HandsOn: A Portable System for Collaboration on Virtual 3D Objects Using Binocular Optical Head-Mounted Display

by

Kevin Winata Wong
S.B. Computer Science and Engineering, M.I.T., 2014

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2015

Signature of Author: ____
Department of Electrical Engineering and Computer Science
May 21, 2015

Certified by: ____
Pattie Maes, Alexander W. Dreyfoos Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by: ____
Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

HandsOn: A Portable System for Collaboration on Virtual 3D Objects Using Binocular Optical Head-Mounted Display

by
Kevin Wong

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2015

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

In this increasingly globalized and technological world, professionals often have to collaborate remotely on 3D content, especially in fields such as urban planning or architectural design. We propose HandOn, a portable system for collaboration on virtual 3D objects. HandsOn leverages the recent advances in optical head-mounted display and finger tracking technology, allowing users to view digital content as if it were merged with the physical reality, while being able to manipulate the objects using simple hand gestures. Supporting both co-located and remote collaboration, this proof-of-concept system is designed to be portable and easy-to-setup, in contrast with many previous systems that require elaborate setup. Our findings suggests that HandsOn greatly increases awareness of remote collaborators and their actions by displaying the remote user's hands while interacting with the virtual objects, and that gestural hand-based interactions felt natural to the users. We also found that the illusion of the physical and the virtual being merged improved the overall experience, even though this aspect could be further improved by using better hardware.

Thesis Supervisor:	Pattie Maes
Title:	Alexander W. Dreyfoos Professor of Media Arts and Sciences

Acknowledgments

I would like to express my deepest gratitude to my thesis supervisor, Prof. Pattie Maes, for her invaluable help and tireless mentorship. She has been thoroughly involved and supported me in the planning, the design, the execution, and the analysis stages of this thesis. She suggested brilliant ideas whenever I needed them the most, while being open to new ideas at the same time. Without all her assistance and guidance, this thesis would not have been possible.

I further want to thank Takako Aikawa, Kanda University of International Studies, and the Sano Foundation for supporting my study in the Master of Engineering program at MIT.

I would also like to thank the people in the Fluid Interfaces Group for their kind support and many thought-provoking discussions regarding my work and other intellectually stimulating topics.

Finally, I would like to thank my parents, my siblings, and my friends for their relentless support and encouragement.

Contents

1 Introduction 9

2 Background 13
2.1 Related Work 13
2.2 A Brief Overview of Optical HMDs 15

3 HandsOn: Designing the User Experience 19
3.1 Design Principles 19
3.2 User Scenario Walkthrough 20

4 HandsOn: System Overview 23
4.1 System Specification 23
4.2 Supported Hand Gestures 25
4.3 Remote User Rendering for Remote Collaboration 27
4.4 Hardware Context 28
4.5 Software Context 31

5 System Implementation 35
5.1 HandsOn Client Application 38
5.2 HandsOn Server Application 39
5.3 HandsOn Display Application 39
5.4 AR Marker-Hands Calibration 41
5.5 Binocular Optical See-Through Calibration 42

6 Evaluation 47
6.1 User Study 47
6.2 Results 48
6.3 Discussions 50

7 Conclusions 53

Appendices
A HandsOn Tracking Configuration 55
B HandsOn Marker Sheet 59

References 61

1

Introduction

Remote collaboration is a challenging topic that has been tackled by researchers for many years. Designers and experts often have to work remotely due to geographical constraints in this increasingly globalized world. While simple audio or video conferencing technology such as Skype or Google Plus Hangout might be sufficient for communicating with families, these tools do not provide professionals with a shared task space where they can collaborate on more complex tasks, e.g. an architectural design meeting in which these experts must be able to visualize digital content at various level of details and modify the content.

Furthermore, many of these tasks require the designers to work with virtual content such as 3D models. This is where augmented reality (AR) comes in. AR interfaces shows great promise for this particular type of interaction since they allow users to interact with the virtual world in ways never previously thought to be possible, by merging the virtual and the physical space. The most popular AR applications currently run on mobile devices that act as a window to the new mixed reality. They do this by augmenting the camera live image with virtual content. Not only does AR allow seamless interaction between the real and virtual environment, it allows delivery of spatial and gestural cues of remote collaborators in ways never thought possible, making it a suitable system for remote collaboration.

Recently, there have been many developments in the field of display technologies, especially head-mounted displays (HMDs). Various display and sensor technology have been made available both to the academic community and the general public, allowing researchers to focus on the interaction aspect of designing an AR-based user interface instead of reinventing the tracking capability or display technology.

In this thesis, we present HandsOn, a proof-of-concept system for remote collaboration on virtual 3D content. Being an AR-based system, HandsOn seeks to bridge the gap between the virtual and the physical by placing virtual objects in the real world that the user can perceive in space by wearing a binocular optical head-mounted display. Moreover, users can simultaneously view and interact with the virtual objects by using their hands. HandsOn also supports remote collaboration, where remote user's hands are also presented to local users to increase gesture and action awareness among collaborators.

This thesis is structured as follows: Chapter 2 will discuss the background work related to HandsOn. The current state of optical head-mounted display technology, which HandsOn uses, is also discussed. Chapter 3 will discuss the design principles on top of which HansdOn is built. These are the guiding principles that manifest themselves in all the design decisions that the author made when implementing the system. Chapter 4 moves on with a high-level overview of the system. This chapter includes the specification of HandsOn based on the design principles outlined in Chapter 3. This chapter also discusses both the hardware context and the software context of the system, since gaining an understanding of these will guide the reader through the next chapter. Chapter 5 discusses the implementation details of HandsOn. In this chapter, all the main software components are discussed in more detail. System setup and calibration procedures are also outlined in this chapter. Chapter 6 discusses the user study that we conducted to test the system and the results that we obtained. Finally, chapter 7 concludes the thesis by providing a summary of the work and outlines future work.

As we shall see in the next chapter, there are many systems that have been created in order to achieve this goal of augmented-reality based collaboration. However, HandsOn's main contribution lies in its portability, ease of use and setup, and its remote collaboration aspect, as opposed to the systems described in the next chapter, which require elaborate calibration and setup procedure. Furthermore, HandsOn also takes advantage of recent advances in the field of optical HMDs and hand tracking, by utilizing commercially available display and sensor technologies.

2

Background

2.1 Related Work

There are many commercially available videoconferencing systems, e.g. Skype, Google Plus Hangouts, Cisco WebEx Conferencing. These systems support communication at a distance by providing users with a live video and audio feed of remote users, enabling face-to-face communication with them. However, these systems do not support collaboration on specific tasks, let alone collaboration on virtual content, as they do not present the user with a shared task space that is crucial for efficient task-based collaboration.

Augmented Reality (AR) is a promising candidate for both collaborative tasks and manipulating virtual contents. AR allows virtual content to coexist with the physical space, creating the illusion that the two are being merged and possibly allowing seamless interaction between the user and the virtual content. Much research has been done in the field of collaborative AR and many research papers have been published in the field of collaboration on virtual content using various types of output devices. A system such as (T)ether [1] employs a touch display and hand-tracking to support co-located collaboration on and manipulation of virtual objects, leveraging proprioception by using touch as an input. A system called Tangible Windows [2] uses a surface that acts as both a window to display additional information and an input device. BeThere [3] explores the use of 3D gestures for remote collaborations on both physical and virtual objects using a mobile device as an output device while utilizing depth sensors to capture and share a 3D representation of the physical space that is shared with a remote user. Other displays such as volumetric displays [4] and projective displays [5] have been explored as well.

Recently, head-mounted displays (HMDs) have become increasingly popular as an output device for AR, even though the first prototype went as far back as the 1968 [6]. Video see-through devices, such as the Occulus Rift, are especially popular. ShowMe [7] is a project similar to BeThere [3] but using a portable video see-through setup that allows a user to communicate with a peer using video, audio, and hand gestures. Using ShowMe, a novice user can receive help from an expert user who can see the view of the novice user, allowing the expert user to reference objects using hand gestures in real-time.

A different type of HMD called optical HMDs may offer better results than its video see-through counterparts. In [8] the advantages and disadvantages of optical versus video see-through HMDs are discussed. The main issues being discussed include system latency, occlusion, the fidelity of the real-world view, and user acceptance. While video see-through allows simultaneous rendering of the real and virtual content—in which latency is not an issue—optical see-through is capable of presenting the user with unobstructed view of the physical reality, making sure that visual and proprioception information is synchronized. These tradeoffs must be considered when designing a system that uses any of these HMDs.

Furthermore, a comparison between video see-through and optical see-through displays for collaborative tasks [9, 10] shows that optical see-through displays are better at communicating gaze and gesture awareness, most likely due to the improved perception of non-verbal cues.

Considering the advantages of optical see-through HMDs, not surprisingly there are a number of systems that have implemented a collaboration system using optical HMDs. The ARTHUR system [11] is a collaborative workspace that supports complex design and planning decisions for co-located architects and urban planners. It uses an elaborate setup that involves tracked see-through HMD, placeholder and pointer tracking as a tangible AR-based user interface, and hand gesture recognition, supporting application scenarios such as urban and city planning and 3D CAD modeling.

Similarly, the Studierstube system [12] is a multi-user augmented reality system for visualization, presentation, and education using see-through HMDs. The Studierstube system is interesting in that it incorporates layering of data in the system, where each collaborator can have a different type of data with the same spatiotemporal structure displayed individually, e.g. plumbers and electricians collaborating on the same building structure, with the plumber viewing the plumbing system and the electricians viewing the electrical system of the building.

HandsOn is unique in that it leverages recent advances in the field of optical HMDs and hand tracking. A reasonably comfortable, commercially available, optical HMD called the Moverio BT-200 is used. Not only does it merge the physical and the virtual space using optical HMD, it also allows users to interact with the virtual objects using simple hand gestures. HandsOn is designed to be easy to setup by using the small hand tracking devices called the Leap Motion sensor, making HandsOn truly portable. Furthermore, unlike most of the systems described above, HandsOn is designed to work for both co-located and remote collaboration on virtual objects, where the remote user hands will be rendered to increase gestural awareness.

2.2 A Brief Overview of Optical HMDs

An optical Head-Mounted Display (HMD) is a type of wearable display that is capable of projecting images into its user’s eyes while allowing the user to see through it. Optical HMDs, especially binocular ones, are central in the development of HandsOn, or any AR-based collaborative applications that aims to overlay virtual content realistically in the physical world. Therefore, it is necessary to gain a better understanding of the engineering behind optical HMD. This section aims to present a brief overview of the state of the art of optical HMD technology.

Many techniques have been implemented for see-through HMD. They can be grouped into two main families: curved-mirror based optical HMD and waveguide based optical HMD. Curved-mirror based optical HMD uses semi-reflective curved mirrors placed in front of the eye with an off-axis optical projection system, such as Vuzix and Laster. The second alternative, waveguide based optical HMD, is more feasible and popular, since the projective light source does not need to be in front of the eyes. However, since the projector is located on the side of the displays, the light needs to be "guided"—hence the name waveguide—so that it can be reflected into the eyes of the user.

