, the distractor is the background map. For the **AS** and **P** tasks the distractors are the background map and the thousands of data points that populate it that are not correct answers to task queries. The probability of successfully completing a task at random is dependent on the ratio of targets versus distractors in each particular dataset. If both targets and distractors were scaled similarly in numbers or size across display conditions, task difficulty would not be affected (since the chance of finding a target *randomly* would remain the same). However, since we only scale the

number or size of distractors, we posit that task difficulty increases linearly. We thus define a second metric, *scaled elapsed time* or **SET**, which assumes a linear scaling in task difficulty based on the size of the task data set and corresponding display condition, and permits us to evaluate the impact of display form factor on performance as the size of the data grows. In order to compare across display conditions, we calculate **SET** as follows. For the **PL100MP** condition, **SET** = **ET**. At **PL300MP SET** = $\frac{\mathbf{ET}}{3}$, at **IM600MP SET** = $\frac{\mathbf{ET}}{6}$ and at **IM1GP SET** = $\frac{\mathbf{ET}}{10}$. Note that the scaling factors are equal to the multipliers of the distractor sizes/populations, as reported earlier. It is worth mentioning that our scaling strategy is similar to that employed by Yost et al. [YN06] for tasks that have a singleton correct answer (**VS**,**C**,**P**).

To evaluate user effort, we administered a paper-based Subjective Mental Effort Questionnaire (**SMEQ**) [ZH93], which generates similar results to a Likert scale while providing additional resolution [SD09]. We also measured the subjects' physical navigation within the facility's ground plane (**TRAVEL**) by simplifying tracking trajectories using the Douglas-Peucker [DP73] algorithm with a tolerance of 0.03.

Overall, each participant performed three trials for every task at every display condition for a total of $4 \times 4 \times 3 = 48$ data points for **ET**, **SET** and **TRAVEL**. A total of $4 \times 4 = 16$ **SMEQ** entries were generated as the test was administered once all three trials were completed.

8.1.7 Protocol

Upon arriving to the experiment location, participants were given an introduction to the immersive gigapixel display, while the system was displaying a sample data set that was not used in the study. They were provided with an explanation of the visualization and introductions to the four tasks that they would be performing. Each participant was allowed approximately five minutes of familiarization with the facility and was given the ability to ask questions regarding the visualization and task descriptions. Each participant was asked to position themselves in the middle of the facility, in front of the operator workstation, prior to starting each trial. After completing all three trials for all tasks at a particular display condition, participants were allowed to take a five minute break. Subjects performed each trial until successful completion. Consequently, accuracy approached 100% for all users. To counterbalance for fatigue and learning effects, the

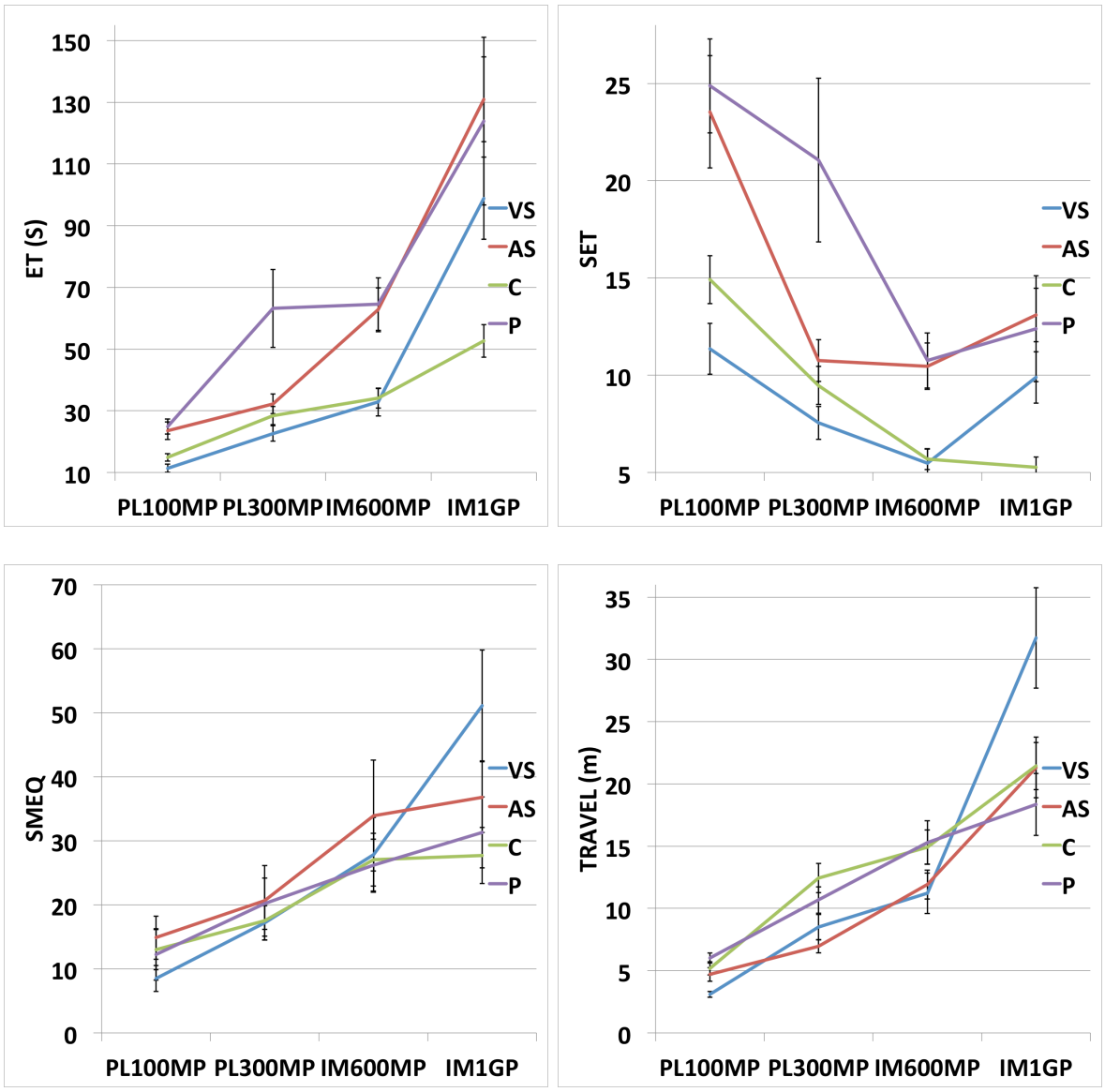


Figure 44: Summarization of the **ET**, **SET**, **TRAVEL** and **SMEQ** metrics from our study. As expected, **ET** increases as the display grows in size. Both **VS** and **C** were faster than **AS** and **P** up until **IM600MP** since the former involved a much smaller number of datums to be discovered and examined. However, at **1G** the mean **ET** for **VS** increases substantially. For **SET**, at the **PL100MP** and up to the **IM600MP** display condition we observe an improvement in user performance for most tasks. At the **IM1GP** no statistically significant improvements are realized and normalized performance for the **VS** deteriorates significantly. An expected increase in **SMEQ** was detected as the size of the display grew. However, we observed a much more significant increase for the **VS** task (more than 6× from **PL100MP**to **IM1GP**). Finally, we observed an expected increase in **TRAVEL** as the display grew in size. The **VS** task at **IM1GP** showed significantly higher amounts of physical navigation (with a mean travel distance of 39.86m versus less than 25m for other task types).

order of exposure to both the display condition and task was determined using a Latin Square design. Each session lasted approximately 1 hour and 30 minutes. At the end of the sessions, we conducted a brief, informal, interview with each participant.

8.2 Results

In this section, we report on the statistical analysis performed on our dependent variables. All analyses involved two-way repeated-measures ANOVAs on the respective variables after *ln*-transformations (to ensure normality). Post-hoc statistics are reported based on pairwise comparisons with Bonferroni correction. The reported means stem from the original (non-transformed) data.

8.2.1 ET

As expected, our analysis of **ET** showed a main effect for **DISPLAY** ($F(3, 141) = 116.850, p < 0.0001$) and **TASK** ($F(2.137, 100.425) = 18.938, p < 0.0001$ with Greenhouse-Geiser (GG) correction). Additionally, we found an interaction effect for **DISPLAY** $\times$ **TASK** ($F(6.341, 298.014) = 5.868, p < 0.0001$ with GG correction). The mean **ET** increased with **DISPLAY** across almost all tasks. A summary of raw performance timings can be seen in Figure 44.

8.2.2 SET

We discovered a main effect on **SET** for **DISPLAY** ($F(3, 141) = 54.885, p < 0.0001$) and for **TASK** ($F(2.139, 100.514) = 18.952, p < 0.0001$ with GG correction). Additionally, we found an interaction effect for **DISPLAY** $\times$ **TASK** ($F(6.346, 298.254) = 5.865, p < 0.0001$ with GG correction). We report post-hoc comparisons with Bonferroni correction after examining differences at the $p < 0.05$ significance level.

Overall, we found a significant difference between **VS** and **AS** ($M = 8.57$ to $M = 14.46$, $p < 0.001$), also between **VS** and **P** ($M = 8.57$ to $M = 17.27$, $p = 0.001$). Additionally, we observed a significant difference between **AS** and **C** ($M = 14.46$ to $M = 8.83$, $p < 0.001$). Overall, faster normalized completion times were recorded for the

visual search and comparison tasks. We also detected significant differences between the **PL100MP**, **PL300MP** and **IM600MP** display conditions, with mean **SET** improving from $M = 18.67$ to $M = 12.21$ ($p < 0.001$) and down to $M = 8.09$ ($p < 0.001$). We observed a decline in normalized performance for **IM1GP** with $M = 10.16$, but it was not statistically significant.

By examining the **DISPLAY** $\times$ **TASK** interaction more closely, we can see that the performance for **VS**, **C** and **P** improves as the resolution increases, up to and including the **IM600MP** condition. However, when transitioning from **IM600MP** to **IM1GP**, the only performance improvement is for **C** (from $M = 5.678$ to $M = 5.267$ which falls within the standard error threshold). Contrarily, we observe a statistically significant drop for **VS** (from $M = 5.476$ at **IM600MP** to $M = 9.888$ at **IM1GP**). **AS** and **P** means also regressed but not in a statistically significant manner. A breakdown of **SET** means can be seen in Figure 44.

8.2.3 SMEQ

For **SMEQ**, we discovered a main effect for **DISPLAY** ($F(3, 45) = 17.126, p < 0.0001$). A breakdown of SMEQ mean responses across tasks and display conditions can be seen in Figure 44. Looking more closely at the mean of SMEQ responses over **DISPLAY**, we can observe a nearly linear scaling. Specifically, at **PL100MP** users felt that tasks were approximately "Not very hard to do" ($M = 12.16$). In comparison, **IM600MP** was significantly different ($p < 0.001$), with the response exceeding the "A bit hard to do" threshold on the SMEQ scale ($M = 28.75$). Finally, at **IM1GP**, we observe that $M = 36.73$ (between "Fairly hard to do" and "A bit hard to do"), significantly different from **PL100MP** ($p < 0.001$).

8.2.4 TRAVEL

Through analysis on **TRAVEL**, we discovered main effects for **DISPLAY** ($F(3, 141) = 147.428, p < 0.0001$), **TASK** ($F(3, 141) = 5.013, p = 0.002$) and an interaction effect between **TASK** and **DISPLAY** ($F(5.995, 281.778) = 8.549, p < 0.0001$ with GG correction). These results are summarized in Figure 44. An examination of the interaction

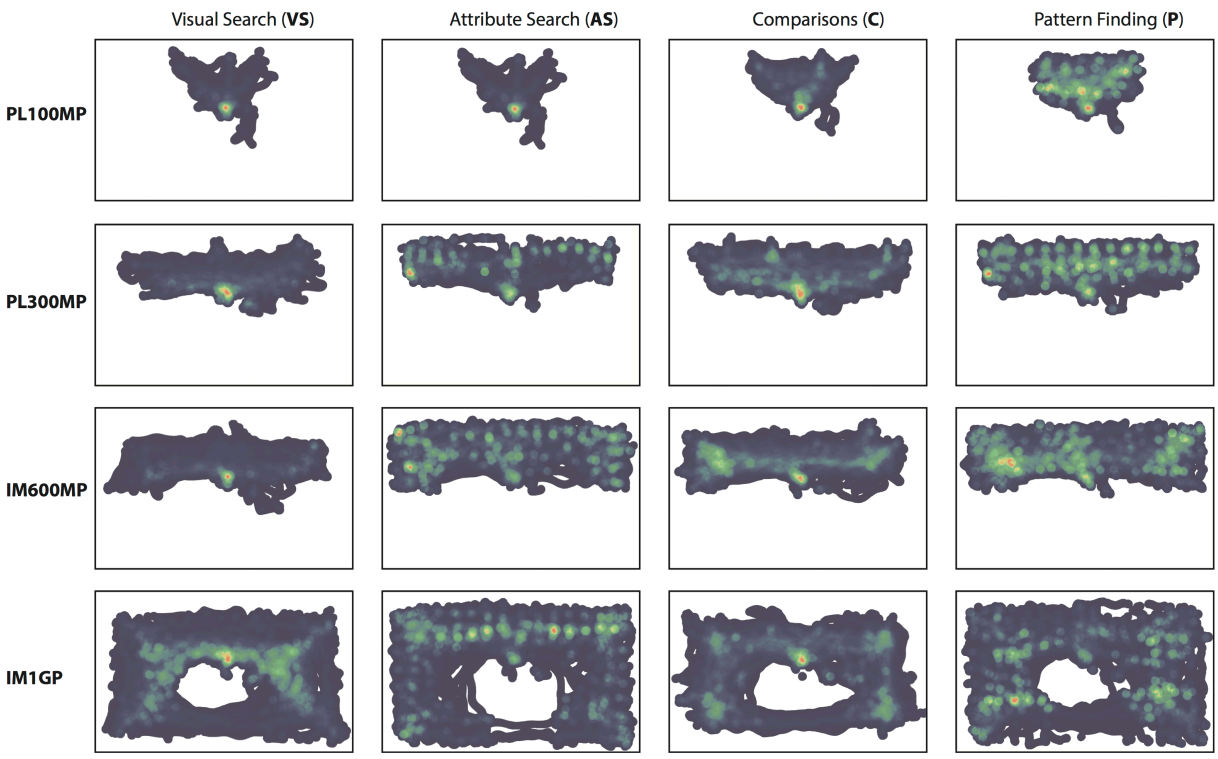


Figure 45: Visualization of user presence within the apparatus space. The images are generated for all combinations of **TASK** and **DISPLAY**. The color mapping corresponds to time spent, with warmer colors signifying more time. The presence hotspots in the middle of the facility denote the starting point for each task, right in front of the operator workstation. A strong "detail" search pattern can be deduced from the **AS** column of visualizations, with multiple presence hotspots aligning roughly with the displays of the facility. Meanwhile, the **VS** task shows the "overview" pattern between the **PL100MP** and **IM600MP** conditions, as users chose to stay near the center of the facility and inspect displays from a distance. This pattern changes at the **IM1GP** level, with users navigating more and examining sections of displays more closely. Somewhat similar patterns are visible for the **C** task, but with additional navigation trails, due to users approaching the displays to examine individual values. The **P** task displays more erratic physical navigation trails.

between display size and task type showed significantly higher amounts of physical navigation for the **VS** task at the **IM1GP** condition ($M = 39.86m$ versus $M = 24.95m$ for other task types).

8.3 Interpretation of Results

A number of insights can be drawn from the statistically significant effects resulting from our study. As expected, we found significant evidence in support of *H1*. Larger task (and display) sizes generally resulted in increased **ET** measurements. The more interesting finding however relates to **SET**. When transitioning from **PL100MP** to **PL300MP** scaled performance improves significantly for **VS**, **AS** and **C**. Some gains are still visible in the **PL300MP** to **IM600MP** transition. The **P**, **VS** and **C** tasks are positively influenced. However, when the display and task space grows to **IM1GP**, performance appears to at least plateau for **AS** and **P** and actually deteriorate for **VS**. Overall while the analysis appears to support *H2* for the first three display conditions, the expected benefits of LHiRDs do not materialize at **IM1GP** and the performance regression for **VS** is particularly striking.

