Holosuite

An Exploration into Interactive Holographic Telepresence

Ermal Dreshaj
B.S. Case Western Reserve University, 2010

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE IN MEDIA ARTS AND SCIENCE
at the **MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

September 2015

Authored by **Ermal Dreshaj**
Program in Media Arts and Sciences
MIT Media Lab
21 August, 2015

Certified by **V. Michael Bove, Jr.**
Principal Research Scientist
MIT Media Lab

Accepted by **Pattie Maes**
Academic Head
Professor of Media Arts and Sciences
MIT Program in Media Arts and Sciences

Holosuite

An Exploration into Interactive Holographic Telepresence

Ermal Dreshaj

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning on the 21st of August 2015,
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE IN MEDIA ARTS AND SCIENCE
at the **MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

Abstract

Research in holographic video technology has made great progress in the past decade, thanks to new advances in hologram computation algorithms and light modulation materials. Due to the niche and inaccessibility of holographic display research, however, literature on applications of holographic display technology remains scarce.

In this thesis, we describe a holographic display application that combines remote telepresence with interaction in 3D space. We propose some key concepts on leveraging the strengths of holographic display technology as a medium for interactive telepresence. These concepts are implemented in a real-time, end-to-end 3D telepresence software application titled “Holosuite.” Our implementation fosters a novel usage of sharing, collaborating with, and visualizing 3D data between users in a highly immersive and realistic way. In doing so, we have created an experience that connects two remote users in a way that is more engaging and provides more affordances than traditional videoconferencing.

Thesis Supervisor
V. Michael Bove, Jr.
Principal Research Scientist
MIT Media Lab

Holosuite

An Exploration into Interactive Holographic Telepresence

Ermal Dreshaj

The following served as a reader for this thesis

Hiroshi Ishii
Professor of Media Arts and Sciences
MIT Program in Media Arts and Sciences

Holosuite

An Exploration into Interactive Holographic Telepresence

Ermal Dreshaj

The following served as a reader for this thesis

Ken Perlin
Professor of Computer Science
NYU Media Research Lab

Acknowledgements

To Babi and Nana for believing in me, inspiring me to do my best, and to accept nothing less. My sisters, Vjosa and Albana for raising me and supporting me throughout my life. My brother, Pleurat, who taught me to never give up and fight through the pain.

Thanks to my advisor, Mike, for being so flexible and accommodating, always giving valuable, constructive feedback and much-needed praise. To my readers, Ken and Hiroshi, for taking valuable time out of their busy schedules to entertain my ideas.

Thank you to Linda and Keira for not being mad about my flip-flopping and for holding my hand throughout the MAS thesis process.

An especially big thanks goes out to Juanita for all of her help in creative direction, for drawing the sketches in this thesis, helping me with the poster, the production and creative vision for the Holosuite video. I couldn't have done it without your help!

To VRAM for walking through the dark tunnel with me. Thank you, Julie, for making the effort to understand me as a human being, for being so kind and compassionate when I needed it most. You guys saved my life.

To my officemates (the 444 crew), I am so thankful for having the opportunity to share a space with you guys for the past year. You all brought cheer, joy and laughter to the office daily. You guys made it easier to wake up and come into the lab every day. Thank you, Bianca, for being patient, diligent, understanding and correcting me when I needed correction. Thank you, Sunny, for inspiring me with your endless knowledge and guidance, and always being ready to answer my millions of questions about holography and physics. Thanks to Valerio for the valuable contributions to my crit. day talk and for being an all-around class act.

Big thanks to all of my group members in OBM—Pip for helping me with my user study, Edwina, Santiago, NovySan, Laura, Nick and Everett for all the drinks at the Muddy and the candid conversations.

To Roz and Karthik for hearing me out, and teaching me that having depression is not shameful.

Thank you Danielle, for making our apartment a true sanctuary.

Finally, the many nameless people at the lab who have had a positive influence on me for the past couple of years, I thank you deeply and wish you all the best in you endeavors.

The young man stepped into the hall of mirrors
Where he discovered a reflection of himself
Even the greatest stars discover themselves in the looking glass

Sometimes he saw his real face
And sometimes a stranger at his place
Even the greatest stars find their face in the looking glass

He fell in love with the image of himself
And suddenly the picture was distorted
Even the greatest stars dislike themselves in the looking glass

He made up the person he wanted to be
And changed into a new personality
Even the greatest stars change themselves in the looking glass

The artist is living in the mirror
With the echoes of himself

Even the greatest stars live their lives in the looking glass
Even the greatest stars fix their face in the looking glass
Even the greatest stars live their lives in the looking glass

Lyrics from “Hall of Mirrors”, by Kraftwerk

1. Introduction

Few scientific endeavors in the past century have managed to capture the public imagination quite like the field of holography. It is a phenomenon that traces its roots to the research in microscopy and optics credited to Denis Gabor and Yuriy Denisyuk in the 1950's and spans to the pioneering work done by Steve Benton at the MIT Media Lab on digital holographic imaging in the 1980's and 1990's. To be more accurate, Gabor's research in microscopy was considered turgid to his peers, and Denisyuk's work of recording optical properties of an object on a shallow plate was largely derided and ignored. It was not until the experiments of Emmet Leith and Juris Upatnieks in 1963, when the laser became available as a coherent light source, that the world would be introduced to a new astonishing form of three-dimensional imagery in which the reconstructed images exhibited depth and parallax with unprecedented realism [1].

The off-axis Leith-Upatnieks hologram method re-framed research in holography as a type of "advanced photography" field, which excited interest in domains far beyonds physics and engineering. Leith and Upatnieks had succeeded in creating a transmission hologram, which could be described by the eloquent metaphor of a "window with a memory" [2]. It was the aesthetic properties of this discovery that captured the imagination of the public, exemplified by the "Princess Leia hologram" in the 1977 movie *Star Wars.* It was also in this tradition that Steve Benton's research at the Media Lab carried the torch of holographic research, specifically with the creation of Mark I, the world's first digital holographic display.

With the development of the Mark I and, subsequently, Mark II holovideo displays, Benton's vision was to turn the "window with a memory" [2] into a living window—a medium that could be used as a real-time, high frame rate, dynamic display with the optical properties of the Leith-Upatnieks transmission hologram that provided a realistic aesthetic [3]. This thesis is a continuation in the tradition set by Leith, Uptanieks, and Benton by exploring an application of holovideo—specifically we demonstrate real-time, two-way, interactive telepresence on a full-color holovideo display based on new guided-wave acousto-optic modulator technology.

1.1 - The definition of "holographic"

Much to the chagrin of holography researchers and artists, the term "hologram" has escaped the clutches of these communities who take pride in the craft and wonder of holography. The usage of the words "hologram" and "holographic" has become diluted and these words are now used to describe a myriad of technologies and methods of display that have little to do with the original intent of the word. The point of contention here lies not with the obsession for semantics, but that these display technologies usually cannot capture the richness of the promise of real holography. Rather, they arguably serve to undermine the efforts of holographic research by painting the technology as being underwhelming, or bogus, even.

Holographic displays can be described as a technology that perform reconstruction of light wavefronts by using the diffraction of coherent light sources. In lay terms, this means that in a holographic display, we are computing the way light interferes with itself to represent the

amplitude and phase of incident light on a scene. This distinction is crucial because while most other methods for displaying 3D imagery typically are concerned with tricking the human visual system into seeing depth and parallax, the holographic method aims to reconstruct the properties of the light signal of a scene that encode these visual cues. Thus, when the holographic theory of wavefront reconstruction is applied, the light wavefronts striking the eye of the user should be theoretically indistinguishable from that of the original scenery.

In this paper, when any terms derivative of “hologram” are used, they are used in the strictest sense—that we are describing the phenomenon of recording, computing, or replaying the wavefront of scenery via diffraction of coherent light. More specifically, when we use the term “digital holographic display” or “holovideo”, we are referring to a machine that can compute the interference pattern from a 3D scene (either captured digitally from the real world, rendered synthetically on a computer, or a combination of the two) and display the wavefront via diffraction of coherent light in real-time.

1.2 - Holography simulation

It’s important to note, however, that in this thesis, we explore not only the practical application, but also the theory of holography as a medium for interactive telepresence. Since we are aiming to explore the practical and theoretical aspects of holographic telepresence together, some of the work can be performed adequately in a theoretical sense with what we refer to as a holographic display “simulation”. Thus, whenever the term “simulation” is used from hereon in conjunction with holographic displays, it is used to describe the approach of using more conventional 3D displays that use stereoscopic vision and motion parallax reprojection—generating an approximation of two very powerful visual cues provided inherently in holographic images.

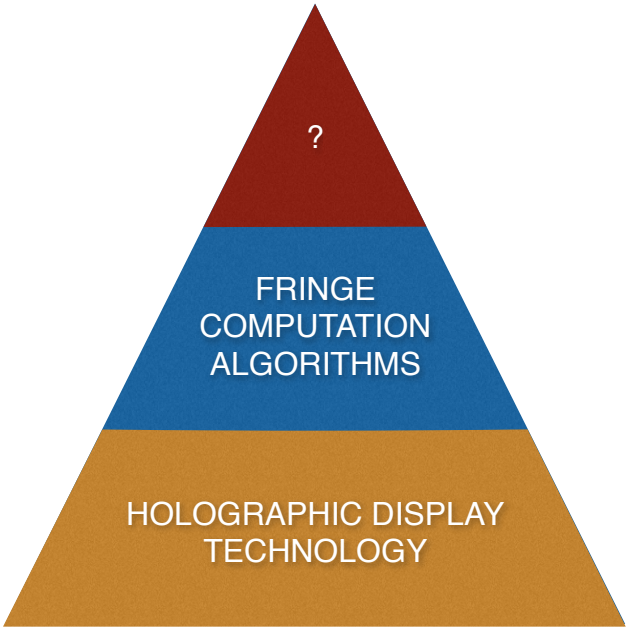


FIG 1.2 - The question mark at the top signifies applications of holovideo research, of which little is done by holographic researchers.

This is done for practical reasons, as the MIT Mark II and Mark IV, for example, are not full-parallax displays—they only provide parallax effect in the horizontal dimension—and the size of Mark II/IV displays and its physical layout on an optical bench do not provide conditions for executing proper user studies.

The approach in this body of work is to implement all aspects of the telepresence software independent of rendering engine (networking, compression, hand tracking, etc.), while creating a simulation renderer made for advanced 3D pixel displays, in addition to a renderer for MIT Mark II and Mark IV holographic video displays. In this architecture, the display method can be easily switched in software to accommodate the display type connected to the computer running

Holosuite, while the experience is consistently across displays. This is described further in the architecture diagrams in the **Development** and **Design** chapters.

1.3 - The definition of telepresence

The definition of telepresence is also a point of contention among academics as it has been used to describe a variety of technologies that allow people to communicate across long distances. For some, telepresence describes the physical manifestation of presence, where physical materials are used to represent presence. For others, the definition can mean something more visual, like video-conferencing.

In Holosuite, we aim to most accurately simulate the visual experience of the presence from someone who is far away. As we are simulating the experience with high realism (depth and parallax), along with audio and hand interactions to manipulate models in a shared space, we contend that this satisfies even the strictest definition of "telepresence." As described in the **Results** section of this thesis, people who have used Holosuite rate it very highly in immersion and realism, so we take this as adequate proof of our assertion that we are properly using the term "telepresence" when describing the Holosuite user experience.

1.4 - What is Holosuite?

The resulting work of this thesis is a software project titled "Holosuite." Coded in C++11, Holosuite is an optimized, multi-threaded implementation of end-to-end 3D telepresence operating on two remote PCs via internet. It can render visual output to holographic displays, as well as advanced 3D displays with motion parallax reprojection.

There is a temptation to label Holosuite as "3D Skype" or "3D Videoconferencing", but this fails to distinguish what is important about Holosuite, as it neglects the collaborative and interactive aspects of Holosuite.

The Holosuite software project merges two 3D worlds seamlessly, so as to simulate the usage where two distant people are merely separated by a metaphorical window. Seamless merging of two 3D environments means that the users are able to share 3D models and data between each others' worlds as if they were both present in the real world, with the virtual 3D models positioned between the users.

Imagine, for example, two people sitting on opposite sides of a table, with a model placed in the middle of the table. In this scenario, both users can see and talk about the model together; they can also pick up the model, look underneath the model, rotate it and point to parts of the model. They may also be able to build the model together, by placing material on the model, or removing material, while picking up visual cues from the opposite user's body language. The Holosuite software project aims to implement this type of experience for two people who are physically separated by a large distance, whilst taking advantage of the visual aesthetic afforded by holographic or advanced 3D displays.

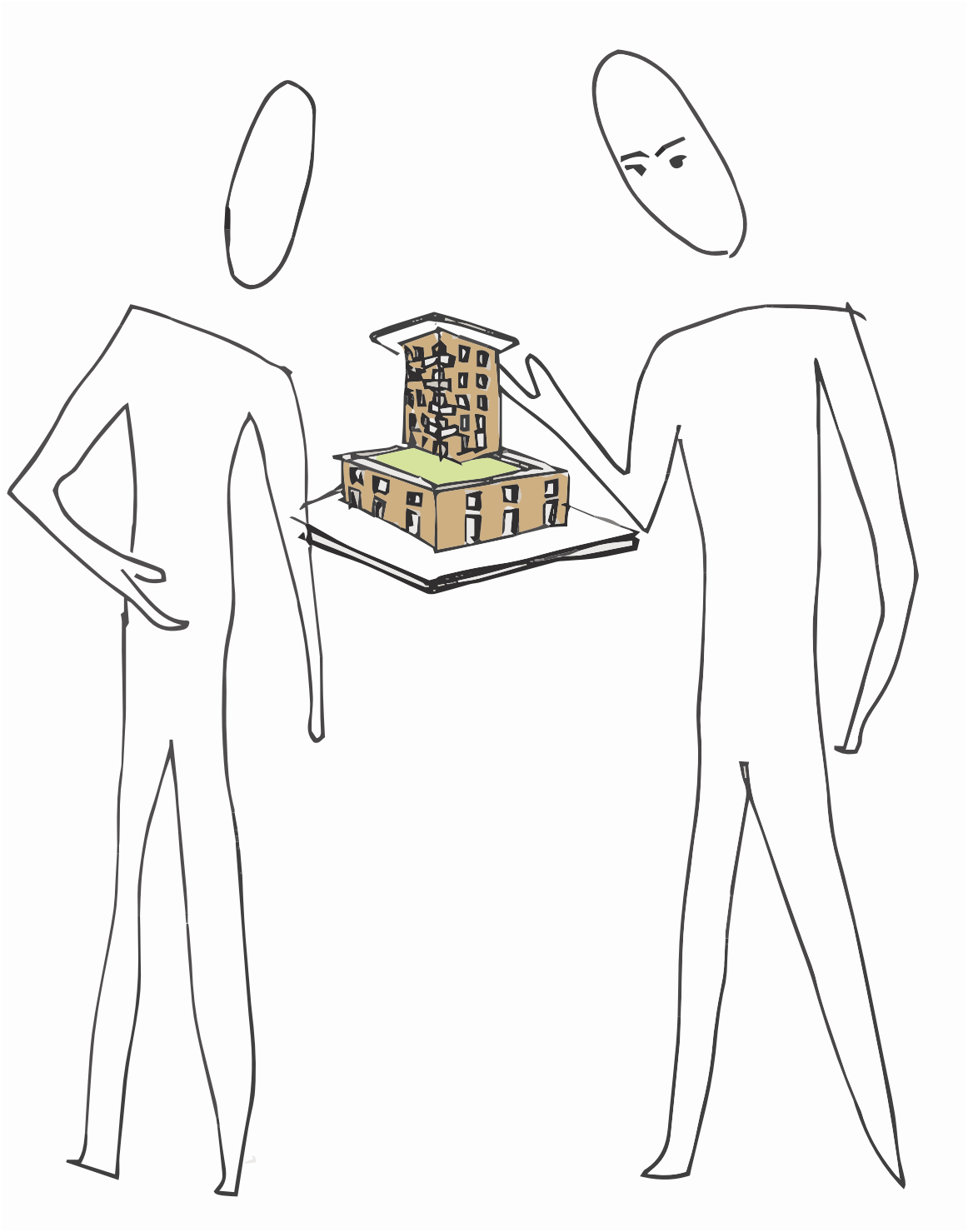


FIG 1.4 - Holosuite implements the usage where two remote users can interact with each other and share 3D data in a unified space. Using holographic and advanced 3D displays to render depth and parallax with high realism.

2. Background and Motivation

The lack of research in digital holographic display applications can be attributed to the following reasons:

First, the technical requirements for computing fringe patterns from 3D data with high frame rates are steep—a problem solved by advancements in modern multi-core CPUs, and many-core GPUs. GPUs were previously locked into fixed pipeline systems, and it was only in the past decade that hardware vendors unlocked them for general purpose computation [4]. Since CGH (computer-generated holography) algorithms are highly parallelizable, we can take advantage of GPU compute power to reconstruct wavefronts in real time.

Second, previous methods of controlling diffraction of light via a digital signal involved the use of bulk-wave acousto-optic modulators and LCOS technology, which provide much smaller space bandwidth product in comparison to newer guided-wave acousto-optic modulators [5]. This advancement allows for larger screen sizes, wider viewing angles and full-color digital holographic displays. Combined with newer, more efficient fringe computation algorithms it gives us the ability to produce a light field which is very compelling to the human visual system [6].

Third, it is only from recent advances in camera capture technology that we are now able to easily capture 3D data from the real world in real time. This is a boon for researching the applications of holography, because 3D data of the real world opens up a myriad of possibilities in visualization and interaction, of which a holographic display can take full advantage. After all, if there is no 3D data of the real world, the benefit of using holovideo as a medium for telepresence becomes dubious. To visualize the real world without 3D data, we have to resort to gimmicks that involve large camera rigs, creating challenging computation and bandwidth problems unsuited for common usage.

Finally, the last technological milestone has to do with the field of human-computer interaction. Recent advancements in computer vision algorithms and camera technology enable us to track a fully-articulated hand skeleton in 3D space in real-time. The availability of robust 3D hand-tracking technology enables us to integrate real-world hand usage metaphors with telepresence interactivity, which is essential for playing with objects in a virtual 3D space.

2.1 - Why holography?

Holosuite is a project that, perhaps opportunistically, aims to take advantage of these recent advancements in technology to implement and prove beyond concept an interactive holographic telepresence application. It is the realization of an idea that has its roots in the fiction and the science surrounding the field of holography. Although interest in holography has waned since the days of Steve Benton, there still exists something in the zeitgeist which permeates the collective imagination—the idea of interacting across long distances with others in a realistic and immersive way that is compelling.

Although the aim for this thesis is to research the applications of digital holographic video displays, we are also interested in developing an application that fosters real-world interaction and provides an engaging experience. After all, the promise of holography is that we can create a visual experience that rivals that of the real world experience. In this capacity, we've established some guiding principles on how to best make use of the affordances of holographic display technology to develop interactive 3D telepresence.

2.2 - Holovideo as an ideal medium for interactive telepresence

Although we have mentioned that the technical hurdles for implementing interactive holographic telepresence have been largely resolved, the question of the value of using holovideo as a medium for interactive telepresence remains unaddressed.

Establishing the feeling of presence of a remote user means that we should be able to see the true 3D image of the person, from all angles, as he or she looks in real life. Holography gives us a theoretical framework and method by which we can accomplish this. If we are to imagine the ideal hologram as a "window with memory" of the scenery behind it, our visual system would not be able to tell the difference between a real window facing the outside world and a hologram substitute [2]. In a theoretical sense, it would mean that reconstructing a wavefront is our best bet at creating the visual representation of a remote user.

Aside from the visual experience, how about interaction? Why should interactivity be any better on a holographic display versus a traditional 2D, or digital 3D display, even?

In the literature for collaborative telepresence, we uncover a more subtle, yet meaningful window metaphor. We imagine two users looking at each other, separated by a window. When the user looks through the window, to the other side, he can see a 3D objects and users, just as he or she does in real life. He or she can look around the objects and experience smooth motion parallax. In this scenario, users can pass objects through the window to the other side, giving the remote user ownership and allowing them to analyze and manipulate the object.

In this imagined scenario, the two users may be only a few feet apart. The power of using holovideo for telepresence is that we can create an experience from this imagined scenario for two people separated across long distances. We can seamlessly merge two distant 3D worlds together as one. In the **Related Works** chapter, we describe the inspiration for this and what it means to create a "seamless" interactive telepresence experience.

3. Related Works

The body of work in this thesis sits somewhere at the intersection of advanced display technology applications and classical human-computer interaction research (specifically in the domains of telepresence and computer-supported cooperative work). The most relevant bodies of work come from Prof. Henry Fuchs, at the University of North Carolina Chapel Hill. His group studies advanced computer graphics technology and telepresence methods related to 3D display technology. Hiroshi Ishii's Tangible Media Group at the MIT Media Lab, which is now more chiefly concerned with telepresence in the 3D physical medium space, has been exploring compelling interactive telepresence applications using shape displays. Although, the most relevant research to this project by Ishii is rooted in work done in the '90s at NTT Japan [7]. Finally, the University of Arizona has been, to our knowledge, the only group to demonstrate a telepresence application that meets our strict definition of what can be considered holographic [8].

Although there are many other examples of similar projects across academia, we have narrowed the most important bodies of work belonging to the four most relevant topics: holographic display applications, end-to-end 3D telepresence, seamless collaboration, and interactive telepresence.