There are several waveguide techniques being employed in currently available optical HMD, including diffractive waveguide technique, holographic waveguide technique, polarized waveguide technique, and reflective waveguide technique, with the most notable ones being the last two techniques.

The polarized waveguide technique, which utilizes embedded polarized reflectors to direct light waves to the pupil, is promising since it does not suffer from small field-of-view (FOV) issues as other waveguide technique does. However, costly manufacturing process and its inability to use plastic as opposed to fragile glass as the transparent material are some of the drawbacks of this technique. This technique is employed by the Lumus prototype.

The reflective waveguide technique, which is employed by the Google Glass and Epson Moverio products, including the Moverio BT-200 that is used in HandsOn, utilizes simple semi-reflective mirrors, which leads to lower manufacturing cost [13]. However, the FOV is limited since it is proportional to the size of the mirrors, which is proportional to the thickness of the glasses. Hence, a large FOV can only be achieved by utilizing thicker glasses.

At the time this thesis is written, there have been many recent developments in the field of displays for AR-based applications, with two most notable ones being the Magic Leap [14] and Microsoft HoloLens [15]. However, currently there has only been self-reported description of people using the system since only a handful of people have seen live demonstrations of the Magic Leap or the Microsoft HoloLens. According to Microsoft, the HoloLens allows users to view virtual content as if they are holograms in the physical world, but it is not clear how it works other than the fact that it uses transparent lens and sensors. A tester who has tried using the HoloLens reported being able to view and manipulate 3D virtual content and perform remote collaboration tasks using it [18]. The Magic Leap claims to achieve similar, if not better, augmentation effect. The same tester reported more vivid, crisper image that blends well with the surrounding world. A closer look at the patents filed by the company reveals that the device works by mimicking 3D patterns of light rays called “light fields” that our eyes receive from real objects, allowing human eyes to focus on the depths of an artificial 3D scene as they would in the real world—hence, a realistic-looking augmented-reality.

3

HandsOn: Designing the User Experience

Before a single line of code is written or any hardware choices are made, it is important to determine exactly what kind of system is being built. Therefore, laying out the design principles and scripting a concrete user scenario walkthrough is critical in the early stages of designing HandsOn. The following two subsections describe the design principles on which HandsOn is built upon and walks through the user scenario that HandsOn is materializing.

3.1 Design Principles

HandsOn is a collaborative system that helps user collaborate using natural gestures on virtual content. Naturally, HandsOn is built on top of the following design principles:

1. *Merging of the Physical and the Virtual*. Augmenting the real world with virtual content will provide the most natural way to visualize and manipulate virtual content. Therefore, HandsOn strives to provide its users with a consistent view of the virtual content as if they are a physical object in the real world with a fixed position and orientation and can be viewed from any angle.

2. *Natural Hand-Based Interaction*. There are many ways to interact with digital content that are overlaid in the physical world, e.g. using voice commands or using less direct control devices. However, in order to achieve a high degree of realism, it is important to be able to directly manipulate the virtual content as a user would manipulate real objects: by using their hands and fingers. Therefore, HandsOn seeks to utilize natural hand-based

interaction as much as possible to allow interaction and manipulation of rendered virtual content.

3. *Collaboration*. HandsOn is envisioned to be a collaborative platform, where multiple users can interact with other users and visualize and manipulate objects together. The collaboration could happen between co-located users or remote users. Therefore, it is important to have a synchronized view of virtual content in the system across all users.

4. *Portability*. HandsOn has to be portable; the system should be easily usable anywhere, and requires minimal and simple procedure to setup. This principle requires that the system is comfortable enough to be worn to perform specific tasks while requiring no or minimal and easily accessible peripheral apparatus.

3.2 User Scenario Walkthrough

The following fictitious scenario describes the ideal representation of a system that achieves the design principles described above.

Joe, who lives in Florida, is a car designer who works for a Boston-based company. Most of the time Joe would design specific components of a car by himself in his house using a 3D modeling software, on which his remote location poses no problem. However, designing a car is a collaborative process and he needs to be able to present his design and completed milestones regularly several times a week effectively from Florida to his colleagues in Boston. Meanwhile, in the Boston office, Joe's colleague Sarah is leading a design meeting that requires complex collaborative efforts from multiple designers and engineers, including Joe himself.

Using HandsOn, Joe and Sarah's team can all view the current 3D models of all the different components during their meeting. All they have to do is run the HandsOn application on their laptops, put on the head-mounted display that has embedded sensor for hand tracking, connect all the devices to the network, and they can suddenly see all the car components in the meeting room as if they are real.

Joe began the meeting by presenting the design of a component that he has been working on. From his home he was able to point at different virtual parts of the component that he designed by using his hands. Meanwhile in Boston, his colleagues can see a representation of Joe's hands rendered using realistic augmented reality, as if they are in the same room as them. Similarly, when Sarah comments on the some parts of the component while pointing at the component, Joe can see what she was doing by looking at the hands labeled as "Sarah."

The meeting proceeds with different team members presenting their components to each other, and Joe can follow everything that is happening since he can see all the components as if they are in his room in Florida.

When adjustments need to be made to some parts, someone would bring the component to the center of the room and start adjusting certain parts of the component as if it were made of clay. This works for both the people in the Boston office and Joe in Florida. Finally, the meeting ended and Joe begins working on the next iteration of the design process, looking forward to his next virtual meeting with his colleagues.

The scenario described above shows the design principles in action. Augmented reality allows users, both Joe and Sarah's team to visualize and manipulate virtual 3D models as if they are real objects. Everyone can point, move, and change the 3D models using their hands; this shows how natural hand-based interaction works in the system. It works for co-located collaboration since everyone in Sarah's team can view and manipulate objects, while the fact that Joe is able to do the same thing shows the remote collaboration aspect of the system. Finally, it is portable since all the users have to do is run the application on their laptop, connect to the network, and put on the head mounted display.

As will become apparent in the upcoming sections, the current implementation of HandsOn has not yet been able to achieve everything that is described in the idealistic user scenario described above. However, it is one step towards the ideal scenario described above.

4

HandsOn: System Overview

As has been previously described in the previous section, HandsOn seeks to enable users to collaborate on 3D virtual content both on co-located or remote scenario using natural hand gestures. To achieve this goal, HandsOn is developed by combining many different types of sensors and system components. This section describes a brief system overview of the entire HandsOn technological ecosystem, and provides an introduction to a detailed system implementation in the following section.

4.1 System Specification

Our current implementation of HandsOn meets the following specifications based on the design principles described in the previous section.

Merging the Physical and the Virtual: Marker-Based Augmented Reality

HandsOn uses marker-based tracking to align virtual content properly on the physical space as viewed by the user. A marker-based tracking is used since it is judged to be the most efficient and reliable way to track the environment.

Fundamentally, in order to merge the physical and the virtual, it is necessary to build a model of the physical workspace, thereby giving the software information of where the display is currently focusing on in the physical workspace. The software can then use this information to properly align the rendering of the virtual content relative to what the user is seeing, therefore creating the illusion that the virtual is merged with the physical.

There are many ways to retrieve this tracking information; one way is by using electromagnetic position and orientation tracking system such as the Polhemus Fastrak used by [12]. Another way is by using computer vision based motion capture system that typically employs reflective marker attached to the tracked objects. These tracking systems, however, requires an elaborate setup which severely limits portability.

Therefore, HandsOn utilizes maker-based tracking for AR applications. The hardware that HandsOn uses has a built-in camera, which makes using this technique feasible. Properly calibrated, it is possible to truly augment the physical reality with virtual contents as if the virtual contents are physical objects.

There are drawbacks to using this computer-vision based tracking: unlike electromagnetic-based tracking system that does not suffer from obstruction of line-of-sight, marker-based tracking is unable to perform continuous tracking properly when the view of the camera is obstructed by the user or objects. However, electromagnetic-based tracking is not an option is contradicts portability.

Natural Hand-Based Interaction: Hand and Finger Tracking

In order to achieve natural hand-based interaction with virtual content, it is necessary for the HandsOn system to track the hand and the fingers of its users. This tracking information can then be used to recognize certain gestures and the positions of the finger joints relative to the sensor. HandsOn uses the commercially available Leap Motion Sensor to track the fingers. The Leap Motion Sensor also comes with a developer SDK that allows easy access to the tracking data, such as joint positions and simple gesture recognition.

Furthermore, it is also necessary to calibrate with respect to the sensor's location and orientation and the marker's position. This calibration step is necessary to properly pinpoint where the hand is located relative to the marker, which in turn is calibrated relative to the physical reality.

Collaboration: Central Server of the Virtual World that Synchronizes All Displays

HandsOn needs to support collaboration between co-located users and remote users at the same time. This means that all users need to be able to see a consistent state of the system—what objects are in the system, where they are located, and their orientation—across all users. This would not be an issue if we were working with physical objects, but it is an important requirement to maintain while working with virtual objects in order to achieve realism.

Portability: Wireless Setup Requiring Local Area Network (LAN) Connectivity and Printable Marker Sheet

In order to support portability, HandsOn should be easy to setup. A printable augmented-reality marker sheet should be easily available to the users. The finger-tracking device should be portable and easy to setup, and calibrating the relative position of the sensor and the marker should be simple and easy to do. In addition, HandsOn should also work seamlessly over simple wireless connection.

4.2 Supported Hand Gestures

Three distinct gestural interactions are supported in HandsOn. See Fig. 1 for pictures. The first interaction is highlighting an object by pointing at the object using the tip of the index finger. When the tip of the index finger is within the interaction range of any object, the highlighted object will be emphasized by using thicker lines. The second interaction is moving an object by pinching using the hand, and releasing the pinch to place the object. Finally, the third interaction is changing the point of view of the entire virtual space. Often users need to view the virtual content in order to understand the 3D structure of the virtual object fully. This can be done by pinching the space by using to hands, and doing a rotating movement as if the virtual space itself is being rotated.