An inspection of the **TRAVEL**-related statistics shows that users do indeed take advantage of the additional display space by navigating more, as hypothesized in *H4*. At **PL100MP** users travelled on average 4.75 meters over all tasks. At **PL300MP**, this metric increased to $9.64m$, roughly a $2\times$ increase in navigation for a $3\times$ increase in display size. A 6-fold increase to the size of the display (to **IM600MP**) resulted in a $2.81\times$ increase in physical navigation distances, with mean distance travelled reaching its maximum at **IM1GP** ($M = 23.209m$). As noted though, **VS** resulted in substantially higher physical navigation than other task types. This finding matches the substantially increased task completion times and negatively impacted **SET** for **VS** at **IM1GP**.

The **SMEQ** findings reflect expectations set in prior studies and outlined in *H5*. Larger displays and task sizes correlate with increases in user mental effort and frustration. Yost et al. [YN06] have found statistically significant increases on a NASA TLX survey for both physical effort and user frustration when scaling visualizations from a 2 megapixel single display to a 32 megapixel 24-monitor LHiRD. Jakobsen and Hornbaek [JH13] have also observed a small increase in mental effort ratings for the large display condition in their variable information space experiment. Interestingly, the **SMEQ** response for **VS** at **IM1GP** was an outlier when compared to other tasks. The deterioration in **SET** and increase in **SMEQ** and **TRAVEL** for this task and display condition leads us to believe that visual search may not be an appropriate task for extremely large display walls. Additionally, all three metrics counterindicated any performance improvements for

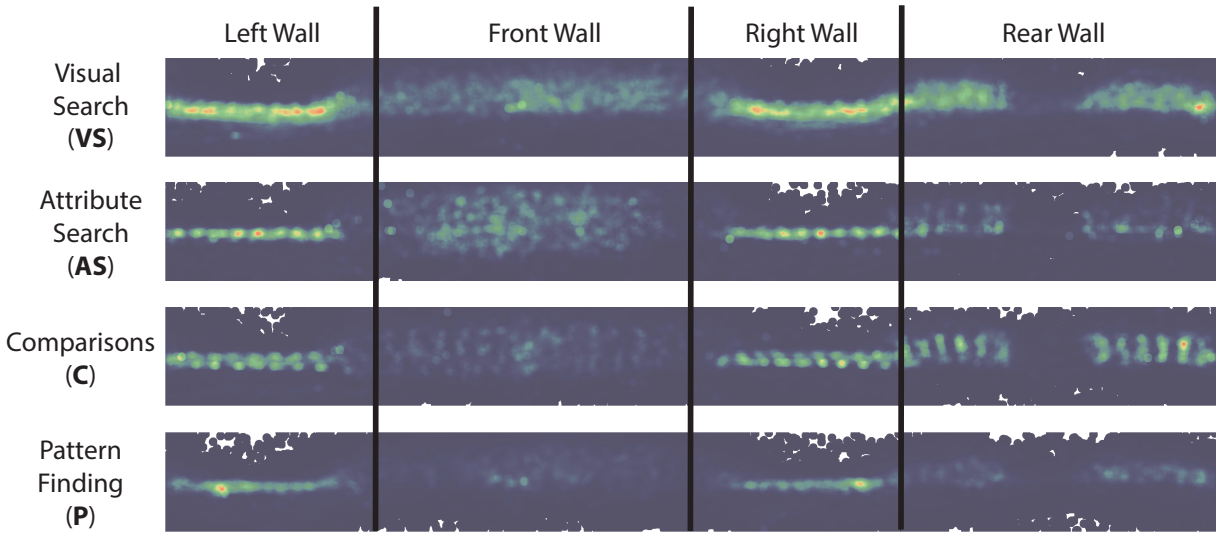


Figure 46: Visualization of estimated participant gaze. The images are generated for all task types at the **IM1GP** display condition. The gaze distribution is presented in the form of a heat map, with normalized gaze time mapping to a truncated HSV color space and warmer hues indicated longer gaze periods. Additionally, the boundaries of the facility walls are annotated by the black lines. Two main patterns can be observed. First, participants commenced the tasks with an overall scan of the front wall, their gaze distributed across most of the display space. Then, they proceeded to more closely focus on the side and rear walls while "narrowing" their search space. Additionally, there is a distinct lack of user attention towards the corners of the front wall. This pattern was also observed by the facility operator during the user study sessions. The gap in the middle of the rear wall corresponds to the section of displays that was not utilized during the study.

the **VS** task at the largest display condition, which implies that *H3* also does not hold for very large systems.

8.4 Physical Navigation Analysis

To gain additional insight on the quantitative results described above, we generated per-task and per-display-condition views of the participants' presence within the facility. These visualizations can be seen in Figure 45.

An examination of the **VS** condition shows that for the **PL100MP**, **PL300MP** and **IM600MP** display form-factors, users primarily chose to remain near the center of the facility and adopt an "overview" search pattern, examining the displays from a distance while looking for the target. However, at **IM1GP**, the motion data indicates that at

least some users adopted a "detail" search pattern, moving around the facility more and taking time to examine individual sections of the displays, which could explain the drop in performance and increase in **TRAVEL** and **SMEQ** that we observed. A more pronounced "detail" search pattern is apparent for the **AS** task at all but the smallest display condition. In this case, it appears that users spent time examining individual columns of displays in an effort to find datums that were valid answers to the task questions. This pattern is particularly strong in the **IM1GP** condition, with presence hotspots approximately lining up with the centers of the display columns of the apparatus facility. For the **C** task, we observe an "overview" pattern, with users opting to search the display space from afar while searching for the small number of datums within the base map. Finally, the **P** task appears more chaotic, with different users adopting different strategies. Some opted for an "overview" approach, searching for the visual patterns generated by the bar charts embedded in our visualization, while other users decided to examine the data labels before making a determination.

To estimate the participants' gaze direction within the visualization space, we use the head orientation approximation. According to Kai and Rainer [NS03], head orientation is a good predictor for eye gaze (with an accuracy of approximately 87% to 89% in pointing-related experiments). We visualize gaze heat maps in Figure 46, calculated similarly to the presence heat maps outlined above. We focus on the **IM1GP** display condition since it generated patterns similar to **IM600MP**. The **PL100MP** and **PL300MP** gaze patterns did not deviate from what has already been presented in the literature. Two main observations can be made. Participants appear to commence a task via an examination of the majority of the front wall display space (evident by the low intensity and evenly distributed concentrations of gaze in the visualization). However, then they begin to more closely examine the other display surfaces in search of the task answer while somewhat limiting their range of search (notice the much stronger gaze patterns on the left, right and rear walls, focused near the vertical middle of the display space). Additionally, one can see that there appears to be a lack of user attention towards the corners of the facility, particularly the ones between the front and side display walls. We observed this behavior during our user study, particularly with some users that adopted a "detail" approach to the **VS** task. They would scan the displays of each wall very closely up until they reached the columns of displays that were adjacent to a corner. At that point, they would rapidly divert their gaze to the next wall, rather than thoroughly scan that last column

of monitors (or the first column of the adjacent wall).

8.5 Discussion

The analysis of performance data from our user study indicated that normalized performance does not measurably improve as the display configuration changes from **IM600MP** to **IM1GP**. Contrarily, normalized performance for the visual search task dropped substantially. Meanwhile, performance in the attribute search, pattern-finding and comparisons tasks remained approximately flat. Why does this happen?

We think that very large LHiRDs can potentially "overwhelm" an user with visual information for certain tasks. For example, in the **VS** task, which required that users potentially examine the entire display space multiple times, we found a significant increase in the SMEQ response at the **IM1GP** level (where as others remained relatively unchanged from the **IM600MP** level). When users are in this state, they seem to adopt the "detail" search pattern, which substantially increased travel distances, compared to other tasks. During the post-interview, users at the **IM1GP** level mentioned that "The display was too large" and that "There were too many data points to inspect". Overall, a large LHiRD may not be the best apparatus for visual search search using physical navigation. Or alternatively, traditional physical navigation could be enhanced by techniques such as "saliency assisted navigation" (introduced by Ip and Varshney [IV11]). In other task types, we observed a normalized performance plateau rather than an outright degradation. Unfortunately, our study can not inform on whether these findings are a result of the sheer size of the data being displayed or if the larger field of view for the **IM1GP** also plays a role.

User performance was correlated to the task difficulty. It would be interesting to study the **AS** tasks at different display conditions and for different densities of correct answers. Anecdotally, during our study design phase, we briefly considered utilizing a total of 10 correct datums as answers to **AS** at all trial stages. This density proved manageable at the **PL100MP** and **PL300MP** display conditions (approximately three to five times slower than the times observed in our full study) but, users required upwards of 20 minutes to locate one correct datum at the **IM1GP** level (30 to 40 times slower than the mean **ET** for **AS**). We feel strongly that this discrepancy in difficulty scaling warrants more

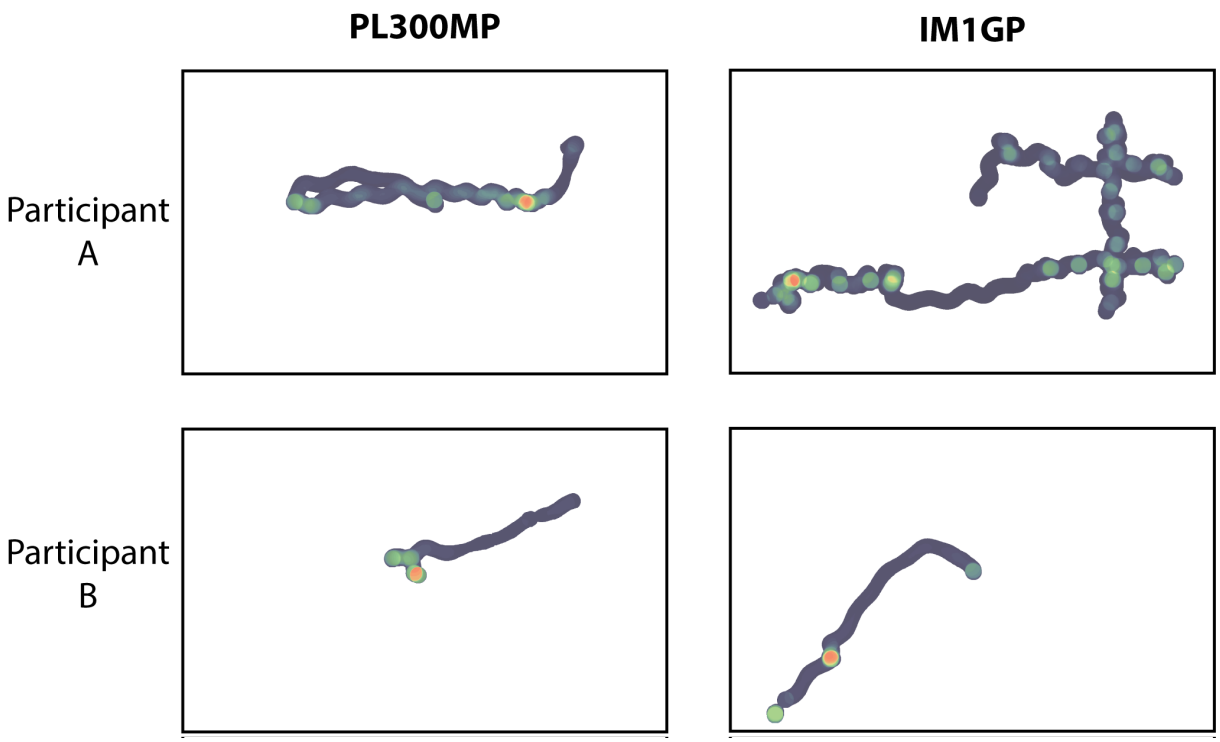


Figure 47: Visualization of the physical navigation of two study participants. These images are generated using tracking data from the **P** task at the **PL300MP** and **IM1GP** display conditions. All visualizations are drawn from the same task trial (hence users were examining the same data set and the targets were at the same locations for both). Participant A demonstrates the "detail" search pattern. Overall, he navigated to different places of the facility and examined separate sections of displays closely, as is evident by the multiple presence hotspots. Participant B adopted the "overview" pattern. He remained near the center of the experimental apparatus and inspected large sections of the displays from a distance, only moving in to verify his answer to the operator. Participant B performed substantially better in both tasks.

investigation.

The emersion of the "overview" and "detail" patterns intrigued us. We thus looked more closely at individual participants and the performance across tasks. In Figure 47, we visualize the movement of two participants for the **P** task across the **PL300MP** and **IM1GP** configurations. Participant A opted for a detail-oriented approach, spending time at different places within the facility in order to closely examine different sections of the display. Meanwhile, Participant B largely remained near the center of the facility and visually inspected the data for patterns, utilizing the embedded bar charts in the visualization. Finally, he approached the screen to verify his answers, as part of our study

protocol. In this example, the performance of participant B was orders of magnitude better than that of participant A ($20s$ versus $261s$ at **PL300MP** and $18s$ versus $771s$ at **IM1GP**). It is also worth noting that both participants had not been exposed to an LHiRD prior to our study.

Additionally, while some users adapted their physical navigation patterns, depending on the task at hand, others utilized the same pattern throughout all tasks. For example, Figure 48 visualizes the search patterns of two participants in the **VS** and **P** tasks. Here participant A chose to transition from a very "detail"-oriented navigation pattern for the **VS** task, to an "overview" approach. He navigated substantially less ($21.44m$ for **P** versus $117.82m$ for **VS**). This change of approach is particularly interesting since both these tasks incorporate a visual search component (since in **P**, participants had to locate the datums prior to comparing). In contrast, Participant B chose a balanced approach for both tasks, inspecting larger sections of the display from a distance.

The takeaway point is that evaluation of user performance in LHiRDs needs to go beyond quantifiable metrics of time and accuracy. A smaller tiled-display affords some physical navigation but the larger workspace of facilities, such as our experimental apparatus, provides room for specific usage patterns to emerge. Through these usage patterns, some users are able to handle visualization tasks better than others. The findings that arose from the visualization of estimated gaze direction give rise to two main questions. Why do users decide to cover the front wall of the visualization facility more thoroughly but then focus their gaze towards the middle of the displays? Also, the lack of user attention towards some of the corners of the facility makes us wonder whether a cylinder-style layout would have been more optimal (even at the expense of overall resolution). In hindsight, if we were to reconduct our study, we would have adjusted the large planar display form-factor to also include the outer-most display columns. Still, it is interesting that this pattern of "corner-blindness" does not appear at the edges of the left/right walls with the rear displays at the IM1GP condition. We feel that this is fertile ground for further visualization and human-factors research, targeted at understanding the root causes of these behaviors and potentially improving user experience within these facilities.

A closing point that we wish to bring up is that our study design (due to the nature of our apparatus) does not decouple of the high display resolution from the immersive experience at the larger display conditions (**IM600MP** and **IM1GP**). This confound is

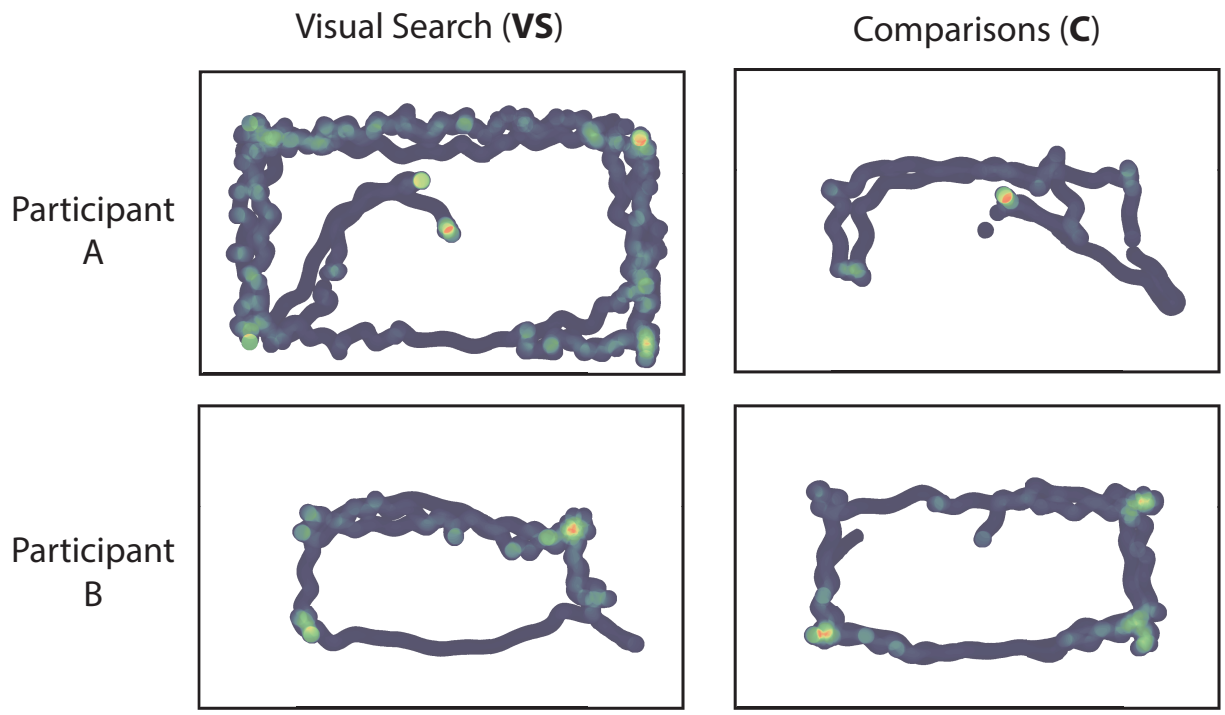


Figure 48: Example of the between-task variation for two different participants. Participant A employed the "detail" pattern for the **VS** task, scanning small sections of displays at a small distance. However, for the **C** task, despite the substantial visual search component, he switched to a more "overview"-centric approach, staying further back from the monitors. Participant B was more consistent in his search patterns between the two tasks, choosing to inspect larger sections of displays from a small number of hotspots. All visualized trials were conducted at the **IM1GP** display condition.

significant and warrants additional investigation. During the engineering stages of our system, 27" monitors with a resolution of 2560×1440 offered the highest pixel density available at a reasonable cost. Nowadays, 4K monitors are quite affordable and multiple of them can be driven by a single GPU. It is not hard to imagine a billion pixel planar LHiRD (with a display area equal to that of our 300 megapixel display condition) being built in the near future. The key difference of such a hypothetical facility, when compared to our experimental apparatus, is that it would not require that the user substantially pan their heads in order to see the entire display space. At that point however, one may be tempted to construct an even larger facility by tiling multiple of these gigapixel resolution walls, resulting in the same conundrum.