3.1 - Holographic display applications

In 2010, Blanche et. al from the University of Arizona published a paper in Nature magazine titled, "Holographic three-dimensional telepresence using large-area photorefractive polymer" [8]. This is largely thought to be the first example of true holographic telepresence achieved, though with some major caveats.

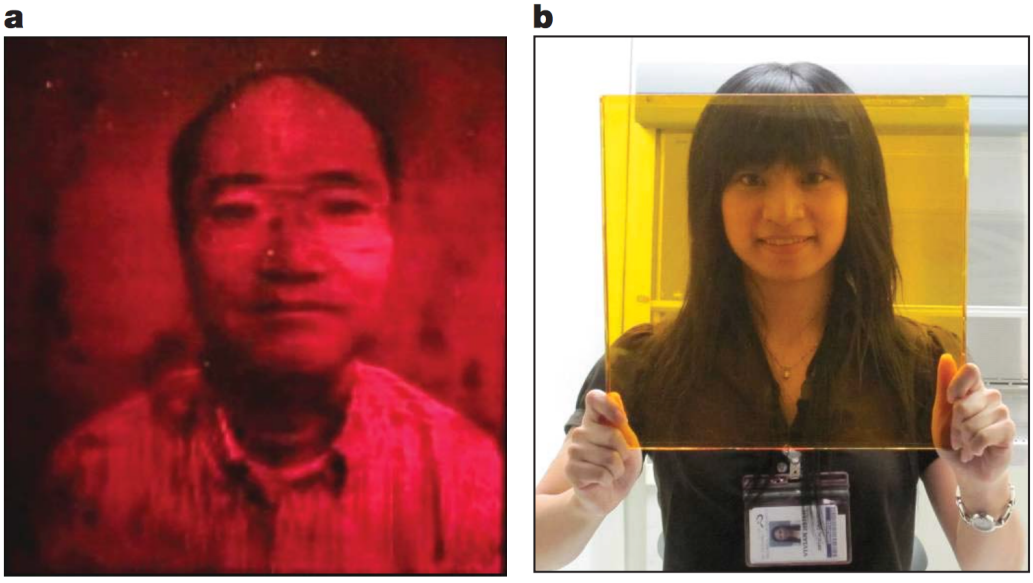


FIG 3.1.1 - An example of Blanche et. al's technique of using large-area photo refractive polymer programmed by 50 Hz nanosecond pulsed laser.

Firstly, this method is not suitable for interactive telepresence as the frame rate is not within the realm of what can be considered real-time (the photopolymer programming method can update at most once every 2 seconds). This not only makes impossible to have meaningful interaction, but as it does not approach tolerable real-time animation frame rate, it is not feasible for simple real-time communication either.

In addition to the display issues, Blanche et. al's capture approach is done via a sophisticated 16-camera array to generate a panorama for viewing motion parallax. This approach has limitations in that what is displayed to the viewer through the holographic pixels is a set of 16 discrete views of the scene, which are not enough to produce the viewing effect of smooth motion parallax (rather the views transition visibly with the viewer motion). It's important to note, also, that this method is monochrome with no foreseeable way to create full-color images.

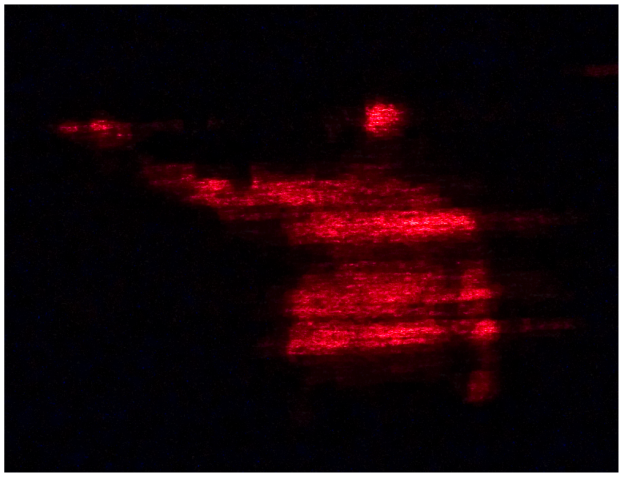


FIG 3.1.2 - Holographic diffraction-specific image of a person pointing with his hand [9].

In 2011, J. Barabas, S. Jolly, D.E. Smalley and V.M. Bove Jr. demonstrated a diffraction-specific method for displaying real-world scenes in real-time on a holographic display at the MIT Media Lab on the MIT Mark II display [9]. Compared to the prior example, this method was fast (17 frames per second) and utilized a hologram computation algorithm that discretized the display into computationally-independent chunks, each chunk being able to display variable light intensity and curvature as a function of viewing direction.

This allowed for a better approximation of ideal wavefronts, providing proper vergence and accommodation cues, resulting in a smooth, continuous motion parallax experience from the left-to-right viewing angles. The MIT Mark II and Mark IV operate via acousto-optic diffraction of coherent light, which has speed limitations dictated by the speed of sound (more than enough for real-time frame rate display) and the ability to modulate light from multiple color channels simultaneously, enabling full color display on the Mark IV.

In addition to the advantages in computation algorithms, the capture method in the diffraction-specific approach is to take data from the real world using depth-sensing cameras, such as the Microsoft Kinect. Instead of generating discrete views from a camera array, this method employs the technique of computing the fringe pattern directly from the point cloud, or the 3D data set of the user as provided by the depth camera. The most obvious benefit of this is that now only 1-3 camera(s) need to be used for capture (depending on the viewer's tolerance for occlusion artifacts), making it more easily accessible as a human interface application, while providing accurate 3D location per voxel in the point cloud. Another, perhaps less obvious advantage, is that depth data is much more compact and easier to compress than 16 discrete color images, allowing for real-time compression and transfer over internet connections (a key requirement for implementing the Holosuite user experience).

Holosuite improves upon this technique by implementing capture, compression, networking and rendering of 3D data on holographic displays for two distant users via internet, using a newer full-color hologram computation algorithm developed for the Mark IV display.

3.2 - End-to-end 3D telepresence

At the University of North Carolina at Chapel Hill, work on 3D telepresence began in the 1990s. Towels et. al [10] in a paper titled, “3D tele-collaboration over Internet2” from 1998 proposed a system very similar to Holosuite which enables end-to-end 3D telepresence with an interactive model shared between the two users.

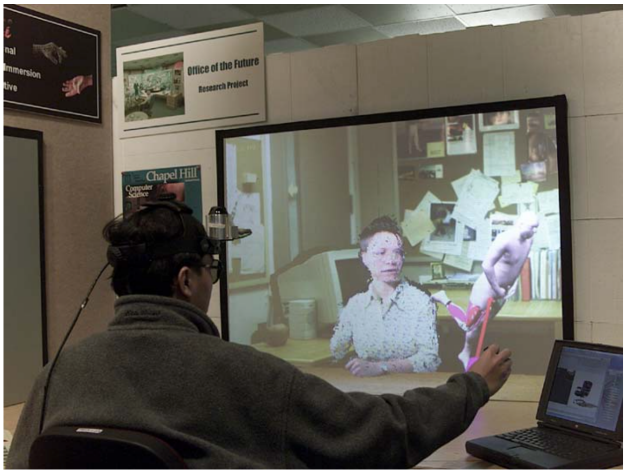


FIG 3.2.1 - End-to-end 3D telepresence in 1998 showing two users manipulating a 3d model with a pointing device [10].

As commodity depth cameras were not available at the time, this technique employed the usage of camera arrays and a pointing device to control and manipulate the 3D object. The image was displayed via a head-tracked stereoscopic display. Interestingly, they did employ 3D reconstruction to generate a point cloud of the user and transmit it over the internet in near real-time from multiple camera arrays on 1990s PC hardware, which was an impressive feat (2-3 fps). This technique generated 75 Mbps of network traffic one way, which is not suitable for even today’s typical home broadband connections.

This implementation was very forward-thinking, in that it combined two 3D worlds with a shared virtual model between the users. Holosuite improves upon this work by implementing the experience on commodity hardware and electronics, capable of operating in real time over home broadband connections.

FIG 3.2.2 - Maimone and Fuchs show two distant users combined in a single 3D scene [11].

A more modern example of the work done at the BeingThere International Research Centre for Tele-Presence and Tele-Collaboration at UNC Chapel Hill is the Maimone and Fuchs 2011 ISMAR paper titled, “Enhanced Personal Autostereoscopic Telepresence System using Commodity Depth Cameras” [11] wherein end-to-end 3D teleconferencing system is implemented by multiple Microsoft Kinects at each end with 3D reconstruction of the full scene transmitted across a network connection from local to remote user.

Maimone and Fuchs's research focuses mainly on the signal processing problems surrounding depth sensor data and layout of multiple Kinect cameras for better dealing with problems such as occlusion and incomplete/noisy sensor data. The main contribution here is that from the point cloud, they are (in real time) rectifying multiple cameras pointed at the same scene, and constructing a mesh from the combined 3D point cloud data from these multiple sources.

Although Maimone does predict that one of the applications is "mixed reality collaboration", where he shows a screenshot of a virtual 3D object in the 3D scene, there is no research done on this topic, nor is there an implementation of such a scenario. Interestingly, on the project page there is an FAQ question titled, "Why does a teleconferencing system need to be in 3D and allow the user to look around the remote scene? Is this just a gimmick?", for which Maimone writes the following response:

> The value of these features is two-fold. First, they increase the sense of "presence" or "being there", the feeling that one is actually co-located with the remote user and his or her environment. This sense helps the user forget he or she is looking at a display screen and communicate naturally as if talking to someone on the other side of a table. Second, the ability to "look around" the scene helps preserve information that is lost during normal 2D video conferencing. For example, imagine that you are seated in a meeting room and someone's head is blocking your view of a whiteboard. In our system, as in real life, you would naturally move your head around for an unobstructed view. In traditional video conferencing, you must interrupt the meeting and ask that the remote user move his or her head. As another example, imagine an engineer is holding up a new prototype part to show to a remote participant. With our system, the remote participant could simply move his or her head around to inspect the part. With traditional 2D video conferencing, the remote participant must communicate back and forth with the engineer regarding the different angles the part should be held until it is fully inspected.[1]

Maimone argues the benefits of having motion parallax for collaboration, especially for mixed reality modes where two users are viewing the same model, being able to look around the scenery does indeed increase the feeling of presence while simultaneously giving the user ability to better and more naturally inspect the virtual 3D model. These assertions are backed by data gathered in user studies, discussed further in the **Evaluation** section of this document.

3.3 - Seamless collaboration

Perhaps one of the most important bodies of work done in telepresence collaboration is the 1992 project, ClearBoard, documented in the proceedings of the SIGCHI conference by Ishii et. al [7]. ClearBoard introduced a new concept of tele-collaboration to the digital world and realized a "seamless shared drawing space and eye contact to support realtime and remote collaboration by two users" [7].

ClearBoard's major achievement was to transpose a traditional human activity for collaboration (two people writing on a whiteboard together) to a visual format suitable for digital display

[1] http://www.cs.unc.edu/~maimone/KinectPaper/kinect.html

technology. It succeeded in abstracting the problem of human collaboration in a way that was not only conducive to a digital/networked display setup, but actually augmented and *enhanced* the experience of the traditional side-by-side method of collaboration. This was done by treating the computer's digital display as a metaphorical "window"— a transparent, interactive surface separating two distantly located users.

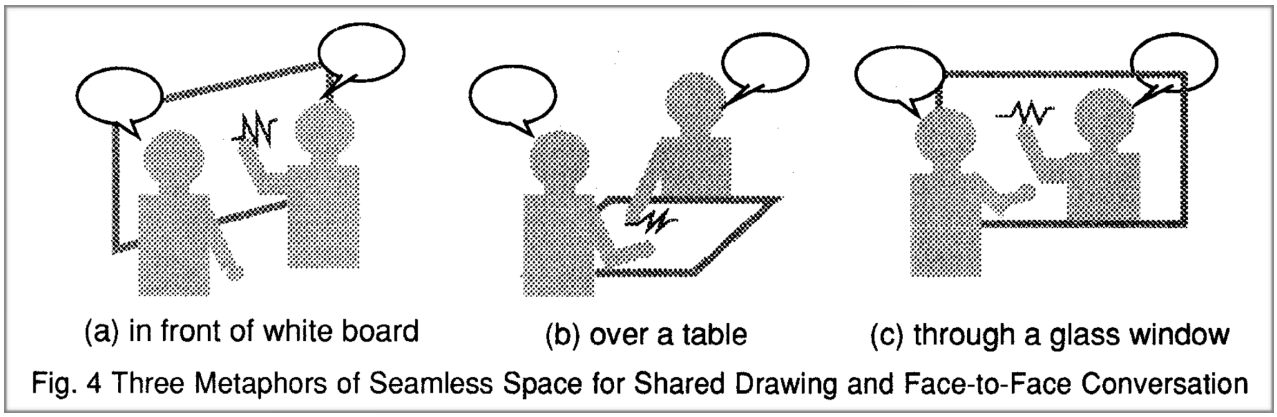


Fig. 4 Three Metaphors of Seamless Space for Shared Drawing and Face-to-Face Conversation

FIG 3.3.1 - The figure from the ClearBoard paper explaining what a "seamless space" is. Drawing C shows the advantage of the "window" metaphor, and is a usage of digital display to enhance the traditional experience, Drawing A.

Although seemingly trivial, the window metaphor was a very creative solution for adapting traditional collaborative interactions to a digital setting. Before this seamless space concept, collaborative applications on digital displays were laid out in a side-by-side tiled format, where each user was given a square alongside a shared space. This setup was the most obvious, but did not allow for an engaging or effective way to collaborate, as the users could not tell where each other were looking, not to mention that it was not an optimal usage of the display space.

ClearBoard was not designed for 3D interaction, nor 3D visualization. Yet, the seamless user interface design from ClearBoard remains an inspiration for Holosuite since this arrangement very effectively provides a realistic interaction method even in 3D. In Holosuite, we can extend this window metaphor into 3D space very easily—by merging two 3D worlds seamlessly, where the users are separated by the holovideo display. The window metaphor is also helpful for describing the advantages of holographic display technology, as we can create the feeling of presence through motion parallax—a natural byproduct of what it feels like to see another person through a window.

FIG 3.3.2 - The ClearBoard seamless layout, showing two remote users collaborating on a single display space.

3.4 - Interactive telepresence

At the MIT Media Lab, Hirosh Ishii's Tangible Media Group is now researching shape display technology, with a primary application focus on interactive telepresence. The inFORM physical telepresence project [12] embodies a system of collaboration where remotely connected users can control objects at the other end-location by utilizing a shape-changing display (both as a system to visualize three-dimensional data, and also to generate hand control as a type of tele-robotic interface).

FIG 3.4 - Shape display tele-robotic application, remote user controls a local object.

One of the great benefits of this technology is the ability to control real-world objects remotely, which is not within the realm of possibility for holographic telepresence applications. While this usage is compelling, holographic display technology is more concerned with reconstructing a wavefront—having the ability to reproduce the light of a scene in a way that is indistinguishable from the real-world experience.

With shape display technology, such as inFORM, there are two problems that need to be addressed before visualization of 3D data can match that of light-based displays: first, the fundamental element of the shape display must be reduced in size by a few orders of magnitude in order to match the visual aesthetic of light displays. Arguably, even for the usability of usages envisioned by by Folmer et. al (such as pointing, for example), require the granularity of light-based displays. This limitation is not just an engineering task, but it is an research problem that deals with finding an actuating mechanism and system that operates on the scale of pixel display technology.

Secondly, and more importantly, such pin-based shape displays operate on a 2.5D axis, on which there is only freedom for pins to move up and down in the Z dimension. This falls somewhat short of the theoretical ideal of interfaces based on "radical atoms" theory. The ideal display built on radical atoms theory would call for a material to be deformable and controllable dynamically in all 3 dimensions to form any arbitrary shape. With the 2.5D limitation, we are limited to convex shapes, or shapes that do not taper at the bottom.

The theory of holographic imaging tells us that there is a phenomenon (diffraction) by which you can store all the information of light from a real-world scene, and replay that information later to faithfully re-create a 3D scene. This theory is well understood, and we are now capable of controlling this phenomenon in a digital sense. For radical atoms, there remains the challenge of finding a material that can practically match the theory, which puts a limitation on what shape displays and other displays based on radical atoms are capable of doing for the foreseeable future.

4. Design

There are two aspects to the design of Holosuite that are important to this document: the first is the human-computer interface model (or the user interface) and the second is the software-hardware architecture required for implementing interactive end-to-end 3D telepresence in real time.

The user interface of the project is heavily inspired by the aforementioned related works, specifically borrowing from the tele-collaboration examples of 3D model manipulation and sharing between two users, while maintaining the seamless layout model. The seamless layout model is useful for both the purpose of maintaining realism and immersion, and because it gives the advantage of being able to see the users' actions and expressions as related to their activities with the 3D model.

4.1 - User interface design considerations

Although it seems fairly obvious to use the seamless layout model for the end-to-end 3D telepresence user interface, some due diligence was done on other layouts to build confidence that the seamless method make sense from the user's perspective. After all, there may yet be some advantages to using a tiled visual layout with seams, where the objects and users are allotted their own spaces on the display.

In order to create an immersive and realistic user experience that takes advantage of holographic display technology we must also provide a natural interaction method for sharing and manipulating virtual 3D data (3D models) between users in a meaningful way. From these goals, we have created a set of criteria that the user interface paradigm must fulfill.

4.2 - Immersion and realism

What provides for an immersive and realistic experience? To create an experience for a local user to feel that the remote user is present, we first ask questions about the basic philosophy of perception. Philosophies of perception can be broken down into internalist and externalist arguments, which hold that perception is either completely within the mind and nervous system, or that it is a result of the mind and external, real-world phenomenon independent of the observer, respectively.

Holography researchers would typically argue from the externalist philosophies of sensing, holding that a realistic experience follows when recreating the conditions of the way light behaves in the real world. Therefore the primary concern in holographic imaging is to recreate the properties of light, not to "trick" the user into perceiving that a 3D object is there. It is to actually reconstruct the light that would be produced when the object is in the presence of a viewer. The externalist argument implies that the requirements for realism and immersion can be fulfilled by a holographic image, since any sensor—whether a human observer or machine—

Monocular Cue	2D Display	Digital 3D Display	Holovideo Display
Occlusion	ø	ø	√
Parallax	ø	ø	√
Accommodation (Focus)			√

TABLE 4.2.1 - Monocular cues that can be provided by display technologies, ø denotes that software assistance/rendering can be used simulate the cues.

Binocular Cue	2D Display	Digital 3D Display	Holovideo Display
Stereopsis (Disparity)		√	√
Convergence			√

TABLE 4.2.2 - Binocular cues that can be provided by different display technologies.

theoretically cannot perceive the difference between a well-made hologram of a real-world scene and the actual scene itself.

However, if the externalist argument fails to convince the reader, we can also argue from a more internalist philosophy by analyzing the human visual system to better understand how humans perceive the 3D world in the mind. This approach can also serve to highlight why holography is an ideal medium for telepresence, speaking in purely visual terms.

Depth perception is the result of a confluence of visual cues that, when processed simultaneously in the brain, result in the understanding of distance of objects and perception of the 3D world. These cues can be categorized into two basic forms: monocular and binocular cues. Monocular cues are the 2D representation of visual information, which can be seen with one eye, while binocular cues are the 3D counterpart that is seen by two eyes. The scope of this thesis does not cover this topic in depth, but a great resource for better understanding depth perception can be found in Chapter 3 of the book Three Dimensional Imaging Techniques by Takanori Okoshi, titled “Physiology and Psychology of Depth Perception” [13].

When analyzing the most important of visual cues for depth perception, the argument in favor of using holographic video for immersion and realism becomes more compelling. As seen in **TABLE 4.2.1** and **TABLE 4.2.2**, holographic video displays are able to accurately provide all of the visual cues important for depth perception, without the necessity to simulate any cues in software, or wearing of glasses. This would imply that applications built on this technology should inherently be more realistic and immersive than those built on traditional displays. This is the true power of what it means to “reconstruct a wavefront” and why we contend that (when speaking in a visual sense) the aesthetic of holovideo is an ideal choice of medium for real-time telepresence.

4.3 - User interface: interaction model

The second goal of defining a meaningful interaction model can be intuited from the way people behave naturally with 3D objects in their presence. To generate a viable interaction model, we first imagine a scenario in which two people standing across from each other are looking at a model together. In such a scenario, the two people are free to grab, rotate and look around the model to get a better vantage point of features and different viewpoints of the 3D model.