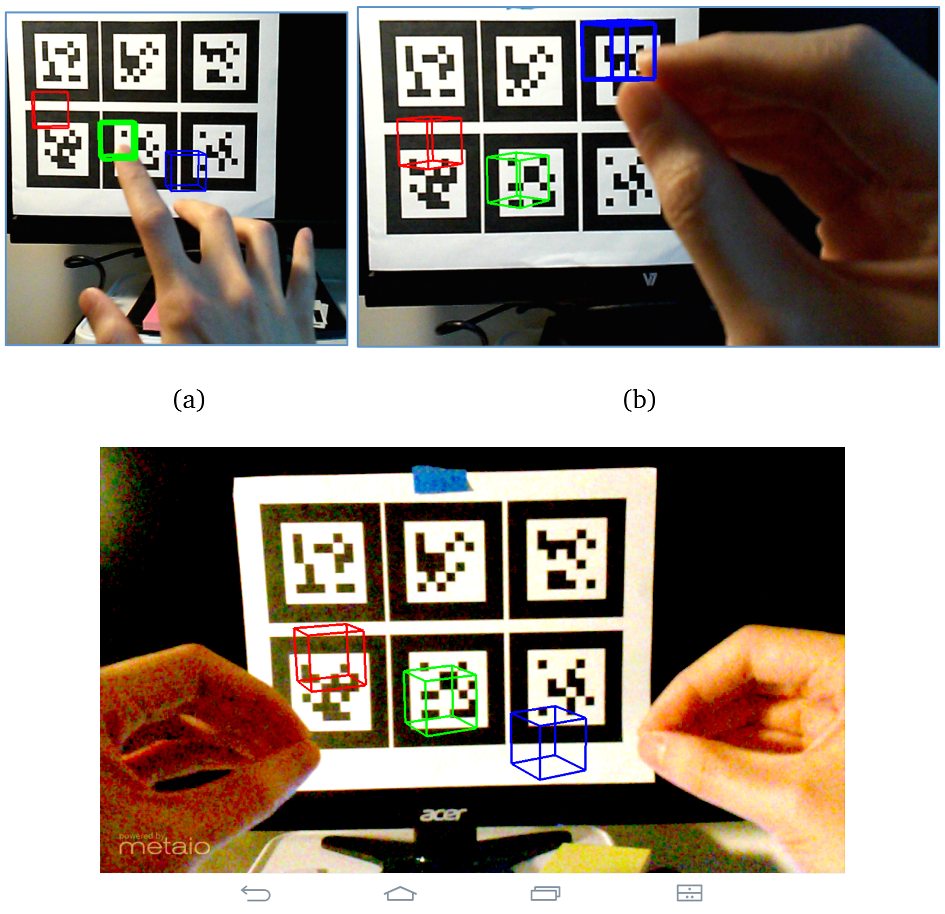


Figure 1. Supported gestural interactions. (a) Highlighting an object by pointing. (b) Moving an object by pinching. (c) Changing the point of view by pinching using both hands. Note that these pictures are captured from a tablet, not from a see-through HMD.

4.3 Remote User Rendering for Remote Collaboration

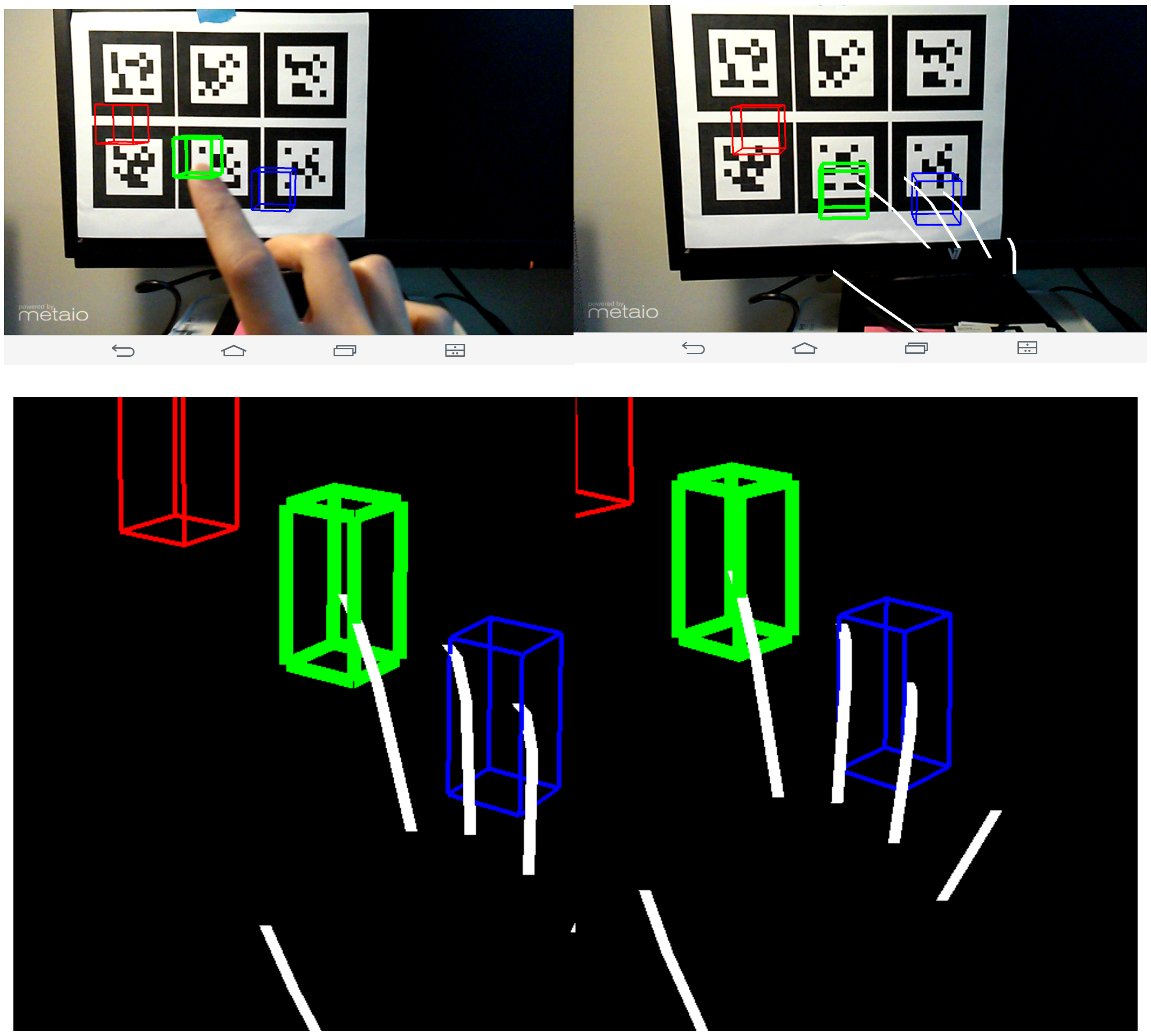


Figure 2. The same action: highlighting an object, as performed by a local hand (top left) and a remote hand (top right) as viewed by a local user. Note that these two pictures are captured from a tablet, not from a see-through HMD. The bottom picture is a screenshot taken from the see-through HMD in the presence of a remote user, as is evident from the hands being rendered. Side-by-side rendering is used, and since the display is semi-transparent, only the virtual content is rendered with no camera feed as the background, as is typical in an AR application running on a tablet.

In the remote setup, remote users have to be represented in some ways to inform the local users of remote actions. HandsOn renders the skeleton of the remote user hands to keep the local users updated on the actions of the remote users. See Fig. 2 for pictures. Note that only the remote users will be rendered this way. Local hands are not rendered since the physical hands are already located in the same space.

4.4 Hardware Context

The HandsOn system is built using many hardware components, each of them are commercially available. Therefore, it is important to understand the overall hardware context of the system before diving into the details. Please see Fig. 4, 5, and 6 for illustrations.

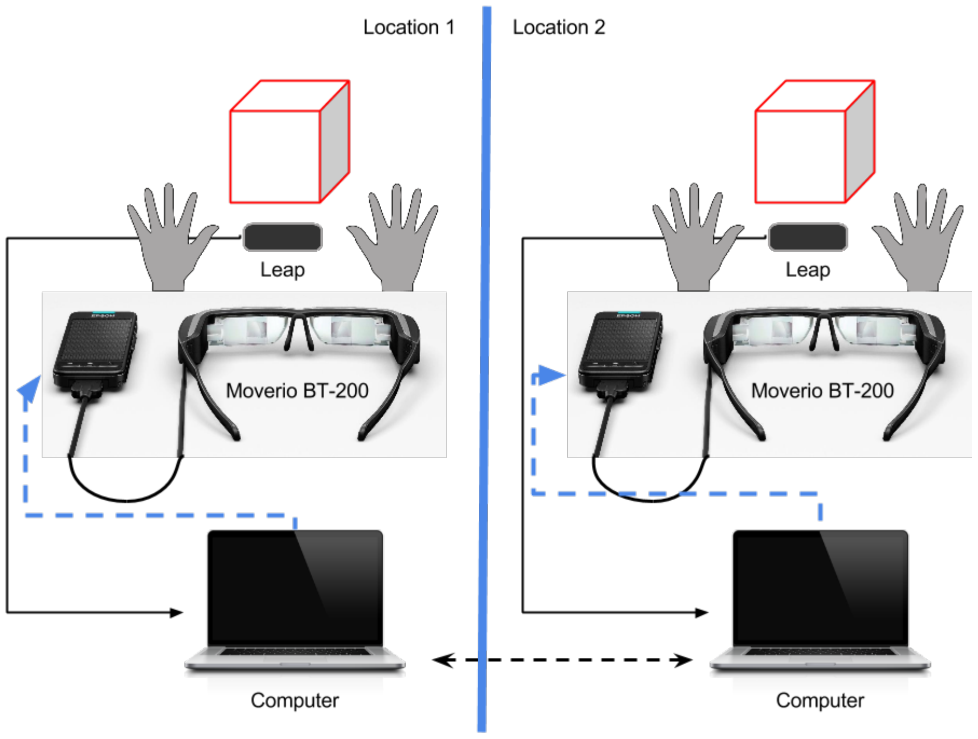


Figure 3. HandsOn system's hardware context diagram for remote collaboration. Each location has one computer running a client, and each client is connected to an output device, which is worn by a user. Dashed lines indicate wireless connectivity.

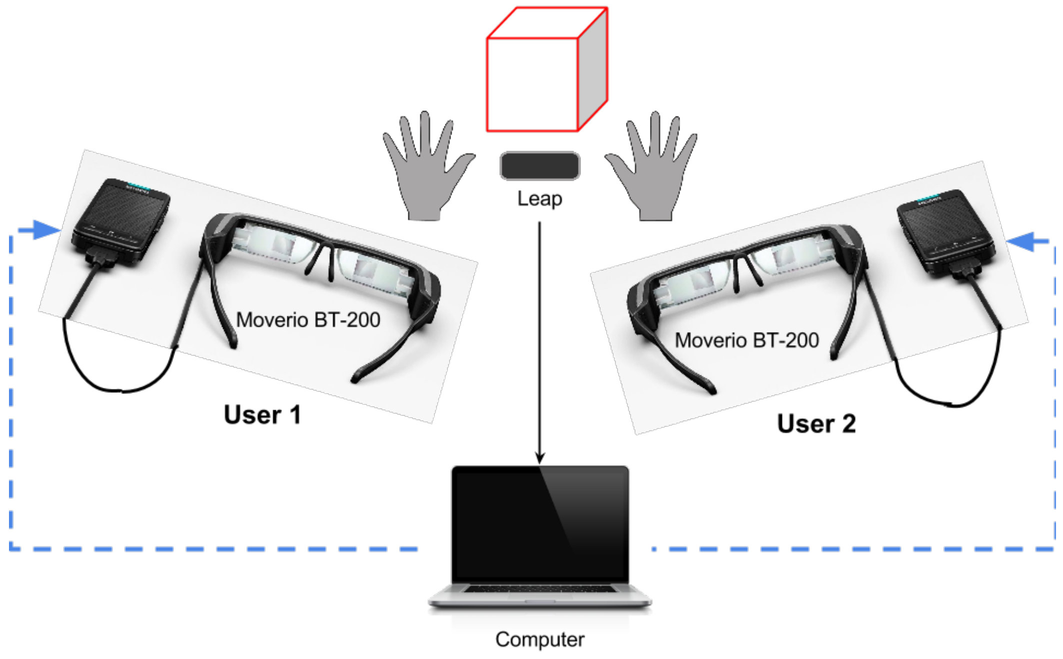


Figure 4. HandsOn system's hardware context diagram for co-located collaboration. Only a single client computer is used, connected to a one Leap Motion sensor. However, the client can be connected to more than one output device, allowing multiple co-located users to view the virtual content by wearing the see-through glasses. Dashed lines indicate wireless connectivity.