8.5.1 Implications for design

Based on our findings we offer the following recommendations to builders, users and researchers of immersive LHiRD systems. The construction of facilities that scale to dimensions and resolutions similar to the **IM600MP** condition of our study appears to be worthwhile, with the expected performance improvements materializing. The layout of an immersive LHiRD is important. While a rectangular layout such as that of our apparatus maximizes the display surface that can fit within an existing physical space, it seems to cause at least a small amount of impact on physical navigation patterns, as found in our gaze analysis. User training and familiarity is important. Even with the simple visualization scenario used in our study, we observed several users who adopted suboptimal navigation patterns, leading to decreased performance timings. Providing LHiRD users with introductory sessions that inform them on usage strategies for these facilities could be beneficial.

8.6 Conclusion

We presented the result of a 16-subject user study targeted at quantifying the limits of large, high-resolution displays (LHiRDs). A statistical analysis of normalized user performance showed that the benefits of LHiRDs plateau past the 600 megapixel point, demonstrating that the benefits of LHiRDs that were established in previous studies do not scale arbitrarily. Meanwhile, an investigation of physical navigation data across display conditions and task types yielded a number of insights. Total movement of the users increased as the size of the display grew. More importantly, users adopted two physical navigation strategies (termed "overview" and "detail") when tackling different types of tasks.

Chapter 9

VEEVVIE - Visual Explorer for Empirical Visualization, VR and Interaction Experiments

9.1 Introduction

Quantitative evaluation is the cornerstone of the scientific discovery process and it is no-less important for relatively young fields, such as visualization, virtual reality and human-computer interaction. Tory and Moller [TM04] have underlined the importance of *testing* visualization systems, as part of human-factors research in visualization. In IEEE's Information Visualization Conference of 2014[1] , roughly half of the accepted papers included some sort of formal evaluation through empirical experimentation. Similar trends can be observed in the majority of seminal conferences and journals that focus on human-centric computing.

These experiments are primarily conducted as evidence supporting the value of a newly introduced technique and/or contrasting the effectiveness of several established methods. These goals are pursued by performing statistical analysis, driven by a set of hypotheses aimed at answering high-level research questions. When presenting the results of these experiments, scientists follow their exposition of the analysis with a discussion, putting

[1] http://www.ieeevis.org/year/2014/

the results in context. Often this process involves inspecting the performance of individual participants under the prism of the analysis results or isolating and visualizing only parts of the experiment design. This post-hoc *exploratory* process is critical for three reasons. Firstly, it further validates the results of the statistical analysis. Secondly, it generates insights that can not be captured directly through statistics (due to the generally low granularity). Finally, it provides future research directions that drive the conduct of more specific or differently targeted experiments through the generation of new hypotheses.

The importance of empirical evaluation has driven researchers to develop visual analysis systems, specifically targeted at the exploration of their experimental data. Blascheck and Ertl [BE13] have suggested a generalizable methodology for visual experiment analysis with a seeming focus on eye-tracking data. Generally, efforts to leverage visualization for the exploration of empirical studies are targeted at specific domains. However, the underlying data (e.g., the experimental design, the types of metrics, etc.) tend to be similar across domains. This uniformity can be leveraged for the development of a generalizable framework for the analysis of empirical experimental data.

In this chapter, we introduce *VEEVVIE*, the Visual Explorer for Empirical Visualization, VR and Interaction Experiments [PGK15]. *VEEVVIE* is a versatile tool that enables the visual analysis of empirical experiments in visualization and human-computer interaction. At its core, *VEEVVIE* is driven by an ontology that can model a wide range of experimental designs, with arbitrary variable, measurement and sample sizes. This ontology can, on its own, capture multiple experimental designs that are based on "common" measurements (e.g., ordinal or interval). Driven by this ontology, *VEEVVIE* generates several familiar linked information visualizations of experimental measurements with support for interactive brushing across different pivots. Additionally, the *VEEVVIE* ontology can be expanded to support domain-specific data types. Through a plug-in architecture, new visualization widgets can integrate in the *VEEVVIE* workspace and ingest this domain-specific data. We demonstrate *VEEVVIE* through several use cases, examining data from a prior visualization experiment.

This chapter is structured as follows. We begin by motivating the design decisions for *VEEVVIE* and then present the ontology that drives our visual analytics framework. Following this groundwork, we describe how *VEEVVIE* leverages this ontology to enable

Table 3: Summary of high-level input experimental data. This data is fed into the *VEEVVIE* ontology in order to appropriately model an experiment.

Experimental Design	Experimental Measurements	Statistical Effects
Independent Variables (and valid levels) Dependent Variables (and type information) Participants (and factor assignments) Trial Grouping Factors	Trial Descriptions (participant, factors, measurements) Experiment-Wide Measurements	Affected Dependent Variable Type (Main or Interaction) Source Independent Variable(s)

extensible visual analysis of experimental data. We conclude by presenting a set of case studies from our usage of *VEEVVIE* in the visual analysis of the study outlined earlier in this thesis.

9.2 Defining the Problem Space

In this section, we outline some examples of visual analytics that are used in a complementary fashion to statistical analysis as part of visualization and interaction experiments. We outline several key "tasks" that emanate from those and other works and also summarize the information that concretely describes a visualization experiment.

9.2.1 Examples of Visual Analysis for Experiments

The design of *VEEVVIE* is based on the utilization of visual analysis experiments conducted by our research group and also those described in the literature. For example, Ball et al. [BNB07] have generated simple visualizations of user movement and gain estimation during different tasks on a tiled display. A sample observation was that during an "insight" task, when participants were provided with a lectern for note-taking, physical navigation was substantially different and clustered around the lectern. This observation led to a recommendation later in the article. They also utilized visualizations to reinforce the statistical findings, which indicated that increases in display width resulted in

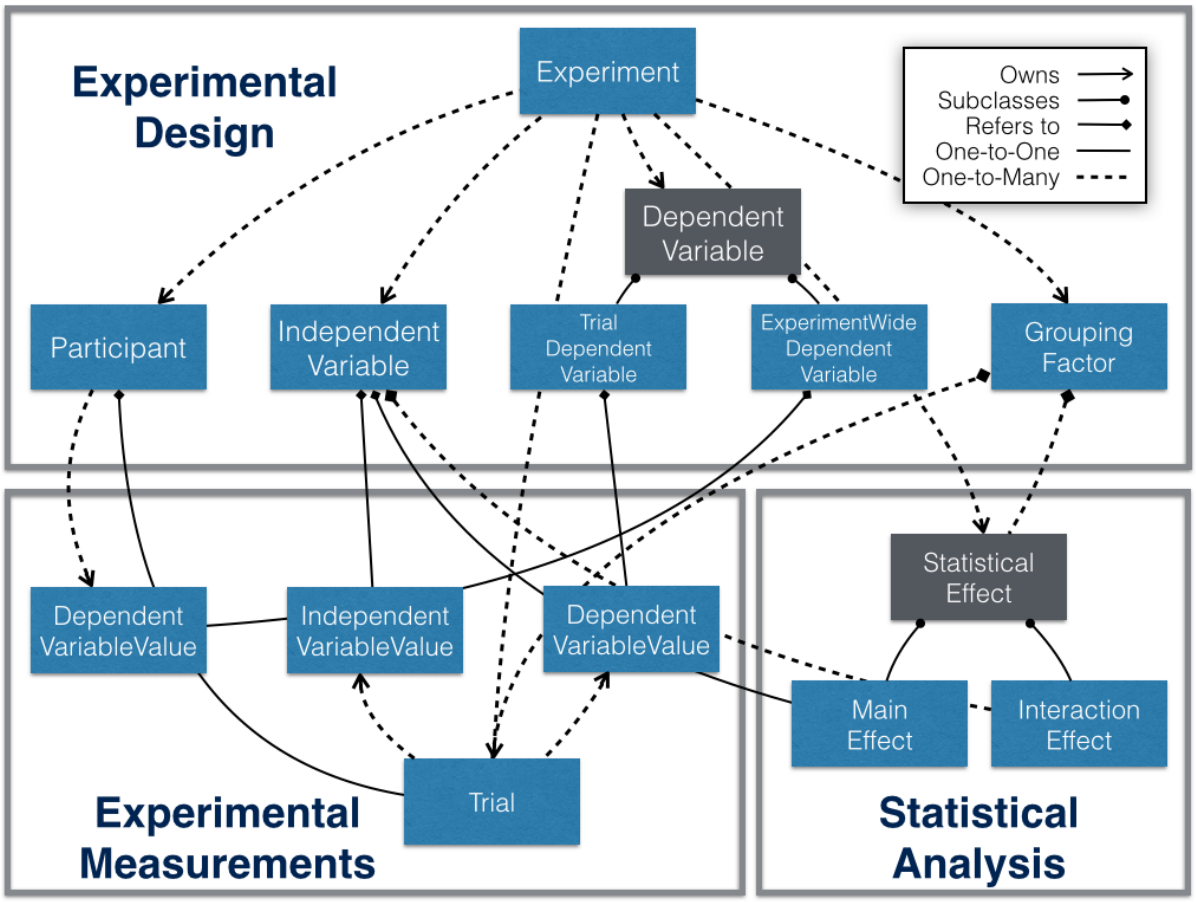


Figure 49: The *VEEVVIE* ontology. Abstract classes are denoted by darker colored backgrounds. Notice the separation between sections of the ontology which capture different aspects of the experimental process, with the referential relationships serving as the connective tissue.

increases in physical navigation. Bezerianos et al. [BI12] utilized motion trajectory visualizations in order to categorize select participants' movement strategies, which then served as input into further statistical analysis. The usage of visualization in the exploration of experimental data clusters around filtering, summarizing, highlighting aspects of the experiment, in a manner driven by the findings of the statistical analysis.

9.2.2 Tasks

Overall we identify the following tasks that researchers might want to perform on their experimental data using a visual analysis tool:

1. *Emergence of the sources of statistical effects.* Given a significant difference between two levels of a factor, or an interaction between two factors, the visualization tool should enable the rapid identification of the subset of the experiment that falls under those factors.

2. *Cross-experiment identification of trials or subjects.* During the visual analysis process, the researcher might identify a set of trials or subjects that exhibit an interesting pattern (e.g., combination of measurements, type of movement, etc). The visual analysis tool should permit the quick investigation of these datums under different visualization modalities.

3. *Summarization of experimental results across different pivots.* The visualizations generated by the tool should enable visual aggregations of the results across different dimensions of the experiment (e.g., across all subjects, across trials, etc).

4. *Filtering experimental data based on the current analytical task.* When investigating a subset of the experiment (e.g., a particular significant difference between two levels of a four-level variable), the tool should allow the removal of non-relevant information. Similar functionality should be enabled across other pivots of the experiment.

9.2.3 Input Data

As described earlier, an experiment is fundamentally a set of *trials*, measurements conducted for participants at different experimental conditions. Trials are the core data that

drives *VEEVVIE* and the ontology that we describe in the next section allows us to uniquely characterize them. Each trial occurs for a particular combination of levels for different independent variables (effectively at a cell of the experimental design matrix). Every trial is associated with a single experiment participant. This is a deliberate design decision since the majority of experiments that we have encountered are non-collaborative. Each trial also corresponds to a number of values for the dependent variables that are being observed. These variables can result from direct observations during a particular trial (e.g., the accuracy of a participant at a certain task) or from post-processing information that was collected during the trial (e.g., calculating travel distances based on motion tracking data). These measurements can be nominal, ordinal or interval, or they can be described in an application-specific data format. Additionally, several experiments (particularly ones that are based on repeated-measures designs) often require more than one trial per participant at each cell of the design matrix. These repeated trials are generally consistent across participants in order to reduce confounds. Thus, each trial is also characterized by a set of *grouping factors* that can describe these correlations. The trial descriptions, along with higher-level experiment meta-data (such as the totality of independent and dependent variables, their types, their potential values), experiment-wide measurements for each subject (e.g., pre- and post- experiment questionnaires) and descriptions of the statistical effects yielded by the analysis are the input data to *VEEVVIE*. It also worth noting that we have identified designs for which the statistical analysis is carried out within individual grouping factors (but without considering the factors themselves as an independent variable). This information is summarized in Table 3.

9.3 An Ontology for Describing Experiments

Given the above task definitions and input data, we designed an ontology that can represent a large number of experiments.

9.3.1 Core Classes

The primary building block of an ontology is the notion of a *class*. The *VEEVVIE* ontology defines several classes that allow the description of experimental data.

- *Experiment* - Instances of *Experiment* are objects that own the totality of information for a particular study.
- *Participant* - Instances of *Participant* describe individuals who partook in the experiment. *Participant* does not explicitly define any properties for demographical information. Rather, these (and other experiment-wide measurements) are modeled as instances of *ExperimentWideDependentVariable*.
- *IndependentVariable* - Class for describing the independent variables of the experiment. The valid levels of each variable are attributes to its instance.
- *DependentVariable* - Instances of *DependentVariable* describe a metric that is substantive to the experiment. The type of the measurement (ordinal, interval, nominal, or other custom types) are appended as attributes to instances of this class. Note that this is effectively an abstract class.
- *TrialDependentVariable* - Subclass of *DependentVariable*. Describes dependent variables, the values of which result from experimental trials (e.g., an accuracy or performance metric).
- *ExperimentWideDependentVariable* - Subclass of *DependentVariable*. Describes metrics, the values of which are only acquired once for each participant during the experiment (e.g., demographical data, pre/post study questionnaires, etc).
- *IndependentVariableValue* - Instances of this class describe a *fixed* value of a certain factor. These are referred to by instances of *Trial* in order to determine membership to a particular cell of the experimental design matrix.
- *DependentVariableValue* - Class for describing a measurement of a particular *DependentVariable*. The instances of this class are owned either by the source *Trial* or the *Subject* to which they correspond to (for *ExperimentWideDependentVariable*).
- *GroupingFactor* - Instances of *GroupingFactor* are referred to by trials in order to denote cross-trial groupings as outlined earlier.
- *Trial* - A trial brings together one or more *IndependentVariableValue* instances (denoting its membership in the experimental matrix), one or more *DependentVariableValue*, one *Participant* instance and, optionally, one or more *GroupingFactor* instances to describe a measurement event.

Table 4: Overview of how the *VEEVVIE* ontology can be used to capture the high-level design of experiments.