In this scenario, we imagine that the two people are an architect and a client, for example. The architect would like to get feedback from the client on the latest model revision, and the client would like to show the architect things about the current revision that he or she likes, or areas

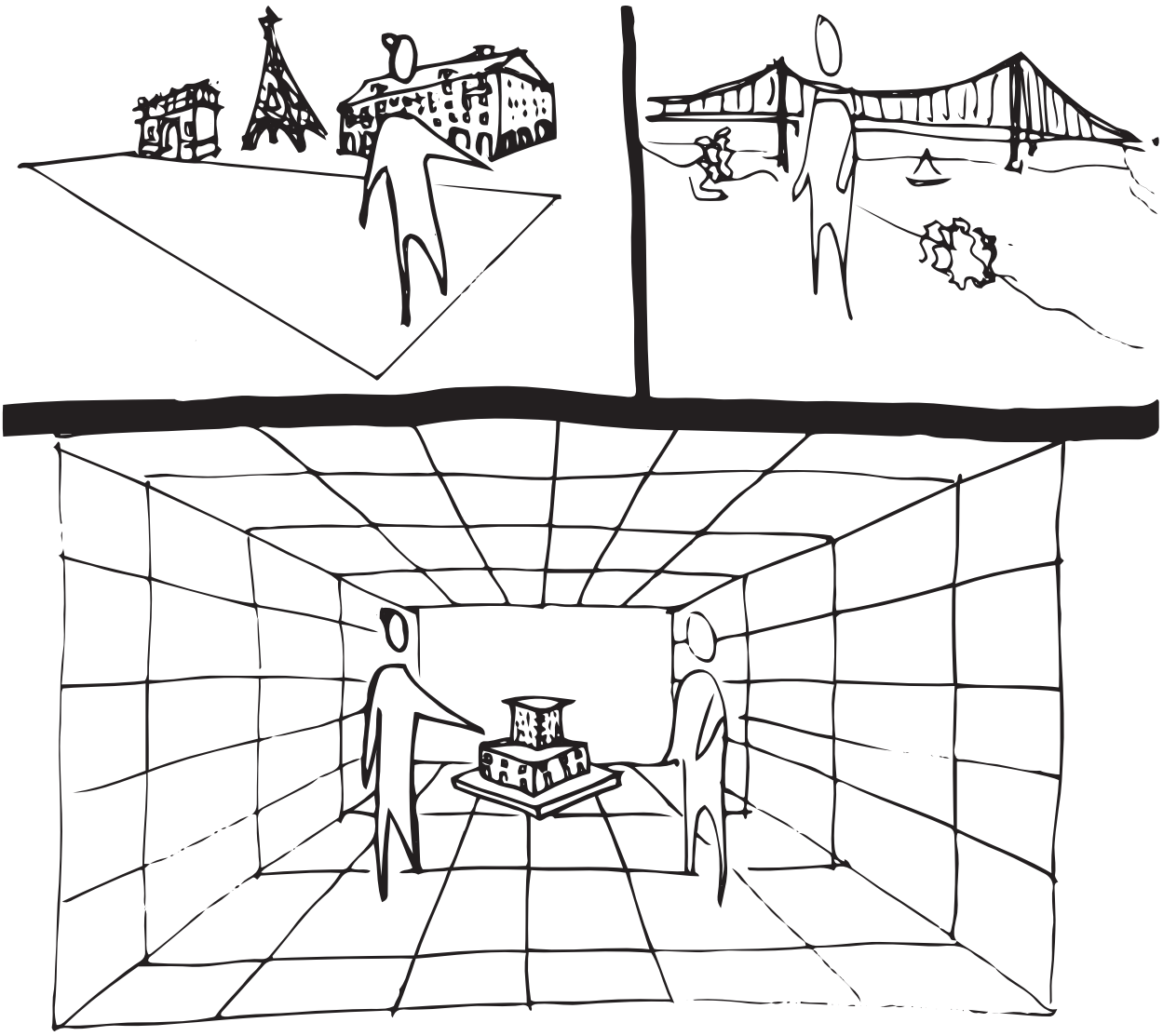


FIG 4.3 - Two distant 3D worlds are merged seamlessly, to create the effect of unified 3D space wherein users can see each other in realistic and visualize a shared 3D model in realistic a 3D projection.

that need improvement. To fulfill such an experience, we require that two usages should be implemented as they are done in the real world. These are: the ability to be aware of where the remote user is looking and where the remote user is pointing. This body language is necessary for getting visual feedback on the remote user's intent.

Further, users must be able to manipulate and share 3D objects with their hands, as they do in the real world. Although there are pointing devices in existence that allow users to manipulate and point to objects in 3D space, we preferred to use a more natural interaction method. Allowing users to use a pinch-to-grasp metaphor for grabbing and rotating a virtual object feels more natural and intuitive for this specific type of activity.

4.4 - User interface: visual layout

The visual layout should be one that can help to fulfill the design criteria of implementing the pinch-to-grasp metaphor of virtual model manipulation, whilst taking full advantage of the depth and parallax afforded by holovideo displays. As such, there are three obvious methods

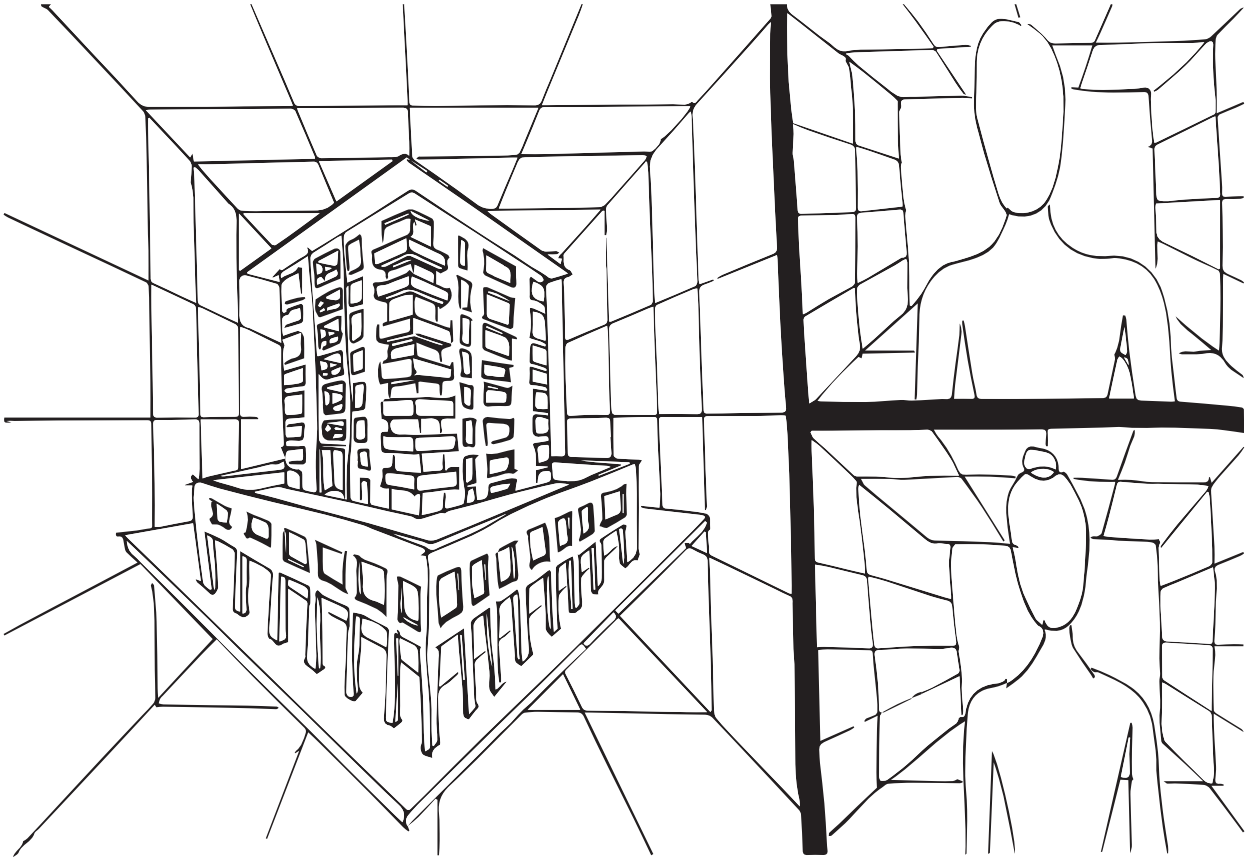


FIG 4.4.1 - An example of a tiled layout, which shows 3 three-dimensional views to the user. The small tiles on the right show the local and remote user. The large tile on the left shows the same perspective view of the shared 3D model to both users. The seams separating the tiles, however, hinder the feeling of immersion, and break the metaphor of two people standing across from each other with a model between them.

considered for building the Holosuite user experience: tiled layout, traditional video-conferencing mode, and seamless layout.

In the tiled layout (see **FIG 4.4.1**), the user sees him or herself and the other user as distinct entities, separated by seams, with the 3D model space as a big tile shared between two people. One of the advantages of this mode is that both users can see the same perspective (same side) of the shared virtual 3D model. However, without some additional interaction methods, in this layout it is not possible to see what the remote user is looking at and which part of the model the remote user is pointing to. Furthermore, the seams take away from the immersive experience, or the “feeling of being there”, as seams separate the local user from a remote user in a way that is not natural or cohesive visually. Therefore this mode was not considered as a viable option.

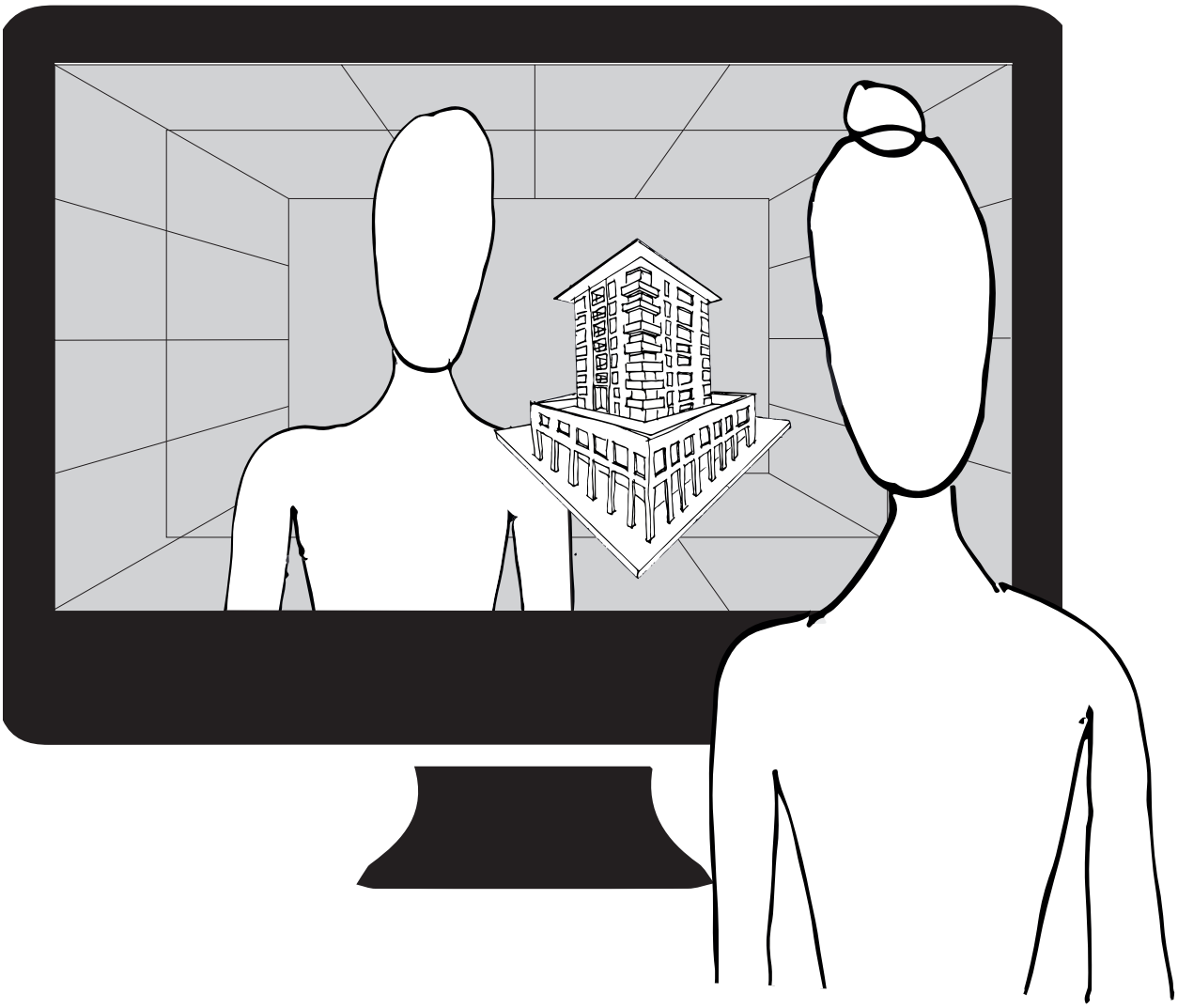


FIG 4.4.2 - Local user and remote user are separated by the display plane, with the 3D model having the ability to cross the display plane. In this representation, there is no visual feedback of local user’s image.

By moving to a seamless design (**FIG 4.4.2**), we give up the feature of both users being able to look at the same viewpoint of the 3D model. The seamless visual layout is a better allegory to the imagined scenario where two people are standing across from each other with a 3D model in the middle, because the display can act as a metaphorical window separating the two users' worlds. In this layout, the 3D model's position and orientation must obey a euclidian 3D rendering, whereby the users see the opposite 180 degree view of the model, or we run the risk of the metaphor falling apart. This allows for the users to understand, as in the real world, at which part of the model the opposite user is looking and pointing. When we lose the ability to gauge a person's gaze and body language, it negatively impacts the realism of the experience.

FIG 4.4.3 - Local user and remote user are separated by a plane, with 3d model having the ability to cross the plane. In this representation, there is visual feedback of local user's image.

Hence, the question we focus on with regards to the visual layout is not so much about the decision between seams vs. seamless design—we have provided good arguments and prior work in the **Background and Motivation** chapter that make us confident in choosing the seamless design. However, in the seamless design there remains a question about whether there is still benefit in seeing oneself (as in traditional videoconferencing layouts) for visual feedback.

Visual feedback of oneself (**FIG 4.4.3**), often implemented in traditional videoconferencing layouts as a box in the corner with an image of the local user's reflection, could be useful for numerous reasons. Perhaps the most important reason is that the local user is unsure whether he or she is within the field of view of the camera. This visual feedback is critical for

understanding what the remote user is seeing during the videoconferencing and helps to build confidence that there is proper engagement from the remote user.

Second, the local user may wish to show something, a 3D model for example, to the remote user. Without proper visual feedback, the local user may not know which viewpoint of the object is visible to the remote user. This is especially challenging in a 2D videoconferencing setting because the camera is only capturing one viewpoint of the object, limiting what the remote user can see. We would like to understand whether this is also true in an immersive 3D environment, or whether the local user can simply "trust" what the other user sees.

Finally, with regards to co-visualizing a 3D model, the local user may wish to understand how the model appears in the 180 degree rotated view to the remote user. This could be helpful if the local user would like to point at a viewpoint of the model that is facing the remote user (in other words, a viewpoint of the model that is behind what the local user can see).

Therefore in the seamless design, there may yet be a benefit to rendering the local user's reflection combined with the reflection of the model, somewhere in the visual layout. It is not immediately apparent whether this layout will offer any benefit, and if it does, whether that benefit will come at the cost of immersion and realism of the experience.

4.5 - Interaction modes

There are two basic types of interaction modes implemented in Holosuite: visualization mode and design mode.

In the visualization mode, we envision that users can grasp at objects, rotate them freely and pass the model back and forth between each other using a pinch-to-grasp action with their hands. The user at the remote end, for example, can rotate and push the model towards the screen, and the local user will see the model coming through the screen and out to them. It is intended to be a first step towards a future where free-space haptics can be added for tactile feedback [14] [15] and tangible holograms [16]. All of this manipulation of the virtual object is done using the right hand.

In design mode, the users can create 3D shapes and transform them, similar to how design is done on traditional 3D modeling software. Either user can switch to design mode during the interaction by tapping the model with his or her left hand. This action freezes the model in 3D space, and pinch-to-grasp now transforms the model (changing the scale of the model in the dimension along the pull direction). The user can also create a shape, by drawing a two-dimensional outline (a circle, for example) using their right hand. This creates a two-dimensional shape, which can be extruded to a cylinder of an arbitrary size using a pinch-to-grasp pulling action.

4.6 - Software architecture

Interactive end-to-end 3D telepresence poses software engineering challenges similar to traditional videoconferencing, but with the added complexity of 3D rendering and holographic video fringe computation. In order to implement this, we must do an analysis and consideration of which basic software components are compute bound and which components are I/O bound. This allows us to properly parallelize the architecture so that, for example, the next audio, depth and color stream frames are being compressed in parallel while the current compressed frames are being sent through the network to the remote user.

Identifying which code blocks are compute bound and which are I/O bound guides us on how to to structure the software architecture so we can maximize CPU, GPU and I/O to provide a real-time high quality experience.

The challenges of architecture implementation lie almost completely within how quickly we can compress, decompress, send, receive and render the relevant streams of data from remote to local user and vice-a-versa. Realism calls for high framerate and high quality for all input streams, while immersion means that GPU-rendered world is implemented in a way such that the remote user appears similar to real life and that the visual aesthetic is engaging for the local user.

To create such an experience, we must architect a thread scheduler that can efficiently parallelize all processing and encoding/decoding of streams, so that data buffers are used

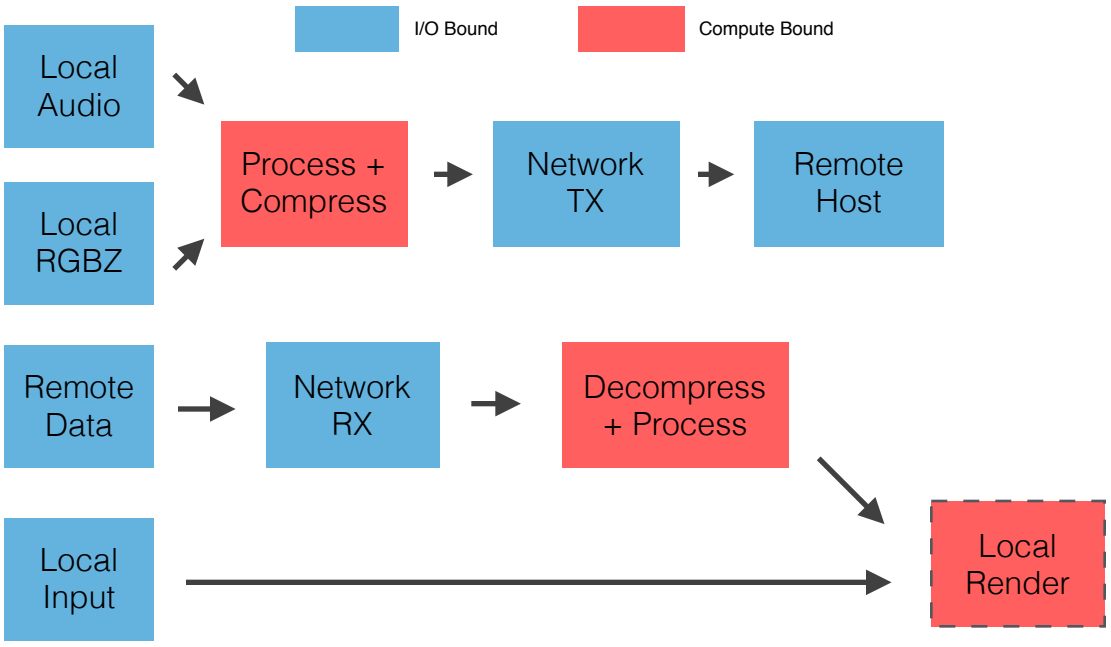


FIG 4.6 - This diagram shows the basic code blocks that must be parallelized and implemented for a telepresence session in Holosuite. Local data from depth, color, audio and hand state input streams must be processed and compressed, while the corresponding remote data must be decompressed and processed for local rendering in real-time, simultaneously.

appropriately and the CPU and GPU are not waiting on any tasks that can be done right away. Hence, we require the use of a threading model that has mutex objects and conditional variables, allowing us to avoid race conditions and wake threads when data is available for processing, respectively.

Furthermore, for evaluation purposes, we require the capability of switching input devices and rendering devices at runtime. This calls for a properly abstracted modular software design, using software libraries that can access different input devices and custom-developed 3D renderers for proper holographic rendering (computer-generated fringe computation) and holographic video simulation (3D rendering with stereo frames and motion parallax reprojection).

Finally, the architecture must accommodate multiple OS platforms, since simulating on advanced 3D displays requires coding with Windows APIs, while the MIT Mark II and Mark IV holovideo displays run on Linux OS. This means any software packages and libraries that are used must be cross-platform compatible.

The software architecture plan for Holosuite encompasses all of these scenarios, and plans for a fully parallelized two-way 3D data and audio stream compression/decompression with a highly optimized pipeline and render engine. We additionally created the requirement that all core components are abstracted as software interfaces, so that they can be changed easily (i.e. the networking stack can be changed without affecting the input or output blocks). This allows the software to be versatile at runtime, with the ability to handle many different kinds of input devices (pre-recorded depth files, or capture devices), while being able to output to multiple types of displays.

5. Development

For implementing interactive end-to-end 3D telepresence, we must create a list of quantitative targets that can fulfill our design criteria listed in the previous chapter. Namely, the real-world capture input has to be of high fidelity, at real-time frame rate, with stereo audio and precision of hand movement that can allow the user to freely rotate, translate and manipulate virtual 3D models. For rendering simulation and holovideo output, we are further constrained by what combinations of CPU and GPU hardware is capable of compressing, decompressing and computing 3D rendering scenes in real time with minimal latency and acceptable real-time frame rates.

