Passive Haptic Feedback for Object Manipulation in Virtual Reality

BY

FRANCESCO MANTOVANI
B.S., Università degli Studi di Modena e Reggio Emilia, Modena, Italy, 2016

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science and Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:

Andrew E. Johnson, Chair and Advisor

Robert V. Kenyon

Fulvio Risso, Politecnico di Torino

ACKNOWLEDGMENTS

First, I want to thank my family who always supported me during my stay in Chicago. I want also to thank all the people I met during this experience in the United States, all the friends that made me feel like home and the people of EVL for giving me the opportunity to work with them.

<div align="right">FM</div>

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

VR   Virtual Reality

HMD   Head Mounted Display

BHI   Bare Hand Interaction

VE   Virtual Environment

SUMMARY

Thanks to technological advancement and lower priced head mounted displays, Virtual Reality is becoming popular in the consumer market. One of the main aspects on which researchers are focusing in these days, beside increasing the resolution of the displays, is finding a way to interact in a way that is as similar as possible to the real world. In this document we will first analyze some previous work done in field. Then, we will show our implementation of a passive feedback system which uses real objects to augment virtual reality experiences. The system will be used on two different sets of real objects. The first is composed of the exact replica of the objects present in the virtual environment, the second is composed by objects with several levels of mismatched physical characteristics. The document will then explain in detail the integration of the system in VR applications using Unity.

In the second part of this document we will describe a user study to compare the execution of object manipulation tasks with the system that we developed to two other state-of-art input devices: Leap Motion and VR Controllers. We will analyze the factors of the task execution in virtual reality, such as the accuracy, velocity, time to execution, level of realism and immersion, and we will compare the results with the execution of the same task in the real world. Finally, we will give suggestions on how to enhance our system in future work.

CHAPTER 1

INTRODUCTION

Virtual Reality has been becoming increasingly popular in recent years. Thanks to technological advancements, head mounted displays are becoming more afforadble and more popular in the consumer market. Its utilizations, however, are not only focused on the gaming and on the entertainment sector. Virtual Reality today is widely used also in scientific visualization, in the military sector, in the educational sector and in the healthcare field.

One of the biggest challenges for developers and researchers in this area of Computer Science is to find a way to enhance the realism and the immersion of a virtual reality simulation by bringing the sense of touch in the interaction. As we will see in this document, there are already state-of-the art devices developed for this purpose, but they have some limitations. In this thesis we describe the implementation of a system that makes possible interaction for object manipulation tasks in a virtual environment as close as possible to the one in the real world. We will also show how we integrated the system in some VR applications to compare it with some state-of-the-art commercially available devices.

1.1 Motivations

The inspiration for this research work comes from working on a project at the UIC Electronic Visualization Laboratory (EVL) commissioned by the Shirley Ryan AbilityLab (formerly Rehabilitation Institute of Chicago), a research rehabilitation hospital classified first nationwide

[1]. The project, done in a team of researchers under the supervision of Dr Andrew Johnson at EVL, consists in developing Virtual Reality applications to help patients in their physical rehabilitation process. A deep analysis of Strengths, Weaknesses, Opportunities, and Threats of Virtual Reality in rehabilitation has been carried by A. Rizzo & G. J. Kim in [1] and numerous research works prove that VR is an effective tool for rehabilitation.

One of the main reasons that encouraged AbilityLab to look at VR is for reducing the boredom of physical therapy. Often patients are requested to repeat the same movement over and over to improve the dexterity of some part of their body. The use of serious games can improve the patient's engagement as shown by J. W. Burke et al. in [2] . Several serious games have been developed specifically for the AbilityLab's necessities (e.g. applications for the stimulation of the balance, for the movement of upper limbs, etc.). One request from the center therapists that at first might seems banal, deserves more attention: the simulation of interaction with objects. What is the best way to interact with objects in VR so that the patient would have the closest experience to reality? This was the question that started this research. The work done is focused not just on VR experiences for rehabilitation, but on general applications that involve the interaction with objects.

1.2 Thesis Structure

In Chapter 2 we will give the reader a brief overview of the technologies involved in this research. We will quickly go through the history of Virtual Reality and illustrate the main devices and concepts exploited in this research.

Chapter 3 illustrates interesting studies on the use of real objects in virtual reality.

Chapter 4 shows how we developed our passive haptic feedback system and the implementation of the virtual reality applications that we used for our user study.

Chapter 5 describes a user study that we conducted to test our system with three virtual reality applications. The first part deals with the procedure and the apparatus involved in the study. The second part shows the collected data and the analysis we performed to discuss the results.

Chapter 6 provides a summary of the thesis findings, discussing how to obtain more consistent results and suggestions on how to apply the results of our work to future applications.

CHAPTER 2

BACKGROUND

In this chapter we will give the reader some general knowlege on the technologies involved in our work. In particular we will first briefly summarize the history of virtual reality and the characteristics of the HTC Vive. In the second part we will deal with the sense of touch and haptics, specifically in virtual reality.

2.1   The virtuality continuum

In this document we will talk about augmenting a virtual environment with real objects. There have been various ways to try to classify such experiences. The first time that the terminology "Mixed Reality" appeared in the literature was in the 1994 article *"A taxonomy of mixed reality displays"* by Milgram and Kishino [3]. In this work, the authors introduced the concept of a "virtuality continuum" of which a simplified representation is illustrated in Figure 1.
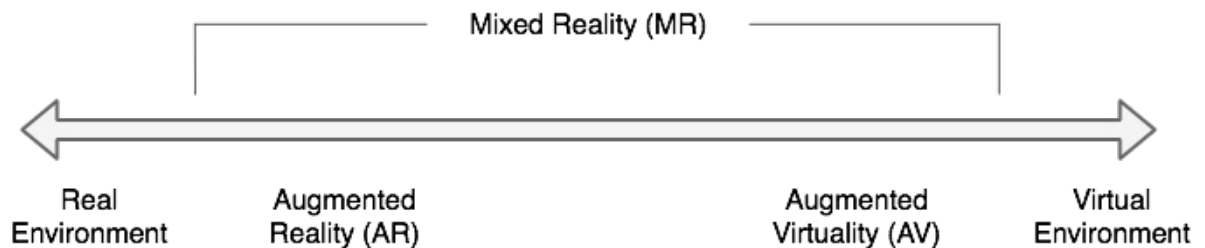
Figure 1: The virtuality continuum

According to the authors, real environments lie on one side of the model, while virtual environments on the other side, everything that falls in between is called Mixed Reality (MR). They also introduce two further categories close to the side of the diagram. Those are the Augmented Reality (AR) and the Augmented Virtuality (AV). In AR the real environment is augmented with computer generated graphics, while in AV the virtual environment is enriched by elements of the real world.