	Ball et al. [BNB07]	Papadopoulos et al. [PPKM15]	Laha et al. [LBS14]
Independent Variable	Display width, task	Display condition, task	Field of regard, stereoscopy, head tracking
Trial Dependent variable	performance timings, pan and zoom counts, physical navigation	Performance timing, scaled performance timing, physical navigation, mental effort	Performance grade, elapsed time, perceived difficulty and confidence
Experiment Wide Dependent Variable	Demographic data	Demographic data, infovis experience	Demographic data, VR experience, Post-experiment questionaire
Grouping Factor	Task repetitions	Task repetitions	Task identifiers

- *StatisticalEffect* - This class serves as a base for subclasses describing statistical effects. It points to a target *DependentVariable* and also refers to a number of *GroupingFactor* instances, for which the statistical analysis was conducted.
- *MainEffect* - This class describes a main effect of an instance of *IndependentVariable*. It also contains as attributes pairs of significant differences, as provided by the statistical analysis.
- *InteractionEffect* - Instances of *InteractionEffect* bring together two or more *IndependentVariables* (as provided by the statistical analysis) to describe an interaction on a *DependentVariable* (pointed by the base class).

The *VEEVVIE* ontology and relationships between classes are schematically illustrated in Figure 49.

9.3.2 Application to Experiments

Our ontology is applicable to experiments in the fields of visualization, HCI and VR. To illustrate this point, we have used it to model the experimental evaluations presented in several pieces of research work. These modelings can be seen in Table 4.

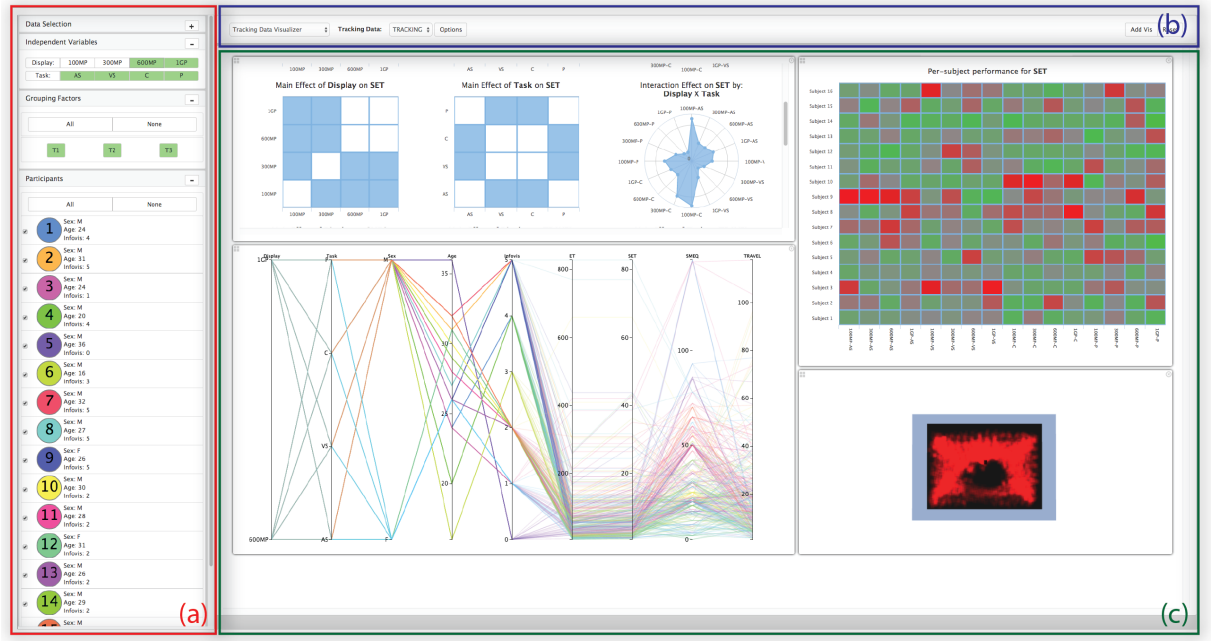


Figure 50: Sample view of the *VEEVVIE* front-end during a visual exploration session. The data filtering controls are visible in the sidebar on the left-hand side (a). These enable researchers to drill down data by toggling on/off factors of independent variables, grouping factors, and/or individual subjects. Visualizations can be generated through the toolbar at the top of the interface (b) by selecting a widget and then binding experimental measurements to dimensions. Widgets appear in the main workspace (c) and are moveable, resizeable and responsive. The four widgets visible in this image are: the statistical-effect explorer, the subject-centric visualizer, a parallel coordinates visualization of the experiment and a presence heatmap visualization.

9.3.3 Implementation

The *VEEVVIE* ontology is accessible through a series of REST-full [RR08] endpoints. The core web application functionality is implemented on top of NodeJS[2] , with the Sails.js[3] framework handling interactions with a MongoDB[4] database and enabling Create, Read, Update, Delete (CRUD) functionality.

9.4 The VEEVVIE Front-end

In this section, we describe the implementation of the *VEEVVIE* front-end, through which researchers can conduct visual analysis of experimental data modeled in our ontology. We illustrate some of the generic visualization modalities, the overall functionality of our interface, technical implementation details and also ways in which *VEEVVIE* can be extended to visualize experiment-specific data.

9.4.1 Layout and Functionality

The *VEEVVIE* front-end application, seen in Figure 50, defines three main interaction areas. On the left-hand side of the screen the user finds the persistent data selection controls, contained in a sidebar. The user can select an experiment to be loaded from the back-end. Following this selection, the left-hand sidebar populates with persistent data-filtering tools. Specifically, *VEEVVIE* allows researchers to drill-down into subsections of the experiment by excluding combinations of levels of independent variables, grouping factors and/or subjects from the visual analysis.

The main *VEEVVIE* workspace occupies the majority of the screen. At the top of the workspace lays the visualization generation toolbar which allows researchers to inject visualization widgets into the workspace. The user binds experiment measurements to different widget dimensions or axes and can also set various options. Widgets declare options and support for different types of measurements as described below. *VEEVVIE* widgets are responsive, resizable and can be moved around the workspace.

VEEVVIE leverages heavily the principles of coordinated multiple views [Rob07]. Through a standardized API, visualization widgets can request highlighting of combinations of trials, subjects, or independent variable factors which are propagated throughout the active view. Additionally, most *VEEVVIE* widgets support transient brushing operations [BC87] which trigger highlight operations. Custom widgets widgets can also access this functionality through a standardized API.

[2] http://www.nodejs.org
[3] http://www.sailsjs.org/
[4] https://www.mongodb.org

9.4.2 Implementation and Extensibility

The *VEEVVIE* front-end is implemented using standards-complaint HTML5, CSS and Javascript. We opted for a browser-based solution rather than developing a native application because it enables rapid development-testing-iteration cycles, cross-platform deployment and access to several open-source and extremely powerful visualization and interaction frameworks. The *VEEVVIE* front-end is structured as an AngularJS[5] application and makes heavy use of its data binding functionality for the propagation of active filtering, highlighting and widget state options. The visualizations are based on the Highcharts SDK[6] , Data-Driven Documents[7] and Kai Chang's D3.js parallel coordinates library[8] .

The architecture of *VEEVVIE* allows the development of visualization plugins without knowledge of the inner workings of the front-end or ontology back-end systems. The plug-in API defines several callback functions for creating a visualization, responding to resize events, reacting to and generating changes in filtering, highlighting, etc. These callbacks are implemented by *delegates*, which define the functionality of new visualizations. Delegates also declare the dimensions and corresponding supported data types for their widget and also any additional options that they might provide to the user. A sample declaration for one of the built-in widgets of our front-end can be seen in Figure 51. The front-end provides each delegate with a DOM element in which it can render its content and also several helper classes for managing access to experimental data, retrieving simple statistics and ensuring a consistent appearance scheme across widgets. We found that this affords flexibility in the widget development process. Widgets can be created separate from the relatively heavy-weight *VEEVVIE* front-end and then ported to the final system. A shortcoming of our current architecture is that widgets can not inject HTML and CSS declaratively into the front-end (this can be done programmatically). We plan on remedying this in the future by allowing delegates to declare paths to HTML templates and stylesheets that *VEEVVIE* injects automatically.

[5] https://www.angularjs.org
[6] https://www.highcharts.com
[7] https://www.d3js.org
[8] https://syntagmatic.github.io/parallel-coordinates/

```
SampleScatterplotWidget.declaration = {
 name: 'Sample Scatterplot Widget',
 delegate: 'SampleScatterplotWidgetDelegate',
 dimensions: [{
   description: 'X axis',
   supportedVariableTypes: ['interval'],
   repeatable: false
  },{
   description: 'Y axis',
   supportedVariableTypes: ['interval'],
   repeatable: false
  }],
 options: [{
     description: 'Brush behavior',
     values: [{
        description: 'Brush to Zoom'
       },{
        description: 'Brush to Highlight'
       }]
 }]};
```

Figure 51: Sample (simplified) declaration of a *VEEVVIE* widget. Widgets can declaratively define dimensions (including repeatable dimensions, e.g., for an arbitrarily dimensional visualization) and supported data types. Additionally, widgets can define custom options (e.g., toggle-able brushing behavior). This declarative approach allows widgets to integrate with and digest the *VEEVVIE* ontology in a way that is agnostics to the inner workings of the application.

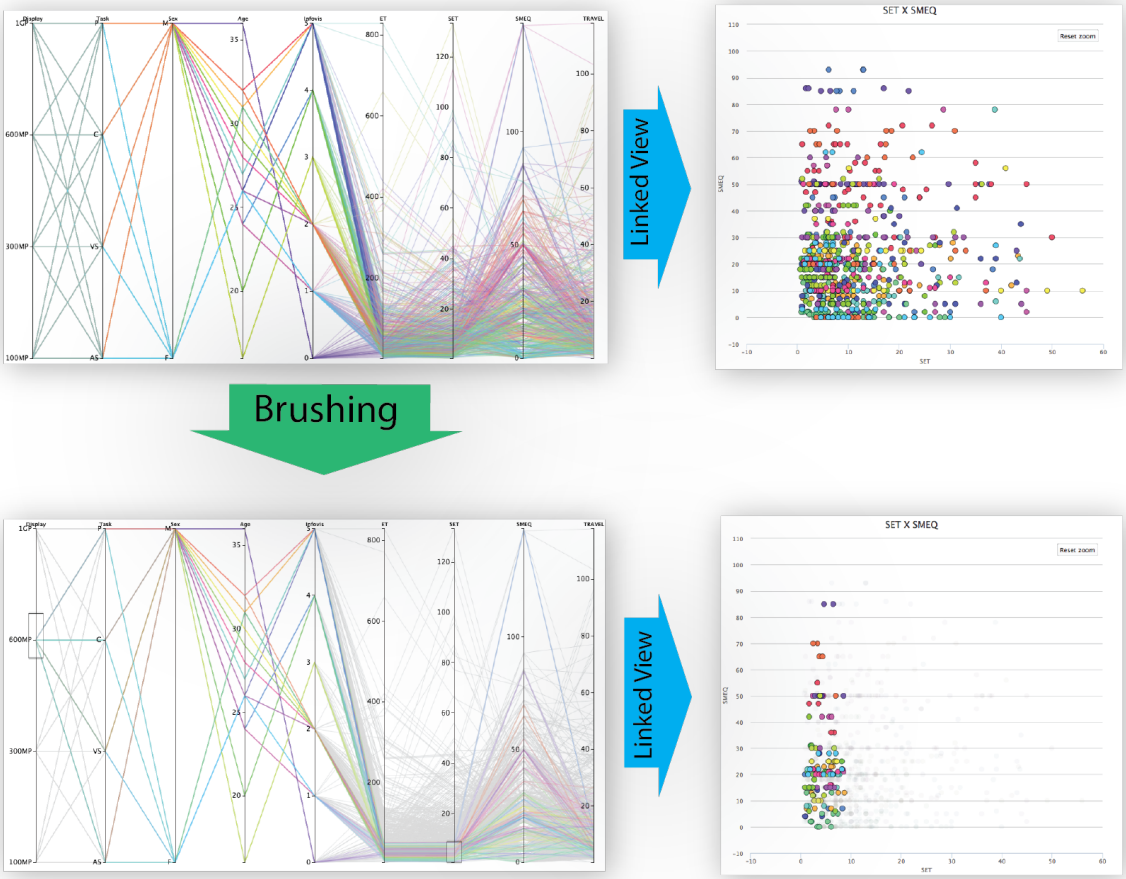


Figure 52: Illustration of coordinated linked views with brushing support. In this example, the parallel coordinates plot presents a high-level visual overview of the experimental data. A linked scatterplot visualizes two dependent variables in more detail. By brushing the parallel coordinates visualization, the user-researcher can drill down within the scatterplot.

9.4.3 Built-in Visualizations

Using the *VEEVVIE* widget architecture, we have implemented several visualization widgets that allow researchers to visually explore experiments that are centered around numerical or categorical dependent variables.

Multidimensional Visualization Widgets: Traditional multi-dimensional visualization is provided through two widgets. Researchers can create interactive scatterplots of trials, across arbitrary pairs of dependent variables (or experiment-wide measurements).

Scatterplots support zooming and brushing. Brushing on scatterplots highlights the selected trials across all active visualizations while hovering over a single datum with the mouse pointer will either highlight that single trial across the workspace or (if a modifier key is active) highlight all measurements traced to the subject of the current trial. Thus, researchers can visually identify outliers in the visualization of two performance metrics and then identify if these outliers also correlate to other aberrations in the source data.

We also provide a parallel coordinates [ID91] visualization widget which can be configured to display all or a subset of the measurements and/or independent variables of the experimental design. Individual polylines correspond to single trials. Experiment-wide measurements are concatenated with the per-trial measurements for the source subject in order to complete the descriptor vector for each polyline. Parallel coordinates provide a very effective way to generate a high-level aggregate view for the entire experiment. Additionally, through axial brushing researchers can highlight increasingly narrow sections of the experimental data (e.g., all trials from a certain combination of independent variable levels that fits within a particular performance threshold) which can in-turn be applied as filtering in order to reduce cognitive load during the analysis. The scatterplot and parallel-coordinates widgets included with *VEEVVIE* are illustrated in Figure 52.

Statistical Effect Explorer: As we described earlier, a critical task for the visual analysis of experimental data is the identification of the source of statistically significant effect. In order to facilitate this, *VEEVVIE* provides a widget that visualizes main and interaction effects that are described using its ontology. The statistical effect visualizer illustrates main effects in the form of a matrix, with cells corresponding on the presence (or not) of a significant difference between pairs of levels for a particular independent variable. Interaction effects are visualized in the form of radar charts. The combinations of levels for the independent variables of the interaction define the axis of the chart while the measurement for each axis corresponds to the mean of the dependent variable at that particular cell of the design. This approach allows to quickly visualize independent variable combinations with interesting measurement characteristics while scaling to (theoretically) arbitrary experimental designs. Both main and interaction effect visualizes support highlighting. For main effects, hovering over a particular cell of the visualization will highlight the two source populations across all supporting widgets in the workspace. Highlighting a

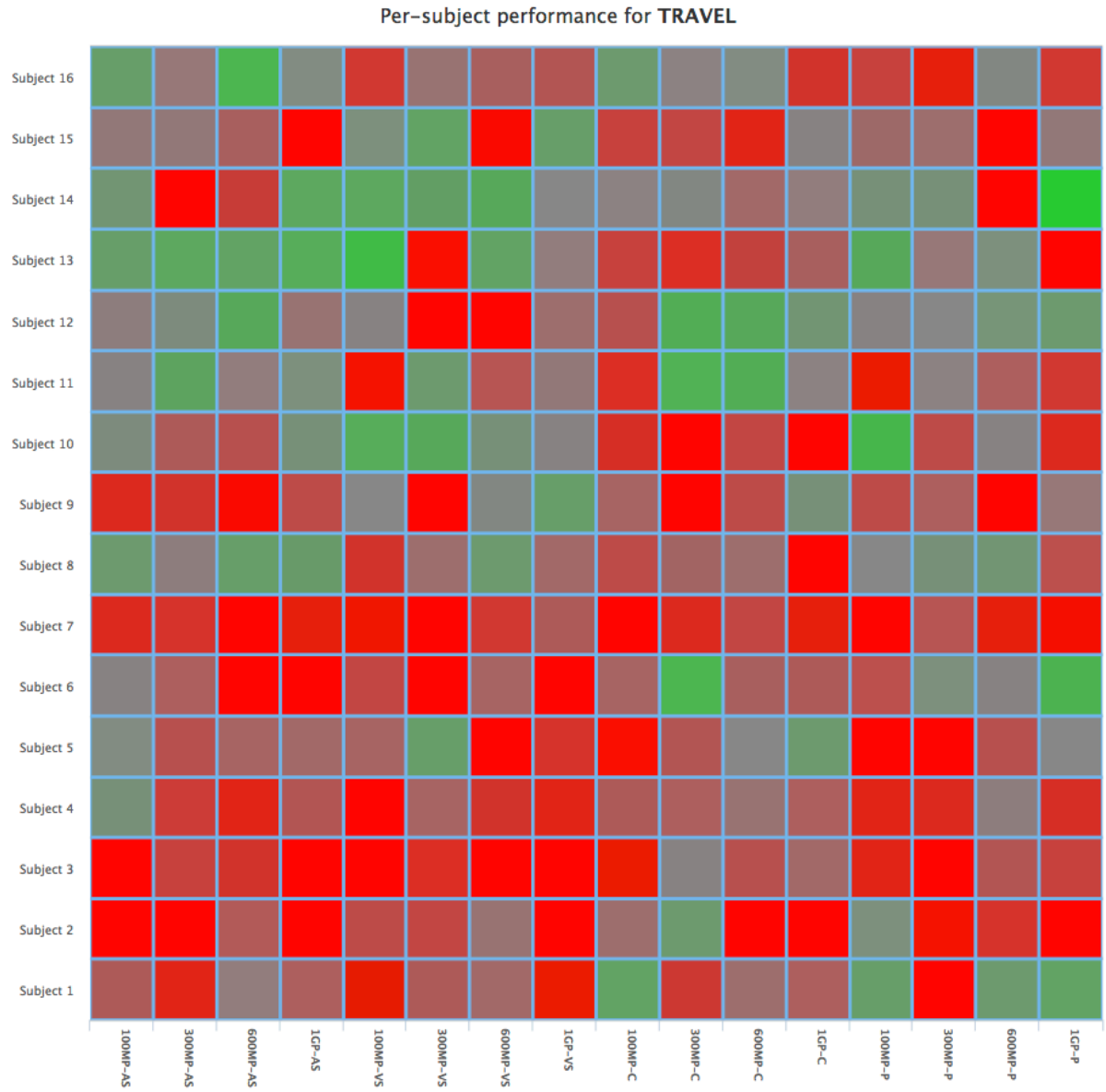


Figure 53: Participant-centric visualization of a physical navigation dependent variable. The columns of the matrix correspond to valid experimental factor combinations while rows correspond to individual experiment participants. Subject performance is color mapped from red to green depending on its deviation from the mean of each experimental condition. By hovering over each cell, the source trials are highlighted on other linked views.