Furthermore, the capture-to-render pipeline must be developed in a way so that while the computer is compressing the current frame of audio or video, another thread can concurrently capture, process and queue the next frame for compression. As stated in the **Design** chapter, identifying which processes are I/O bound and which are compute bound allows us to implement concurrency wherever possible, so that the CPU and GPU are being utilized and not remaining idle when there is data that is available to be processed. For low latency, however, we can only implement at most just two stages of pipelining, as each additional stage of pipelining will add a frame of delay to the processing and rendering of video and audio data, which can be detrimental to the user experience (especially considering that at a 30 fps capture rate, we are delayed by at least 33 ms capture of real-world 3D data input per frame).

5.1 - Quantitative targets for building an immersive and realistic experience

The following tables describes the quantitative targets used as our guide for hardware/software implementation decision-making, to make maximum use of hardware capabilities.

	3D Capture	3D Simulated Render	Holovideo Render
Frame rate	~30fps	~60fps (120 stereo)	~30fps
Spatial Resolution	640x480	1920x1080 Stereo	600x468 HPO Wafels
Depth Resolution	16-bit (w/ 1 mm accuracy at 1 m)	16-bit	-
Color Resolution	24-bit	24-bit	-
Latency	< 34 ms	< 17 ms	< 34 ms

TABLE 5.1.1 - Video capture must be low latency, high frame rate with high accuracy and high depth resolution to create a realistic 3D scene of the user. Simulated rendering requires the ability to render high frame rate in stereoscopic output. Holovideo rendering targets are limited by the Mark IV display.

	Audio Capture	Audio Output
Frequency	44.1 KHz	44.1 KHz
Resolution	16-bit	16-bit
Channels	2 (Stereo)	2 (Stereo)
Latency	< 20ms	< 20 ms

TABLE 5.1.2 - High fidelity audio capture and playback in stereo, based on 960-sample frames.

	Hand State Input
Frequency	120 Hz
Accuracy	0.1mm
# Hands	2
Latency	< 9 ms
Gestures	Pinch and Rotate

TABLE 5.1.3 - Hand input should have high accuracy and low latency for providing proper feedback of hand state through 3D model state.

5.2 - 3D capture device

Although there are various capture solutions available to academic researchers, we decided to use a commodity depth camera for Holosuite development, since we could meet the

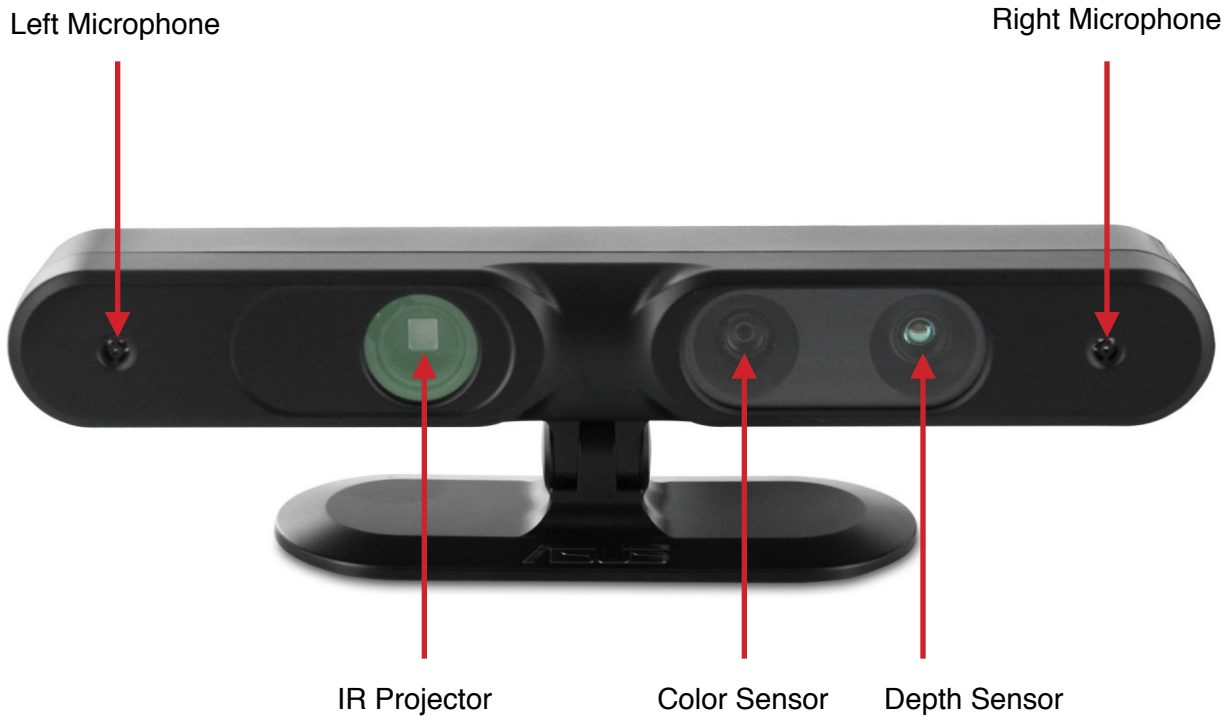


FIG 5.2 - ASUS XTION Pro Live is a commodity depth-sensing camera with stereo microphones, a structured-light depth sensor operating in IR spectrum with an RGB sensor for color image capture. The unit is self-powered via USB 2.0 port, giving data similar to the Microsoft Kinect with a smaller footprint and no need for external power supply.

development targets with the added benefit of being able to capture with hardware available to consumers.

The ASUS XTION Pro Live includes sensors for RGB and depth that give us data of real-world scenery at VGA (640x480x24) and QVGA (320x240), respectively. Although the depth sensor is only QVGA, it is upscaled by the firmware to VGA resolution. The depth camera gives 12-bit depth resolution, with an accuracy of 1 millimeter at 1 meter distance.

The ASUS XTION PRO Live uses a structured light approach to measure the depth of a scene, with a field-of-view 58˚ horizontal and 45˚ vertical, and a minimum distance of 0.85 meters. These specifications will be able to capture an accurate 3D model of a person at 1 meter distance from the camera, with the caveat that it cannot capture scenery outdoors, which our design specifications do not require.

In addition to the depth and color sensors onboard, the camera includes stereo microphones for audio capture at 48 KHz sampling rate and 16-bit resolution, which is ideal for high-quality voice capture.

5.3 - Hand-tracking device

Although it would have been possible to perform hand-tracking with the 3D capture solution, the low frame rate and minimum distance issues facing commodity depth-sensing cameras make it difficult to perform fluid and natural hand pose and tracking estimation. Hence, we looked toward a solution that could accurately track hands with minimal latency and high frame rate.

FIG 5.3 - The LEAP Motion is an advanced hand-tracking solution, operating over USB 3.0 to give 120 FPS tracking of two hands in 3D space, with full skeletal articulation and position for every joint in the hand, including the ability to detect some discrete gestures, such as swipe, circle and tap.

The LEAP Motion, another commodity device is capable of tracking two hands, giving 0.1mm of accuracy position tracking for every joint in two hands simultaneously. The LEAP SDK, which provides access to the device, can also recognize a few simple gestures, such as pinch, circle and tap.

5.4 - The program: using C++11 and Holosuite executable

Implementing the architecture of Holosuite requires a programming language that compiles fast native CPU code, has native threading support, with a programming paradigm that is conducive to development of interchangeable I/O code blocks for interfacing with hardware (object-oriented programming model). In addition to this, the programming language must be able to access a large number of cross-platform libraries for hardware SDKs, audio and video codecs, computer vision processing and 3D rendering.

For these requirements, C++ is an obvious choice, as it gives the power of being able to compile code to fast assembly, while maintaining properties of high-level abstraction for complex software engineering projects, which low-level languages such as C are lacking. The creation of the C++11 standard introduced native threading support to C++, without which coding a responsive user experience would have been much more difficult.

The programming effort involved in the implementation of the architecture, as imagined by **FIG 4.6**, and defined in **Appendix A**, **Appendix B**, and **Appendix C** was considerable, resulting in over 17,000 lines of code. It was created as a cross-platform CMAKE project, which could compile on Windows and Linux primarily, though it was also tested to compile on OS X. It was coded using Visual Studio 2013 and Eclipse IDEs.

The resulting Holosuite C++11 project compiles into a command line executable that has program options for starting a session as a server, client, or loopback (which simply takes the input devices, compresses, transmits, receives and decompresses input streams locally and renders the resulting output on one machine). The loopback mode was created for the purpose of testing the networking stack, along with codecs and renderer without the necessity to debug and develop on two machines.

With the Holosuite executable, the user can also set a variety of input options, such as what type of camera input to use, the resolution settings for capture devices, the frequency for audio, or codec settings for compression rates and render options.

5.5 - The program: open source software libraries

Open source libraries were used extensively in the development of Holosuite. Relying on community-built software allowed the project to be feature-complete and cross-platform, with the ability to import various 3D model formats and perform video, audio and depth data compression. The table in **Appendix D** denotes which open source software libraries are used in Holosuite, their version and the respective purposes.

Libraries such as Point-Cloud Library and OpenCV were strategically chosen, as they provide many different functions for digital signal processing of depth and color streams, which can be further utilized for improving the fidelity of the image, rendering and segmentation of the user.

5.6 - Audio compression

For the sake of making Holosuite a realistic experience, we wanted to take advantage of the fact that the ASUS Xtion Pro Live has a stereo microphone layout. Using stereo audio input, we can get an audio signal with better fidelity than normal video conferencing, along with high sampling frequency to match CD quality.

For these requirements, some common codecs were considered, such as MP3, OGG and even lossless audio codecs such as FLAC. Lossless codecs would prohibit usage across common internet connections, since the bitrate for lossless audio is too high and the benefit of lossless audio is dubious, as speech frequencies comprise only a small band of frequencies in the auditory spectrum (usually quoted as being between 20 Hz - 22 KHz). Furthermore, the traditional PC speaker system does not reproduce the low end range of frequencies which can be sampled by CD-quality audio.

For these purposes, we made the decision to use a newer, open-source audio codec built specifically for high-quality, low-bitrate real-time telecommunications, called Opus. The description of Opus on the website is as follows:

> Opus can handle a wide range of audio applications, including Voice over IP, videoconferencing, in-game chat, and even remote live music performances. It can scale from low bitrate narrowband speech to very high quality stereo music. Supported features are:
>
> - Bitrates from 6 kb/s to 510 kb/s
> - Sampling rates from 8 kHz (narrowband) to 48 kHz (fullband)
> - Frame sizes from 2.5 ms to 60 ms
> - Support for both constant bitrate (CBR) and variable bitrate (VBR)
> - Audio bandwidth from narrowband to fullband
> - Support for speech and music
> - Support for mono and stereo
> - Dupport for up to 255 channels (multistream frames)
> - Dynamically adjustable bitrate, audio bandwidth, and frame size
> - Good loss robustness and packet loss concealment (PLC)
> - Floating point and fixed-point implementation

Started as an attempt to make a low-bitrate, high-quality audio codec for telecommunications, Opus has turned into a competitive general-purpose audio codec, rivaling that of the well-

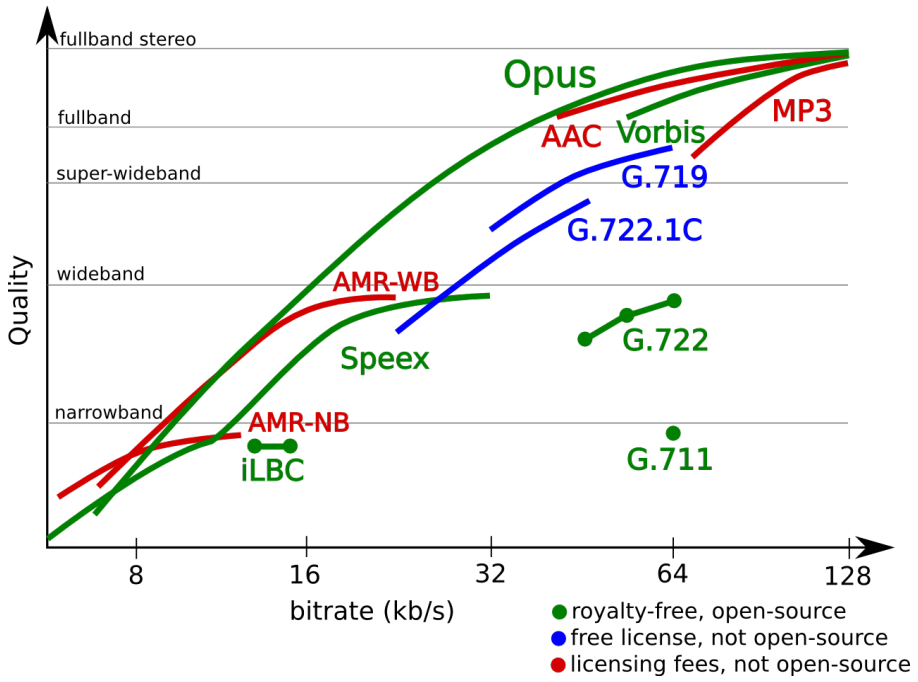


FIG 5.6.1 - Listening tests show Opus covers from narrowband to fullband stereo at competitive quality across low and high bitrates.

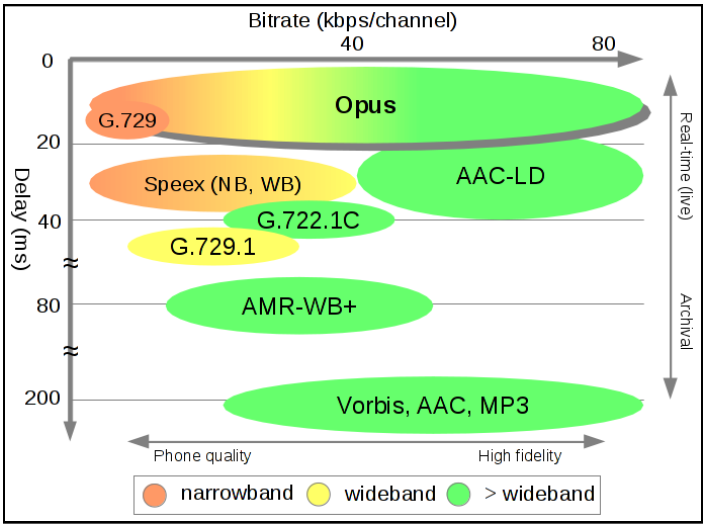


FIG 5.6.2 - Opus covers low-latency encoding with a wide bitrate range, for high quality encoding.

established MP3, Vorbis and AAC in listening tests, with the added advantage of being low-latency (which is critical to telecommunication).[2]

The Opus API offers a flexible selection of bitrates and encoding qualities, which we programmed to be changeable options as part of the Holosuite main executable.
The default settings we used were 48 KHz sampling rate, fullband stereo with an encode frame size of 960 samples. This resulted at high quality stereo audio with low latency, which was determined to be sufficient for the purposes of understanding one another clearly, and to be able to distinguish sounds coming from the left vs. right channel.

5.7 - Virtualizing the user from real-world data to 3D rendering

To visualize the remote user, it is not only necessary to have color information, but also a depth component for each color element. This is because depth information is required for rendering stereo and parallax effects.

Fortunately, using a depth-sensing camera solution, it is somewhat trivial to end up with real-world 3D data of the remote user. However, there was a question of whether to create a mesh of the user or to use a straightforward point-cloud rendering approach. A mesh can be helpful in compression as it is inherently a low-information data structure, whereas a point cloud data structure is more faithful to the original raw captured data coming from the camera. To further complicate matters, the method of visualizing the user can also affect which segmentation method, compression algorithms to use and the rendering approach, which makes the decision somewhat tricky.

One of the benefits of using Point Cloud Library is that it allowed us to test various scenarios to determine the best structure and method for virtualizing the real-world 3D data captured from the camera. Ideally, we can state that a 3D mesh of a person would offer the best visual quality and be most appropriate for rendering, as modern GPUs are made strictly for the purpose of rendering a high number of triangles in real time.

This assumption, however, seems to ignore some other other important considerations. We tested the fast meshing function in Point Cloud Library to get an idea for how meshing would look, and discovered several drawbacks: first, the quality was not so visually appealing since we were using only one camera, holes and rough edges in the mesh were quite apparent and difficult to tolerate visually. Although this could be solved by adding additional cameras, and additional image processing, it would add further burden to the CPU and USB bus which was of great concern.

Second, the visual quality of the meshing function had the negative effect of putting the person's likeness into the "uncanny valley", presumably because the reduction of data points and averaging of colors and depth values created an unrealistic model of the user which appeared artificially generated, almost as if it was drawn by a 3D artist, not sampled by an image sensor from the real world.

---

[2] Opus specs and figures from http://www.opus-codec.org/comparison

Finally, meshing algorithms also take a lot of compute power, which would hurt the performance of compression and decompression of audio, depth and color streams on the CPU. Even if done on the GPU, it would undoubtedly limit the compute power for performing fringe computation for holovideo displays, so for these reasons, so creating a mesh was an undesirable option.

FIG 5.7.1 - This scene is taken of a user facing the depth camera, and the resulting 3D data is rotated to visualize the side angle of the face. On the left is the simple point cloud representation of a segmented user, the right image shows the scene from the same angle using fast mesh implementation from Point Cloud Library. Notice that the edges and holes on the mesh are much more difficult to tolerate visually. In the point cloud we can cover and smooth these simply by changing the size of the points.

It turned out that segmenting the user by doing a simple threshold of depth data within a reasonable range (assume that the user sits stationary somewhere between 1 and 1.5 meters of the camera and there are no other objects in the field of view of the camera at this depth range) combined with a point cloud data structure rendering was enough to create a convincing and realistic result.

Conversion to a point cloud is somewhat trivial, as the ASUS Xtion Pro Live gives us a depth image which is registered to the color image, meaning that for each X, Y in the color and depth streams, the color and depth data is aligned and matching. The major caveat here being that depth data from an image sensor is given in pixel coordinates, not in real-world coordinates, which means that we need to convert to real-world coordinate space from a 640x480 grid of depth values.

Therefore, in order to build a point cloud, we need a robust algorithm that combines red green blue and depth information for each color and corresponding depth image element to create an individual voxel in real-world 3D space, while also re-projecting each voxel into real-world coordinates. This must be done for every frame, at a rate of ~30 frames per second, as this is the design target for creating realistic and fluid presence of the remote user.

To do this, we start by capturing two images (one color image of 640x480 elements with an RGB value, and a depth image of 640x480 elements of 12-bit Z, or depth, values). This must be converted to a data structure that can be processed easily by a GPU shader, where the X, Y, Z field of each voxel contains its position in 3D space (not the same as the depth value's position in the pixel grid). This, called the pixel spread function, can be calculated by a trigonometric function relating the spread of a voxel as its distance gets further from the camera, and the camera's field-of-view.

FIG 5.7.2 - The voxel data structure is a vector composed of 32-bit floating point values in X, Y, Z for the position and 8-bit color values for red, green, blue and alpha of each voxel in the image. This is a dense point cloud structure, meaning that for each X*Y resolution frame in the scene, we end up with X*Y number of voxels. However, because some data points from the depth sensor are unreliable, or bad (0 depth value), we must set X, Y, Z to NaN (Not a number), so our renderer knows to not process such voxels.

The basic voxel structure, pictured above in **FIG 5.7.2**, is repeated for every pixel element in the captured scene. In our capture method, we have 640x480 image sensor resolution, which amounts to 307,200 individual voxels in 3D space, rendered as points with 8-bit red, green and blue values. In practice, however, we end up rendering many fewer points, as segmenting the user and removing the background during a pre-processing step reduced the number of points for processing. This also sped up our rendering frame rate and helped to creating a more immersive experience where the background can be drawn artificially in a way that emphasizes the subjects.

The pseudo-algorithm for determining the real-world position of a voxel in a right-handed coordinate system from *x*, *y* in the pixel map is as follows:

```
foreach depth_value in depth_image
   if depth_value == 0
            voxel.x = voxel.y = voxel.z = FLOAT_NaN;
            voxel.r = voxel.g = voxel.b = 0;
            continue;
   else
            voxel.z = depth_value * 0.001f;   //convert from mm to meters
            voxel.x = (y / y_image_resolution - 0.5f) * voxel.z * 2 * tan(vov/2);
            voxel.y = -(x / x_image_resolution - 0.5f) * voxel.z * 2 * tan(hov/2);
            voxel.r = color_image[x,y,red];
            voxel.g = color_image[x,y,green];
            voxel.b = color_image[x,y,blue];
   end if
   voxel.a = 255;
end foreach
```

The voxel Z position is determined simply the depth sensor value at a given x, y coordinate. This is converted to meters (the raw data is given in millimeters)

$$z(x,y) = v_z = 0.001z \quad \text{(Eq. 1)}$$

The voxel X position is determined by the equation

$$v_x = 2 \cdot \left(\frac{y}{\text{res}_y} - \frac{1}{2} \right) \cdot v_z \cdot \tan\left(\frac{fov_y}{2} \right) \quad \text{(Eq. 2)}$$

where y is the pixel coordinate in the vertical dimension of the image, res_y is the resolution in the vertical dimension, and fov_y is the field-of-view of the capture device in the vertical dimension and v_z is given by **Eq. 1**.

Likewise, the voxel Y position is determined by the equation

$$v_y = -2 \cdot \left(\frac{x}{\text{res}_x} - \frac{1}{2} \right) \cdot v_z \cdot \tan\left(\frac{fov_x}{2} \right) \quad \text{(Eq. 3)}$$

where x is the pixel coordinate in the horizontal dimension, res_x and fov_x are the resolution and field-of-view of the image sensor in the horizontal dimension, respectively, and v_z is given by **Eq. 1**.