The work that we are going to present in this document is related to the topic of enriching virtual environments with real-world content to enhance the user's experience by providing a passive haptic feedback. Integrating real objects in a virtual environment classifies our work in the macro-category of Mixed Reality and since the environment in which our tasks are executed is mostly virtual, we should consider our work to be part of the field of the Augmented Virtuality. However, this term has not received much popularity in the literature so we will keep using the phrase (enhanced) Virtual Reality.

2.2   Virtual Reality

Virtual Reality is defined as "a three-dimensional, computer generated environment which can be explored and interacted with by a person" [4], the key elements of VR are the immersion and the interactivity. The term "Virtual Reality" was first coined in 1987 by Jaron Lanier, founder of the VPL (Visual Programming Lab) [5], but even before that we have some examples of interactive immersive experiences.

In 1962 the cinematographer Morton Helig developed the *Sensorama* (Figure 2), a mechanical machine known to be one of the earliest examples of immersive multi-sensorial experiences.

This device was able to display 3D images with a wide field of view enhanced by a vibrating chair, fans and smell generators. Helig also patented the first head mounted display (the *Telesphere Mask*), but he could not obtain founding to further develop it.

Figure 2: The Sensorama by Morton Helig - Credits[Minecraftpsyco [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], from Wikimedia Commons]

One of the most famous attempts to create an head mounted display is the one of Ivan Sutherland in 1968 [5] who created the *Sword of Damocles*, a large and scary looking device

considered the first AR / VR HMD. For the first time interactive head-tracked graphics were computer generated but rather basic. It was so heavy and bulky that it had to be hung from the ceiling and laced to the user's head.

Skipping forward, we have the founding of the *VPL Research* company by Jaron Lanier in 1985. This company gave birth to some virtual reality devices like the *EyePhone* head mounted display and the *DataGloves* and *DataSuit* input systems.

In 1989 Fake Space Labs developed the *BOOM* (Binocular Omni-Orientation Monitor), a stereoscopic display device that was attached to a multi-link arm providing tracking. The user needed his hands to move the device.

In 1991 the Virtuality Group launched some virtual reality arcade machines, it was the first time that VR devices conquered public spaces. These devices consisted in stereoscopic visors an joysticks for the interaction, both magnetically tracked. Some of them were also networked together for multiplayer games.

In 1992 at the Electronic Visualization Laboratory (EVL) at The University of Illinois at Chicago Carolina Cruz-Neira, Daniel J. Sandin and Thomas A. DeFanti developed the first *CAVE (Cave Automatic Virtual Environment)*. CAVE was the first cubic immersive environment based on rear-projection screens making shared virtual reality experiences possible.

In 1992-5 there have been attempts to bring virtual reality headsets to the consumer world. SEGA announced the *SEGA VR* headset which never made it to the market due to development issues. Nintendo launched the *Virtual Boy*, a 3D gaming console that was a failure dut to the lack of full color graphics and the scarce ergonomics of the device.

In the 21st century we start to see the major and rapid improvements in the VR field. In 2010 came to the world the first prototype of the *Oculus Rift* by Palmer Luckey which brought to the consumer market the first head mounted display with a 90° field of view. Facebook purchased the company Oculus VR in 2014 for $2 Billions. In 2012 the EVL developed the second version of CAVE, the CAVE2, a hybrid reality environment composed of 72 passive stereo flat panels, optical tracking and 21.1 sound system.

Google launched *Google Glass* in 2013. They were one of the first low cost, lightweight augmented reality devices.

In the period between 2013 and 2015 many companies like Google, HTC/Valve, Sony, Microsoft and Oculus announced their HMD devices. In 2016 many of these products appeared on the consumer market. Today a wide range of VR HMDs are available which are mainly differenciated from each other by the resolution of the display, the tracking system and the location of the computational unit (i.e. on the device itself or if they have to be wired to a PC). Virtual Reality is again going through an hype phase after the one in the 90's. This time, thanks to the lower cost of the technology, it is getting more widespread and adopted by consumer users.

2.2.1 VR Hardware

They key elements of a VR system are:

- Display

- Tracking systems

- Image generator

The displays used for a virtual reality experience characterize its type. There are four different kinds of hardware currently used:

- Head Mounted Display

- BOOM Mounted Display

- Fish Tank

- Large Format Display

BOOM mounted display are not used anymore as they have been outclassed by Head Mounted Displays.

Fish Tank VR consists of a "stereo image of a three dimensional (3D) scene viewed on a monitor using perspective projection coupled to the head position of the observer" [6]. This provides however the lowest level of immersion. Large format displays (projector or flat panel based) are used to create shared virtual reality experiences. Systems like the CAVE2 use this technology.

Head Mounted Displays are the ones that interest us here. HMDs provide the highest levels on immersion and are those on which our work focuses. As we have seen in the previous section, the history of VR is basically based on the advancements of in the technology of head mounted displays. The objective of VR displays is to create stereo visuals feeding to each eye an image from a different perspective. In older HMDs two displays (one per eye) were used, while with current technology it is possible to use a single high pixel density display, half per each eye.

Tracking in a VR is used mainly to track the user's head position and rotation in order to display the image from the correct perspective. The tracking systems currently available are:

- Magnetic

- Mechanical

- Acoustic

- Optical

- Inertial

Magnetic tracking was used in the first version of the CAVE, its main drawback is that it suffers from interference from nearby metal objects. Mechanical tracking is used in BOOM Mounted Displays, it is accurate, but it limits the movements of the user. Acoustic tracking uses sound-waves to calculate the position of an object. Optical and inertial tracking are the most used in VR today due to their precision and their relatively low cost. We will discuss optical tracking in detail in Chapter 4, but it generally uses computer vision algorithms to calculate an object's position. Intertial tracking uses data from accelerometers and gyroscopes placed on the tracked object to keep track of its position and rotation.

The image generator is the unit responsible for generating the graphics. It can be on the device itself (e.g. *Oculus GO*) or a separate PC (e.g. *HTC Vive, Oculus Rift*). In general, VR simulations are computationally expensive and for a complex experience a PC with a good GPU is required.