It is important to understand that HandsOn works for both co-located collaboration and remote collaboration, with each of them having slightly different hardware context. In a remote collaboration setup, there is a client computer on each location. Each location will have displays that are connected to a client computer equipped with a finger-tracking sensor. In a co-located setup, there will only be one client computer with a finger-tracking sensor but is connected to possible multiple displays.

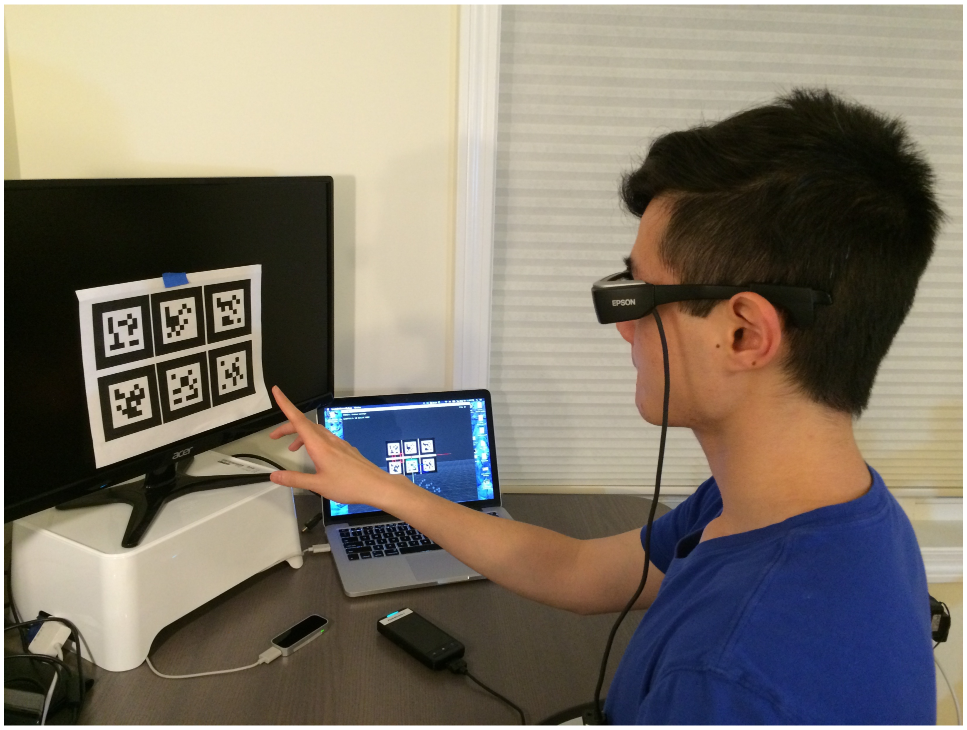


Figure 5. A user is testing the system. Note all the hardware elements that are shown in the picture: Moverio BT-200 worn by the user, Leap Motion sensor connected to the client computer, a computer running the client application, and a marker sheet attached to a vertical surface.

Moverio BT-200 glasses are used for the display. It is a commercially available optical see-through head-mounted display with built-in camera and it is commercially available. HandsOn is able to merge the virtual and the physical through marker-based AR tracked by the camera, and since there are two separate displays for each eye, the Moverio BT-200 can render stereoscopic 3D images, allowing realistic depth perception. Since the display is very personal—it is in the form of glasses—the virtual content is only visible to someone who wears these glasses. Another important thing to note is that the glasses do not operate independently; it is necessary to connect the glasses to a specialized Android device through a proprietary connector. Therefore, all rendering has to be done in the Android device poses important constraints to the system.

Leap Motion Sensor is used for hand and finger tracking. Together with the provided SDK, this small device allows precise enough tracking to allow simple but natural hand-based interaction. It provides finger joint information and simple pinch and grab gesture recognition.

A general-purpose computer is used to connect all the pieces together. The glasses are connected to this computer wirelessly, and the Leap Motion is connected over serial connection to the computer. In a remote setup, the computer also connects to a remote computer.

The current implementation of HandsOn works over wireless LAN connection, where two computers are connected over wireless, one being a client, and the other one being both a central server and a client in a remote setup. Another device that needs to be connected is the display, and at the moment, the displays are connected to the server computer.

Another important piece of the system is not electronic; it is the AR marker sheet. This sheet needs to be printed out and placed on a vertical service in order to use HandsOn. The marker will be tracked by the camera of the Moverio BT-200 glasses to compute the relative position of the glasses and the world.

4.5 Software Context

In addition to understanding the numerous hardware components, it is also important to consider the software ecosystem of HandsOn before jumping into the details of the system architecture. Fig. 6 describes the software context of HandsOn.

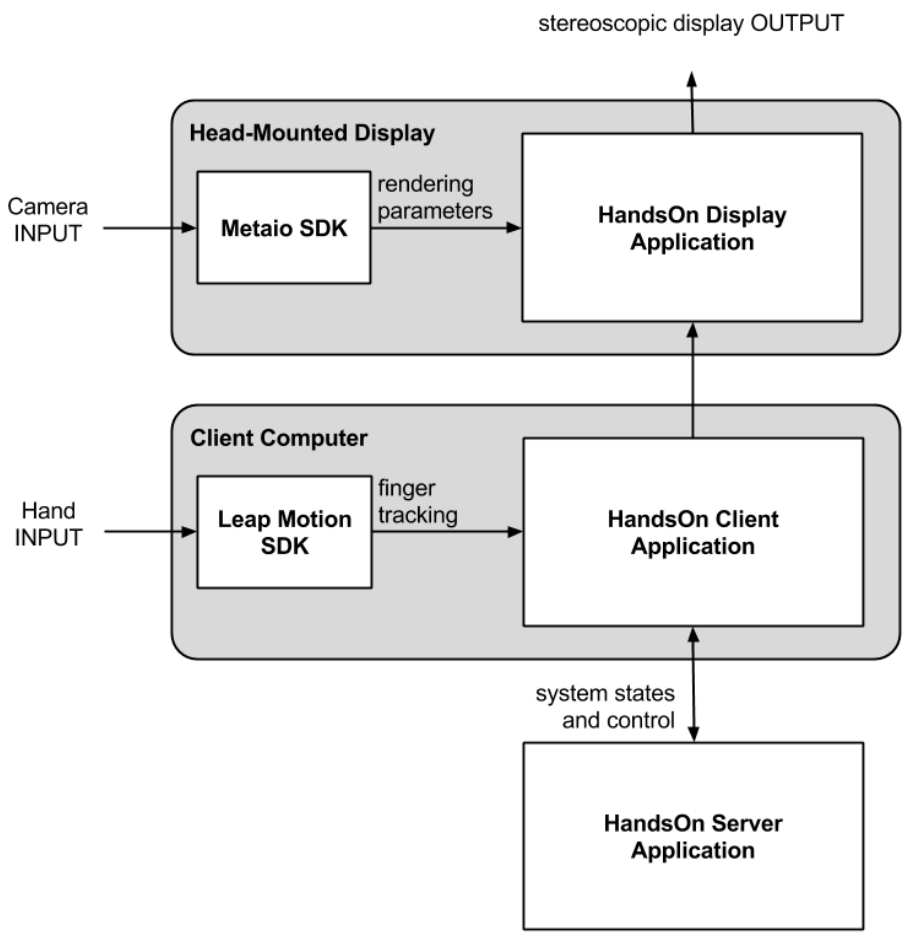

stereoscopic display OUTPUT
Head-Mounted Display
Camera INPUT
Metaio SDK
rendering parameters
HandsOn Display Application
Client Computer
Hand INPUT
Leap Motion SDK
finger tracking
HandsOn Client Application
system states and control
HandsOn Server Application

Figure 6. HandsOn system's software context diagram. Note the flow of the data from the input to the visual output. Also note how the Client Application runs on the client computer while the Display Application runs on the Moverio BT-200.

There are two different hardware devices involved, each running an application that is part of the HandsOn system. First, there is the Android device that is connected to the Moverio BT-200. This Android device runs the HandsOn Display Application that takes rendering matrices computed by the Metaio SDK using the live camera input. Second, there is the client computer that runs the HandsOn Client Application that takes tracking data from the Leap Motion SDK, send that information to the HandsOn Server Application, which may or may not be running on the same device, and synchronizes all displays connected to the client computer.

The **Leap Motion SDK** is an easy-to-use library to retrieve inferred finger-tracking information for the Leap Motion Sensor. It provides real-time joint positions and recognizes simple gestures such as pinching and grabbing, which is very useful for implementing natural hand-based interaction.

The **Metaio SDK** is an Augmented Reality SDK that is supports the Moverio BT-200 glasses, complete with a calibration routine that can be performed in order to perfectly register the marker on each of eye. In HandsOn system's software context, the Metaio SDK runs on the Android device that is connected to the glasses. The Metaio SDK is capable of accessing the camera live image and, with the calibration data and tracking configurations, compute the position and orientation of the glasses relative to the marker. With this information, the Metaio SDK can generate the model, view, and projection matrices that are used to render the virtual content using OpenGL ES 2.0 stereoscopically.

Each of the applications and libraries described above will be described in more details in the following section.