In Holosuite, the point cloud generation function can either run as the last step before rendering the scene, or it can be done at capture-time. This is because Holosuite implements two different types of compression that can be chosen at runtime, one that operates on point cloud data and another that operates on raw color and depth streams. Holosuite will choose at which step in the capture-to-render pipeline to generate the point cloud, based on which compression algorithm is selected by the user at runtime.

5.8 - Depth and color image compression

Depth and color image data combined account for the vast majority of data that must be transferred over the network, both ways, in real time to create the experience of end-to-end 3D telepresence. There are two methods which we explored and eventually implemented for compression: a lossy point-cloud based compression and hybrid image-based compression scheme.

Point Cloud Library contains a compression scheme for data already in a point cloud structure which performs a spatial decomposition on the dataset by converting the point cloud structure into a hierarchical data structure (octree data structure). It also exploits temporal correlation (motion estimation) in the point clouds between captured frames, allowing the algorithm to only send the differences between key frames to save bandwidth [17].

Octree compression has some key advantages, in that changing to a hierarchical data structure allows us to perform a fast voxel filter on the data set, greatly reducing the number of data points that must be transmitted and rendered on the remote end, at the cost of spatial resolution of 3D data. This is typically an ideal situation if there will be some post-processing done for creating a mesh on the GPU on the remote end, as the remote connection will receive a sub-sample of the point cloud, a minimum amount of 3D points determined to triangulate the cloud and turn it into a mesh.

For transmitting a dense point cloud, however, this method is not very effective, as the implementation in Point Cloud Library takes great compute power to compress each frame. For this reason, we also implemented a hybrid 2D signal image compression method, which creates a lossy color image packet per frame combined with a lossless compressed depth image.

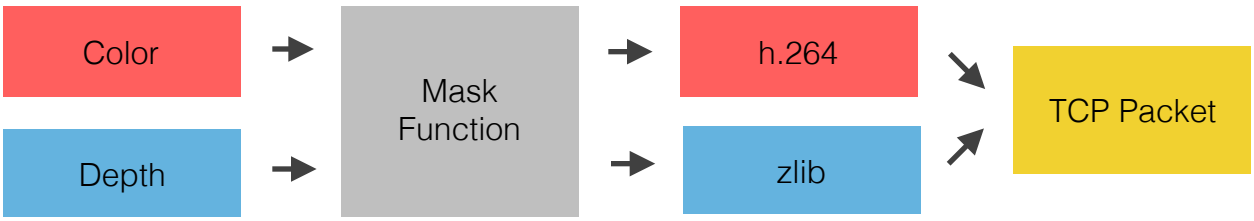


FIG 5.8 - The hybrid image-based compression scheme compresses the color frame using lossy H.264 low-latency video compression, and ZLIB for lossless depth stream compression. these two compressed frames are combined into one TCP packet to be sent to the remote host.

This hybrid approach works by using the zero-latency compression option with FFMPEG/ libx264, an open-source h.264 codec, along with zlib open-source library for lossless compression of depth data. Before the color and depth frame are compressed, they are processed through a masking function, which sets all of the background pixels (the segmented pixels not related to the user) in the color image to null-value, reducing entropy of the data. This effectively allows the depth to be compressed to a smaller size, and the h.264 compression to focus most of the encoding effort on the user's face and body.

In the hybrid compression technique, Holosuite will generate the point cloud on the receiving end by using the remote user's camera intrinsics (sent during a handshake packet when the network connection is formed between two Holosuite sessions). For a detailed explanation on the handshake information, see the **Networking** section of this chapter. This approach is desirable as it perfectly retains the 3D position data for each voxel in the point cloud, along with industry-standard color video compression, which can be further optimized as many devices today carry h.264 codecs built into hardware. It also places low burden on CPU and GPU, as the compression is done with least effort and lowest latency, while allowing the receiving end to take the burden of generating the 3D data for rendering.

5.9 - Shared virtual 3D model

The shared virtual 3D model data is imported from a file in one of the users' computer, sent once as a collection of vertices, normals and vertex colors from the owner to the remote user when the connection is established. No compression is done on the 3D model, as it is sent only once.

However, when the model is rotated, translated or scaled by the local user, the state of the 3D model is sent to the remote user. This allows the remote user to see the state change of the 3D model in real time, without having to receive the vertices of the 3D model every time the state of the model changes.

3D model files are imported via "assimp", an open-source library that can read many different 3D asset files, such as OBJ, PLY, 3DS, etc. The assimp API gives developers the ability to import vertices, normals, and colors of the model in a standard format, as well as the ability to simplify and triangulate a mesh so that duplicate vertices are removed. We implemented these features so that any arbitrary 3D model can be imported and stored in CPU memory in an efficient manner.

Once the 3D model is imported, but before it is rendered, we have to normalize the dimensions, so that the initial size of the 3D model is reasonable for visualizing and manipulating. For this, we employed the "Miniball" algorithm [18], which finds the smallest bounding sphere of all points in an N-dimensional space. The bounding sphere of the 3D model allows us to determine the scaling factor necessary to resize any arbitrary 3D model to a normalized scale.

5.10 - Networking

Holosuite's networking component uses TCP and runs on its own thread for transmitting packets and in the background will schedule and combine data for packets to be sent. Receiving is done synchronously, parsing incoming data and queuing each packet to separate threads for further processing, which is stored in a standard Type-Length-Value (TLV) format.

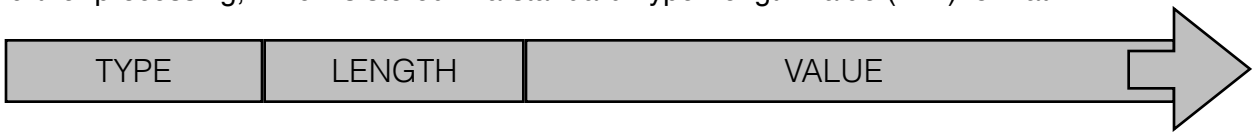


FIG 5.10 - One packet in Holosuite has a 32-bit value for TYPE, and LENGTH, and packet data is stored in VALUE, which is a number of bytes determined by the LENGTH field. A type can be audio, color, depth, 3D model state, 3D model vertices, client name, capture resolution, audio resolution, etc.

TLV format allows us to send different types of data of arbitrary length through a TCP socket, so that the remote end can easily parse and process the incoming data. For telepresence, we can send any number of "Types" of packets, such as depth data, color data, audio data, 3D model vertices state, etc. Also, since we are using compression on many of these streams, we don't know how big the frames for audio, depth and color streams will be until just before they are

sent to the remote end. The variability in the length of the data for different data types makes it necessary to employ a system such as TLV.

Holosuite can be run as a client, or a server. The client mode will require the user to enter an IP address or hostname of a machine, and the server is responsible for accepting a socket connection from clients. Once the connection is formed, a handshake packet is sent by both machines to each other, giving some important specifications about the data format and client, such as protocol version, number of audio channels, audio sampling frequency, depth and color resolution, camera field of view, etc.

The structure of the handshake packet is as follows:

```
struct HoloNetProtocolHandshake
{
        uint32_t magicNumber;
        uint32_t protocolVersion;
        uint8_t clientName[HOLO_NET_NAME_STR_SIZE];
        int videoCodecType;
        int audioCodecType;
        uint32_t rgbazWidth;
        uint32_t rgbazHeight;
        float captureFPS;
        float captureHOV;
        float captureVOV;
        int audioNumChan;
        int audioFreq;
        int audioBitDepth;
};
```

This handshake information is used to appropriately parse and decompress the incoming audio, color and depth streams, and gather information necessary for generating a point cloud of the received depth and color streams.

We considered other networking protocols, such as UDP, and another lesser-known protocol called “UDT”, which is built on top of UDP, but with TCP-like properties. UDP was not implemented in the initial stage of development because we preferred to first implement a networking stack that could reliably send full data packets across the internet. UDP is connectionless and data is not guaranteed to arrive, nor is it guaranteed to arrive in order, which complicates the networking development. When we tested an implementation of the TCP protocol, it was good enough to do live streaming of all the streams simultaneously in real-time with minimal latency, so we opted to maintain the networking stack on a TCP socket connection. Holosuite could theoretically take advantage of UDP by implementing more robust frame-dropping behavior, which could help when there are moments of bandwidth degradation.

For testing the networking component, we implemented a “loopback” session mode, which runs a server with the input capture devices and connects a client session with the output (renderer and audio) devices. This was implemented to verify the correctness of the networking code without the necessity for running a server and client on separate machines, while also testing the encoding and decoding of depth, color and audio data.

5.11 - Holographic video simulation rendering

In the **Introduction** chapter of this thesis, we described the necessity for implementing a simulation renderer—i.e. a renderer that can produce some of the most important cues of holographic display rendering to simulate the ideal Holovideo display. As we did not particularly aim to take advantage of accommodation cues because of the relative shallowness and simplicity of the scenes, we employed the usage of advanced 3D displays made by ZSpace, Inc. which provide reliable stereoscopic and motion parallax tracking.

FIG 5.11.1 - The ZSpace display, with glasses and a 3D stylus used for interaction in ZSpace application demos (not used in Holosuite).

The ZSpace advanced 3D display requires the user to wear glasses, which provide left and right passive stereoscopic vision by alternating stereo frames with reverse polarity, allowing the left eye of the user to see only the left frame of the scene and the right eye to see the right frame. The glasses are also designed with 5 white reflective dots on the front, which reflect illuminated IR light and are captured by IR cameras built into the display for tracking the user's head position and orientation in 3D space, relative to the display.

The tracking system on the ZSpace display is very robust, and the ZSpace API is able to give us not only the position and orientation of the user's head, but conveniently provides the

transformation matrix, which we can apply to the projection matrix in OpenGL to create the stereo frames and motion parallax reprojection for every frame in the render loop.

Thus, in setting up the simulation renderer, there were two simple steps invloved: first was to create an OpenGL renderer from scratch, one that would visualize the remote user's point cloud along with the shared model and an arbitrary background. This OpenGL component was written to take advantage of OpenGL 3.0 calls that allow us to quickly copy large chunks of vertex, normal and color buffers and draw them all at once, instead of the traditional method of making a function call for each vertex, color and normal in the 3D dataset.

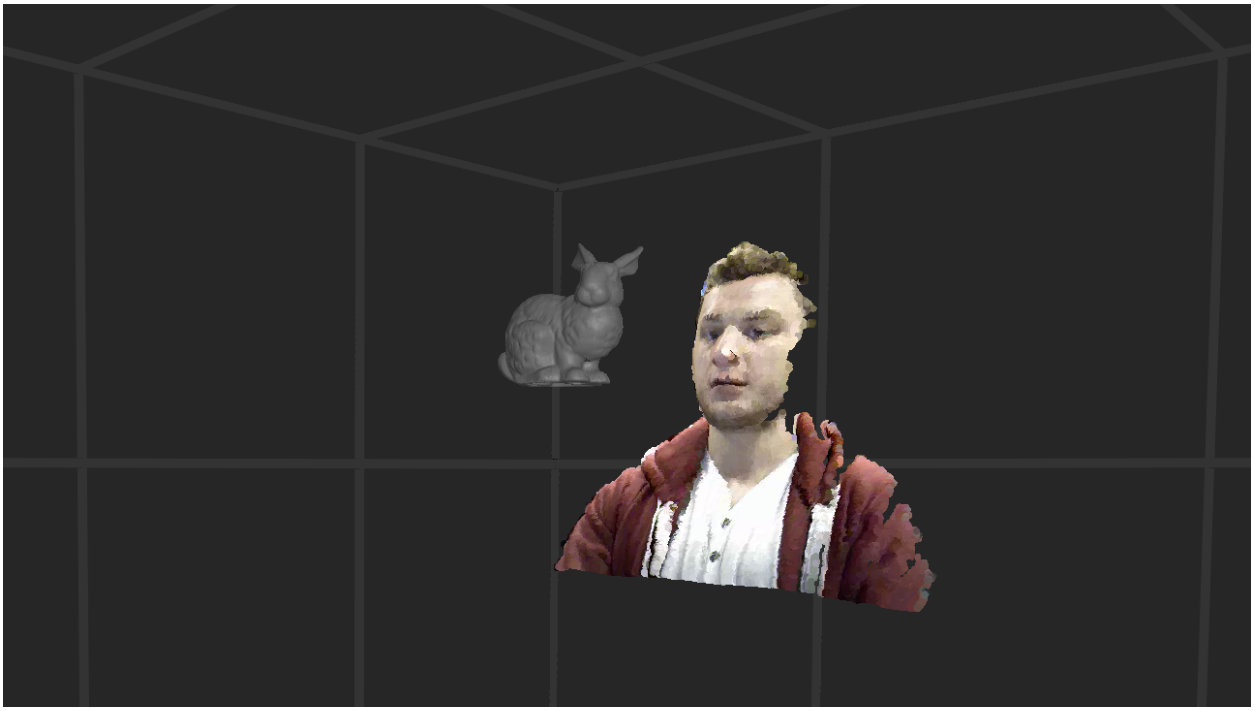


FIG 5.11.2 - The simulation OpenGL renderer in perspective projection, showing a segmented 3D point cloud of the remote user from a 3/4 angle, along with the Stanford Bunny scanned 3D model, imported from a file. The background is a simple 3D grid of a box—the lines in the background help to emphasize the motion parallax effect, while creating a sense of immersion for the viewer, drawing his or her attention only to the subjects in the rendered 3D scene.

The second step was to access the ZSpace API for the head tracking information, to get the transformation matrix and apply it to the projection matrix. This was a relatively straightforward procedure, although we had to tweak the position of the remote user's point cloud and model to put the user behind the display plane and the model in between (in the middle) of the display plane. Once we had the transform matrix, we could create stereo frames, using quad-buffered stereo buffer flipping built into OpenGL, while reprojecting the view of the scene every frame draw for motion parallax with the new transform matrix.

The OpenGL simulation renderer was also incredibly useful for debugging and testing, as development could be done on a normal LCD display. We implemented a switch in the Holosuite program to enable ZSpace rendering, so that any changes that were made to the renderer in OpenGL could easily be tested after some development, without the necessity of the ZSpace display.

5.12 - CGH (Computer Generated Holography) rendering

There are a few methods we considered for generating holographic images through computation. Traditional methods, such as Fresnel computation, involve simulating the physics of the way light interferes with a 3D scene to produce an digital representation of the interference pattern. This method is akin to modeling the physical creation of an analog hologram on a glass coated with light-sensitive material. Fresnel computation has been known since at least the late ‘70s, and can construct a highly realistic image, albeit at the cost of incredible computation power. Consider that such a method typically must model the behavior of light interfering at a single 3D point in a scene, as it it interferes with every other 3D point in the scene, repeated *for every* 3D point in the scene. This has a high computational complexity, making it impractical for real-time computation of high-resolution 3D scenery—taking minutes to compute a single holovideo frame on contemporary GPU hardware. For this reason, we did not consider Fresnel computation as a viable option for Holosuite.

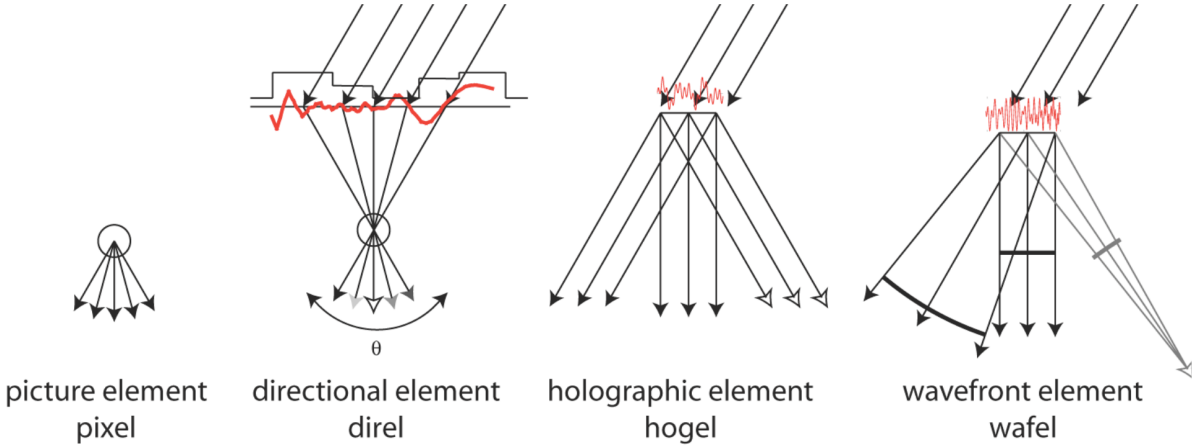


FIG 5.12.1 - This figure from J. Barabas et al. [9] shows the difference between pixels, hogels and wafels. note that wafels are able to produce curvature of a wavefront, which enables the accommodation cue for the user (ability for the eye to focus).

Newer approaches, such as the diffraction-specific hologram computation approach, do not attempt to simulate the interference pattern through a physics model, rather they solve the problem of displaying a holographic image in a backwards way. That is to say, the diffraction-specific approach aims to discretize the holographic image into fundamental elements first, solving for what these discrete elements on the display would show to the user. This is similar to how pixels on a normal 2D display are generated, but with a directional vector dimension in addition to the 2 spatial dimensions. This fundamental element, called a “hogel” (short for holographic element), is visually similar to a pixel in that it constructs an image as a grid of discrete elements in the horizontal and vertical space. Unlike a pixel, however, a hogel can vary its color value along the incident viewing angle. From the viewer’s perspective, this is experienced as the image changing depending on the angle at which he or she is looking at the holovideo display.

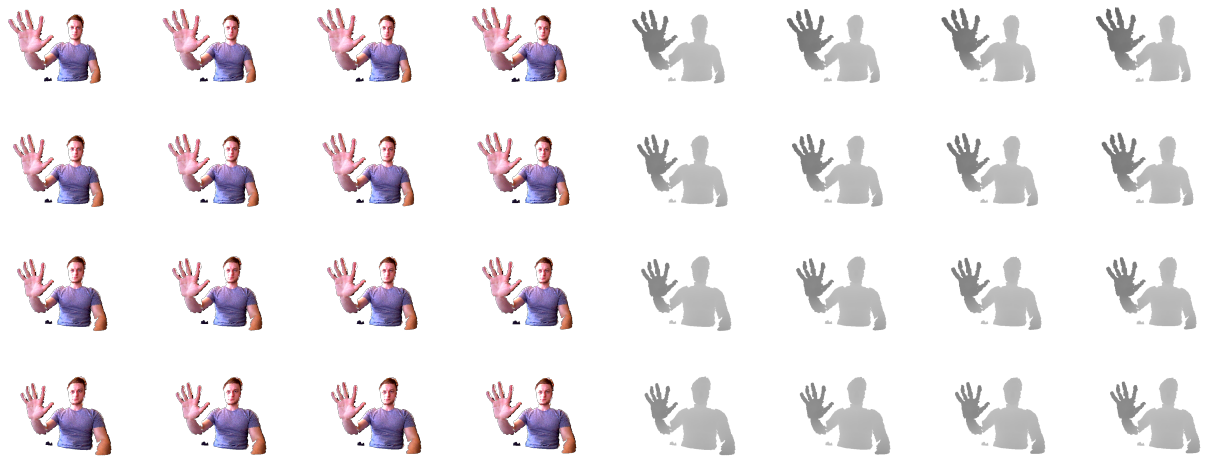


FIG 5.12.2 - The first stage of diffraction-specific algorithm for the Mark IV: the parallax view set is generated from the 3D scene, composed of 16 discrete views which represent the basis of the "light-field". The left set is the color component of the 3D scene, while the right set is the corresponding depth component. The view set begins at the top-left, moves left-to-right, where the bottom-right is the last image in the view set.

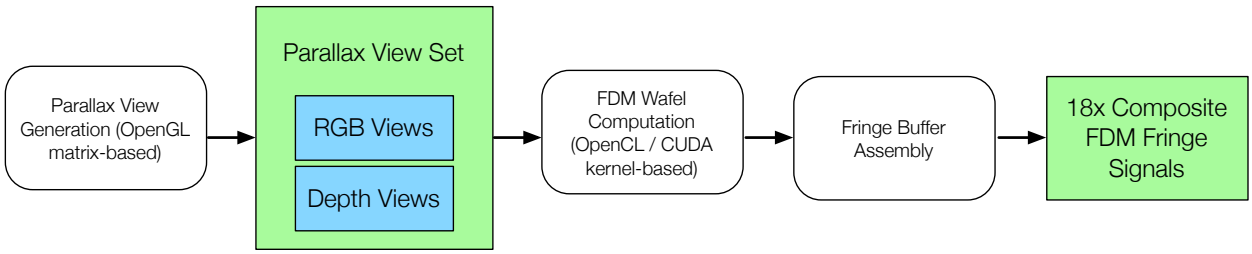


FIG 5.12.3 - Figure from S. Jolly, E. Dreshaj and V. M. Bove, Jr. [19] showing the stages of GPU implementation of full color diffraction-specific hologram image creation used in the Mark IV display.