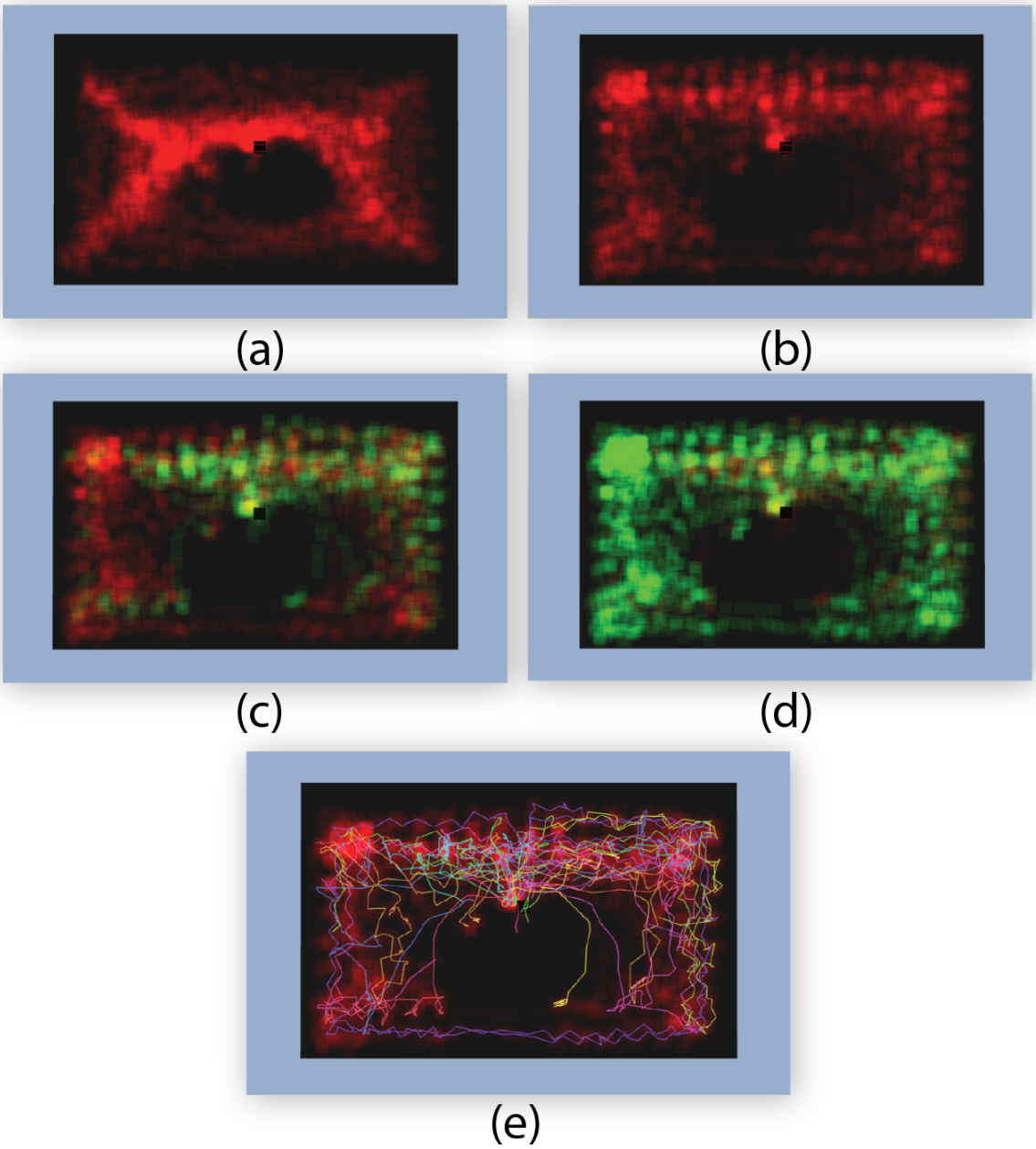


Figure 54: Sample views of the physical navigation visualization widget. This widget was developed using the *VEEVVIE* framework in order to visually analyze tracking data from our case study. (a) Presence heatmap for the **VS** task. (b) Presence heatmap for the **AS** task. (c) Heatmap highlight of a good performer for the **VS** task. (d) Heatmap highlight of a poor performer for the **VS** task. (e) Motion tracking paths for all participants for the **VS** task. All visualizations are generated for the **1G** display condition.

particular factor combination brings out the source data. In both cases, tooltips provide simple descriptive statistics. The functionality of the main effect visualizer is illustrated in Figure 55 while both sub-widgets can be seen also in Figure 50.

Subject-centric Visualizer: A frequent task during the exploration of experimental data is the identification of performance patterns across participants. For example, a researcher may want to quickly identify if some participants deviated significantly from the median of a measurement and then follow that up with an investigation of the root case. To facilitate this analysis approach, *VEEVVIE* provides a subject-centric visualizer. The researcher can select a dependent variable and is then presented with a heatmap visualization of the performance of all (filtered) participants across all experimental design levels. The heatmap cells are color-coded (red to green) to indicate deviation from the mean of the selected dependent variable at each particular experimental condition while hovering over a cell highlights the trials that satisfy the subject $\times$ condition constraint. An example of the subject-centric visualizer is shown in Figure 57.

9.5 Usage

In this section, we demonstrate the usage of *VEEVVIE* for the purposes of analyzing the experiment described in Chapter 8. We illustrate the application of *VEEVVIE* for three use-cases: *hypothesis validation*, *insight gathering* and *hypothesis generation*.

9.5.1 Experiment Description

The overarching goal of our experiment was to evaluate the scalability limits of large, high-resolution, immersive displays. We were motivated by prior research that illustrated the benefits of large displays in data visualization, but which did not scale those benefits to the absolutely massive display sizes and resolutions that are currently feasible. In pursuit of our goal, we designed an experiment which asked of 16 participants to complete four visualization tasks -attribute search (**AS**), visual search (**VS**), pattern finding (**P**) and comparisons (**C**). These tasks were conducted under four display resolutions, sizes and horizontal immersion fidelities - 100 megapixel planarly (**100MP**), 300 megapixel planarly

(**300MP**), 600 megapixel partially immersive (**600MP**) and 1 gigapixel fully immersive (**1GP**). Our main metrics concerned user performance, specifically elapsed time (**ET**) and scaled elapsed time (**SET**. We also measured subjective mental effort [ZH93] (**SMEQ**). Finally, we captured physical navigation data and calculated total physical navigation distance (**TRAVEL**). For a thorough exposition on the experimental design, tasks, data and analysis we refer the reader to Chapter 8.

9.5.2 Implementing a Custom Visualization Widget

The *VEEVVIE* widget architecture and ontology allowed us to develop an interactive visualization tool for displaying the physical navigation trajectories captured during our experiment. We defined an additional dependent variable using the *VEEVVIE* ontology (termed **TRACKING**) which also had a custom data type *trackingURI*. We then developed a visualization widget that declared its support for the *trackingURI* variable type. The *VEEVVIE* back-end stored and delivered the Unique Resource Identifiers that corresponded to the tracking trajectory files. These files were served via another web server and our new widget handled their download. The raw trajectories (which were recorded at 120Hz) were simplified using the Douglas-Peucker algorithm [DP73] in order to reduce file size and also remove jitter. The visualization was implemented using WebGL and the BabylonJS[9] scene graph. Our tracking visualizer widget can display trajectories and also generate simple heatmaps of user presence via quad-splatting. It also supports the *VEEVVIE* highlighting functionalities. Samples views of the tracking data visualizer widget can be see in Figure 54.

9.5.3 Case 1: Hypothesis Validation

Prior to our experiment, we hypothesized that larger (in terms of resolution and total visualized data) displays would enable users to more efficiently tackle core visualization tasks. Our hypotheses were largely motivated by prior experiments in the space. Our statistical analysis showed that performance, when scaled against the size of the visualization, did improve up until and including the **600MP** display condition. However the hypothesis did not hold for the **1GP** display condition. These results can be visually

[9]http://www.babylonjs.com

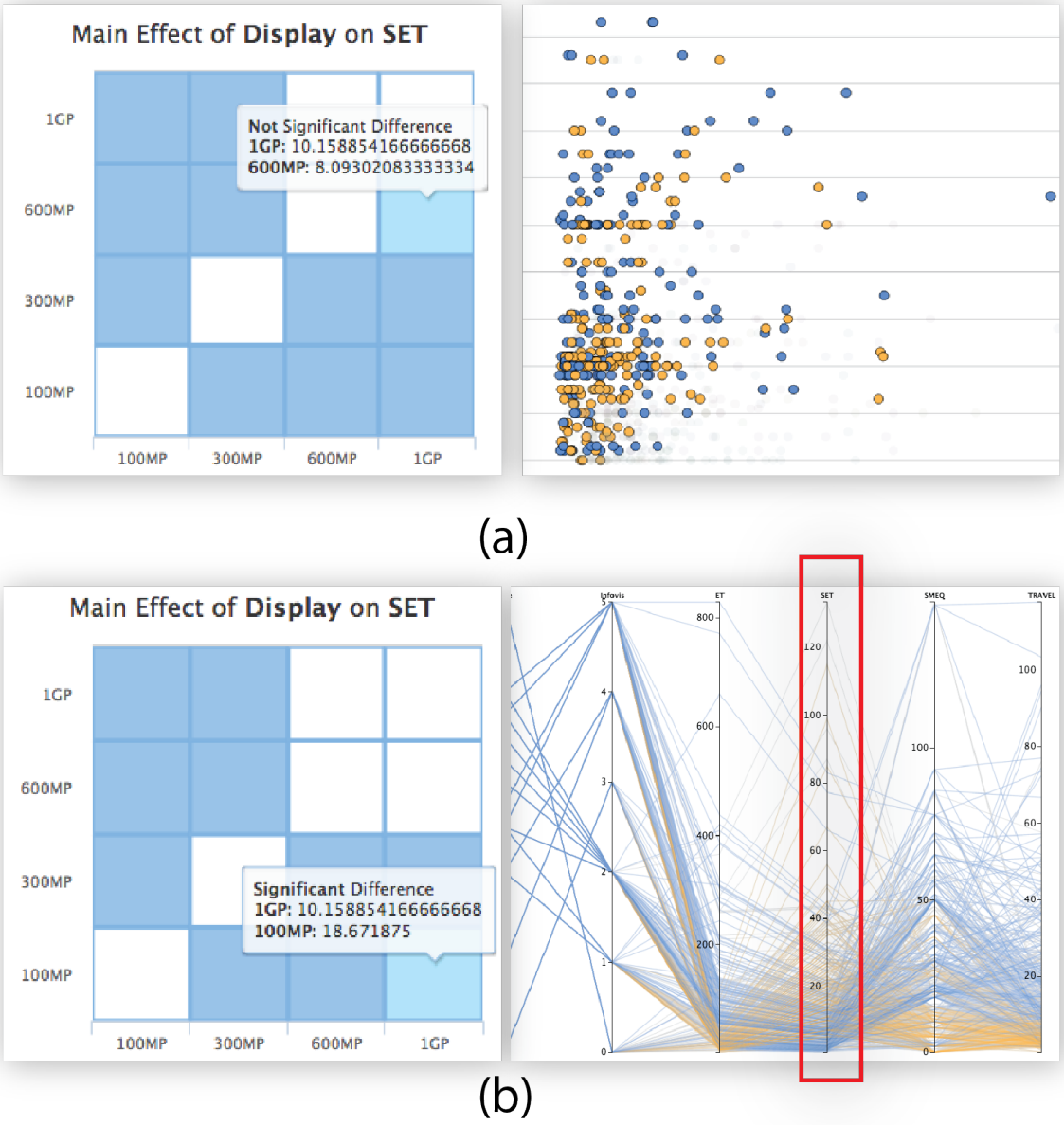


Figure 55: Illustration of utilizing *VEEVVIE* for the hypothesis validation. In this example, the analysis did not indicate a significant difference between the **600MP** and **1GP** display conditions. By utilizing the statistical effect explorer, the two populations can be highlighted in the linked scatterplot shown in (a) (**600MP** in orange, **1GP** in blue). Conversely, the **100MP** condition is highlighted in orange in the parallel coordinates plot (b) along with the significantly different **1GP** condition (visible in orange). The SET axis has been highlighted for emphasis and the visual pattern of the difference is visible.

verified using *VEEVVIE*. In Figure 55 we show results generated using *VEEVVIE*, by highlighting combinations of experimental conditions and looking at the distributions of the **SET** measurement. Notice the relatively clear boundary between the **100MP** and **1GP** populations but the much more diffuse distribution when looking at the **600MP** and **1GP** conditions. Similar visual patterns can be identified for most results of the statistical analysis. These patterns (and their outliers) motivate the insight gathering process. Visual hypothesis validation is illustrated in Figure 55.

9.5.4 Case 2: Insight Gathering

During the visual analysis of the results of our study, we wanted to delve into the driving factors behind substantial differences in performance across our sample, particularly in the largest display condition, for which our analysis did not indicate an overall improvement in performance. Using the filtering controls, we narrow the data scope to the **1G** display condition. Using the interaction effect explorer we can see that mean **SET** regressed for the **VS** task substantially. We further drill down in this particular task and attempt to understand this performance regression. The participant-centric visualizer widget offers a "compare against worse case" option which we leverage in order to quickly identify the most and least performant subjects in this experimental condition. We identify subjects 2 and 3 as being good performers and subjects 9 and 15 as performing poorly.