5

System Implementation

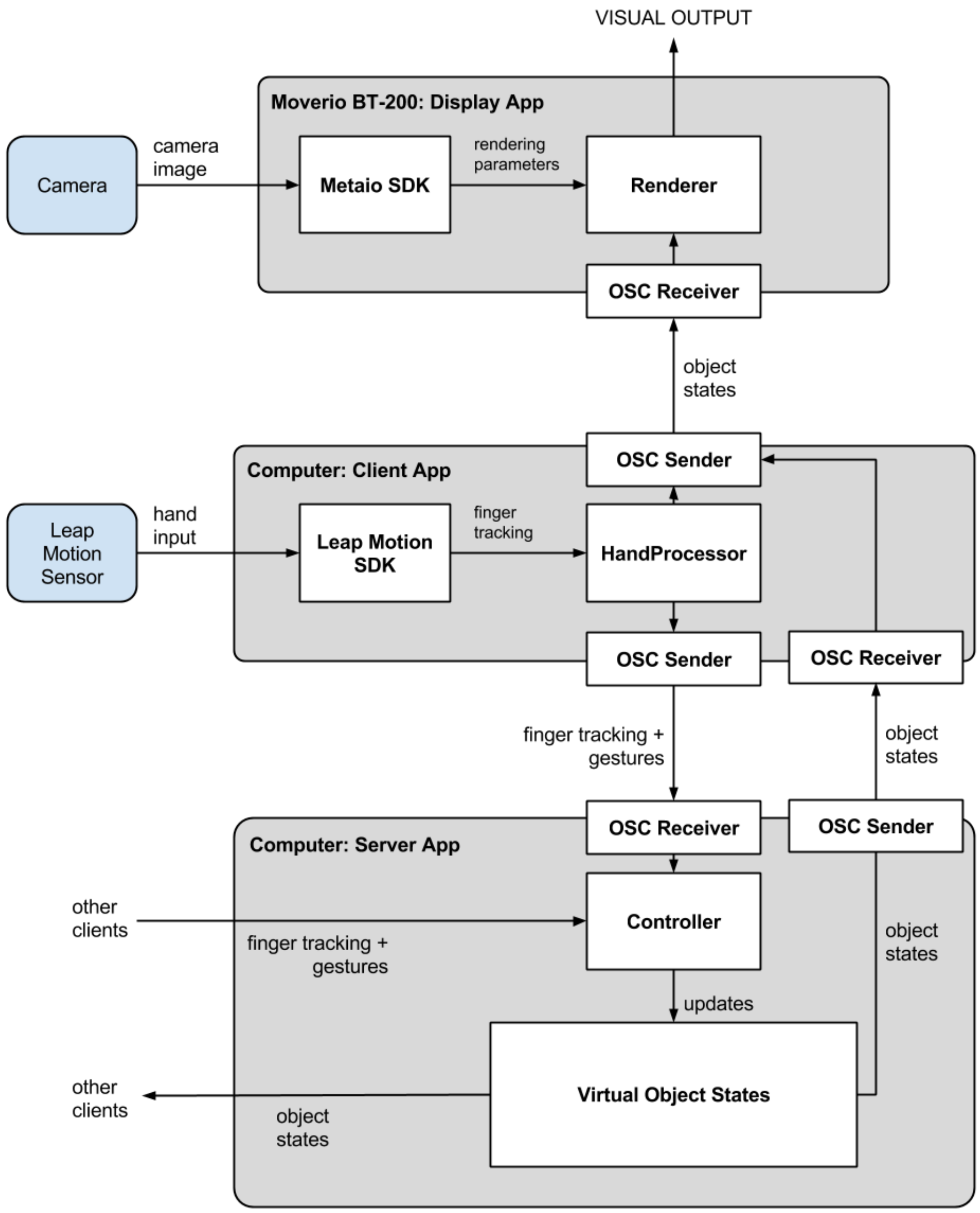


Figure 7. The system architecture of HandsOn. See text for detailed explanation.

Before we begin discussing the implementation details of HandsOn, please see Fig. 7, which outlines the system architecture of HandsOn. Note that the diagram combines the major features of the hardware context diagram and the software context diagram from the previous section. There are three major components in the system: HandsOn Display Application, HandsOn Client Application, and HandsOn Server Application, each serving a clearly defined role. The Display Application runs on the Android device connected to the Moverio BT-200, receives data from the Server Application, and renders the virtual content for the user. The Client Application retrieves hand-tracking information from the Leap Motion sensor, sends it to the server, receives virtual content state from the server, and sends that information to the Display Application to be rendered. Meanwhile, the server receives all input from the Client Applications, updates the system states and virtual content, and synchronizes all the updates with the Client Applications. The following subsections will describe each of these applications in detail.

It is important to note the multiplicity of each component in the system. Fig. 8 describes the multiplicity of the Display, Client, and Server Applications. In any collaboration session, there can only be one Server Application running, since there has to be one entity that keeps track of everything that is happening, and is able to update anyone requesting information about the state of the system. In other words, the server is the central brain of the system. A Server Application takes input from any number of Client Applications and updates each of them accordingly. This multiplicity supports the remote collaboration aspect of HandsOn, since typically, a single workspace is represented by one Client Application that retrieves hand-tracking input and synchronizes it with the server. Furthermore, a Client Application can be connected to any number of Display Applications. This multiplicity supports the co-located collaboration aspect of HandsOn, since multiple users can view the virtual content on the same workspace at the same time. This is due to the fact that a user needs to wear the Moverio BT-200, which runs the Display Application, in order to view the virtual content.

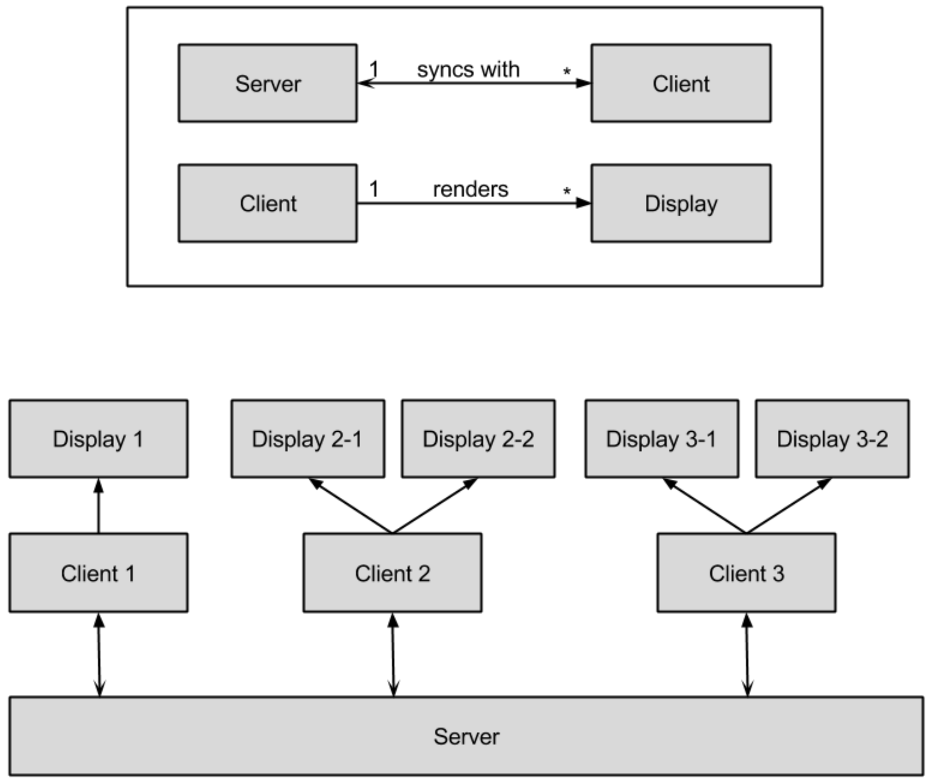


Figure 8. The multiplicity of the HandsOn Client, Server, and Display Applications. A server can synchronizes with multiple clients, while each client can render to multiple displays. A sample instance of the HandsOn system with three client applications running and a total of five displays is depicted.

In the current implementation of HandsOn, both the Server Application and the Client Application runs on the same machine. Furthermore, the Server Application synchronizes with all connected HMDs to simplify the software system. Audio communication is also not implemented, as this can be easily done using third-party video-conferencing system, such as Skype or Google Hangout. Furthermore, this proof-of-concept implementation, unlike the system described above, works only over LAN but produces the exact same behavior, which is sufficient to simulate and evaluate the system. A router is used to connect all the devices wirelessly.

5.1 HandsOn Client Application

The HandsOn Client Application is the main input of the application. Running on a computer connected to a Leap Motion sensor, the Client Application retrieves hand-tracking data from the Leap Motion Sensor through the Leap Motion SDK. The Client Application is implemented on OpenFrameworks, an open source framework that suits building highly visual application and has support for many popular libraries.

The Client Application takes advantage of the Leap Motion SDK to access the hand and finger tracking data from the sensor. These joint positions are then stored and sent to the Server Application to be processed and synchronized with all connected clients. The Leap Motion SDK also recognizes simple pinch and grab gesture, which are then linearly combined in HandsOn to form a robust pinching recognition using empirically determined parameters. This pinching recognition is used by the server to update the state of the virtual world, as pinching an object allows the user to move the object. As the current Leap Motion SDK does not allow data retrieval from more than one Leap Motion Sensor, each computer can only run a single Client Application.

One important role of the client application is to send the processed hand information to the Server Application, where this information is used to update the state of the virtual content, based on the user interaction with the virtual object. The Server Application in turn sends the most current state of the system, which includes object positions and orientations, back to the client, which will then be forwarded to the Display Application so that the virtual world can be stereoscopically rendered for the user.

5.2 HandsOn Server Application

The HandsOn Server Application is the central brain of the system—the ones that controls and maintain the state of the system. The Server Application receives hand input from all connected clients and updates the virtual world accordingly. Implemented using OpenFrameworks, it uses Open Sound Control (OSC) protocol to communicate with the clients.

The Server Application also manages the interaction between users and virtual objects. Given the finger positions and hand poses, the server moves an object whenever a user pinches inside an object. Another possible interaction with the virtual world is rotating the point of view so that the user can view the virtual scene from a different angle, reinforcing the 3D feeling of the virtual objects. The current implementation is limited to moving objects and changing the point of view of the user, but it is possible to extend the implementation to include other interactions, such as rotating an object.

Ideally, in order for the user to see the virtual world merged with the physical world, the Server Application would send the virtual content states to the Client Applications, which in turn forwards it to the Display Applications. However, in the current implementation of HandsOn, the Client Application is skipped since it is assumed that all clients are connected to the same LAN, which yields the same behavior in simulated remote collaboration.

5.3 HandsOn Display Application

The HandsOn Display Application is the main output of the system, through which the users can view the virtual content overlaid on top of the virtual world. Running on the Android device connected to the Moverio BT-200 glasses, this application renders the virtual scene twice, once for each eye, yielding a stereoscopic view of the virtual 3D objects. The Moverio BT-200 has a total resolution of 960x540 per eye, with support for side-by-side 3D mode, meaning the left half of the screen will be stretched to fill the left eye display, and the right half of the screen will be stretched to fill the right eye display, which means only half of the horizontal resolution is available in 3D mode.

In order to render the virtual content as if it was merged with the physical world, precise tracking of the position and orientation the glasses is necessary. This is done by using marker-based AR tracking, which is provided by the Metaio SDK. Using the Metaio SDK, it is possible to define a coordinate system based on the relative positions of the markers, and by processing the live camera image, the Metaio SDK can provide the necessary transformation matrices that is used to render the virtual content.

As has been mentioned above, we can define a coordinate system based on the relative positions of the markers, and this can be done by using a pre-defined tracking configuration. A tracking configuration contains all the relative positions and orientations of all the markers in the system. See Appendix A for more details. Another important component of the marker-based tracking system is a marker sheet that can be easily printed by the user, and this marker sheet has to be precisely as defined by the tracking configuration. See Appendix B for details. It was found that using many of smaller markers—six markers is used—works better than using one big marker since the user’s hand sometimes occludes the marker as viewed from the camera if the marker is too big.