The latest variants of diffraction-specific algorithms allow us to further refine the visual quality of the image by producing a new type of fundamental element, called a "wafel" (or wavefront element). One of the key benefits of using wafels over hogels is that while hogels rely on generating hundreds of views to produce a smooth motion parallax experience, wafels can produce smooth motion parallax with much fewer views (as few as 16). This is because wafel computation can better approximate wavefront curvature, saving precious compute power that can be used for other stages in the hologram generation.

$$Q = \begin{bmatrix} \frac{2(f+p)}{X_{right}-X_{left}} & 0 & \frac{X_{right}+X_{left}}{X_{right}-X_{left}} & \frac{X_{right}+X_{left}}{X_{right}-X_{left}}p \\ 0 & -\frac{2p}{Y_{top}-Y_{bot}} & 0 & 0 \\ 0 & 0 & \frac{f+n+2p}{f-n} & \frac{2fn+p(f+n)}{f-n} \\ 0 & 0 & 1 & -p \end{bmatrix}$$

f: far clip plane
n: near clip plane
p: center of projection

X_{left}	left clip plane
X_{right}	right
Y_{top}	top
Y_{bot}	bottom

FIG 5.12.5 - The viewing double-frustum projection matrix used to generate each view in the parallax view set as explained by Smithwick, Barabas, Smalley and Bove, Jr. [4].

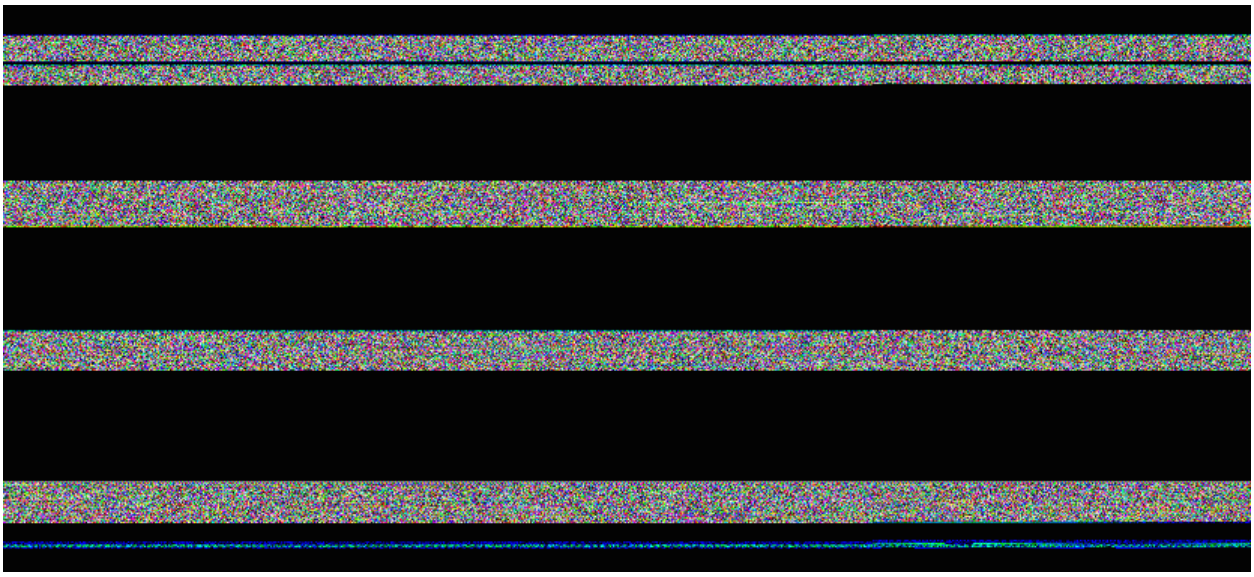


FIG 5.12.4 - The second stage of diffraction-specific computation for the Mark IV: the hologram is generated from the parallax view set, by computing abutting wafels in every scanline. The figure shows a segment of the resulting computation, output as a frame buffer signal to the Mark IV holovideo display via the GPU VGA port. Each line in the image corresponds to parts of analog signal sent to the acousto-optic modulator, which diffracts laser light to produce a 3D image of the user's point cloud.

In Holosuite, we employed a diffraction-specific rendering algorithm for two different holovideo displays: the Mark II monochromatic red-laser display and the Mark IV full-color display. As any other diffraction-specific algorithms, the process of generating the hologram for each frame follows a basic two stage process: generating the parallax view set and computing the hologram from the parallax view set.

The first stage, similar to normal GPU rendering, involves creating the light field by way of transforming the geometry of the 3D scenery to create the parallax view set. In our approach we form a collection of renderings of the same scene, from different perspectives in the horizontal dimension taken across a viewing angle. Since both displays are HPO (horizontal parallax only), we need only compute a parallax view set in the horizontal dimension; however, we must apply a transformation to the scenery so that the view set is rendered orthographically in the horizontal dimension, but in perspective format in the vertical dimension. This is because in the next stage of generating a hologram, we must use the resulting parallax view set to compute the hologram with accurate to real-world geometry in the horizontal dimension. See **Fig 5.12.2** for an example image of the view set.

Note that the first step of the first stage is nearly identical to the OpenGL simulation renderer discussed in the previous section—we create a 3D scene with a lighting model, occlusion, and draw our point cloud of the user. However, in addition to the normal rendering pipeline, we create the parallax view set of our scene from many angles, to prepare for the next stage in hologram generation.

The second stage, known as the hologram computation, is where the light field information is encoded in a digital signal from the parallax view set, along with the physics of optical propagation of the display—the "diffraction signal". In literature this is also referred to as

computing the “fringe pattern” or “interference pattern”. The signal is computed on GPUs using GPGPU (OpenCL or CUDA), and output through the analog VGA port, connected to the light modulator (in our case, a guided-wave acousto-optic modulator).

In the hologram computation stage, we use CUDA to compute a “chirp” for each wafel in the holovideo display (600 per hololine) from the Z-value of the view set generated in the first stage. The chirp can be described as the signal that encodes the depth of a wafel along a directional vector, representing a diffraction grating. The transmittance function for an unmodulated chirp is given by

$$t(x) = cos\left[\frac{2\pi}{\lambda}\left(\sqrt{(x - x_0)^2 + z_0^2} - x_0 + x\ sin\ \theta_r\right)\right] \qquad \text{(Eq. 4)}$$

where x is the position on the composite hologram transmittance function, (x_0 , z_0) is the position of a scene point to be reconstructed, and θ_r is the angle of the reconstruction beam relative to the normal of the hologram plane, and λ is the wavelength of the illumination beam.

This signal, however, has to be further modulated to represent color information (varying intensity of light along a directional vector) from the red, green and blue channels of the view set. For monochromatic displays, such as Mark II, the chirp can easily be modulated by the luma signal of the view set. Having a color holovideo display (such as Mark IV), means that one channel in the holographic display must carry the signal from all three color components of the 3D rendering. Therefore, when modulating a chirp for the Mark IV, we have to employ a signal processing technique known as SSB (single side-band modulation) to combine the frequency contributions from red, green and blue into one channel.

The transmittance function for an SSB-modulated chirp is given by the equation

$$t_{SSB}(x) = \cos\left[\frac{2\pi}{\lambda}\left(\sqrt{(x - x_0)^2 + z_0^2} - x_0 + x(\sin\theta_r + \lambda f_0)\right)\right] \qquad \text{(Eq. 5)}$$

where the terms are the same as in **Eq. 4**, and f_0 is the material carrier frequency described by S. Jolly et. al [19].

The final wafel generation for Mark IV is given by

$$W_{FDM}(x) = \sum_{i=1}^{3}\sum_{j=1}^{16} m_{ij}\ cos\left[\frac{2\pi}{\lambda}\left(\sqrt{(x - x_0)^2 + z_i^2} - x_i + x(sin\theta_r + \lambda_j f_j)\right)\right] \qquad \text{(Eq.6)}$$

where $f_j = Nf_{jT}/Pw$ is the j-th spatial upconversion frequency corresponding to the j-th temporal upconversion frequency f_{jT} , x is the spatial position on the composite hololine, x_i is the wafel center position on the composite hololine, θ_r is the angle of the reconstruction beam relative the

hologram plane normal, λ_j is the wavelength of the j-th color channel, and (m_i, z_i) are the color and depth values retrieved from the i-th parallax views in RGB and depth [19].

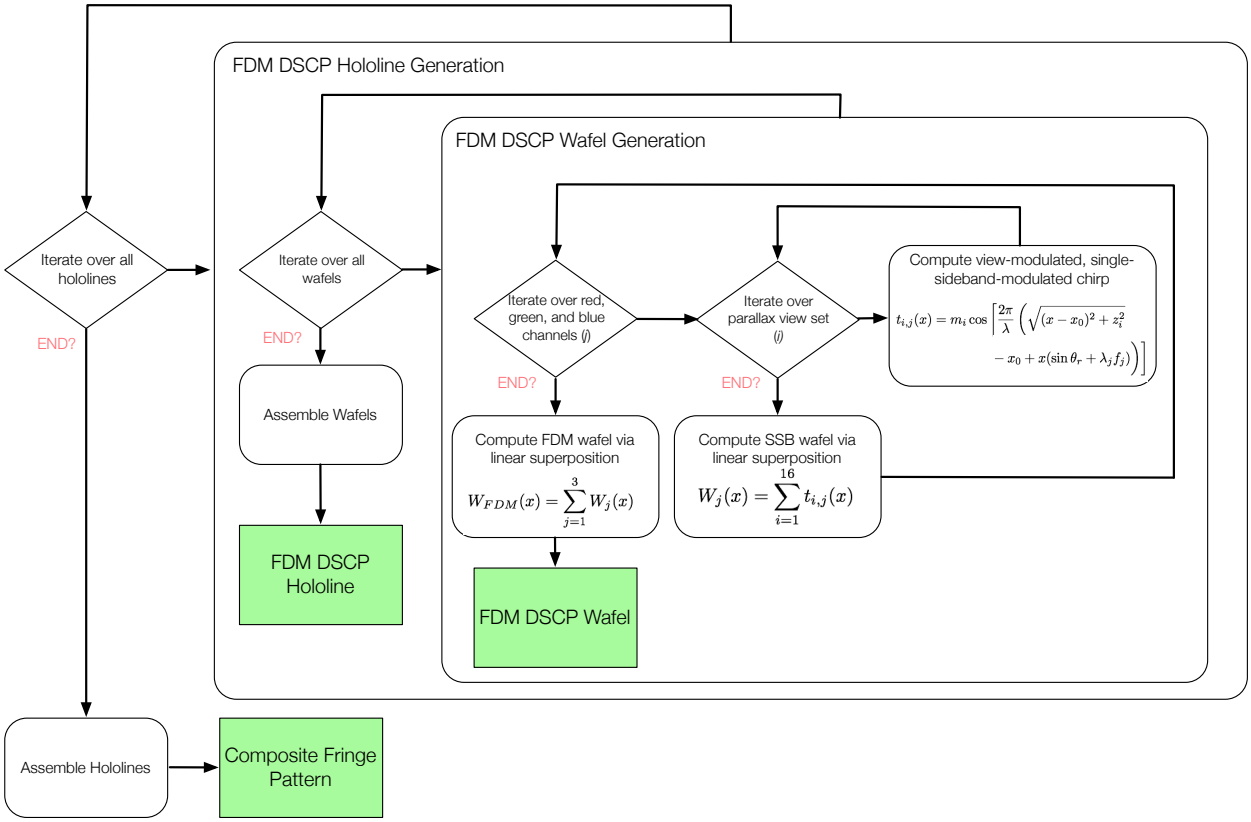


FIG 5.12.6 - Figure from S. Jolly, E. Dreshaj, V. M. Bove, Jr. [19] describing the pipeline for generating abutting wafels that make up the entire display of the Mark IV.

6. Evaluation

Since the scope of the development of Holosuite covers solving technical problems, as well as implementing an HCI (human-computer interface), we performed a hybrid quantitative-qualitative analysis to evaluate the result of the application.

For the quantitative analysis, we were chiefly concerned with whether we could perform compression, transmission and rendering for end-to-end 3D telepresence in real time. The compression levels would need to be acceptable for an average home broadband connection (in the range of 10-20 Mbps), audio, depth and color frames must be encoded with low latency and rendered at high frame rates (between 20-30 fps).

6.1 - Operating hardware specs

Holosuite was evaluated on two PCs built with the same component build. We used high-end GPUs, necessary to render complex point clouds, and also to compute holograms for each frame in real-time. We chose CPUs that could handle many threads, with large cache sizes as we would be concerned about a memory copy operation wiping out the entire cache space every time a frame is copied.

Part	Type	Description
CPU	Intel i7-4820K CPU	4 Cores / 8 Threads, 10 MB Cache @ 3.7 GHz
GPU	NVIDIA Quadro K6000	Quadro required for OpenGL Quad-buffered Stereo
RAM	8 GB DDR3	800MHz, RAM size is more than enough to process and compress

TABLE 6.1 - PC specs for evaluating Holosuite.

As we were not I/O limited from the capture side so much, nor memory constrained, our most important considerations were compute power for GPU and CPU, with other specs being secondary. The table above shows the basic computer configuration which should be easy to reproduce on any commodity PC hardware setup.

Although Holosuite was developed to be cross-platform, we evaluated interaction on Windows 8.1 OS, as ZSpace displays do not operate with other operating systems, leaving us with Windows as our only choice. We do not expect that other operating systems will differ significantly in the experience. We evaluated the rendering algorithm for holovideo displays in Windows and Linux.

6.2 - Quantitative analysis: compression, networking and latency

In the **Development** section, we mentioned that Holosuite contains a loopback mode, used to develop, debug and test the networking and compression components on a local machine.

Using the loopback mode, we are also able to quantify the overall latency of the compression-to-render pipeline without the variable latency of remote networking over an internet connection. Using C++ 11 chronology functions, we were able to measure the CPU time before and after compression and decompression of color and depth frames, along with other functions that contribute to latency time.

We found that point cloud generation was done in 2 ms per frame, while rendering a full frame (populating vertex buffers and swapping OpenGL buffers) took just 13 ms. At 30 fps capture, the process time for capturing and segmenting each depth and color frame was 35 ms.

	Color	Depth	Audio
Compression	< 1 ms	15 ms	1 ms
Decompression	1 ms	5 ms	< 1 ms
Compressed Size	1-4 KB	20-30 KB	254 bytes
Frame Dimensions	640x480 (24-bit RGB)	640x480 (16-bit depth)	2 x 16-bit @ 44.8 KHz
Theoretical Bandwidth	30-120KB/s	600-900 KB / sec	8 KB / sec

TABLE 6.2.1 - Audio, color and depth are compressed and decompressed on separate threads, so the function of their latency put together is determined by the slowest component, which is the depth compression and decompression (also known as the critical path). Note that compressed depth is the largest bandwidth bottleneck as well.

From the critical path, shown in **FIG 6.2.2**, we calculated approximately 68 ms time to capture, process, compress, decompress and render a 3D scene.

As we are using TCP networking, there is also additional latency to consider, as TCP requires acknowledge packets to be received by the sender, which means the round-trip (commonly referred to as “ping” time) of the connection must be added to 68 ms to measure the total latency, before pixels or wafels are seen by the remote viewer.

After the capture-to-render pipeline, there is, of course, one additional measurement of latency which has to do with the display technology. The display lag of the ZSpace display is unknown, but we assume that since the ZSpace was built for real-time interactivity and works well for this purpose, the display lag can be considered negligible for simulation rendering. Likewise for the Mark IV, we assume that the 30 fps scan rate is sufficient to account for the latency.

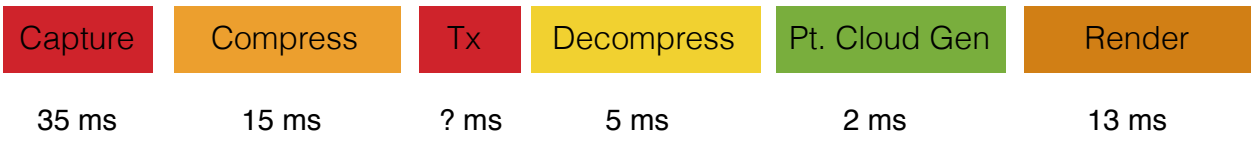


FIG 6.2.2 - The capture-to-render pipeline shown above is the critical path to displaying a frame of a compressed 3D scene in an OpenGL 3D environment. The total latency is 68 milliseconds, not including the network latency, which can vary widely across the internet. red signifies the highest lag, followed by orange, yellow and green being the lowest lag.

6.3 - Quantitative analysis: responsiveness and frame rate (simulation)

Since the focus of the Holosuite user experience is mainly on the visual aspect of telepresence, we wanted to be sure that the rendering stage of the pipeline was convincingly realistic and immersive, as per the specifications laid out in the **Design** section of this document. This means that the rendering frame rate should be high and consistent (minimum 20 fps, ideally 30 fps or higher). In addition to the visual experience, the experience of the interaction (manipulating the model) should be smooth and responsive. Since the Leap Motion is capable of sampling at 120 fps, there is yet more benefit from rendering higher than 30 fps, despite that the user's 3D model is only refreshing at 30 fps. Finally, rendering at 60 fps means that motion parallax reprojection will occur on every frame re-draw, adding to the fluidity of the experience, even though the user capture is done at a maximum of 30 fps.

1080p @ 60 fps Simulation Render	**Capture**	**Audio**	**Input**	**3D model**
Sampling	640 x 480 @ 30 fps	Stereo, 44.1 KHz 16-bit	120 fps, 2 hands	20,000 vertices
Compression	~8 Mb/sec	64 Kbit/sec	N/A	N/A

TABLE 6.3.1 - The above variables for input and output processing can be changed as options in Holosuite. We were able to achieve the maximum 60 fps rendering with high-quality video compression, audio compression, and complex 3D models, while maintaining smooth hand tracking responsiveness.

Initially we had implemented the point cloud rendering method, background and virtual 3D model as legacy OpenGL calls by writing a function call for each vertex, color and normal element. In this first iteration, however, we noticed that the rendering tended to be jerky and inconsistent (well below the 60 fps upper limit dictated by the ZSpace display). As a result, we investigated and found that changing the rendering API to OpenGL 3.0 gave us access to function calls that allow us to designate chunks of CPU memory, specify the data structures for the vertices, normals and colors of the point cloud and 3D model to efficiently copy the point cloud and shared 3D model into GPU memory for drawing. These calls were used extensively from the OpenGL 3.0 spec: *glBufferData* to copy CPU memory to GPU memory, *glVertexPointer*, *glColorPointer*, *glNormalPointer* to define the data structures and starting positions of the vertices, colors and normals, and *glDrawArrays* to issue the drawing command to the GPU pipeline.

With these changes, we were able to produce a consistent, fluid and realistic holovideo simulation experience. The rendering on the ZSpace display was done with 1080p resolution at 60, which felt very responsive to interaction, motion parallax reprojection whilst showing smooth animation of the remote user's point cloud stream.

6.4 - Quantitative analysis: responsiveness and frame rate (holovideo)

Using a diffraction-specific approach to generate the holographic image gives us some unique control over how the Mark IV holovideo display renders content. In the Mark IV architecture, we are locked to 468 hololines (scan lines) in the vertical dimension, but we have some freedom in defining how a hololine appears to the viewer. Namely, we can determine the viewing angle in the horizontal dimension for generating parallax, the granularity of the panoramagram (number of horizontal views in the view set used to compute the hologram) and the number of wafels per hololine, which gives the effective horizontal spatial resolution of the image.

Holovideo Rendering, K6000 GPU	Number of Views	Number of Wafels per Hololine	Number of Hololines	Viewing Angle
22 fps	16	600	468	30º Horizontal
12 fps	36	600	468	30º Horizontal
7 fps	64	600	468	30º Horizontal

TABLE 6.4.1 - Our implementation of diffraction-specific panoramagram computation for mark iv can perform real-time computation at 16 views, which is sufficient granularity for smooth motion parallax.

However, it is important to note that the architecture of the Mark IV is somewhat limiting in that there is a tradeoff between the viewing angle and horizontal resolution, meaning that the more wafels we generate per hololine, the narrower the viewing angle we can project and fewer wafels per hololine can project a wider viewing angle.

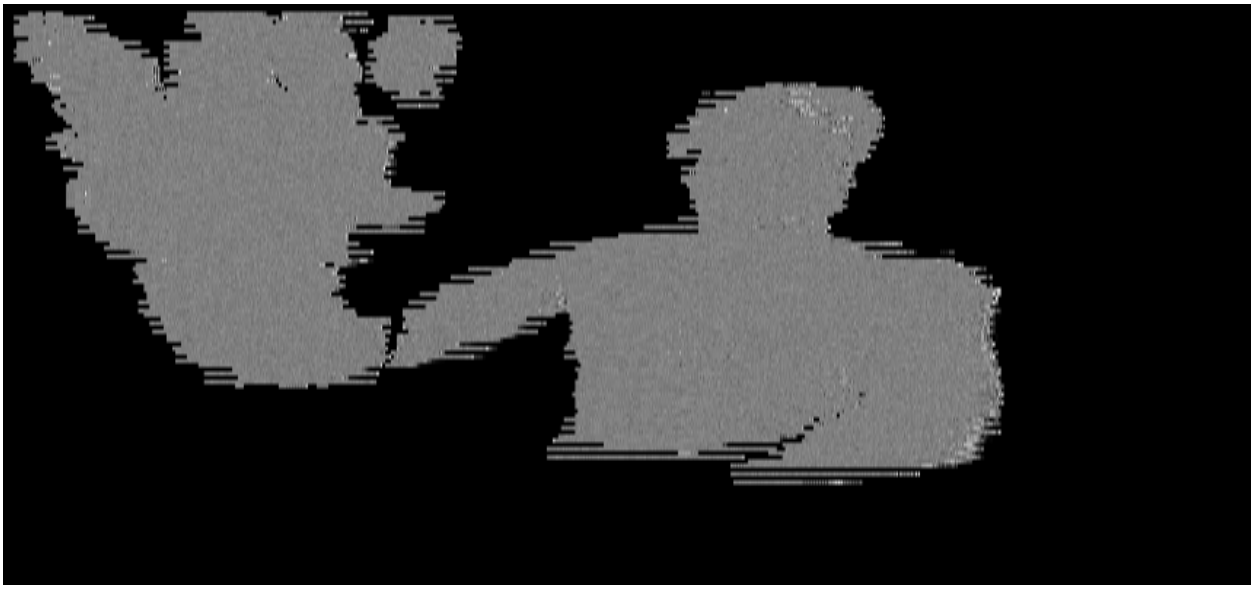


FIG 6.4.2 - The unpacked data from the diffraction-specific signal generated of a user's 3D point cloud, this is a verification of the frequencies from depth and color components which contribute to the holographic image displayed on a Mark IV display.