We then proceed to visualize the motion trajectories and presence heatmaps of these individuals. We observe that subjects 2 and 3 navigated substantially less. Contrarily, the poor performers travelled much more throughout the visualization facility, often several "rountrips" around the space and seemed to inspect individual columns of displays closely. This initial insight along with additional observations led the hypothesis generation use case that we describe

9.5.5 Case 3: Hypothesis Generation

The statistical analysis reported a main effect of the display condition on the **SMEQ** metric. We begin by examining the significant difference between the **300MP** and **1GP** display conditions and drill down to the **AS** task. A scatterplot of **SMEQ** $\times$ **TRAVEL** shows that several participants exhibited high **SMEQ** measurements at the **1GP** display

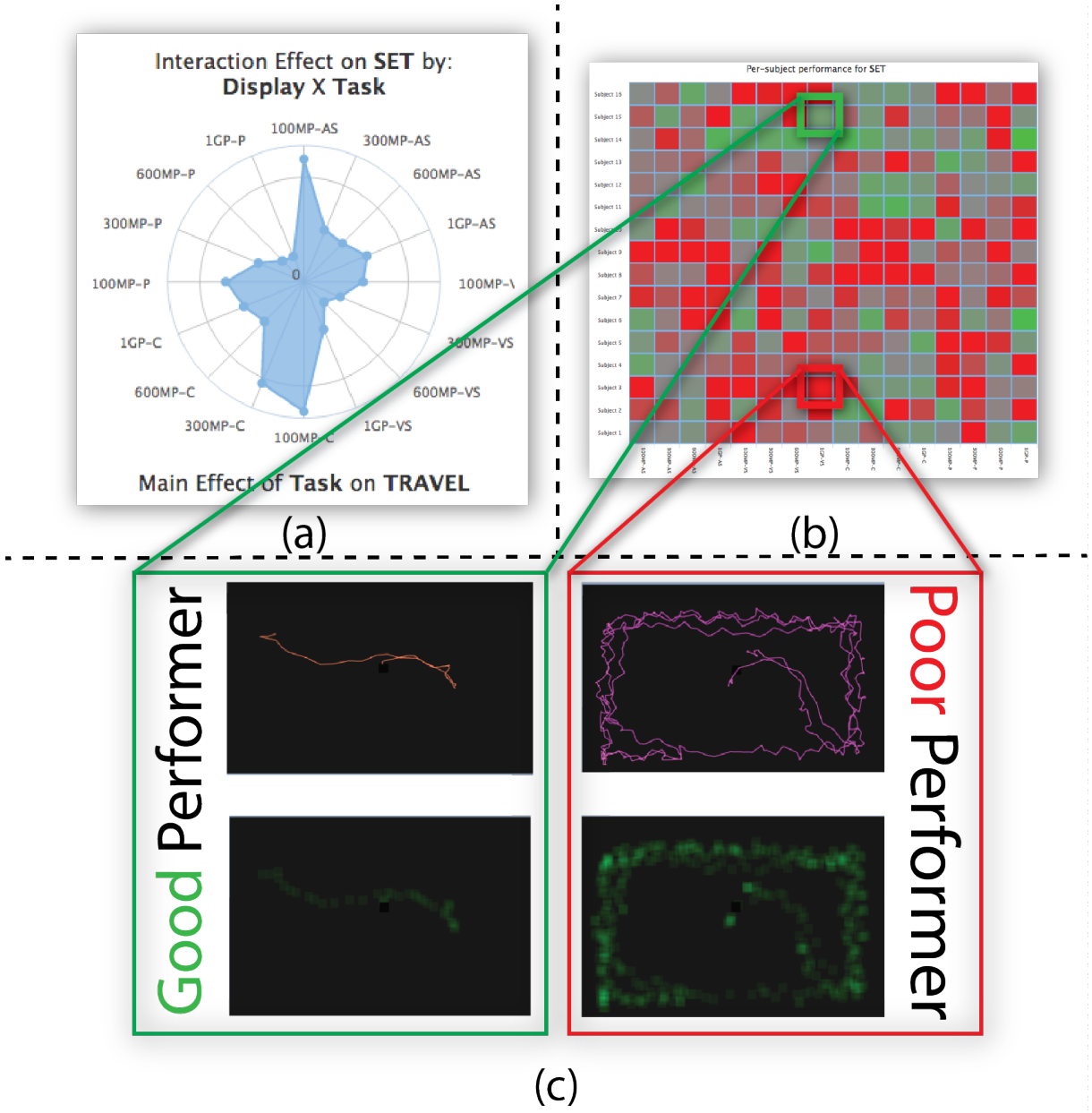


Figure 56: Leveraging *VEEVVIE* for insight development. The interaction effect explorer points out a regression in scaled performance (**SET**) for the **1GP-VS** independent variable level combination (a). This leads us to examine the worse-case performance across subjects and find exemplar good and poor performers (b). Sample physical navigation trajectories and presence heatmaps from the selected participants are visualized in (c). We notice that the good performer only navigates very little, standing near the center of the visualization space and only moving to confirm his response. Contrarily, the poor performer navigates much more as they frantically inspect individual columns of displays within the visualization facility, potentially resulting in poor performance.

condition. However, these measurements do not seem to tightly correlate with the amount of **TRAVEL**. In fact, we observe that several participants self-reported high **SMEQ** values ($\approx 70-80$) but within a wide range of physical navigation measurements (between $10m$ and $75m$), all generated at the **1GP** display condition. The three participants that generated those trials spanned a range of self-reported visualization experience (1, 2 and 5 on a $0-6$ scale). The subject-centric visualizer indicated that their performance was worse than the average within that particular design cell. Looking across the rows of the visualizer, we also observe that these participants generally underperformed in the **VS** task across all display conditions.

By visualizing the movement trajectories we can see a clear difference in how the approach to the task affects user frustration. In this case, we contrast one of the participants that self-reported high frustration with the task versus another participant that reported lower than average frustration. The non-frustrated user preferred to look for targets that answered the task query by observing displays from a distance. However, the highly-frustrated participant approached the displays and basically aligned themselves with the columns of the facility, individually inspecting visualization glyphs in search of an answer. Both participants reported the same experience in visualization.

This leads us to believe that the strategy with which users tackle problems within large display environments might be highly correlated to their eventual frustration. In this case, neither participant had prior exposure to any large-scale visualization system. Both participants performed approximately the same. However, one "enjoyed" their time in the facility much more than the other and, consequently, would be more likely to use it in the future. Thus, we feel that a future experiment should investigate this interrelation between large-display *strategies*, user performance and frustration. This process of visually driven hypothesis generation using *VEEVVIE* is illustrated in Figure 57.

9.6 Conclusion

This chapter introduced *VEEVVIE*, the Visual Explorer for Empirical Visualization, VR and Interaction Experiments. Our experience with *VEEVVIE* has allowed us to generate new insights and provided us with several investigative avenues for future research on human factors for large visualization systems. We hope to eventually release *VEEVVIE*

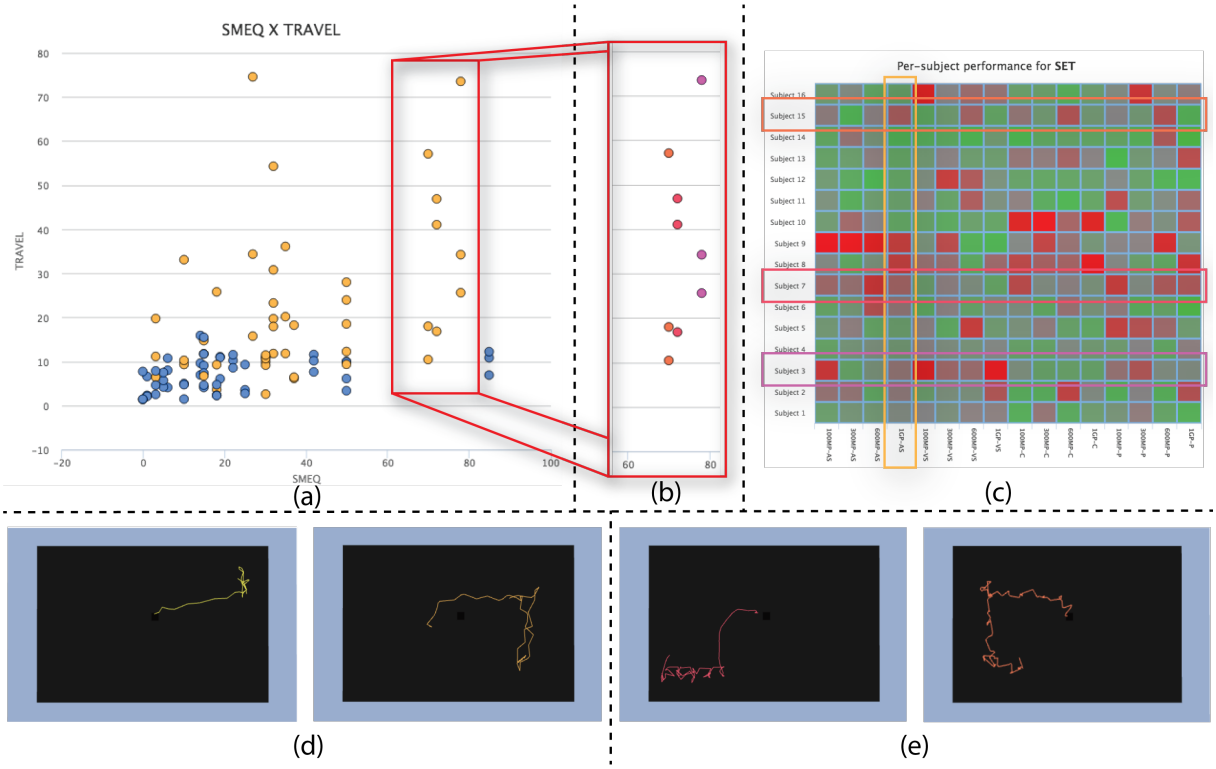


Figure 57: Illustration of the generation of a new hypothesis using *VEEVVIE*. An **SMEQ** × **TRAVEL** scatterplot (a), highlighted per display condition (orange for **1GP**, blue for **300MP**) shows that the larger display condition three users (b) exhibited similarly high frustration for varying levels of physical navigation. Using the participant-centric visualizer we can observe that overall performance for all these subjects was below average (c). The physical movement of these frustrated performers (e) is contrasted with that of some other below-average performers that reported substantially lower **SMEQ** readings (d). The patterns in (d) are much simpler while the patterns in (e) appear substantially more involved, with users moving close to the displays. These different "strategies" in tackling the visual search problem and their correlation to user frustration can form the basis for a future experimental hypothesis.

to the visualization, VR and HCI communities so that our colleagues can utilize interactive visual analysis as a tool in their experimental processes.

Chapter 10

Conclusions

The goal of this chapter is to briefly summarize the contributions outlined through this dissertation and provide short and long-term directions for future research that build upon this body of work.

10.1 Summary of Contributions

This dissertation has presented contributions focusing on several aspects of user interactions with room-sized, gigapixel-resolution, immersive displays. These contributions are enabled by the Reality Deck facility, a gigapixel resolution display which is described in detail in Chapter 3. Additionally, the majority of this work is implemented on a flexible distributed visualization software stack that is described in Chapter 4. This combination of cutting-edge hardware and software has allowed the author of this thesis to make the following contributions:

- The NuNav3D body-driven navigation (described in Chapter 5), which was implemented on a Kinect sensor and evaluated for two different task types. NuNav3D exposed 4 degrees of freedom simultaneously, allowing users to perform complex navigation maneuvers in 3D without requiring a dedicated controller device. While the evaluation of NuNav3D revealed certain shortcomings (particularly with regard to navigating in constrained spaces), it proved the viability of body-driven, unencumbered navigation interfaces.

- A chirocentric (or hand-centric) user interface platform for immersive environments, exposed in Chapter 6. Motivated by the shortcomings of NuNav3D (particularly the unintended navigations observed during our user study), this platform utilizes machine learning algorithms for the detection of hand poses and the identification of hand gestures (trajectories of the user's hands over time). Hand poses are correlated to certain interaction and navigation modalities, allowing users to explicitly declare navigation intent while gestures can trigger actions in the visualization. Using this platform, we implemented a chirocentric 3D user interface prototype which was deployed in the Reality Deck.

- A unique optimization algorithm that leverages the de-facto occurring physical navigation within an LHiRD to improve performance. This acuity-driven gigapixel visualization system (described in Chapter 7) takes advantage of the perceptual limitations of the human visual system in order to dynamically adjust level of detail settings in different rendering modalities. Through its perceptual basis, this technique operates in a manner that is mostly invisible to the user, without affecting their ability to appreciate finer details within the visualization (as was shown by our user study). Meanwhile, it provides significant frame-rate and streaming savings through its operation.

- A user study aimed at quantifying the scalability limits of room-sized immersive displays. Prior work which evaluated the scalability of these systems was only applicable to relatively small designs (of roughly 100 megapixels). In Chapter 8 we describe a user study which attempts to quantify the scalability of immersive, high-resolution displays, starting from a planar, 100 megapixel form-factor and scaling to a near fully-immersive arrangement with a resolution of 1 gigapixel. Our results showed that performance does not scale arbitrarily, but rather that there exists a point of diminishing returns, after then 600 megapixel resolution point, past which performance does not improve in a significant way. Additionally, by visualizing movement patterns within the Reality Deck, we uncovered two distinct approaches to these visualization tasks, termed "overview" and "detail" with implications on user performance.

- The commonplace application of visualization as part of the analysis of the results of empirical experiments is the driving force for the final contribution of this thesis. In

Chapter 9 we describe *VEEVVIE*, the Visual Explorer for Empirical Visualization, VR and Interaction Experiments. *VEEVVIE* is a visual analytics framework which allows researchers to explore data collected from hypothesis-driven experiments in an interactive and extendable way. *VEEVVIE* is driven by an ontology which can model several common experimental designs, while the web-based front-end offers several familiar visualization modalities and allows researchers to implement widgets that can ingest their custom data. *VEEVVIE* is demonstrated through use cases that we draw from the study presented in Chapter 8.

10.2 Future Work

As a conclusion to this thesis, the author presents several directions for future research work, both short-term (as a continuation of the contributions presented herein) and also longer term, within the broader area of visualization and interaction with room-sized displays.

10.2.1 Short-term Guidance

With regard to chirocentric user interfaces, there exists relatively little research in the "best" or most "natural" hand-centric user interactions. What are the best gestures that a system should support? Which hand pose is appropriate to activate a flythrough interaction? Such notions are relatively well established for two-dimensional UIs (especially on touch-enabled devices), however the research in the 3D user interface domain barely scratches the surface. Our chirocentric UI implementation offers a platform on which this sort of investigation can be conducted. On the technical side, our system can benefit from greater robustness, particularly in the recognition of gestures. An additional concern is the determination of the "end-point" of a gesture. Currently our system triggers the corresponding action as soon as the current frame window gets classified by the SVM, which can result in inaccuracies when gestures have similar dynamics. Work by Hoai and De la Torre [HDlT12] addresses this problem and we plan on exploring its applicability to gesture recognition in an interactive setting. Finally, the chirocentric user interface can be expanded so that includes support for additional types of gestures. More complex gestures or even chords of gestures can greatly expand the interaction opportunities of the

system. Similarly, an expanded hand pose dictionary can allow the manipulation of various non-binary visualization parameters. For example, in a volume rendering scenario, a certain hand pose can enable a clipping plane manipulation mode, which is then modified through the user moving their secondary hand. Naturally, expanding the gesture and pose vocabulary requires the acquisition of additional training data for our classifiers. We wish to explore the avenue of synthetic training data (based on exemplars), which has proven to be quite powerful in a number of applications and would allow our techniques to generalize to a wider audience.

The acuity-driven gigapixel visualization scheme system can be enhanced in a number of ways. For instance it could be expanded in order to support multi-user scenarios. Another research avenue is the implementation of a predictive approach to LoD selection, accounting for the user's trajectory within the visualization system. Finally, an unexplored, yet quite-relevant area is that of gigapixel resolution video streaming over bandwidth constrained networks. Sensors such as the AWARE cameras [MST+11] are currently being realized and will soon be commercially available. The ulterior goal of these research projects is the capture of real-time video and naturally a gigapixel resolution display is an obvious visualization system for such an image stream. Our acuity-driven gigapixel level-of-detail algorithm could be used to reduce bandwidth costs in these situations.

While the display scalability study described in this dissertation showed diminishing returns in user performance past a certain resolution, it does not offer a definitive answer to the question "Does size really matter?". Our study did not decouple resolution from immersion (since those two variables are physically coupled together for the Reality Deck). A future study on this topic would probably be of tremendous help to builders of new LHiRDs and other large visualization systems. Additionally, the "overview" and "detail" navigation patterns, and the between-user differences in choosing these patterns for different tasks, present an interesting human-factor research avenue. Finally, our study focused on single-user visualization sessions. Another promising avenue for future evaluation is that of collaborative data exploration within the confines of a room-sized LHiRD such as the Reality Deck. If a one gigapixel display is overwhelming for one user but four 250 megapixel displays are not, then maybe the right way to use these large facilities is "divide and conquer".