2.2.2   HTC Vive

In this section we will describe the HTC Vive system and we will explain the motivations which lead to the choice of this device for our research. When we started our project three major consumer virtual reality PC headsets were available on the market: *HTC Vive, Oculus Rift, Microsoft Mixed Reality.*

HTC Vive and Oculus Rift have the same characteristics for display resolution (2,160 x 1,200 pixels total, 1,080 x 1,200 pixels per eye) and same field of view of 110°, while Microsoft Mixed Reality devices have a lower resolution of 1440 x 1440 (20 x 1440 pixels per eye). What influenced our choice the most was the tracking system. HTC Vive uses a proprietary tracking system called *Lighthouse* which permits room-scale tracking (we will explain this next). Oculus Rift uses a optical tracking (called *Constellation*) where one or more cameras detect the position of some IR LED on the device. It has however some limitations when used for room-scale scenarios and it performs best in small areas and seated experiences. Both Vive and Rift use outside-in tracking. Microsoft Mixed Reality devices use another kind of optical tracking called outside-in to track the controllers and intertial tracking for the headset which also sense some external sensors. This system is less recommended for accurate room-scale experiences. (Note: inside-out and outside-in tracking are explained in 4.2.3.2)

The HTC Vive is a VR head mounted display system developed by Valve Corporation and HTC. It was announced in March 2015 during HTC's Mobile World Congress and commercialized in April 2016. In its basic configuration the system is composed by the headset, two hand-held controllers and two base station for the tracking.

The headset has an OLED display of 2160×1200 pixels and the lenses in front of it create a field of view of 110° with a resulting pixel density of 11 pixels/° per eye and a refresh rate of 90Hz. The device is also equipped a front camera and accelerometers and gyroscopes to support the tracking system.

The tracking technology used is unique and it is not found in any other device. It is based on IR laser beams and photosensors. Here we explain the basic functioning of the V1.0 of this technology (Note: V2.0 is already being deployed in newer devices). On the headset and on the controllers some photosensors (around 24) are present in certain positions which are sensitive to IR light. In the room are placed two base stations (called Lighthouses) at ceiling height in the opposite corners of the play area. The base stations are synchronized (wirelessy or with a cable). In those devices there are theree elements: a IR illuminator, a vertical IR beam emitter, a Horizontal IR beam emitter. The tracking works in this way:

1. The lighthouses emit an IR flash which illuminates the room.

2. Photosensors on the tracked device detect the flash and start a timer.

3. Lighthouses sweep the room horizontally with a IR laser beam.

4. Photosensors on the tracked device detect the beam when it reaches them and stop the timer.

5. The lighthouses emit another IR flash which illuminates the room.

6. Photosensors on the tracked device detect the flash and start a timer.

7. Lighthouses sweep the room vertically with a IR laser beam.

8. Photosensors on the tracked device detect the beam when it reaches them and stop the timer.

These phases are repeated at a frequency of 60Hz. Knowing the speed at which the laser beam sweeps the room it is possible to calculate the relative position with respect to the base station of the sensor. The system then calculates the position of the tracked device using the data coming from its photosensors [7]. It gives good results, but it is impossible to determine the actual position of a tracked device at a certain instant of time. The headset also uses inertial tracking to improve the system.

2.3   The Sense of Touch

This thesis will deal with the sense of touch in a virtual environment, so we think that it is useful to give to the reader the basic concepts on how our body perceives tactile stimulation. For this section we took inspiration from the interesting book *"In touch with the future: The sense of touch from cognitive neuroscience to virtual reality"* written by Alberto Gallace and Charles Spence [8].

We want to start from our favorite citation from this book: *"The sense of touch is the one that contributes most to making things "real" to us, the one that cannot be fooled, and perhaps even the most arousing of our senses."* [8]. This is true. All the actions that we normally do every day are deeply based on the sense of touch, we take our decisions based of what we feel. The somatosensory system is extremely complex for us that are not in the medical field, but it is basically composed by tactile receptors that innervate different parts of the skin and neural pathways which transmit the stimuli to the central neural system. The receptors are not

uniformly distibuted on the body surface, indeed they are more concentrated on areas like the hands where our sense of touch is more acute. The tactile sensation is composed of different properties of the stimuli:

- Microgeometric properties: for example the texture, roughness, stickiness, and spatial density of the surface.

- Macrogeometric properties: for example the shape, size, and structure of an object.

- Spatial properties: the location of the stimuli according to a reference point.

All these properties together makes the sense of touch.

2.3.1   Meaning of 'haptic'

In this document we will often use the word "haptic" and we understand that it is not such a common word, so here we try to explain what we are refering to. The word *Haptic* comes from the Greek word *haptikos*: "able to touch or grasp" and it has gained the meaning of "Relating to the sense of touch, in particular relating to the perception and manipulation of objects using the senses of touch and proprioception" [9]. With haptics we mean the process of recreating the sense of touch by applying forces, vibration or motion to the user. There are several ways to do this:

- Vibration: using eccentric rotating mass actuator (ERM) or linear resonant actuator (LRA) to move a mass to create a vibration.

- Force Feedback: using external forces to manipulate the movement of the device held by the user.

- Non-contact: using ultrasounds or air to create the sense of touch.

2.3.2 The sense of touch in virtual reality

Bringing tactile sensation has been one of the biggest challenges in virtual reality. People in this field are realizing that focusing on the enhancement of the visual part of the experience (e.g. by increasing the display resolution and the quality of 3D renderings) is not enough to augment the realism. As it has just been said, the sence of touch is what makes things real. For this reason many VR system try to replicate touch sensations in the most sensitive part of our body, the hands. The most common input device in VR is a set of controllers which can not really simulate touch, but they act more like instruments that the user can use to grab or poke an object in the virtual environment. In this way a third part is introduced in the interaction and that can result in loss of realism and immersion making everything less convinving.

On the market there are solutions that allow the reproduction in the virtual environment of hands movement, and some of them can also provide the user with tactile information by means of vibration. Another solution is the use of force feedback devices, namely devices which produce forces to limit and resist the user's movement. *HaptX*[1] and *VRgluv*[2] are some examples, however, those instruments tend to be bukly and complex from an engineering point of view. In addition, it is nearly impossible to replicate all of the information transmitted in act of touching an object including the characteristics of its surface and its weight. Other solutions foresee the

---

[1]https://haptx.com/

[2]https://vrgluv.com/

use of grounded force feedback haptic interfaces like the one presented in[10] by Xu and Wang, but those do not lend themselves to the use in VR because the movements would be too limited.