Once the Display Application knows how to render the virtual content, it needs to know what kind of virtual content to render. For example, the application needs to know where the remote fingers are located, what kind of virtual objects to render, and where to place them on the scene. The Display Application receives this information from the Client Application. JavaOSC library [16] is used to mediate the communication between the two applications. Remote hands are rendered as simple lines to indicate where the fingers are located (Fig. 2), whereas the actual hands, or so called local hands, are not rendered, since it is physically in the same space of the user, which means it is entirely visible to the user.

Since all rendering work has to be done on the Android device, the rendering capability is very constrained with respect to memory and versatility, leading to rendering of simple 3D object primitives only, such as lines or cubes. However, we realized that this is due to the limitation of the hardware, and using a different type of optical HMD would alleviate or solve the problem completely, so we decided not to pursue rendering complex objects. A partial solution to this problem on the Moverio BT-200 is discussed in the discussion section of the thesis.

5.4 AR Marker-Hand Calibration

In order to accurately align the hand positions as returned by the Leap SDK, which in turn is required to properly merge the virtual content and reality, it is necessary to perform a simple calibration step that takes into account the infinitely many different possible marker and leap relative position. This calibration procedure greatly supports the portability aspect of HandsOn, as there is no need for a fully calibrated setup; all the user needs to do is mount the marker sheet on a vertical surface, place the Leap Motion Sensor on a horizontal surface, and proceed with the calibration step on the Client Application.

The goal of the calibration procedure is to obtain a vector from the center of the Leap Motion Sensor and the center of the marker sheet. In order to do this, the user needs to point at the center of the marker sheet and press the designated key ‘c’ while the Client Application is running. A proper calibration is achieved when the visualization matches the real position of the hand joints. Note that the marker needs to be at a 90 degrees angle with respect to the Leap Motion Sensor for the calibration procedure to work. This calibration procedure only need to be performed once after every start-up, or after the sensor has been moved.

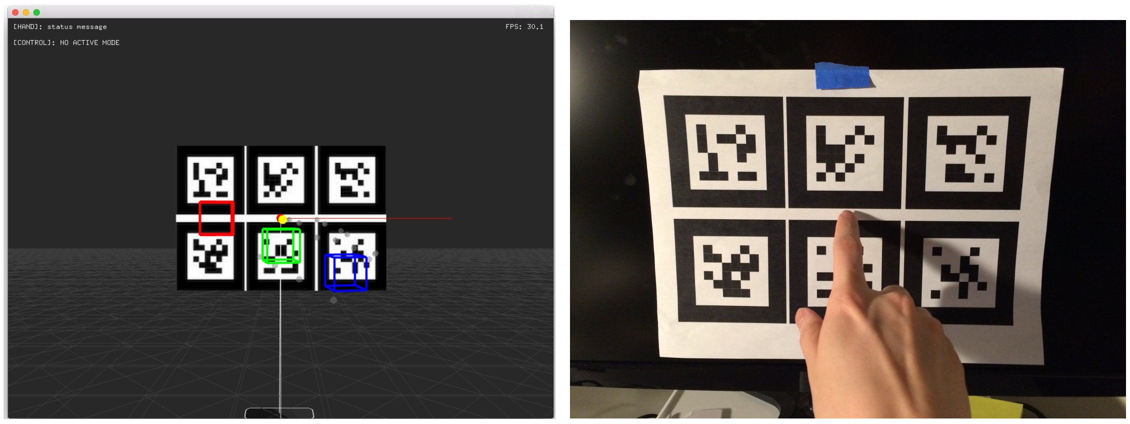


Figure 9. What the user have to do to calibrate (a) as visualized on the client (b) in the real world, i.e. pointing at the center of the marker sheet and pressing the calibration key ‘c’.

5.5 Binocular Optical See-Through Calibration

In order to achieve realism, it is important to render virtual content as if it is part of the physical world. Being part of the physical world means that any virtual content should be viewed as if it is a physical object, thereby having a consistent position and orientation and can be consistently viewed by the user from any point of view at any point in time. See Fig. 10. Therefore, it is necessary to properly calibrate the display.

The goal of this calibration procedure is to obtain a precise representation of the camera frustum relative to the right and left displays of the glasses. These parameters can be represented in many different ways, but the calibration procedure of the Metaio SDK yields a translation offset and a rotation offset for each of the display relative to the camera. These offsets are then used by the rendering module to place the virtual camera in the scene to render the virtual content properly.

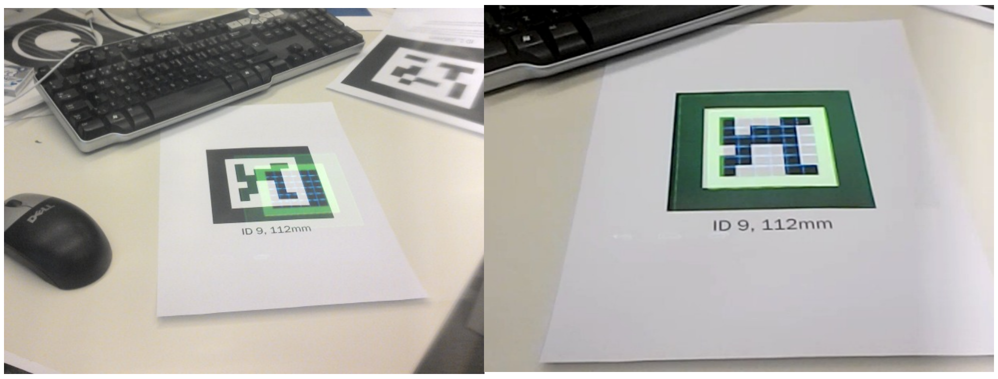


Figure 10. Virtual content rendering when not calibrated (left) and calibrated (right). Not calibrated: virtual content does not have consistent position and orientation, which is supposed to be immediately above the marker—when viewed from different angles. Calibrated: virtual content has consistent position and orientation when viewed from any angle. Images taken from [17].

```
<?xml version="1.0"?>
<HandEyeCalibration>
      <Metadata>
        <ToolboxVersion>6.0.1, v6010 (metaio SDK v6.0.1)</ToolboxVersion>
        <Device>EPSON_embt2 (embt2; EPSON; embt2; D1.0.5)</Device>
        <Date>2015-03-26 23:28:12.315-0400</Date>
      </Metadata>
      <!-- Transformation from camera to eye (display) coordinates -->
      <Transform for="ECT_RENDERING_STEREO_LEFT">
        <!-- Translation in millimeters -->
        <TranslationOffset>
                <X>107.973</X>
                <Y>-10.3894</Y>
                <Z>-30.7591</Z>
        </TranslationOffset>
        <!-- Rotation specified as unit quaternion -->
        <RotationOffset>
                <X>-0.0171863</X>
                <Y>0.0146289</Y>
                <Z>0.00643986</Z>
                <W>0.999725</W>
        </RotationOffset>
      </Transform>
      <Transform for="ECT_RENDERING_STEREO_RIGHT">
        <!-- Translation in millimeters -->
        <TranslationOffset>
                <X>46.5881</X>
                <Y>-6.80305</Y>
                <Z>-7.4931</Z>
        </TranslationOffset>
        <!-- Rotation specified as unit quaternion -->
        <RotationOffset>
                <X>-0.016325</X>
                <Y>0.0139466</Y>
                <Z>0.00442303</Z>
                <W>0.99976</W>
        </RotationOffset>
      </Transform>
</HandEyeCalibration>
```

Figure 11. Sample calibration parameters as obtained after completing the calibration procedure.

Ideally, the calibration should have been performed by the manufacturer, with calibration parameters readily available online. However, the manufacturing process of the Moverio BT-200 is not precise enough to yield to same calibration parameters for each pair. Hence, each pair needs to be calibrated individually, and the built-in calibration parameters would only work in a small number of devices, yielding an inexact overlay in most pairs.

Fortunately, the Metaio SDK provides a simple calibration routine that is available in a downloadable Android application called Metaio Toolbox. The calibration routine is a simple four-step process [17]. During the calibration procedure, the user will be prompted to match a transparent grid overlay of a certain size to the user's own view of a real marker with a known physical size. This is repeated for each eye, left and right, and for two different marker sizes. Once completed, a calibration file is generated and stored in the device.

It is important to realize that a calibration procedure is not necessary for video see-through application, where the virtual content is rendered on top of camera feed. In this case, the marker location and orientation is conveyed to the user through the camera feed instead of directly through the eyes of the user, so overlaying the virtual content on top of the marker is relatively simple. However, video see-through AR does not achieve the same kind of realism that optical see-through AR can, as the user does not perceive the mixed-reality world directly but through a camera feed.

One obvious limitation of this calibration procedure is that it needs to be performed for each pair, thereby requiring a new user with non-calibrated glasses to execute the calibration routine before they can use any binocular see-through application. However, it is reasonable to expect that future devices can be manufactured in a more precise fashion, therefore eliminating the need to calibrate each pair individually.

Another limitation that is more challenging involves user-specific calibration. Accurate rendering of virtual content on the real world involves precise measurement of the relative position and orientation of the eyes of the user, the camera used to track the markers, and the see-through displays. While this can be estimated to a certain degree, variations due to anatomical variation between users will change both the relative position of the display and the eye, which can lead to less-than-realistic perception of mixed-reality. A headset with eye-tracking capability can improve the situation, but this is beyond the scope of this thesis.

6

Evaluation

6.1 User Study

We conducted a user study to test the usability and effectiveness of the system as a gestural remote collaboration system on virtual content. We invited 6 subjects (2 males, 4 females) to participate in a formal user test in a controlled environment in which they have to perform specific tasks in pairs and collected their feedback. Before the main user study, we also conducted an informal pilot test of the system in an uncontrolled environment with no specific tasks to perform, and collected the testers' feedback. None of the participants in the formal user study had seen the optical HMD or the Leap Motion sensor prior to this study.

During the formal user study, the remote setup for HandsOn is used, where the HandsOn system is setup in two separate rooms and connected wirelessly. Each pair of participants was briefed on how to interact with the system before they started. During the study, the participants were given specific written instructions to take turns pointing at one of the displayed objects, e.g. colored cubes, move them around to a certain location, observe the remote users execute the tasks, and communicate with the remote users using gestures. All pairs of users were able to complete the tasks in less than 20 minutes.