Although the Mark IV allows us some flexibility over the horizontal spatial resolution, we chose to target a spatial resolution close to VGA (640x480) which would encompass the resolution of the captured 3D data, while giving a reasonable viewing angle of 30º in the horizontal dimension, enough for one user to experience motion parallax.

At 600 wafels per hololine with 16 views, we were able to achieve 22 fps rendering, close to the maximum rendering capability of the Mark IV holovideo display (30 fps). This includes both stages of the holographic rendering process: creating the parallax view set for 16 views, and computing the hologram for each frame using a CUDA kernel.

6.5 - Qualitative analysis: user experience study and methodology

In order to better understand the user experience of Holosuite, we performed a live user study (N = 17) of Holosuite running on the simulation renderer, using ZSpace displays. The user study was performed with the investigator at one end of the session, connected to a Holosuite session being used by the subject, located in another location (different building) in the MIT campus. See **Appendix E** for a figure detailing the configuration used for the user study. The investigator and participant were connected on different Wi-Fi access points, with a TCP connection formed via internet provided by MIT. Participants were recruited via e-mail forum posts on MIT campus, word-of-mouth and through the social networking site, Tinder.

FIG 6.5.1 - These two pictures show the scenario of the user study. An investigator and a participant (2 users) using Holosuite holovideo simulation mode with Zspace displays, in separate locations over the internet.

In the user study, we asked participants to fill out a questionnaire before performing the user study, and again after performing the study (see **Appendix F** and **Appendix G** for the questionnaires). The nature of the study was to first gather some information about the participant's familiarity with 3D display technology and video conferencing. This was followed by a tutorial session that explained to the participant how to use the pinch-to-grasp action to manipulate the object and motion parallax to see around the 3D scenery. The user was then given a task to complete, while using Holosuite with the investigator as the remote user, in real time. In a post-study questionnaire, we asked the participants to rate how realistic and how immersive their experience was, and asked participants to give some qualitative feedback on the interaction and impression of using Holosuite.

The task we assigned to the participant was one that would require manipulation of the 3D object and usage of motion parallax to complete properly. To create such a task, we formed a 3D model of a building with openings on the sides of the building, allowing one to see through the building from one side, all the way to the opposite side. Inside building model, we placed a colorized Stanford Bunny 3D model. After giving the participant a brief introduction to the features of Holosuite, the investigator instructed the user to see if he or she could find an object inside the building and identify it. If the user had trouble finding the bunny, the investigator would offer a suggestion to look for any openings in the building around the sides.

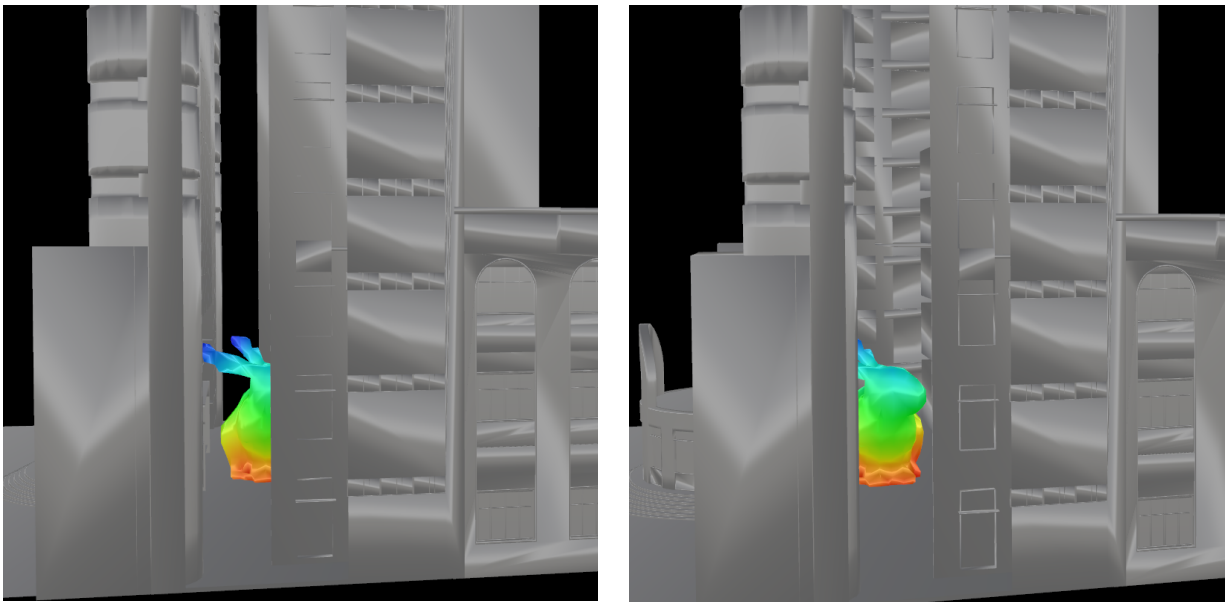


FIG 6.5.2 - Rendering of the virtual building model placed between the investigator and test subject in the user study. Notice the openings of the building, there is a small Stanford Bunny model within the building, occluded by side of the building. The two pictures show the bunny model from different angles of the same side of the building. The user must rotate or look around the corners of the openings to fully see the bunny.

The initial orientation of the building was such that the participant was facing the front of the building, and there was no way to find the bunny inside the building without doing some manipulation and/or careful peering into the side of the building. The expectation was that users would rely on motion parallax naturally to see the model of the bunny, which was partially occluded by the side of the building, to highlight the visual aesthetic of holographic rendering. Once the participant was able to find and identify the bunny, he or she was then asked whether he or she could see the front, or the back of the bunny. After identifying the bunny, the participant was asked to show the investigator where the bunny is.

The task of finding and identifying the bunny was performed twice. Once (Task A) with the user seeing a some visual feedback of their own point cloud (the point cloud and model from the perspective of the investigator), and again without any visual feedback of the self (Task B). In Task A, the user was asked to point to where the bunny was, while looking at their own reflection. The purpose of asking the users to look at their own reflection was to better understand whether this would be distracting to them. This was executed as a randomized

control trial, where half the users performed Task A first, and the other half started with Task B first.

6.6 - Qualitative analysis: user experience study results

The question of highest interest to us was whether participants considered using Holosuite to be a realistic and immersive experience. Our results showed that participants, indeed, overwhelmingly reported being quite immersed in the experience and found it to be realistic.

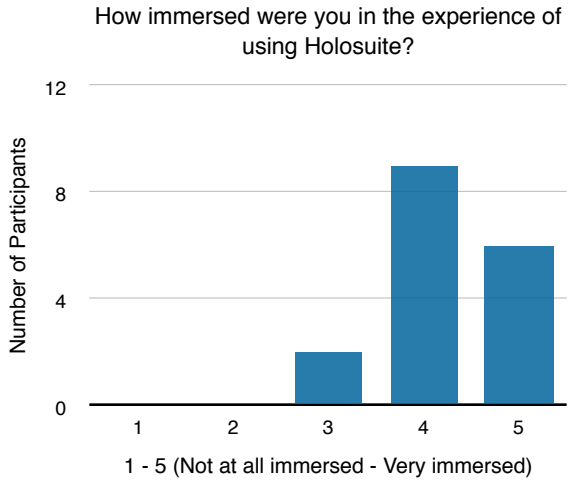


FIG 6.6.1 - Participant response on immersion, after performing the study.

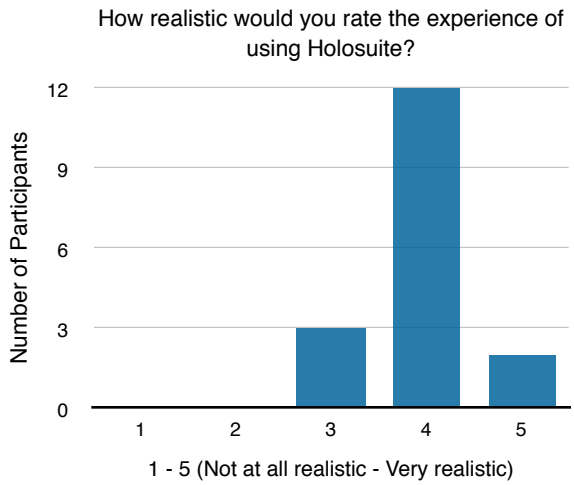


FIG 6.6.2 - Participant response on realism, after performing the study.

When participants were asked to rate how realistic the experience of Holosuite was, 14 out of 17 participants rated the experience as being realistic or highly realistic (~82% rated the experience as 4 or 5 out of 5), with none responding with a rating less than 3.

Similarly, on the question of how immersed participants found themselves with the experience of using Holosuite, 15 out of 17 participants rated the experience as being immersive or highly immersive (~88% giving at least a 4 out of 5), with none responding with a rating less than 3. Interestingly, the distribution of responses to this question was a bit different than the question of realism, showing that 6 people felt “highly immersed”, as opposed to just 2 people who felt it was “very realistic”. This would suggest that the Holosuite experience could be considered somewhat realistic, and highly immersive.

All the participants were able to find the bunny object within the building and identify it. Users were also asked how they were able to find the object inside the building, with 11 people (~65%) reporting that they used a combination of motion parallax and hand manipulation to rotate the building, while the remaining 5 people (~35%) reporting that they only rotated the building with their hands to find the object.

~52% of users (9 users) reported being distracted by seeing their 3D reflection on the screen, and ~35% (6 people) reported that seeing themselves diminished the immersion or realism of the experience. Although it is important to note that this figure is similar to the number of participants that found traditional videoconferencing visual feedback to be distracting (10 users, or ~58%).

From this data, we can assume that having visual feedback in the visual layout is, at best, a divisive feature for Holosuite, and can be detrimental to the experience for some.

When asked to write about their impressions on the experience and how it compares to traditional videoconferencing, user response was overall positive and encouraging about their experience. One user wrote:

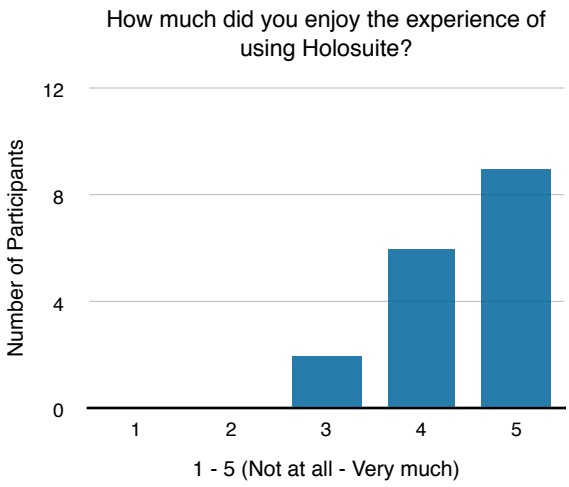


FIG 6.6.3 - Participant response on enjoyment, after performing the study.

> “Even the simple functionality exhibited in the test cases were engaging and immersive. Being able to engage in a live conversation while manipulating an object felt very intuitive. The test case -- pointing out an object in a 3-d model to a colleague -- is impossible to imagine happening fluidly through screencasting or videoconferencing, especially in a collaborative "here, have the ball" kind of way that this system encourages.”

Another user wrote:

> “Definitely much richer than normal video conferencing, and the ability to share a common object back and forth made it seem more like we're in the same space. Probably also the fact that we weren't seeing our backgrounds also created the impression that we were in some "other" in-between space, sort of like being on the telephone.”

When asked if the participants enjoyed the experience of using Holosuite, ~88% (15 people) responded with a 4 or 5 out of 5, with the other 2 people giving a rating of 3 (somewhat enjoyed the experience). It is safe to conclude that the experience was quite enjoyable for a vast majority of the participants.

As for critical feedback, many participants reportedly had trouble using their hands to manipulate the virtual 3D object. They reported difficulty in Holosuite being able to detect the grasp gestures, and intermittent quality of tracking issues. Although the Leap motion algorithm is fairly advanced in its features, we observed that it has severe reliability issues, working well for some users’ hands, while even just detecting the presence of a hand can be difficult for other users. We suspect that the discrepancy in hand size between subsequent users and the users’ lack of experience with the Leap motion could be the major contributors to the difficulty.

Other participants were concerned about the visual aesthetic, as the noisy depth sensor data was reported to take away a little from the feeling of immersion. This can be corrected in the future by implementing more advanced segmentation algorithms, that clean the edges of the users' point cloud and fill missing holes.

Furthermore, some participants also noted that seeing their own image reduced immersion because it broke the metaphor of merging two 3D worlds together seamlessly. One participant wrote:

> "It took my attention away from the 3D object more toward myself.
>
> For normal video collaboration (e.g. Skype), I do prefer to see video of myself because there's no sense of shared space in that interaction, so I need to make sure that I convey what I intend to the collaborator. For instance, I wouldn't want half my face to not be showing because my computer is pointed the wrong way. In this case, I preconsciously bought in to the virtual "physicality" so I didn't need the feedback, much like a constant feedback of how I look in "real life" would be super distracting. This shared space borrows enough from the real life feeling of space that my mental conception of how to convey intention in that space is retained."

Another participant wrote:

> "As opposed to a normal video conferencing system, I felt like this was a much more transparent interaction with the other user, so I sort of trusted the system more and assumed that the other user was just seeing my side of the screen. I didn't feel like I relied on the self-image as much as I do during a skype video call."

The suggestion that having visual feedback of the local user would hinder the immersive experience for some, combined with the unenthusiastic feedback about this feature suggests that it can be removed with minimal negative impact to the overall experience.

7. Conclusion

In this thesis document we have detailed the creation of an end-to-end 3D telepresence software program, written in C++11, that provides an interactive, realistic and immersive experience for two users to communicate, and collaborate over the internet. Our application can compress and transmit 3D captured data of users at 30 fps over typical home broadband connections (9-17 Mbps full-duplex), allowing users to manipulate and share virtual 3D models in a life-like fashion with low latency (68 ms capture-to-render pipeline latency).

Our application can decompress live 3D streams and render in real-time on advanced 3D displays (ZSpace) using OpenGL, providing smooth motion parallax at 120 fps (60 fps stereo), or at 22 fps on a full-color MIT/BYU Mark IV holographic video display using a diffraction-specific holographic image computation algorithm at 600x468 spatial resolution using CUDA.

Qualitative analysis via a user study showed that participants rated the experience of our application as being both very realistic and highly immersive (82% and 88%, respectively) and very enjoyable (88%).

7.1 Future Work

The development of Holosuite opens up avenues of further research for applications of holovideo display technology. There may yet be other collaborative or productive usages for motion parallax that have not been highlighted here. One of the differentiators of holovideo display technology is also the ability to provide accommodation, the usages for which were not explored at all in our application, but which could be a topic for future research.

As 3D capture techniques continue to improve, our application architecture is written to accommodate future sensors, which can easily be integrated into the software for capture and point cloud generation. Multiple camera layouts and other novel methods of 3D capture could improve the visual aesthetic and increase the feeling of realism when using Holosuite.

Holosuite can benefit greatly from future depth compression techniques, as today the literature on depth and point cloud compression is scarce—perhaps future video compression standards may include profiles for lossy depth compression, reducing the bandwidth for communication.

7.2 Closing Remarks

Our demonstration has shown, for the first time known to us, the implementation of real-time two-way interactive holographic telepresence. We hope that this body of work serves to educate and inspire the public in the way that the idea of holographic telepresence has done in the zeitgeist for the past few decades. Our approach shows a usage which is possible on commodity hardware, made for researching consumer holovideo displays. When such displays become available to the general public, we hope this thesis serves as a guide on how to implement usages, telepresence specifically, to take full advantage of the display technology.

References

[1] Caulfield, H. J. and C. S. Vikram (2008). New Directions in Holography and Speckle, American Scientific Publishers. pp. 1-14.

[2] Benton, S. A., V. Michael Bove, Jr. (2008). Holographic Imaging, John Wiley & Sons.

[3] St-Hilaire, P., S. A. Benton, M. E. Lucente, M. L. Jepsen, J. Kollin, H. Yoshikawa and J. S. Underkoffler (1990). Electronic display system for computational holography. OE/LASE'90, 14-19 Jan., Los Angeles, CA, International Society for Optics and Photonics.

[4] Smithwick, Q. Y., J. Barabas, D. E. Smalley and V. M. Bove Jr (2009). Real-time shader rendering of holographic stereograms. SPIE OPTO: Integrated Optoelectronic Devices, International Society for Optics and Photonics.

[5] Jolly, S., D. Smalley, J. Barabas and V. M. Bove (2014). Computational architecture for full-color holographic displays based on anisotropic leaky-mode modulators. SPIE OPTO, International Society for Optics and Photonics.

[6] Smalley, D., Q. Smithwick, V. Bove, J. Barabas and S. Jolly (2013). "Anisotropic leaky-mode modulator for holographic video displays." Nature 498(7454): 313-317.

[7] Ishii, H. and M. Kobayashi (1992). ClearBoard: A seamless medium for shared drawing and conversation with eye contact. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM.

[8] Blanche, P.-A., A. Bablumian, R. Voorakaranam, C. Christenson, W. Lin, T. Gu, D. Flores, P. Wang, W.-Y. Hsieh and M. Kathaperumal (2010). "Holographic three-dimensional telepresence using large-area photorefractive polymer." Nature 468(7320): 80-83.

[9] Barabas, J., S. Jolly, D. E. Smalley and V. M. Bove Jr (2011). Diffraction specific coherent panoramagrams of real scenes. SPIE OPTO, International Society for Optics and Photonics.

[10] Towles, H., W.-C. Chen, R. Yang, S.-U. Kum, H. F. N. Kelshikar, J. Mulligan, K. Daniilidis, H. Fuchs, C. C. Hill and N. K. J. Mulligan (2002). 3d tele-collaboration over internet2. In: International Workshop on Immersive Telepresence, Juan Les Pins, Citeseer.

[11] Maimone, A., J. Bidwell, K. Peng and H. Fuchs (2012). "Enhanced personal autostereoscopic telepresence system using commodity depth cameras." Computers & Graphics 36(7): 791-807.

[12] Follmer, S., et al. (2013). inFORM: dynamic physical affordances and constraints through shape and object actuation. UIST.

[13] Okoshi, Takanori (1976). Three-dimensional Imaging Techniques, Academic Press.

[14] Wilson, G., Carter, T., Subramanian, S., & Brewster, S. A. (2014, April). Perception of ultrasonic haptic feedback on the hand: Localisation and apparent motion. In Proceedings of the 32nd annual ACM conference on Human factors in computing systems (pp. 1133-1142). ACM.

[15] Plesniak, W. and R. Pappu (1999). Spatial interaction with haptic holograms. Multimedia Computing and Systems, 1999. IEEE International Conference on, IEEE.