Finally, in recent years, some suits have appeard on the market that seem to come straight from a science fiction movie. These are haptic suits, devices aimed to provide haptic feedback to the whole body by the means of vibration or electic impulses. In a certain way that is even easier that bringing the touch sensation to the hands as tactile receptive fields are larger on certain areas of the body [8] and fewer actuators would be needed. This is certainly way to increase realism and the immersion but it would not add much more to interaction.

CHAPTER 3

RELATED WORK

In this chapter we will illustrate and discuss some studies that are relevant to and inspired this research.

3.0.1   Tactile Augmentation and the Realism of Virtual Environments

One of the first studies in this field is the one performed by Hunter G. Hoffman in 1998 [11]. In his study, he explored the impact of physically touching a virtual object on the perceived realism of the virtual environment for the user. It is worth mentioning that in 1998 the VR hardware and the computer graphics were at a complete different level compared to today's technologies. Hoffman in his experiment used a Division ProVision 100 system with a Division dVisor™ HMD and a Polhemus branded magnetic sensor to track the position of the object.

Figure 3: The apparatus used by Hoffman. © 1998 IEEE

17

Figure 3 shows the hardware for the study, we can notice the difference with todays' apparatus. Attached to the plate is the sensor for the tracking.

The study consisted of an experiment where people were divided in two groups: the first group would experience the tactile augmentation, while the second group would only see the virtual environment and interact with it through a controller. The virtual environment was a representation of a kitchen with some objects. Among those object there was a plate which participants could interact with. Participants in the first group were equipped with a tracked glove to show the hand's virtual counterpart in the VE and were asked to touch a physical plate which was mapped in VR. Participants in the second group were asked to interact with the plate using a tracked controller. After the VR phase, the subjects were asked via a written questionnaire to make predictions about other objects present in the VE but which they never interacted with. From the elaboration of the results it came out that subjects of the first group predicted other objects of the VE to be more "real", meaning that they were perceived more solid, heavier and more affected by the gravity, compared to subjects of the second group.

This study is important because it was the first to demonstrate in an empirical way the validity of how tactile augmentation can be a smooth, inexpensive approach for adding physical texture and haptic feedback hints to virtual environments.

3.0.2 Substitutional Reality

One of the studies that inspired us the most for this thesis is the work conducted by Adalberto Simeone, Eduardo Velloso and Hans Gellersen in *"Substitutional Reality: Using the Physical Environment to Design Virtual Reality Experiences"* [12]. In this article they introduced the

concept od Substituational Reality, a virtual environment where every physical object around the user is paired to a virtual counterpart. In the study the authors present a model to substitute virtual objects with real objects with different level of discrepancy. They conducted two user studies to investigate the factors that affect the suspension of disbelief in the users and the level of engagement as the physical proxy varies.

In the first user study they used as physical proxy a real mug and then they substituted for it in the virtual environment with 10 different models belonging to different substitutional categories such as:

- Aesthetic: material alteration to also simulate a different temperature.

- Addition/Subtraction: size variation (bigger and smaller).

- Function: combination of factors that makes the object different (different material, size and features).

- Category: different shape (box and a sphere).

Users were allowed to interact with the object in the virtual environment without a specific task for one minute and then were asked to answer some questions on the physical characteristics of the object they interacted with. Full results are illustrated in the article [12], but in general the objects that users found to be breing down their "suspension of disbelief" were those with a mismatch in shape or temperature.

In the second user study, the subjects were on a virtual spaceship and their task was to hit a moving sphere with a virtual light-saber. The test was executed three times using different

physical proxies: a model of a light-saber, a torch and an umbrella. The time to completion of the task was recorded for each proxy and then the subjects were asked to answer to some questions about their engagement, preference and physical and mental exertion. Results surprisingly showed that the torch performed better and was the preferred among the objects.

That is their research quickly summarized, but this article will be often cited in our document for the guidelines that it gives in the realization of substituational reality experiences which is related to what we have done.

3.0.3 Other examples of integration of physical objects in virtual reality

In this section we will briefly present some others relevant studies into the integration of physical objects in a virtual environment.

3.0.3.1 Catching a Ball in Virtual Reality

A study conducted by Pan and Niemeyer [13] at Disney Research realized a system that allows a user to accurately catch a physical ball while in a virtual environment. They implemented and tested three different ways of visualization:

- Displaying a virtual ball which always matched the position of the real ball.

- Displaying only the predicted trajectory of the ball. A physical model was used to calculate the trajectory in real-time.

- Displaying only the target point for an under-hand catch. Using the same physical model as before a point indicating also the direction of the ball was displayed where the user had to put his or her hand.

The system was implemented using OptiTrack cameras to track the ball and Unity for the graphics. It was used to conduct a user study in order to investigate the subjects' behavior in those three different conditions.

This is not directly related to the work that we will present, but it is a good example of the potential of integrating real objects in virtual reality such as using virtual augmentation to assist users in the execution of a task.

3.0.3.2   TactileVR

The work presented by Shapira, Amores and Benavides in *"TactileVR: Integrating Physical Toys into Learn and Play Virtual Reality Experiences"* [14] is a virtual system for children aged 5–11 which uses physical objects and toys tracked in a virtual playground. Researchers purposely decided to test their system on children as they "are a tough audience for VR". Four game applications were developed and tested. The team wanted to investigate the level of enjoyment and the speed and accuracy of handling objects with tactile augmentation versus the case without tactile augmentation (just using hands, not controllers).

Overall, results showed that using tactile augmentation improved the accuracy in handling objects and the levels of enjoyment were higher than in the non-tactile experience.

3.0.3.3   Using Real Objects for Interaction in Virtual Reality

One last work that we want to mention is the one of Yoshimoto and Sasakura in *"Using Real Objects for Interaction in Virtual Reality"* [15]. This is a much simpler implementation of a system which uses physical objects in virtual reality when compared to the studied listed above. The two authors wanted to compare the play of a tower defense game using virtual objects in

VR to the same game using a mouse as input device. The developed application was running on a Samsung Gear VR HMD and the physical objects were equipped with markers similar to QR codes tracked using the device's camera. This implementation had some issues with tracking during the user studies such that users on average said that the ease of use was higher using the mouse. The interaction method which involved the physical objects was rated more enjoyable to use.