6.2 Results

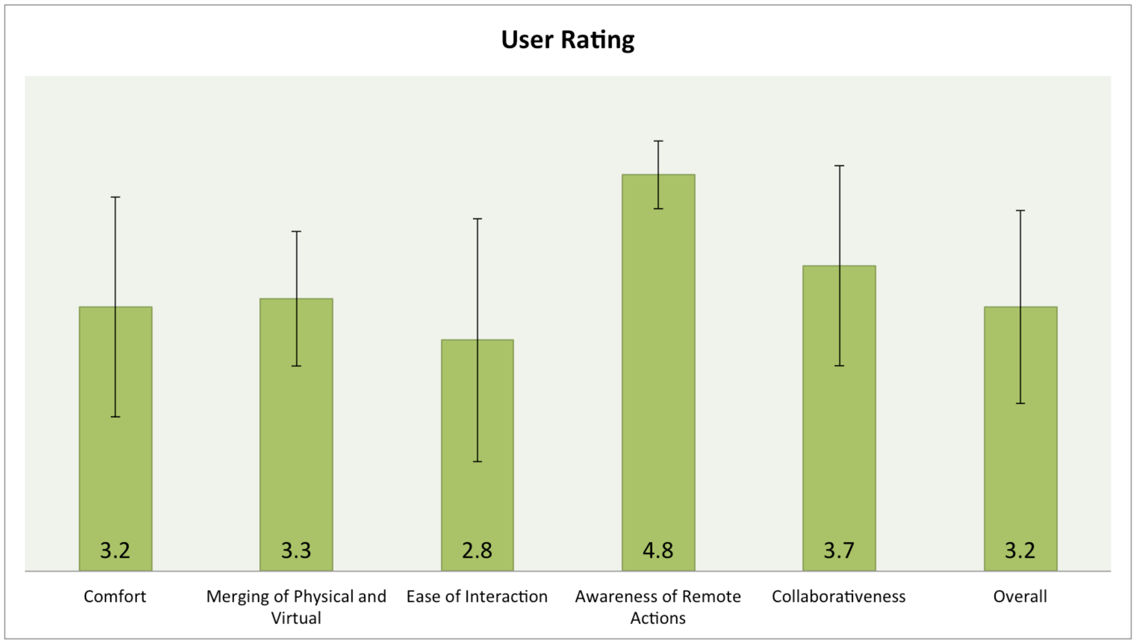


Figure 12. Average of user ratings on a 5-point Likert scale with 5 being the highest rating for each respective aspect. The error bars are +/- 1 standard deviation from the average.

Overall, we received positive feedback from the users. We found that users were very satisfied with HandsOn when it comes to communicating gestural information with remote users. They expressed that the rendering of the remote user's hands greatly increased the sense of co-presence and general awareness of the remote actions, as reflected by the consistently high ratings for this particular aspect. Furthermore, the system is also rated quite highly, 3.7 out of 5.0, in terms of collaborativeness since they were able to complete the designated tasks with minimal problems (see Fig. 12).

Users felt reasonably comfortable using the Moverio BT-200—a rating of 3.2 out of 5.0—with occasional complaints about how the glasses are too heavy or will not stay properly in place, mostly for participants who were wearing glasses. Many of them also commented on how they felt that the stereoscopic rendering really adds to the experience and how it gives the illusion of the virtual coexisting with the physical. Many users complained about how the virtual image is shaky and would not stay in place, which does not ruin the entire experience but definitely a nuisance to the users. This is caused by low resolution of the built-in camera of the Moverio BT-200, since the same problem is not observed when a tablet with higher resolution camera is used. Another interesting observation that we made was that a small number of users found it difficult to discern the 3-dimensional aspect of the rendering, and this phenomenon was observed in both our formal and informal testing sessions.

Users also felt that the interaction with the virtual objects felt natural, as pointing and pinching are natural actions a person would do to manipulate objects. While about half of the users found it easy to interact with the virtual objects, the other half of the users found it hard to initiate the interaction with the virtual objects, yielding lower ratings—2.8 out of 5.0—for ease of interaction. This was due the inability of the Leap Motion sensor to track the fingers of the users well outside of its optimal tracking region, which is a small region about 20 centimeters above the sensor. Many times, the hands are lost by the tracking system or the system would return unreliable finger joint positions. This severely limits the effective interaction range of the system, as the users have to position and orient their hands and the virtual objects in a space larger than the Leap Motion sensor's optimal tracking region. A better sensor, such as a high-resolution depth camera, would help track the hands in a larger region of space. However, finger tracking is a difficult problem to solve and it is not the focus of HandsOn. Another possible solution is to use marker-based tracking, e.g. by attaching color markers on the finger and performing computer vision analysis to track finger positions and poses.

When asked about possible user scenarios, many users connected the system immediately to their backgrounds. One of the participants is a chemical engineer and expressed how this system would be very useful for her, as her team often have to design a chemical plant collaboratively with a remote team, often the clients. Her current solution includes the use of video conferencing system or remotely logging into the remote team's computer, but a more extensive version of HandsOn could allow both teams to actively participate and visualize the end product in a decision-making meeting. She imagined how she could decide where to place new equipment or piping in a chemical plant expansion using HandsOn, which are often ambiguous and difficult to decipher from 2D drawings and layouts. Another participant who is an architect expressed similar ideas for meetings with clients or design sessions with fellow architects, which require many participants to highlight different aspects of one key model with a lot of details. Other ideas such as a learning tool for the study of 3D geometry in mathematics and interactive games were also mentioned by users.

6.3 Discussion

HandsOn has proved to be a helpful system for remote collaboration on virtual objects. However, we realized that there are rooms for improvements. Some of these are due to the limitation of the device that we are currently using, the Moverio BT-200, while some others fell in the general challenge of designing and implementing an AR-based system.

6.3.1 3D Depth Perception and Depth Occlusion

Some users mentioned that it is difficult to pinpoint where the virtual object is located even though it has been stereoscopically rendered. What is interesting is that not all user responses are the same—some reported that it is very easy to pinpoint where the virtual object is located and that the merging between virtual and the physical is very good, i.e. the user can easily select, pinch, and move an object as if they were in real objects.

One tester said that the tester personally is not very adept at 3D depth perception, self-explaining the tester's difficulty at using the system. Another commented that it is confusing to use the system because it is possible to see the virtual image even though the hand is supposed to be occluding the object, i.e. the lack of depth occlusion in the current implementation of the system. This problem of implementing depth occlusion is a common challenge in implementing AR-based system, since the system requires extensive sensing of the surrounding environment in real-time in order to know which object is occluding another.

One possible solution that will alleviate the problem is to head-mount a depth sensor to build a depth map of the physical world which is aligned with the user's own view. This will allow the system to render virtual objects selectively: objects that are behind the obstructing objects will not be rendered. A different placement option of the depth sensors can also be explored—such as placing the depth sensor on the workspace instead of head-mounting it—but this will require possibly complicated setup procedure, thereby limiting the portability aspect of HandsOn.

6.3.2 Client Rendering

The fact that the rendering is performed on the Android device that is connected to the Moverio BT-200 severely limits the range of objects that can be rendered. For example, using the current version of the Metaio SDK, it is not a simple task to render 3D model when in see-through mode, which is one of the reason that only simple objects are implemented. Furthermore, the rendering capability is limited by the amount of available memory on a mobile device. One possible solution is to use a hardware that can be connected directly to a general-purpose computer and displays whatever is on the screen, at the expense of portability, but this contradicts the portability aspect of HandsOn.

One solution that is likely to work is by letting the Client Application perform the rendering on the computer, and then streams the resulting video to the Display Application, which then simply displays the received video. However, a two-way communication is necessary because the Client Application requires the model, view, and projection matrices for each eye, which depends on the location of the marker on the live camera image, to be sent by the Display Application. Therefore, in this case, the role of the Display Application is reduced to tracking the marker, sending that information, and receiving the video stream from the Client Application, but this will lead to an improved rendering and development environment, as the computer that runs on the Client Application does not have the constraints of a mobile device like the Android device running the Display Application.

There are, however, issues that might arise in this new setup: sending the marker tracking information, rendering on the Client Application, and sending the resulting video back, may introduce unacceptable latency to the system, leading to loss of consistency in the merging of the physical and the virtual world.

7

Conclusions

We proposed HandsOn, a portable system for collaboration on virtual objects that supports both co-located collaboration and remote collaboration. HandsOn takes advantage of the most recent advances in display and sensing technology, combining optical see-through HMD and finger tracking to merge the physical space with virtual objects, which users can manipulate using simple hand gestures. This proof-of-concept system is designed to be portable, to run wirelessly, and to be able to be easily setup using a simple calibration procedure as described in section 5.4 and 5.5.

We received positive feedback from our user study participants. HandsOn did well on increasing user awareness on remote user and actions, as well as creating the illusion of the physical and the virtual space being merged. However, users found it difficult at times to initiate object manipulation using hand gestures. This is mostly due to the limited tracking capabilities of the Leap Motion sensor, which often yields inaccurate finger tracking. Furthermore, the HMD's built-in camera's limited resolution sometimes lead to unstable virtual rendering in poor lighting conditions.

Even though most of the limitations of the current system are due to the limitation of the Moverio BT-200 optical HMD that we are using, future work should not focus on overcoming these barriers, since better display technologies, such as the Microsoft's HoloLens and the Magic Leap, and better finger tracking technology will eventually solve the problem in the future. Instead, future research should focus on exploring ways to design a truly 3D user interface for interacting with virtual objects. Challenges will be abundant in this topic since the general population is not yet accustomed to truly natural, 3D user interfaces, especially when there is no haptic feedback. Therefore, haptic feedback for AR-based interfaces is also another research subject worth exploring. Implementing occlusion for AR-based interfaces is also a very important subject to pursue, since true AR would require occlusion to create the perfect illusion of the virtual being merged with the physical reality.

Appendix A

HandsOn Tracking Configuration

The following is the content of the tracking configuration file used by the Metaio SDK to track the AR markers described in Appendix B.

```
<?xml version="1.0"?>
<TrackingData>
      <Sensors>
      <Sensor type="MarkerBasedSensorSource">
                <SensorID>HandsOnMarkerTracking</SensorID>
                <Parameters>
                        <MarkerTrackingParameters>
                                <TrackingQuality>robust</TrackingQuality>
                                <ThresholdOffset>128</ThresholdOffset>
                                <NumberOfSearchIterations>2</NumberOfSearchIterations>
                        </MarkerTrackingParameters>
                </Parameters>

         <SensorCOS>
                <SensorCosID>Marker1</SensorCosID>
                <Parameters><MarkerParameters>
                <Size>80</Size><MatrixID>1</MatrixID>
                </MarkerParameters></Parameters>
        </SensorCOS>
        <SensorCOS>
                <SensorCosID>Marker2</SensorCosID>
                <Parameters><MarkerParameters>
                <Size>80</Size><MatrixID>2</MatrixID>
                </MarkerParameters></Parameters>
        </SensorCOS>
       <SensorCOS>
                <SensorCosID>Marker3</SensorCosID>
                <Parameters><MarkerParameters>
                <Size>80</Size><MatrixID>3</MatrixID>
                </MarkerParameters></Parameters>
         </SensorCOS>
        <SensorCOS>
                <SensorCosID>Marker4</SensorCosID>
                <Parameters><MarkerParameters>
                <Size>80</Size><MatrixID>4</MatrixID>
                </MarkerParameters></Parameters>
         </SensorCOS>
         <SensorCOS>
                <SensorCosID>Marker5</SensorCosID>
                <Parameters><MarkerParameters>
                <Size>80</Size><MatrixID>5</MatrixID>
                </MarkerParameters></Parameters>
          </SensorCOS>
          <SensorCOS>
                <SensorCosID>Marker6</SensorCosID>
                <Parameters><MarkerParameters>
                <Size>80</Size><MatrixID>6</MatrixID>
                </MarkerParameters></Parameters>
           </SensorCOS>
            </Sensor>
      </Sensors>

      <Connections>
        <COS>
```