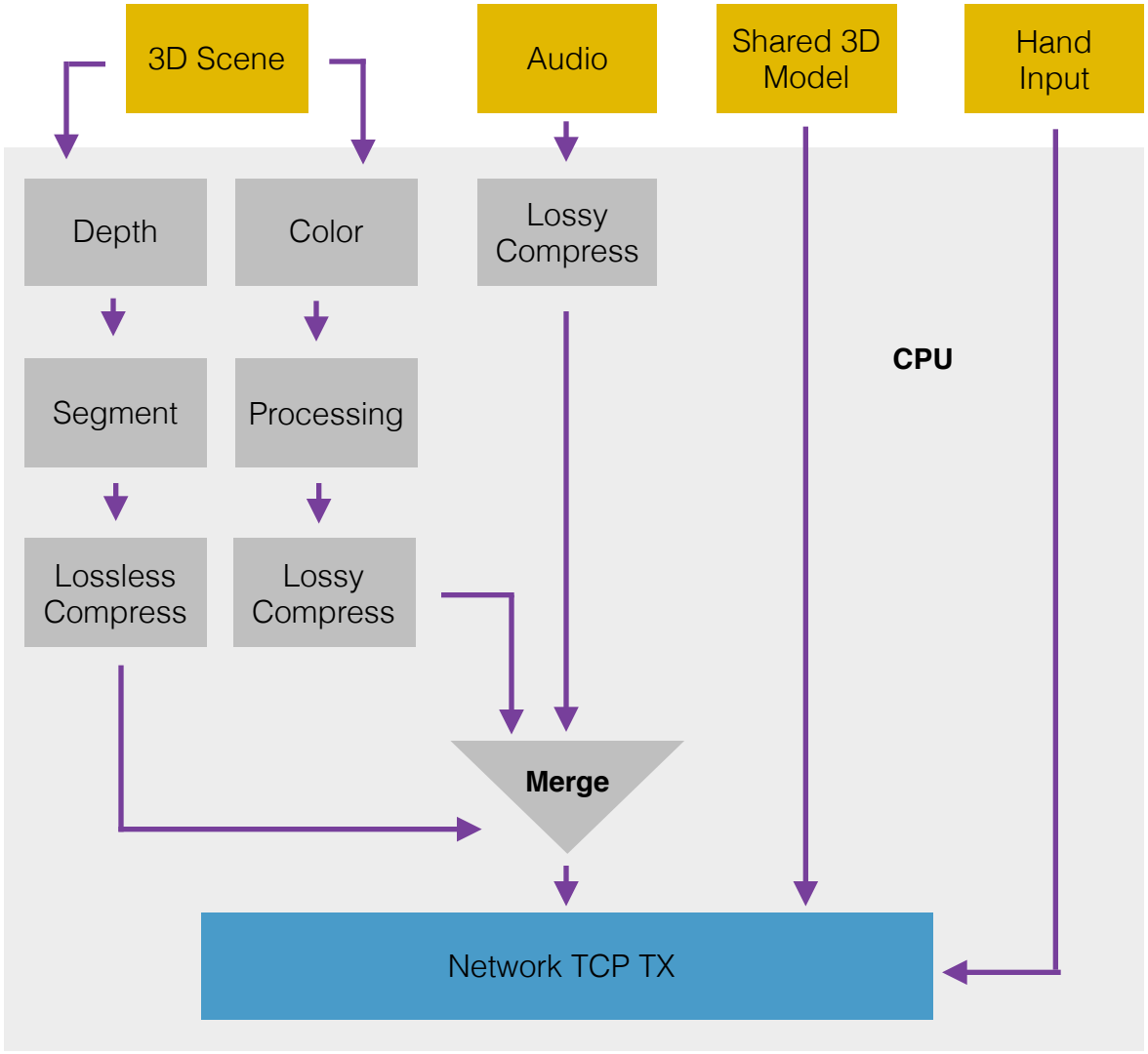
[16] Plesniak, W. J. and M. A. Klug (1997). Tangible holography: adding synthetic touch to 3D display. Electronic Imaging'97, International Society for Optics and Photonics.

[17] Kammerl, J., N. Blodow, R. Bogdan Rusu, S. Gedikli, M. Beetz, E. Steinbach (2012). Real-time compression of point cloud streams. Proc IEEE ICRA, Minnesota, USA.

[18] Gärtner, B. (1999). Fast and Robust Smallest Enclosing Balls. Proc. 7th Annual European Symposium on Algorithms (ESA), Lecture Notes in Computer Science 1643, Springer-Verlag, pp.325-338.

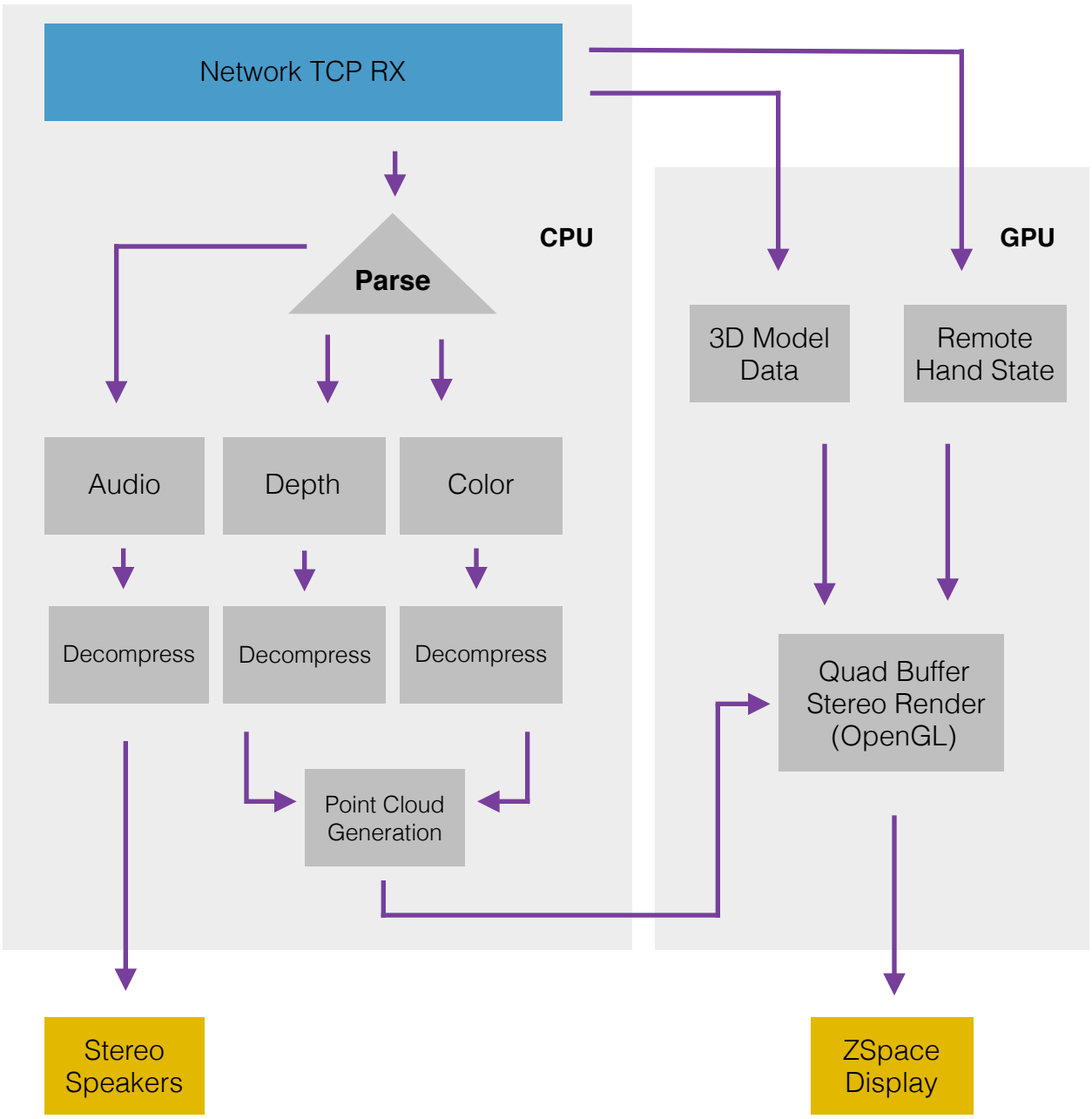
[19] Jolly, S., E. Dreshaj, V. M. Bove Jr. (2015). Computation of Fresnel holograms and diffraction-specific coherent panoramagrams for full-color holographic displays based on anisotropic leaky-mode modulators. Proc. SPIE Practical Holography XXIX, 9386.

Appendix A - Local Session Architecture



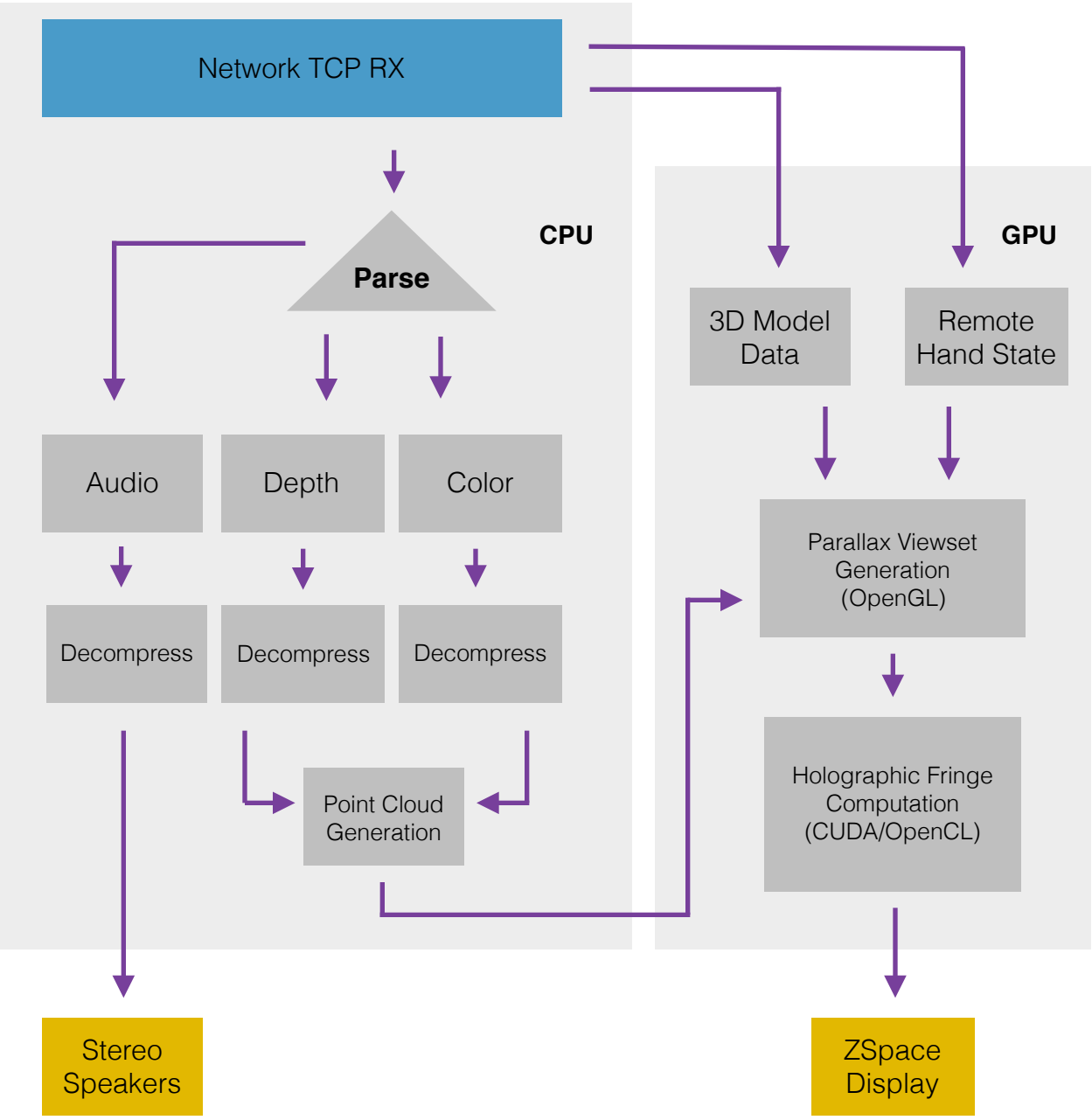
Local Holosuite session architectural model - Each gray block in the model is a thread of its own on the CPU. Using condition variables, the threads wait on data for processing/compressing. The merge block is responsible for combining the compressed streams into a packet. The Network TCP TX block schedules the packets from the 3D scene, audio, local user's hand input state and shared 3d model data (vertices and color information) as FIFO.

Appendix B - Remote Session Architecture and Rendering (Simulation)



Remote Holosuite session architectural model - Audio and 3D streams from the remote connection are parsed, and decompressed on the CPU. The point cloud is generated from color and depth stream, reprojected to perspective coordinates and the remote user's image is rendered along with the 3D model and the state of the model, which is captured from the remote user's hand.

Appendix C - Remote Session Architecture and Rendering (Holovideo)

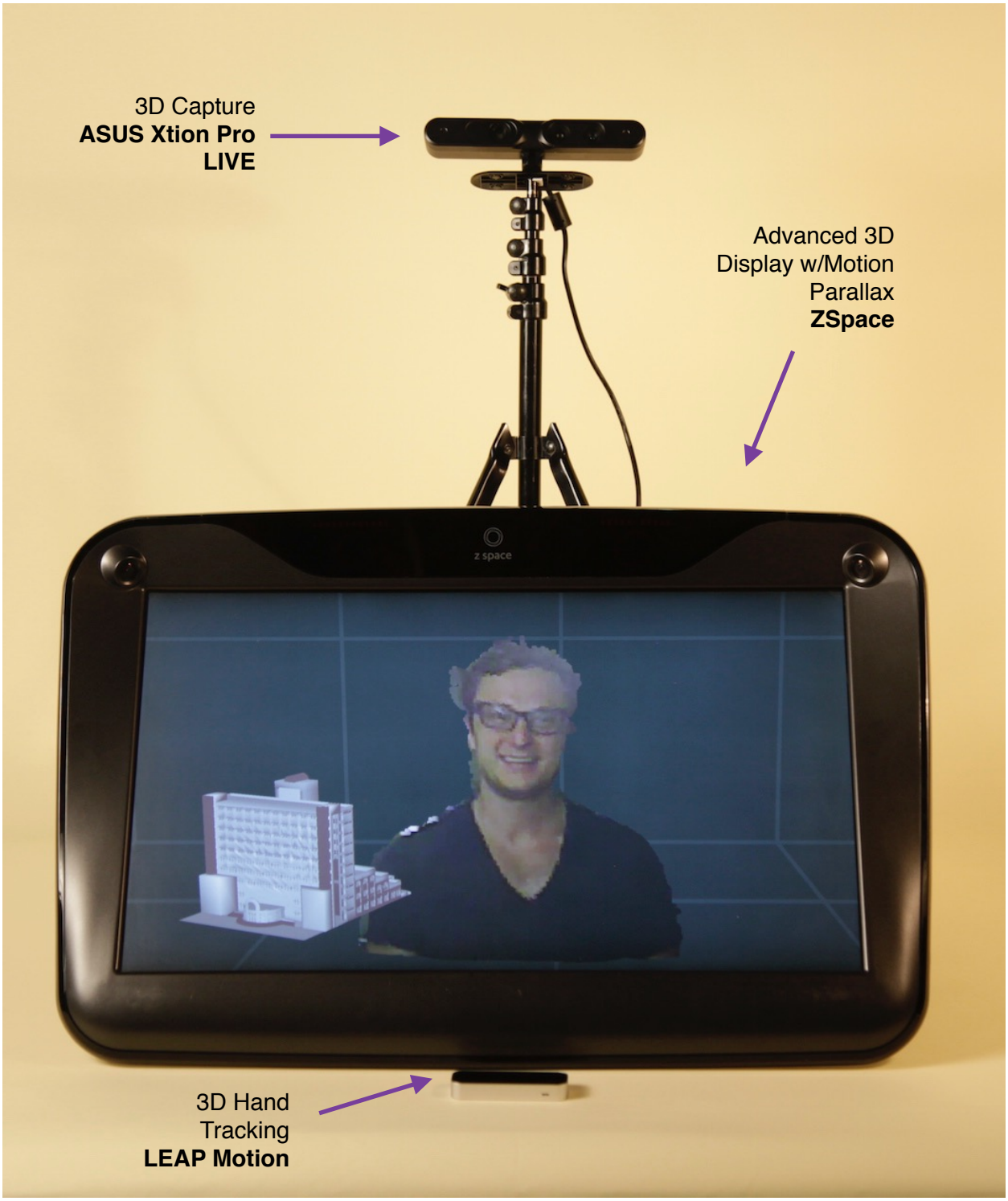


Remote Holosuite session architectural model - Similar to Appendix B diagram, however instead of quad-buffered stereo OpenGL rendering, we generate a parallax views of the rendered scene (point cloud of the remote user and 3D model) and compute the holographic fringe pattern in CUDA or OpenCL for output to a holographic video display.

Appendix D - Software Libraries

	Version	Purpose
assimp	**3.1.1**	3D Model importing
Boost	**1.55.0**	Standard TCP networking, serialization, etc.
FFMPEG	**20140626**	H.264 compression
FreeGLUT	**1.2.8**	OpenGL window initialization
GLM	**0.9.6.1**	Computer graphics math for computing matrices
LeapSDK	**2.2.4.26750**	Hand tracking
Log4Cxx	**0.10.0**	Logging
OpenCV	**2.4.8**	Image processing and compression
OpenGL	**3.1**	3D rendering
OpenNI	**2.2.0.33**	Depth camera input
Opus	**1.1**	Audio codec
Point Cloud Library	**1.4.7**	Point cloud codec, point cloud generation
Portaudio	**r1891**	Audio input and output
ZSpace SDK	**3.0.0.318**	ZSpace display rendering

Appendix E - Holovideo Simulation Setup



Appendix F - Pre-User Study Questionnaire

1. **Have you ever used video conferencing software such as Skype or FaceTime?**
   - Yes/No

2. **If so, how often do you use video conferencing to communicate with others?**
   - Daily
   - Weekly
   - Monthly
   - Rarely

3. **For which usages do you partake in video conferencing?**
   - Meetings
   - Collaboration
   - Education (Teaching/Learning)
   - Talking with friends and family
   - Interviews
   - Other:

4. **When video conferencing with others, how important is it for you to be able to see visual feedback of yourself on the display?**
   - Crucial to the experience
   - Very important
   - Somewhat important
   - Not at all important

5. **If so, why is it important to see yourself?**

6. **When video conferencing with others, how distracting is it to see visual feedback of yourself on the display?**
   - Extremely distracting
   - Very distracting
   - Somewhat distracting
   - Not at all distracting

7. **Do you ever use online collaboration tools, such as Google Apps (Google Docs, Slides, Forms, etc.), to work with others simultaneously?**
   Any hardware/software that allows you to communicate with others while doing work together.
   - Often
   - Sometimes
   - Not often/rarely
   - Never

8. **If so, which online collaboration tools do you use?**
   - Google Apps (Google Docs, Slides, Forms, etc.)
   - Microsoft 365
   - Slack
   - Trello

- Other:

9. **Have you ever experienced digital 3D display technology?**
   This includes active and passive digital 3D display solutions, such as RealD (digital 3D films at the cinema) or modern 3D HD televisions
   - Yes/No

10. **If so, which 3D software/technology have you used or consumed in the past?**
    - RealD (3D films at the cinema)
    - Glasses-based Digital 3D TV (Active/Passive stereo glasses consumer TVs)
    - Glasses-free 3D displays (such as Nintendo 3Ds, light-field displays and some TVs)
    - ZSpace/Head-tracked motion parallax displays
    - Holographic (diffraction-based) displays
    - Other:

11. **How much do you enjoy watching entertainment in digital 3D format?**
    - Very much
    - It's enjoyable
    - Somewhat enjoy it
    - Do not enjoy at all
    - I avoid it

12. **Are you prone to discomfort when viewing 3D display technology?**
    This typically includes motion sickness, nausea, headaches, dizziness and related symptoms.
    - Yes/No

13. **Have you ever used advanced 3D display technology that simulates motion parallax?**
    This includes displays that track the location of the user's head to reproject 3D scenery, multi-view stereo or light-field displays
    - Yes/No

14. **Do you ever use 3D CAD tools?**
    This includes software such as Sketch-up, Blender, Rhinoceros, SolidWorks, etc.
    - Often
    - Sometimes
    - Not often/rarely
    - Never

15. **How familiar are you with gesture-based interaction technology?**
    - Advanced
    - Experienced
    - Novice
    - Somewhat familiar
    - Not at all familiar

16. **If so, which technologies have you used in the past?**

17. **This includes any technologies that enable user interface control by hand/body movement and gestures**
    - Kinect (XBOX One/XBOX 360)
    - Kinect (PC)
    - OpenNI/NITE
    - Leap Motion controller
    - Nintendo Wii
    - Face tracking
    - Other:

18. **When using gesture-based interaction technology, how important is it for you to see visual feedback of your hand/body state?**
    - Crucial to the experience
    - Very important
    - Somewhat important
    - Not at all important

19. **Have you ever used gesture-based interaction in conjunction with 3D display technology?**
    - Yes/No

20. **What is your experience and impression of 3D display technology?**
    Please write about your pre-conceptions, the good and bad experiences, and why they were good/bad

Appendix G - Post-User Study Questionnaire

1. **How realistic would you rate the experience of using Holosuite?**
   - Not at all - Highly Realistic (1-5)

2. **How immersed were you in the experience of using Holosuite?**
   - Not at all - Very much (1-5)

3. **Do you feel that Holosuite exhibits a desirable usage for remote collaboration?**
   - Yes/No

4. **Based on your previous answer, explain why you think it is or isn't desirable**

5. **What sorts of usages for yourself can you envision in Holosuite?**

6. **Did you find it distracting to see your own image while collaborating with the other user?**
   - Yes/No

7. **Did you find that seeing yourself diminished the realism or the feeling of being immersed in the telepresence experience?**
   - Yes/No

8. **Explain the things you did or did not like about having 3D visual feedback of yourself**

9. **Were you able to find and see the object inside the building?**
   - Yes/No

10. **What method did you use to find the object in the building?**
    - Mostly by moving my head and looking around and through the structure of the building
    - Mostly by rotating the building with my hand
    - Both by looking around the object and rotating the object with my hand

11. **How much did you enjoy the experience of using Holosuite?**
    - Not at all - Very much (1-5)

12. **How do you compare using Holosuite to your previous experience with using 3D display technology?**

13. **How do you compare using Holosuite with your previous experience in online collaborative environments?**

14. **How do you compare using Holosuite with your previous experience with video conferencing?**

15. **How would you improve the experience of Holosuite?**