From this study we can appreciate the simplicity of the implemetation and the compactness of the system (it was contained in the headset), but we must notice that to bring real objects into virtual reality it is of extreme importance to use a reliable tracking system.

3.0.4   Training in Virtual and Real Environments

A study conducted by Kenyon and Afenya in 1995 at the University of Illinois at Chicago described in the article *"Training in Virtual and Real Environments"* [16] wanted to investigate the transfer of training between real and virtual environments using a pick-and-place task. The virtual environment system in this case was not a head mounted display, but the original CAVE [17], a projection based system where the user is surrounded by four screens in a 10-ft cube and the illusion of 3D is given by providing to each eye a different image using stereo shutter glasses synchronized with the projectors.

The task had two levels of difficulty and consisted in minimizing the time to move cans between two rows using color coded locations. In the first level of difficulty the origin and destination targets were aligned on the two rows while in the second they were randomly placed.

For both the tests in real and virtual environments the magnetic tracking system of the CAVE was used to record users' head and hand position. In the VE the virtual cans were grabbed using a *"Grasper"*, a device of shape similar to a can with a switch that had to be pressed and held to grab and move the virtual object. The position of the hand was displayed as a cube in the virtual environment and that visual was the only kind of feedback provided to users. The experimenters used two ways to display the cursor (attached: same position, detached: with some offset). The tracking system suffered of a 150 ms delay in displaying the position in the virtual environment and that in the end influenced the execution of the tasks.

The experiment consisted in dividing subjects in four groups, half of those were trained in the virtual environment and tested in the real one. The other half was trained in the real environment and tested in the virtual one. Both the training and the test consisted in 30 trials of the same task.

From the analysis of the results it came out that the training of a task in the virtual world can improve the execution of the same task in real world. But the significance was not constant during the experiment. The researchers hypothesized that the results were not so robust because of the difficulties in producing a sensory parity between the virtual environment and the real world. The sensory mismatch in this test of speed influenced the transfer of training between the two worlds. Also, it was noticed that in the virtual world the movements were more deliberate. That was due mainly to "poverty of the sensory feedback available in that environment" [16] and to the 150 ms delay of the system in visualizing the hand position which induced users to adjust their control to match the system dynamics. No significant transfer of training was found in

the performance improvement for subjects trained in the virtual world and tested in the virtual environment. That means that any gain received with the training was voided by the need to adapt to the virtual world different sensory and interaction system.

This research is important to our study because it empirically demonstrated that transfer of training from a virtual world to the real world is possible and this is what we want to achieve with physical rehabilitation in VR. It also highlights the importance of reducing the sensory mismatch between the real world and the simulation in the virtual environment. This work was conducted in 1995 with state-of-the-art devices for the time. In more than twenty years a lot of technological advancements have been made and nowadays the available computational power makes possible to considerably reduce the latency of tracking devices and render some more realistic graphics. We will take advantage of these technologies to reduce the sensory mismatch in our system.

CHAPTER 4

IMPLEMENTATION

In this chapter we will illustrate the system that was built for this research. Based on the related literature we decided to test four different methods of interaction in virtual reality. These are:

- Bare Hands Interaction

- HTC Vive controllers

- 1:1 real object replica mapped in VR

- Different object replica mapped in VR

The reasons of the choice of these specific methods will be explained in the next sections. We created several VR applications in Unity for the HTC Vive head mounted display integrating the listed interaction methods. In this chapter we will first deal with the Unity applications and then we will discuss the interaction methods used and how they have been implemented in the applications.

4.1   Unity Applications

To conduct a study on the chosen interaction methods we had to create some test applications with Unity. The purpose of these applications is to collect data in order to quantitatively compare the execution of tasks with different interaction methods. The applications were developed using the game engine *Unity 2017.3.0f3*.

Three applications were created. The first one consists of the test of the Fitts' Law in VR. The second and the third are based on the manipulation of objects to achieve simple everyday tasks: one consists in making a bowl of cereal and milk, and the other to move objects from point A to point B.

Next follows a detailed description of the implementation of the scenes.

4.1.1   Fitts' Law Test
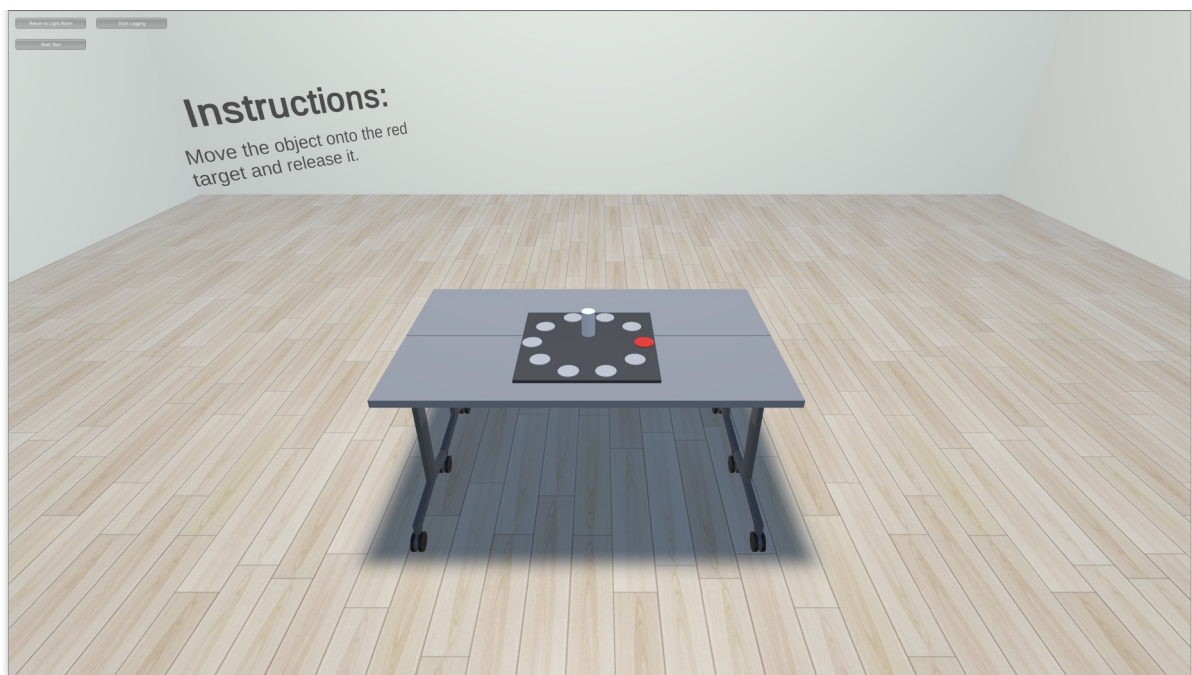
Figure 4: Screenshot of the application used for the Fitts' Law Test