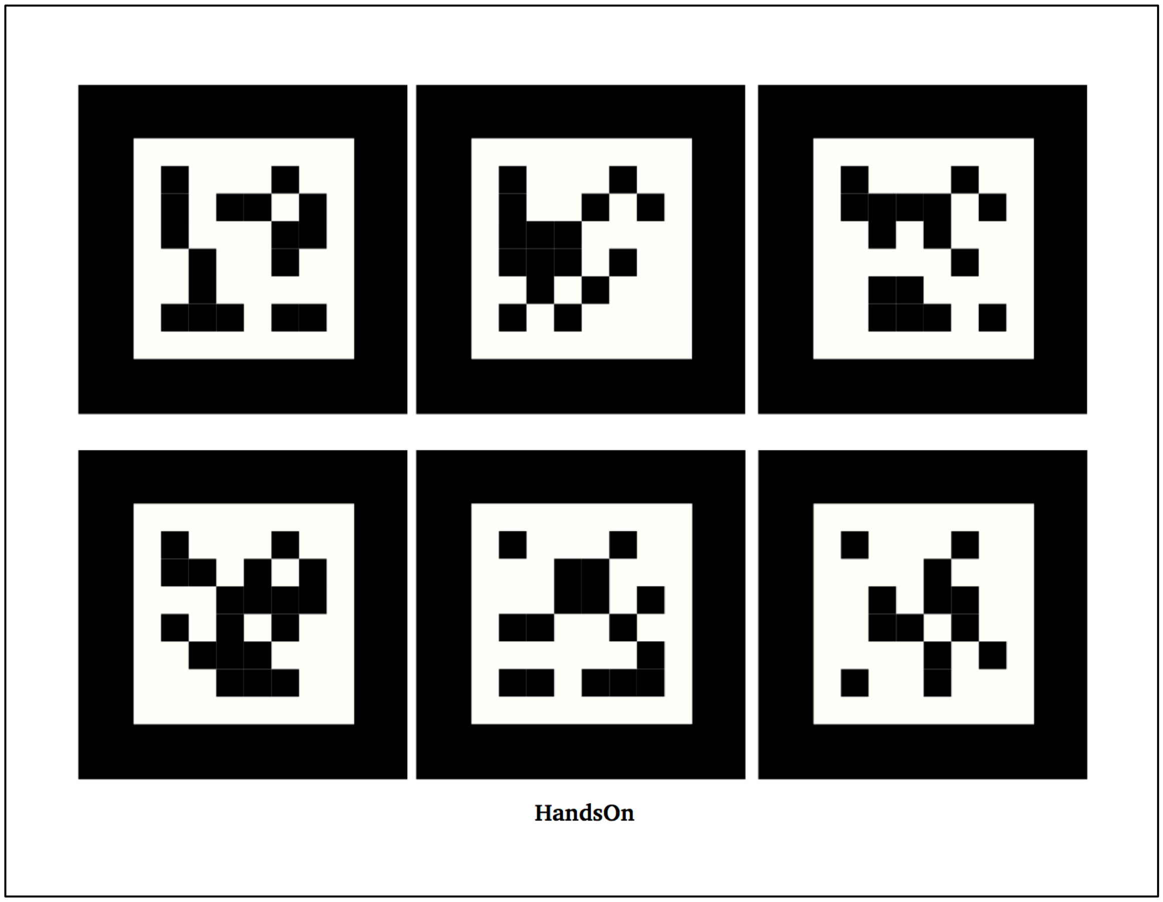
```
              <Name>HandsOnCOS</Name>
              <Fuser Type="SmoothingFuser">
                     <Parameters>
                            <KeepPoseForNumberOfFrames>3</KeepPoseForNumberOfFrames>
                            <GravityAssistance></GravityAssistance>
                            <AlphaTranslation>0.8</AlphaTranslation>
                            <GammaTranslation>0.8</GammaTranslation>
                            <AlphaRotation>0.5</AlphaRotation>
                            <GammaRotation>0.5</GammaRotation>

     <ContinueLostTrackingWithOrientationSensor>false</ContinueLostTrackingWithOrientatio
nSensor>
                     </Parameters>
              </Fuser>

              <SensorSource>
                     <SensorID>HandsOnMarkerTracking</SensorID>
                     <SensorCosID>Marker1</SensorCosID>
                     <HandEyeCalibration>

     <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>

     <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                     </HandEyeCalibration>
                     <COSOffset>
                   <TranslationOffset><X>82</X><Y>-45</Y><Z>0</Z></TranslationOffset>
                   <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                     </COSOffset>
              </SensorSource>

              <SensorSource>
                     <SensorID>HandsOnMarkerTracking</SensorID>
                     <SensorCosID>Marker2</SensorCosID>

               <HandEyeCalibration>
                   <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>
                   <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
               </HandEyeCalibration>

               <COSOffset>
                   <TranslationOffset><X>0</X><Y>-45</Y><Z>0</Z></TranslationOffset>
                   <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
               </COSOffset>
              </SensorSource>

           <SensorSource>
               <SensorID>HandsOnMarkerTracking</SensorID>
               <SensorCosID>Marker3</SensorCosID>

               <HandEyeCalibration>
                   <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>
                   <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
               </HandEyeCalibration>

               <COSOffset>
                   <TranslationOffset><X>-83</X><Y>-45</Y><Z>0</Z></TranslationOffset>
                   <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
               </COSOffset>
           </SensorSource>
           <SensorSource>
               <SensorID>HandsOnMarkerTracking</SensorID>
               <SensorCosID>Marker4</SensorCosID>

               <HandEyeCalibration>
                   <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>
                   <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
               </HandEyeCalibration>
```

```
                <COSOffset>
                    <TranslationOffset><X>82</X><Y>45</Y><Z>0</Z></TranslationOffset>
                    <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                </COSOffset>
            </SensorSource>
            <SensorSource>
                <SensorID>HandsOnMarkerTracking</SensorID>
                <SensorCosID>Marke5</SensorCosID>

                <HandEyeCalibration>
                    <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>
                    <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                </HandEyeCalibration>

                <COSOffset>
                    <TranslationOffset><X>0</X><Y>45</Y><Z>0</Z></TranslationOffset>
                    <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                </COSOffset>
            </SensorSource>
            <SensorSource>
                <SensorID>HandsOnMarkerTracking</SensorID>
                <SensorCosID>Marker6</SensorCosID>

                <HandEyeCalibration>
                    <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>
                    <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                </HandEyeCalibration>

                <COSOffset>
                    <TranslationOffset><X>-83</X><Y>45</Y><Z>0</Z></TranslationOffset>
                    <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                </COSOffset>
            </SensorSource>


        </COS>
      </Connections>
</TrackingData>
```

Appendix B

HandsOn Marker Sheet

The following is a scaled-down version of the HandsOn Marker Sheet.



Unfortunately, the full-sized version cannot be included in this thesis due to margin constraints. A ready-to-print version in PDF format is available here:

https://drive.google.com/file/d/0B2l223gZ6JZSWGhiUk0yNVlUV0k/view?usp=sharing

Please make sure to print the marker sheet on letter-sized paper with no scaling applied. Otherwise the tracking configuration file described in Appendix A will not work properly.

References

[1] David Lakatos, Matthew Blackshaw, Alex Olwal, Zachary Barryte, Ken Perlin, and Hiroshi Ishii. 2014. T(ether): spatially-aware handhelds, gestures and proprioception for multi-user 3D modeling and animation. In Proceedings of the 2nd ACM symposium on Spatial user interaction (SUI '14). ACM, New York, NY, USA, 90-93.

[2] Spindler, M., Büschel, W., and Dachselt, R. Use Your Head: Tangible Windows for 3D Information Spaces in a Tabletop environment. Proc. ITS '12, 245-254.

[3] Rajinder S.Sodhi, Brett R.Jones, David Forsyth, Brian P.Bailey, and Giuliano Maciocci. BeThere: 3D Mobile Collaboration with Spatial Input. In Proc. CHI 2013, ACM Press (2013), 179-188.

[4] Grossman, T. and Balakrishnan, R.. Collaborative interaction with volumetric displays. Proc. CHI '08, 383-392.

[5] Agrawala, M., Beers, A., McDowall, I., Fröhlich, B., Bolas, M., and Hanrahan, P. The two-user Responsive Workbench: support for collaboration through individual views of a shared space. Proc. SIGGRAPH '97, 327-332.

[6] Ivan E.Sutherland. A head-mounted three- dimensional display. In Proc. AFIPS 1968, ACM Press (1968), 757-764.

[7] Judith Amores, Xavier Benavides, and Pattie Maes. A Remote Collaboration System that Supports Immersive Gestural Communication. Work In Progress, CHI 2015.

[8] J.P. Rolland and H. Fuchs, "Optical Versus Video See-through Head-Mounted Displays in Medical Visualization," Presence: Teleoperators and Virtual Environments, vol. 9, no. 3, June 2000, pp. 287-309.

[9] Kiyokawa, K., Takemura, H., Yokoya, N. "SeamlessDesign for 3D Object Creation," IEEE MultiMedia, ICMCS '99 Special Issue, Vol.7, No.1, pp.22-33, 2000.

[10] Mark Billinghurst , Hirokazu Kato, Collaborative augmented reality, Communications of the ACM, v.45 n.7, July 2002.

[11] Broll, Wolfgang, Irma Lindt, Jan Ohlenburg, Michael Wittkämper, Chunrong Yuan, Thomas Novotny, Chiron Mottram, A. Fatah gen Schieck, and A. Strothman. "Arthur: A collaborative augmented environment for architectural design and urban planning." (2004).

[12] Schmalsteig, D., Fuhrmann, A., Szalavari, Z., Gervautz, M., Studierstube - An Environment for Collaboration in Augmented Reality. In CVE '96 Workshop Proceedings, 19-20th September 1996, Nottingham, Great Britain.

[13] Sarayeddine, K., & Mirza, K. (2013, May). Key challenges to affordable see-through wearable displays: the missing link for mobile AR mass deployment. InSPIE Defense, Security, and Sensing (pp. 87200D-87200D). International Society for Optics and Photonics.

[14] Magic Leap (2015). http://www.magicleap.com.

[15] Microsoft HoloLens (2015). https://www.microsoft.com/microsoft-hololens/en-us.

[16] Illposed Software – Java OSC (Download) (2015).
http://www.illposed.com/software/javaosc.html

[17] Metaio SDK: Stereo See-Through Calibration (2015).
http://dev.metaio.com/sdk/documentation/standalone-apps/toolbox/stereo-see-through-calibration/

[18] Reality Check: Comparing HoloLens and the Magic Leap (2015).
http://www.technologyreview.com/news/535806/reality-check-comparing-hololens-and-magic-leap/