Finally, we have identified several features that we would like to incorporate into future

versions of *VEEVVIE*. For example, the current filtering and highlighting functionality can be enhanced by providing users with an interface for managing complex queries. Additionally, visual analytics applications have sometimes offered domain-specific query languages which can allow expert uses to more quickly delve into sections of the data. The *VEEVVIE* ontology can be used to support such a language for experimental designs. Alternatively, such functionality could be exposed through a visual query builder. We plan on expanding the functionality of the built-in visualization widgets and also incorporate new ones based on feedback. Additionally, our current system is geared towards single-user experiments, since they are most commonly encountered in literature. While collaborative experiments could be modeled with small modifications to our ontology, we feel that their visualization presents unique challenges and would like to explore it in the future. Finally, we wish to identify experimental designs that can not be captured by the existing *VEEVVIE* ontology, in an effort to make our contribution even more generalizable.

10.2.2 Long-term Guidance

With the advent of consumer-priced, acuity-saturating head-mounted displays, one might be inclined to dismiss LHiRDs as a "dying breed" of visualization facility. Why spend hundreds of thousands of dollars of a tiled display when you can simulate one with a $500 HMD? In fact, during the six years of PhD studies that culminated in to this thesis, the author was asked several times if he thought that LHiRDs were soon to become obsolete by the advent of cheap and high-quality VR hardware.

While the author of this dissertation is fairly confident in the proliferation of HMDs over the coming years, these devices present (and will continue to present) several fundamental presence issues which hamper ergonomics and constrain opportunities for collaboration. Additionally, even if those issues were to be solved, and HMDs eventually simulated room-sized visualizations, the fundamentals of interactions with these virtual LHiRDs would probably be similar to those of their real-world counterparts. Thus, the author of this dissertation feels that LHiRDs will prove to be a fruitful long-term research topic.

If anything, existing work in the field of visualization and interaction with LHiRDs barely scratches the surface of what these devices can offer. Most evaluations are centered around simple visualizations in single-user situations. When these systems are deployed in production settings (industrial or otherwise) they are often used as demonstration

platforms or simply as large-surface displays. However, this thesis and other research works have shown that LHiRDs are fundamentally different than a desktop display with a wall-sized surface area. Different users approach them in different ways, their size requires that users move around in order to ingest the totality of the data and enables new ways of interactions with the displayed content. Thus, why is it that, despite their increased flexibility, LHiRDs are used mostly as large desktop displays? The author of this thesis feels that there exist three causes to this issue. The first reason involves the lack of robust tools for developing applications for these systems. While there exist a wide range of frameworks for distributed rendering and visualization, they all present different shortcomings (in terms of complexity, ease of development, portability, etc). Second, there is no unified hardware platform for LHiRDs, since they are either designed and built as one-off research projects or in small numbers by specialized vendors. Third, there exists no consensus on what an "application" looks and works like on an LHiRD. While the desktop computer settled on the WIMP (Window Icon Menu Pointer) paradigm and mobile devices settled on single-window multi-touch, this uniformity has not been reached for LHiRDs. Applications can range from large-scale WIMP-style designs to first-person VR-style viewers, to gesture or proxemics-controlled prototypes to systems that utilize a tablet for input. These three shortcomings of the platform, feed into each other, in a sort of vicious circle which prevents the mass proliferation of LHiRDs.

In this author's opinion, the single most important research direction for LHiRDs is standardization. What is desperately needed is the "CocoaTouch" or "Win32" for large, high-resolution, displays. A single software platform and underlying hardware specification that will allow developers to build fully fledged applications for these systems and integrators to construct LHiRD designs that can ingest content from a variety of sources. It is through standardization that PCs became a powerful appliance, accessible and useful to billions of people. As the hardware that comprises LHiRDs becomes more and more affordable, it is through this sort of standardization that these wonderful systems can affect the lives of those that do not live next door to a top tier university or a national research laboratory.

Bibliography

[AABW12] Gennady Andrienko, Natalia Andrienko, Michael Burch, and Daniel Weiskopf. Visual analytics methodology for eye movement studies. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2889–2898, 2012.

[ABM+97] Maneesh Agrawala, Andrew C. Beers, Ian McDowall, Bernd Froehlich, Mark Bolas, and Pat Hanrahan. The two-user responsive workbench: support for collaboration through individual views of a shared space. *24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 327–332, 1997.

[ACP10] Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. High-precision magnification lenses. *28th International Conference on Human Factors in Computing Systems*, pages 273–282, 2010.

[APPC12] Rodrigo A. de Almeida, Clement Pillias, Emmanuel Pietriga, and Pierre Cubaud. Looking behind bezels: french windows for wall displays. *International Working Conference on Advanced Visual Interfaces*, pages 124–131, 2012.

[AR11] JK Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43(3):16, 2011.

[BBGV11] Amartya Banerjee, Jesse Burstyn, Audrey Girouard, and Roel Vertegaal. Waveform: remote video blending for vjs using in-air multitouch gestures. *Human Factors in Computing Systems (Extended Abstracts)*, pages 1807–1812, 2011.

[BBL93] Thomas Baudel and Michel Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Communications of the ACM*, 36(7):28–35, 1993.

[BC87] Richard A Becker and William S Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[BCF+08] Doug A. Bowman, Sabine Coquillart, Bernd Froehlich, Michitaka Hirose, Yoshifumi Kitamura, Kiyoshi Kiyokawa, and Wolfgang Stuerzlinger. 3d user interfaces: New directions and perspectives. *IEEE Computer Graphics and Applications*, 28(6):20–36, 2008.

[BE13] Tanja Blascheck and Thomas Ertl. Techniques for analyzing empirical visualization experiments through visual methods. *KI Workshop on Visual and Spatial Cognition*, pages 44–51, 2013.

[BGS+12] David J. Brady, Michael Gehm, Ronald Stack, Daniel Marks, David Kittle, Dathon R. Golish, Esteban Vera, and Steven D. Feller. Multiscale gigapixel photography. *Nature*, 486(7403):386–389, 2012.

[BH06] Leonard D. Brown and Hong Hua. Magic lenses for augmented virtual environments. *IEEE Computer Graphics and Applications*, 26(4):64–73, 2006.

[BI12] Anastasia Bezerianos and Petra Isenberg. Perception of visual variables on tiled wall-sized displays for information visualization applications. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2516 – 2525, 2012.

[BJ05] Rajvikram Singh Andrew Johnson Jason Leigh Byungil Jeong, Luc Renambot. High-performance scalable graphics architecture for high-resolution displays. Report, Electronic Visualization Laboratory - University of Illinois, Chicago, 2005.

[BJH+01] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. VR juggler: A virtual platform for virtual reality application development. *IEEE Virtual Reality*, pages 89–96, 2001.

[BN05] Robert Ball and Chris North. Effects of tiled high-resolution display on basic visualization and navigation tasks. *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 1196–1199, 2005.

[BN07] Robert Ball and Chris North. Realizing embodied interaction for visual analytics through large displays. *Computers & Graphics*, 31(3):380–400, 2007.

[BN08] Robert Ball and Chris North. The effects of peripheral vision and physical navigation on large scale visualization. *Graphics Interface*, pages 9–16, 2008.

[BNB07] Robert Ball, Chris North, and Doug A. Bowman. Move to improve: promoting physical navigation to increase user performance with large displays. *SIGCHI Conference on Human Factors in Computing Systems*, pages 191–200, 2007.

[Bol80] Richard A Bolt. put-that-there: Voice and Gesture at the Graphics Interface. 14(3):262–270, 1980.

[BRP05] Ragnar Bade, Felix Ritter, and Bernhard Preim. Usability comparison of mouse-based interaction techniques for predictable 3D rotation. *International Symposium on Smart Graphics*, pages 138–150, 2005.

[BSH09] Gerd Bruder, Frank Steinicke, and Klaus H. Hinrichs. Arch-explore: A natural user interface for immersive architectural walkthroughs. *IEEE Symposium on 3D User Interfaces*, pages 75–82, 2009.

[BSP+93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. *SIGGRAPH*, pages 73–80, 1993.

[CAN14] Haeyong Chung, Christopher Andrews, and Chris North. A survey of software frameworks for cluster-based large high-resolution displays. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1158–1177, 2014.

[CBP08] Marcelo Cohen, Ken W. Brodlie, and Nick Phillips. The volume in focus: Hardware-assisted focus and context effects for volume visualization. *ACM*

Symposium on Applied Computing, pages 1231–1235, 2008.

[CL11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[CLP04] Sheelagh Carpendale, John Ligh, and Eric Pattison. Achieving higher magnification in context. *17th Annual ACM Symposium on User Interface Software and Technology*, pages 71–80, 2004.

[CM01] Sheelagh Carpendale and Catherine Montagnese. A framework for unifying presentation space. *14th Annual ACM Symposium on User Interface Software and Technology*, pages 61–70, 2001.

[CMN11] Oliver Cossairt, Daniel Miau, and Shree Nayar. Gigapixel computational imaging. *IEEE International Conference on Computational Photography*, pages 1–8, 2011.

[CNSD+92] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Communications of the ACM*, 35(6):64–72, 1992.

[CW11] Douglas W Cunningham and Christian Wallraven. *Experimental Design: From User Studies to Psychophysics*. CRC Press, 2011.

[CWG+02] Han Chen, Grant Wallace, Anoop Gupta, Kai Li, Tom Funkhouser, and Perry Cook. Experiences with scalability of display walls. *Immersive Projection Technology Symposium*, 2002.

[DDS+09] Thomas A. DeFanti, Gregory Dawe, Daniel J. Sandin, Jurgen P. Schulze, Peter Otto, Javier Girado, Falko Kuester, Larry Smarr, and Ramesh Rao. The StarCAVE, a third-generation CAVE and virtual reality OptIPortal. *Future Generation Computer Systems*, 25(2):169–178, 2009.

[DK10] Kai-Uwe Doerr and Falko Kuester. Cglx: A scalable, high-performance visualization framework for networked display environments. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):1077–2626, 2010.

[DP73] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[DRCB05] Piotr Dollr, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. *International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

[EALN11] Alex Endert, Christopher Andrews, Yueh Hua Lee, and Chris North. Visual encodings that support physical navigation on large displays. *Graphics Interface*, pages 103–110, 2011.

[EBZ+12] Alex Endert, Lauren Bradel, Jessica Zeitz, Christopher Andrews, and Chris North. Designing large high-resolution display workspaces. *International Working Conference on Advanced Visual Interfaces*, pages 58–65, 2012.

[EFH79] Ruth B Ekstrom, John W French, and Harry H Harman. Cognitive factors: Their identification and replication. *Multivariate Behavioral Research Monographs*, 1979.

[EMP09] Stefan Eilemann, Maxim Makhinya, and Renato Pajarola. Equalizer: A scalable parallel rendering framework. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):436–452, 2009.

[ETO+10] Achim Ebert, Sebastian Thelen, Peter-Scott Olech, Joerg Meyer, and Hans Hagen. Tiled++: An enhanced tiled hi-res display wall. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):120–132, 2010.

[Fis92] Ronald A Fisher. *The Arrangement of Field Experiments*, pages 82–91. Springer, 1992.

[FNM+14] Alessandro Febretti, Arthur Nishimoto, Victor Mateevitsi, Luc Renambot, Andrew Johnson, and Jason Leigh. Omegalib: A multi-view application framework for hybrid reality display environments. *IEEE Virtual Reality*, pages 9–14, 2014.

[FNT+13] Alessandro Febretti, Arthur Nishimoto, Terrance Thigpen, Jonas Talandis, Lance Long, JD Pirtle, Tom Peterka, Alan Verlo, Maxine Brown, and Dana Plepys. Cave2: a hybrid reality environment for immersive simulation and information analysis. *IS&T SPIE Electronic Imaging*, pages 864903–864903–12, 2013.

[Fur86] George W. Furnas. Generalized fisheye views. *SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, 1986.

[GD95] Dariu M. Gavrila and Larry Davis. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. *International Workshop on Automatic Face and Gesture Recognition*, pages 272–277, 1995.

[GFD+12] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM Transactions on Graphics*, 31(6):164, 2012.

[GKN05] Emden R. Gansner, Yehuda Koren, and Stephen North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, 2005.

[GWB04] Tovi Grossman, Daniel Wigdor, and Ravin Balakrishnan. Multi-finger gestural interaction with 3d volumetric displays. *ACM Symposium on User Interface Software and Technology*, pages 61–70, 2004.

[HBEH00] Greg Humphreys, Ian Buck, Matthew Eldridge, and Pat Hanrahan. Distributed rendering for scalable displays. *ACM/IEEE Conference on Supercomputing*, 2000.

[HDKG08a] Martin Hachet, Fabrice Decle, Sebastian Knodel, and Pascal Guitton. Navidget for easy 3D camera positioning from 2D inputs. *IEEE Symposium on 3D User Interfaces*, pages 83–89, 2008.

[HDKG08b] Martin Hachet, Fabrice Decle, Sebastian Knodel, and Pascal Guitton. Navidget for immersive virtual environments. *Proceedings ACM symposium on Virtual Reality Software and Technology*, pages 47–50, 2008.

[HDlT12] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. *IEEE Conference on Computer Vision and Pattern Recognition*, pages

2863–2870, 2012.

[Hec83] Paul S Heckbert. Texture mapping polygons in perspective. *NYIT Computer Graphics Lab Technical Memo*, 13, 1983.

[HH10] Charles Han and Hugues Hoppe. Optimizing continuity in multiscale imagery. *ACM Transactions on Graphics*, 29(6):1–10, 2010.

[HHN+02] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter Kirchner, and James T. Klosowski. Chromium: A stream processing framework for interactive rendering on clusters. *ACM Transactions on Graphics*, 21(3):693–702, 2002.

[HMB11] Georg Hackenberg, Rod McCall, and Wolfgang Broll. Lightweight palm and finger tracking for real-time 3d gesture control. *IEEE Virtual Reality*, pages 19–26, 2011.

[HPLVdW10] Charles Hollemeersch, Bart Pieters, Peter Lambert, and Rik Van de Walle. Accelerating virtual texturing using cuda. *GPU Pro: Advanced Rendering Techniques*, pages 623–641, 2010.

[ID91] Alfred Inselberg and Bernard Dimsdale. *Parallel coordinates*, pages 199–233. Springer, 1991.

[IV11] Cheuk Yiu Ip and Amitabh Varshney. Saliency-assisted navigation of very large landscape images. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1737–1746, 2011.

[JH13] Mikkel R. Jakobsen and Kasper Hornbaek. Interactive visualizations on large and small displays: The interrelation of display size, information space, and scale. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2336–2345, 2013.

[JJR+05] Byungil Jeong, Ratko Jagodic, Luc Renambot, Rajvikram Singh, Andrew Johnson, and Jason Leigh. Scalable graphics architecture for high-resolution displays. *IEEE Information Visualization Workshop*, 2005.

[KBF+95] Wolfgang Kruger, Christina-A. Bohn, Bernd Frohlich, Heinrich Schuth, Wolfgang Strauss, and Gerold Wesche. The responsive workbench: a virtual work environment. *Computer*, 28(7):42–48, 1995.

[KG04] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568, 2004.

[KH14] Torsten Wolfgang Kuhlen and Bernd Hentschel. Quo vadis cave: Does immersive visualization still matter? *IEEE Computer Graphics and Applications*, 34(5):14–21, 2014.

[KKB+11] Alexander Kulik, Andr Kunert, Stephan Beck, Roman Reichel, Roland Blach, Armin Zink, and Bernd Froehlich. C1x6: a stereoscopic six-user display for co-located collaboration in shared virtual environments. *ACM Transactions on Graphics*, 30(6):1–12, 2011.

[KKKA13] Cem Keskin, Furkan Kra, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. *Consumer Depth Cameras for Computer Vision*, pages 119–137, 2013.

[KLBL93] Robert S Kennedy, Norman E Lane, Kevin S Berbaum, and Michael G Lilienthal. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3(3):203–220, 1993.

[KLKB05] Arie E. Kaufman, Sarang Lakare, Kevin Kreeger, and Ingmar Bitter. Virtual colonoscopy. *Communications of the ACM*, 48(2):37–41, 2005.

[KMS+11] Arie E. Kaufman, Klaus Mueller, Dimitris Samaras, Hong Qin, Amitabh Varshney, Charilaos Papadopoulos, and Kaloian Petkov. The realitydeck - immersive giga-pixel display. *CEWIT International Conference & Expo on Emerging Technologies for a Smarter World*, 2011.