This first scene is set in a empty room, whose design is kept minimal on purpose to avoid distractions for the user. The only furniture present is composed of two tables in the center of the room. On the tables there is a darker square on which all the actions of this scene are focused. We created a target acquisition task that follows the recommendations of Soukoreff and Mackenzie [18] to test Fitts' Law [19]. The recommendations are for a 2D pointing task, not 3D and so during the analysis of the collected data we had to proceed with a different strategy rather than the classical approach. Fitt's Law is one of the foundations of Human Computer Interaction (HCI) and describes the relationship between the time needed to acquire a target, its width and distance. It can also be used to compare different input devices. In our applications there are 10 circular targets arranged around the circumferance of a circle. The recommendations suggest varying the width and the distance between the targets. We proposed to the user 9 unique combinations of width and distance:

- Widths (diameter): *5.5 cm, 8.0 cm, 12.0 cm*

- Distances: *32.0 cm, 45.0 cm, 50.0 cm*

On a table in the scene there is a white cylinder of 5.5 cm of diameter. Our pointing task consists of the user grabbing the cylinder, moving it to the target highlighted in red and then releasing it. We created a script which manages the interaction. The script is attached to this document in A. Its main functions are to:

- *Create targets*: instantiate the 10 target of the selected width and distance.

- *Highlight next target*: when the user places the cylinder on the red target and releases it, a confirmation sound is emitted and the target diametrically opposed is highlighted.

- *Manage target sets*: once the user has moved the cylinder on all the targets of the set, another combination of widths and distances is loaded and the targets are recreated. The order execution of the set is randomized so that to each user is presented with the same combinations but always in a different order to avoid biases.

- *Log data*: the program records, for each move, the time from when the cylinder leaves the start target to when it is released on the end target, the width and distance of the target, and a timestamp of when the cylinder is released on the target. The data is exported in a CSV file.

In addition, we created a data logger to record the positional and rotational data of certain GameObjects (such as the user's head and hands and the cylinder) and export it in a CSV file.

4.1.2 Breakfast Time

This scene is set in a virtual kitchen (Figure 5). The user is located in front of a table on which are present three models:

- A milk box

- A bowl

- A box of cereal

Figure 5: Screenshot of the application set in a virtual kitchen

The kitchen model comes from the Unity Asset Store [1]. We created the other three models using the 3D modelling software *StetchUp* [2]. The task in this application was to make a bowl of milk and cereal. The user was requested to move the bowl to the middle of the table and then pour one ingredient a a time. For this we needed a fluid simulator to represent the milk and a particle

---

[1] *Studio Krokidana* – Kitchen Creation Kit https://assetstore.unity.com/packages/3d/environments/kitchen-creation-kit-2854

[2] https://www.sketchup.com

generator for the cereals. We decided to use the *Obi Fluid*[1] plugin for Unity. With that it is possible to create physically realistic fluid simulations. *Obi Fluid* provides a particle emitter, a solver, which deals with the physics and the interaction of particles, an element to enable Unity colliders to interact with Obi particles, and a renderer to render the fluid surface between the particles. Obi allows the customization of physic characteristics of the fluid material that we want to use (e.g smothness, surface tension, etc.). The computation is entirely done on the CPU for each particle and sice the workload on the computer is already intensive during the execution of a VR aplication, we decided to use a low-fidelity fluid material for the milk which does not take into consideration several physical features to avoid drops in the framerate. For the cereal we used always Obi, but we set the material to be granular and not be rendered as a fluid. Finally, we placed the particle emitters on the emission point of the milk and ceral boxes with a script that regulates the emission speed based on the rotation of the object so when the user rotates the object in order to pour milk or cereal the corresponding particles are emitted (result shown in Figure 6) and a sound effect of the content is reproduced. For our experiment we limited the number of particles emitted to be just right to fill the bowl so that the criterion to judge if the task was achieved would always be the same. When all the particles (milk and cereal) are in the bowl the task is considered accomplished and the time spent to do it is recorded (Note: time starts when the user touches an object). Also in this application we

---

[1]https://assetstore.unity.com/packages/tools/physics/obi-fluid-63067

used a data logger to record the positional and rotational data of the objects and the user's hands and head and export them in a CSV file.
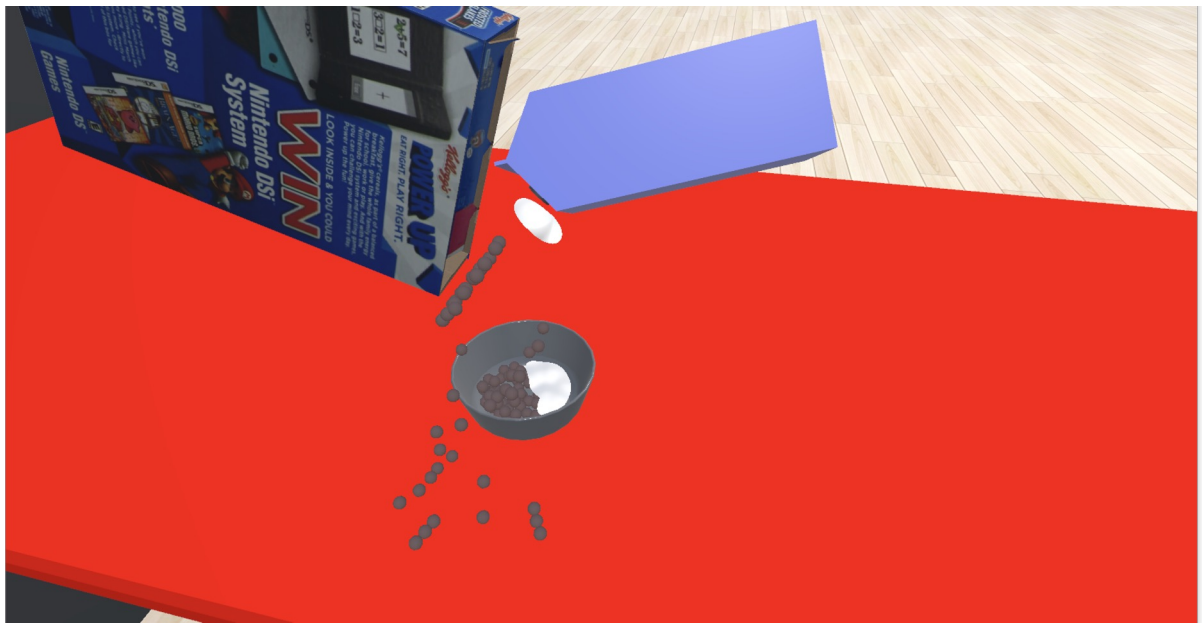
Figure 6: Screenshot of the pouring of milk and cereals in the bowl

4.1.3   Colored Objects

The third application is set in a room with three tables arranged as horseshoe around the user (Figure 7). On each table is placed a colored pad (black, blue and white) and on the right-hand side of table in front of the user are placed five objects of the color of one of the pads:
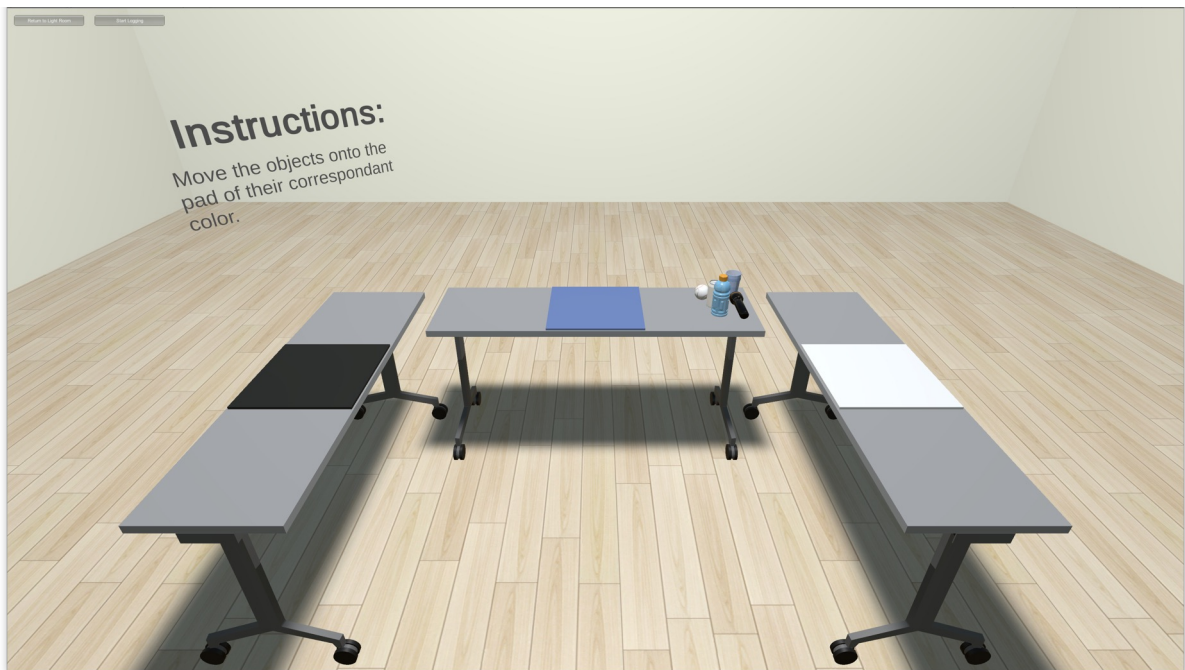
Figure 7: Screenshot of the application used for the Colored Objects scene

- Soda can (*white*);

- Baseball (*white*);

- Plastic bottle (*blue*);

- Plastic glass (*blue*)

- Flashlight (*black*);

Here the task consists in simply moving all the objects onto the pad of their corresponding color and then to bringing them back to their original location. We created a script to manage the interaction and keep track the objects position. We put trigger colliders on the pads and on the

starting area to dynamically start and stop the logging of their positional and rotational data. As a confirmation of a correct move, a sound effect was played when the object was released in the correct location (if it was release on a wrong location, a different sound was played). Also in this application, in addition to the objects data, we used an additional data logger to record the positional and rotational data of the user's hands and head, and export it in a CSV file.

4.2   Interaction Methods

In this section we will talk about the interaction methods that we have chosen to compare in this study. For each case we will explain our motivations for the choice and we will illustrate how it has been implemented in details.

4.2.1   Bare Hands Interaction

To better understand the implications of having haptic feedback while interacting with virtual objects in virtual reality, we thought that it was important to study the behavior of the users in a condition where feedback was absent. For this reason, we used the Leap Motion, an hand tracking camera mounted in front of the HTC Vive (Figure 8).   The device is composed mainly of two cameras and three IR LEDs. Some part of the elaboration is done on the raw sensor data direclty on the device, but then the image data is streamed to the computer where the *Leap Motion Service* software processes it through computer vision algorithms to reconstruct a 3D representation of the user's hand (Figure 9). The main drawback of this device is the limited field of view. Figure 10 shows the area in which hands are recognized correctly. Given that the device is attached in front of the HMD, users have to always keep their hands in front of their face. That could be a problem, for example, when they have a

Figure 8: HTC Vive HMD with the Leap Motion device installed on it

virtual object in their hands and then they move them out of the field of view, the system loses tracking and the object falls on the ground.

Leap Motion provides an SDK for Unity that we used to integrate this technology in our applications. The SDK components that are relevant to our work are two modules:

- **Hands Module**: it provides some sets of rigged hand models optimized for the use in virtual reality.