[KOJL+14] Paul Klemm, Steffen Oeltze-Jafra, Kai Lawonn, Katrin Hegenscheid, Henry Volzke, and Bernhard Preim. Interactive visual analysis of image-centric cohort study data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1673–1682, 2014.

[KRF11] Anette von Kapri, Tobias Rick, and Steven Feiner. Comparing steering-based travel techniques for search tasks in a cave. *IEEE Virtual Reality*, pages 91–94, 2011.

[KUDC07] Johannes Kopf, Matt Uyttendaele, Oliver Deussen, and Michael F Cohen. Capturing and viewing gigapixel images. *ACM Transactions on Graphics*, 26(3):93, 2007.

[LBS14] Bireswar Laha, Doug A. Bowman, and John J. Socha. Effects of vr system fidelity on analyzing isosurface visualization of volume datasets. *IEEE Transactions on Visualization and Computer Graphics*, 20(4):513–522, 2014.

[LCBL+14] Can Liu, Olivier Chapuis, Michel Beaudouin-Lafon, Eric Lecolinet, and Wendy E Mackay. Effects of display size and navigation type on a classification task. *32nd International Conference on Human Factors in Computing Systems*, pages 4147–4156, 2014.

[LCC+00] Kai Li, Han Chen, Yuqun Chen, Douglas W. Clark, Perry Cook, Stefanos Damianakis, Georg Essl, Adam Finkelstein, Thomas Funkhouser, Timothy Housel, Allison Klein, Zhiyan Liu, Emil Praun, Rudrajit Samanta, Ben Shedd, Jaswinder Pal Singh, George Tzanetakis, and Jiannan Zheng. Building and using a scalable display wall system. *IEEE Computer Graphics and Applications*, 20(4):29–37, 2000.

[LDK+08] Qing Luan, Steven M. Drucker, Johannes Kopf, Ying-Qing Xu, and Michael F. Cohen. Annotating gigapixel images. *ACM Symposium on User Interface Software and Technology*, pages 33–36, 2008.

[LFKZ01] Joseph J. LaViola, Daniel Acevedo Feliz, Daniel F. Keefe, and Robert C. Zeleznik. Hands-free multi-scale navigation in virtual environments. *Proceedings Symposium on Interactive 3D Graphics*, pages 9–15, 2001.

[Lin72] James Lind. *A Treatise on the Scurvy.* S. Crowder, 1772.

[LMSR08] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[LW90] Marc Levoy and Ross Whitaker. Gaze-directed volume rendering. *SIGGRAPH, Computer Graphics*, 24(2):217–223, 1990.

[MBS09] Meinard Mller, Andreas Baak, and Hans-Peter Seidel. Efficient and robust annotation of motion capture data. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 17–26, 2009.

[McM11] Ryan Patrick McMahan. Exploring the effects of higher-fidelity display and interaction for virtual reality games. *Virginia Polytechnic Institute and State University Thesis*, 2011.

[MEN+11] Syed Zain Masood, Christopher Ellis, Adarsh Nagaraja, Marshall F Tappen, JJ LaViola, and Rahul Sukthankar. Measuring and reducing observational latency when recognizing actions. *International Conference on Computer Vision Workshops*, pages 422–429, 2011.

[MGT+03] Tamara Munzner, Francois Guimbretiere, Serdar Tasiran, Li Zhang, and Yunhong Zhou. Treejuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. *ACM Transactions on Graphics*, 22(3):453–462, 2003.

[Mit08] Martin Mittring. Advanced virtual texture topics. *ACM SIGGRAPH Games*, pages 23–51, 2008.

[MMY11] Al Mansur, Yasushi Makihara, and Yasushi Yagi. Action recognition using dynamics features. *IEEE International Conference on Robotics and Automation*, pages 4020–4025, 2011.

[MMYT01] Tsuyoshi Minakawa, Toshio Moriya, Masami Yamasaki, and Haruo Takeda. A multi-detailed spatial immersive display. *11th International Conference on Artificial Reality and Tele-existence*, pages 48–55, 2001.

[Mon08] Douglas C Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2008.

[MPL11] Maud Marchal, Julien Pettre, and Anatole Lecuyer. Joyman: a human-scale joystick for navigating in virtual worlds. *IEEE Symposium on 3D User Interfaces*, pages 19–26, 2011.

[MRC05] Meinard Mller, Tido Rder, and Michael Clausen. Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics*, 24(3):677–685, 2005.

[MST+11] Daniel L. Marks, Hui S. Son, Eric J. Tremblay, Joseph E. Ford, Paul O. McLaughlin, Michael Gehm, Ronald A. Stack, Steven D. Feller, Jungsang Kim, and David Brady. Optical testing of the AWARE wide field 2-gigapixel multiscale camera. *Frontiers in Optics*, 2011.

[NBC06] Tao Ni, Doug A. Bowman, and Jian Chen. Increased display size and resolution improve task performance in information-rich virtual environments. *Graphics Interface*, pages 139–146, 2006.

[NS03] Kai Nickel and Rainer Stiefelhagen. Pointing gesture recognition based on 3D-tracking of face, hands and head orientation. *5th International Conference on Multimodal Interfaces*, pages 140–146, 2003.

[NSMG04] Michael Nielsen, Moritz Strring, Thomas B Moeslund, and Erik Granum. A procedure for developing intuitive and ergonomic gesture interfaces for HCI. *Gesture-Based Communication in Human-Computer Interaction*, pages 409–420, 2004.

[NWFF08] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

[PA08] Emmanuel Pietriga and Caroline Appert. Sigma lenses: Focus-context transitions combining space, time and translucence. *26th Annual SIGCHI Conference on Human Factors in Computing Systems*, pages 1343–1352, 2008.

[PBA10] Emmanuel Pietriga, Olivier Bau, and Caroline Appert. Representation-independent in-place magnification with sigma lenses. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):455–467, 2010.

[PBO+14] Gregorio Palmas, Myroslav Bachynskyi, Antti Oulasvirta, H-P Seidel, and Tino Weinkauf. Movexp: A versatile visualization tool for human-computer interaction studies with 3d performance and biomechanical data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2359 – 2368, 2014.

[PCS95] Catherine Plaisant, David Carr, and Ben Shneiderman. Image-browser taxonomy and guidelines for designers. *IEEE Software*, 12(2):21–32, 1995.

[PCS+13] Charilaos Papadopoulos, Ho Jin Choi, Joydeep Sinha, Kiwon Yun, Dimitris Samaras, and Arie E. Kaufman. Gestural interfaces for the reality deck. *Center for Dynamic Data Analytics (CDDA) Workshop*, 2013.

[PCS+15] Charilaos Papadopoulos, Ho Jin Choi, Joydeep Sinha, Kiwon Yun, Arie E. Kaufman, Dimitris Samaras, and Bireswar Laha. Practical chirocentric 3dui platform for immersive environments. *IEEE 3D User Interfaces Symposium*, pages 31–34, 2015.

[PDK10] Kevin Ponto, Kai Doerr, and Falko Kuester. Giga-stack: A method for visualizing giga-pixel layered imagery on massively tiled displays. *Future Generation Computer Systems*, 26(5):693–700, 2010.

[PGK15] Charilaos Papadopoulos, Ievgeniia Gutenko, and Arie E. Kaufman. VEEVVIE - Visual Explorer for Empirical Visualization, VR and Interaction Experiments. *Submitted to IEEE Visual Analytics Science and Technology*, 2015.

[PK13a] C. Papadopoulos and A. E. Kaufman. Building the reality deck. *POWERWALL: International Workshop on Interactive, Ultra-High-Resolution Displays, SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[PK13b] Charilaos Papadopoulos and Arie E Kaufman. Acuity-driven gigapixel visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2886–2895, 2013.

[PMG+15] Charilaos Papadopoulos, Seyedkoosha Mirhosseini, Ievgeniia Gutenko, Kaloian Petkov, Arie E. Kaufman, and Bireswar Laha. Scalability limits of large immersive high-resolution displays. *IEEE Virtual Reality*, 2015.

[PMK14] Charilaos Papadopoulos, Seyedkoosha Mirhosseini, and Arie E. Kaufman. Immersive visualization of storm-surge simulations. *CEWIT International Conference & Expo on Emerging Technologies for a Smarter World*, 2014.

[Pop10] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[PPK12] C. Papadopoulos, K. Petkov, and A. Kaufman. 1,500,000,000 pixels on a budget - building the realitydeck. *CEWIT 2012*, 2012.

[PPK13] Kaloian Petkov, Charilaos Papadopoulos, and Arie E. Kaufman. Visual exploration of the infinite canvas. *IEEE Virtual Reality*, pages 11–14, 2013.

[PPKM15] Charilaos Papadopoulos, Kaloian Petkov, Arie E. Kaufman, and Klaus Mueller. The Reality Deck - Immersive Gigapixel Display. *IEEE Computer Graphics and Applications*, 35(1):33–45, 2015.

[PRS10] Mark W. Powell, Ryan A. Rossi, and Khawaja Shams. A scalable image processing framework for gigapixel Mars and other celestial body images. *IEEE Aerospace Conference*, pages 1–11, 2010.

[PSK12a] Charilaos Papadopoulos, Daniel Sugarman, and Arie Kaufman. Nunav3d: A touch-less, body-driven interface for 3D navigation. *IEEE Virtual Reality Poster Session*, 0:67–68, 2012.

[PSK12b] Charilaos Papadopoulos, Daniel Sugarman, and Arie E. Kaufman. Body-driven navigation for 3D visualization using NuNav3D. *IEEE Pacific Visualization*, 2012.

[QZP+08] Feng Qiu, Bin Zhang, Kaloian Petkov, Lance Chong, Arie Kaufman, Klaus Mueller, and Xianfeng David Gu. Enclosed five-wall immersive cabin. *4th International Symposium on Advances in Visual Computing*, pages 891–900, 2008.

[RFT+13] Roy A Ruddle, Waleed Fateen, Darren Treanor, Peter Sondergeld, and Phil Ouirke. Leveraging wall-sized high-resolution displays for comparative genomics analyses of copy number variation. *IEEE Symposium on Biological Data Visualization*, pages 89–96, 2013.

[Rob07] Jonathan C Roberts. State of the art: Coordinated & multiple views in exploratory visualization. *International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71, 2007.

[RR08] Leonard Richardson and Sam Ruby. *RESTful web services*. O'Reilly Media, Inc., 2008.

[RTR+15] Roy A Ruddle, Rhys G Thomas, Rebecca S Randell, Philip Quirke, and Darren Treanor. Performance and interaction behaviour during visual search on large, high-resolution displays. *Information Visualization*, 14:137–147, 2015.

[SA82] Bob Spence and Mark Apperley. Bifocal display, 1982. www.interaction-design.org/encyclopedia/bifocal_display.html.

[SADK+09] Lauren Shupp, Christopher Andrews, Margaret Dickey-Kurdziolek, Beth Yost, and Chris North. Shaping the display of the future: The effects of display size and curvature on user performance and insights. *Human Computer Interaction*, 24(1-2):230–272, 2009.

[SCP95] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual reality on a wim: interactive worlds in miniature. *SIGCHI Conference on Human Factors in Computing Systems*, pages 265–272, 1995.

[SD09] Jeff Sauro and Joseph S Dumas. Comparison of three one-question, post-task usability questionnaires. *SIGCHI Conference on Human Factors in Computing Systems*, pages 1599–1608, 2009.

[SD10] Martin Spindler and Raimund Dachselt. Exploring information spaces by using tangible magic lenses in a tabletop environment. *28th International Conference on Human Factors in Computing Systems (Extended Abstracts)*, pages 4771–4776, 2010.

[SGS04] Karan Singh, Cindy Grimm, and Nisha Sudarsanam. The ibar: a perspective-based camera widget. *ACM Symposium on User Interface Software and Technology*, pages 95–98, 2004.

[Sho92] Ken Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. *Graphics Interface*, pages 151–156, 1992.

[SLC04] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. *International Conference on Pattern Recognition*, 3:32–36, 2004.

[SMvB+10] Martijn D. Steenwijk, Julien Milles, Mark A. van Buchem, John H.C. Reiber, and Charl P. Botha. Integrated visual analysis for heterogeneous

datasets in cohort studies. *IEEE VisWeek Workshop on Visual Analytics in Health Care*, 2010.

[SPSS11] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from RGBD images. *AAAI Workshop: Plan, Activity, and Intent Recognition*, 2011.

[SSJ+11] Brian Summa, Giorgio Scorzelli, Ming Jiang, Peer-Timo Bremer, and Valerio Pascucci. Interactive editing of massive imagery made simple: turning atlanta into atlantis. *ACM Transactions on Graphics*, 30(2):7, 2011.

[SSK+13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[Sut65] Ivan E. Sutherland. The ultimate display. *IFIP Congress*, pages 506–508, 582–583, 1965.

[TGSP06] Desney S. Tan, Darren Gergle, Peter Scupelli, and Randy Pausch. Physically large displays improve performance on spatial tasks. *ACM Transactions on Computer-Human Interaction*, 13(1):71–99, 2006.

[TM04] Melanie Tory and Torsten Moller. Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):72–84, 2004.

[TMJ98] Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. The clipmap: A virtual mipmap. *25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 151–158, 1998.

[TS99] Masashi Toyoda and Etsuya Shibayama. Hyper mochi sheet: A predictive focusing interface for navigating and editing nested networks through a multi-focus distortion-oriented view. *SIGCHI Conference on Human Factors in Computing Systems*, pages 504–511, 1999.

[WCL03] Grant Wallace, Han Chen, and Kai Li. Color gamut matching for tiled display walls. *Workshop on Virtual Environments*, pages 293–302, 2003.

[Wil83] Lance Williams. Pyramidal parametrics. *SIGGRAPH, Computer Graphics*, 17(3):1–11, 1983.

[WL95] Colin Ware and Marlon Lewis. The dragmag image magnifier. *Human Factors in Computing Systems*, pages 407–408, 1995.

[WO90] Colin Ware and Steven Osborne. Exploration and virtual camera control in virtual three dimensional environments. *SIGGRAPH, Computer Graphics*, 24(2):175–183, 1990.

[WP09] Robert Y Wang and Jovan Popovi. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28(3):63, 2009.

[WWH97] Benjamin Watson, Neff Walker, and Larry F. Hodges. Managing level of detail through head-tracked peripheral degradation: a model and resulting design principles. *ACM Symposium on Virtual Reality Software and Technology*, pages 59–63, 1997.

[YB98] Yaser Yacoob and Michael J Black. Parameterized modeling and recognition of activities. *International Conference on Computer Vision*, pages 120–127, 1998.

[YCF+00] Chen Yuqun, D. W. Clark, A. Finkelstein, T. C. Housel, and Li Kai. Automatic alignment of high-resolution multi-projector displays using an uncalibrated camera. *IEEE Visualization*, pages 125–130, 2000.

[YDK11] So Yamaoka, Kai-Uwe Doerr, and Falko Kuester. Visualization of high-resolution image collections on large tiled display walls. *Future Generation Computer Systems*, 27(5):498–505, 2011.

[YGVG12] Angela Yao, Juergen Gall, and Luc Van Gool. Coupled action recognition and pose estimation from multiple views. *International Journal of Computer Vision*, 100(1):16–37, 2012.

[YHC+12] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L Berg, and Dimitris Samaras. Two-person interaction detection using body-pose features and multiple instance learning. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 28–35, 2012.

[YHN07] Beth Yost, Yonca Haciahmetoglu, and Chris North. Beyond visual acuity: the perceptual scalability of information visualizations for large displays. *SIGCHI Conference on Human Factors in Computing Systems*, pages 101–110, 2007.

[YN06] Beth Yost and Chris North. The perceptual scalability of visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):837–844, 2006.

[ZBL+10] Jamie Zigelbaum, Alan Browning, Daniel Leithinger, Olivier Bau, and Hiroshi Ishii. g-stalt: a chirocentric, spatiotemporal, and telekinetic gestural interface. *Fourth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 261–264, 2010.

[ZDlTH13] Feng Zhou, Fernando De la Torre, and J Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2013.

[ZH93] Ferdinand Zijlstra and Rudolf Hendrikus. *Efficiency in work behaviour: A design approach for modern tools*. Delft University Press, 1993.

[ZMB11] David J. Zielinski, Ryan P. McMahan, and Rachel B. Brady. Shadow walking: An unencumbered locomotion technique for systems with under-floor projection. *IEEE Virtual Reality*, pages 167–170, 2011.