- **Interaction Engine**: it manages the physics and the interaction of virtual hands with virtual objects. It is, for instance, the component which recognizes the hand gestures (e.g. the grab gesture).
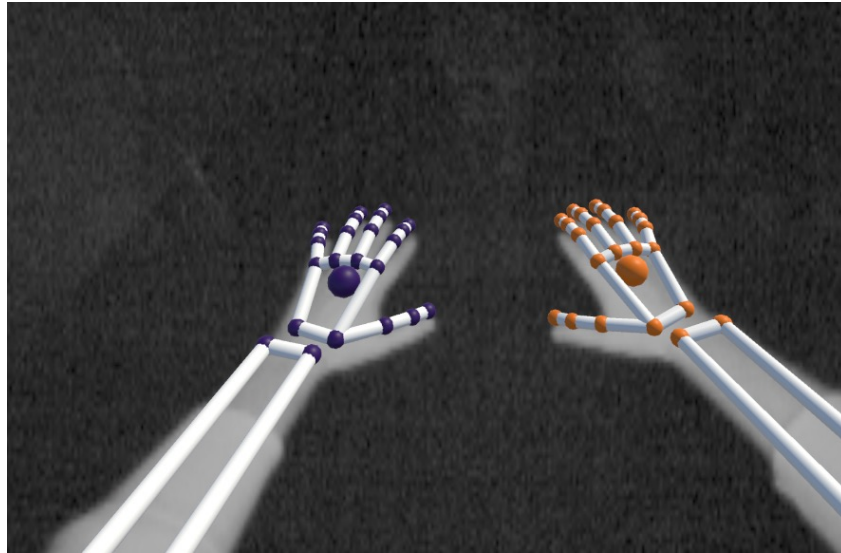
Figure 9: Screenshot of what Leap Motion sees and 3D reconstructed hands on overlay

Interaction Area
2 feet above the controller, by 2 feet wide on each side
(150° angle), by 2 feet deep on each side (120° angle)
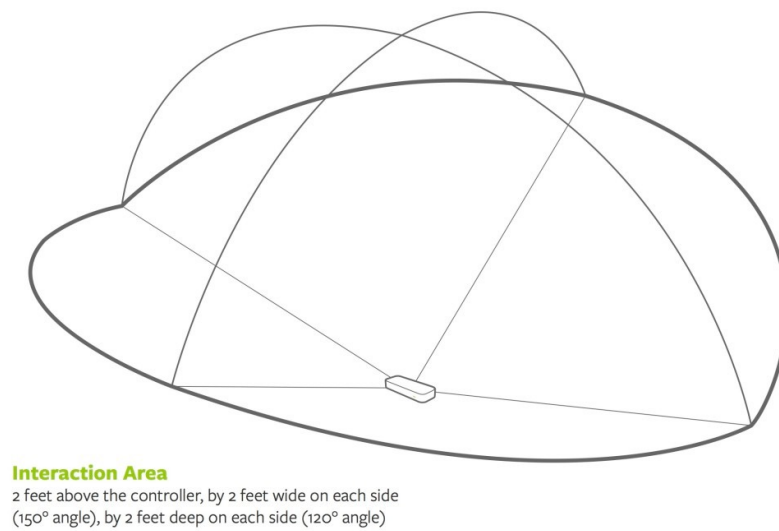
Figure 10: Leap Motion camera field of view. The device is here shown unmounted from the HMD, but once mounted on the front the long side is parallel to the ground – [20]

In our implementation we used the *Hands Module* for the rendering of a pair of hand models and the *Interaction Engine* for the interaction with virtual objects.

The SDK provides a Unity prefab called *LMHeadMountedRig* which automatically manages the configuration and the calibration of the HTC Vive with the Leap Motion device and the connection to the *Leap Motion Service* provider. The other prefab that we used is the *Interaction Manager* which handles all the internal logic that makes interactions possible. We need one of each for every scene. To make objects interactable with the *Interaction Engine* we had to attach the `InteractionBehaviour.cs` on eah of them. This script requires that on the object there is a Unity Rigidbody and at least one Collider. In the GameObject's inspector we set the approximate object's weight for a realistic physic simulation under the Rigidbody and under the *Interaction Behaviour* we set the *Grasp Movement Type* as "Nonkinematic", which means that the interaction object receives "a velocity and angular velocity that will move it to its new target position and rotation on the next physics engine update, which allows the object to interact and collide against other objects in the scene before reaching its target grasped position" [21]. In this way we integrated the Leap Motion system in our work, making possible to interact in a virtual environment with our bare hands (from here BHI: Bare Hands Interaction).

4.2.2 HTC Vive Controllers

The fastest way to develop an interaction in virtual reality is to use the devices that are already included in the head mounted display set that has been chosen to use. In our case it was the HTC Vive, which comes with two controllers (see Figure 11). The tracking of the controller is managed directly by the HTC Vive system and its functioning has already been

Figure 11: One of the two controllers included in the HTC Vive system

explained in 2.2.2. To integrate the controllers in our application we used the SteamVR Unity plugin created by Valve software [22]. This plugin, available in the Unity Asset Store [23], is an SDK that allows developers to target a single interface that works with all principal VR headsets (e.g. Oculus Rift, Samsung Gear VR, etc.). Although the SDK provides a wide range of functionalities to build VR games/applications, for this interaction method we just needed a few components.

The main component is the Unity prefab *CameraRig*. It is composed by three elements that are automatically connected to the corresponding components of the HTC Vive:

- *Camera (head)*: mapped to the HMD position, renders the stereoscopic camera image to display in the headset.

- *Controller (left)*: mapped to the left Vive controller.

- *Controller (right)*: mapped to the right Vive controller.

We designed our interaction method such that to grab a virtual object the user has to make the controller collide with the object and then press the Trigger button on the controller. Vive controllers also have a haptic actuator which consists of an Eccentric Rotating Mass Motor (ERM). This actuator is controllable via script with the SDK. Applying the concepts acquired from [24], we decided to use the haptic vibration to provide a vibrotactile feedback to the user when the controller makes contact with a virtual object.

Implementation

The implementation consist in just one C# script attached to both the controllers. The code comes in part from the Eric Van de Kerckhove's guide [25]. The code is explained next:

```
1  /*
2   * Copyright (c) 2016 Razeware LLC
3   *
4   * Permission is hereby granted, free of charge, to any person obtaining a
         copy
5   * of this software and associated documentation files (the "Software"), to
         deal
6   * in the Software without restriction, including without limitation the
         rights
7   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8   * copies of the Software, and to permit persons to whom the Software is
9   * furnished to do so, subject to the following conditions:
10  *
11  * The above copyright notice and this permission notice shall be included
         in
12  * all copies or substantial portions of the Software.
13  *
14  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
         OR
15  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
         THE
17  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

```
18   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
        FROM,
19   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20   * THE SOFTWARE.
21   */
22
23   using UnityEngine;
24
25   public class ControllerGrabObject : MonoBehaviour
26   {
27       private SteamVR_TrackedObject trackedObj;
28
29       private GameObject collidingObject;
30       private GameObject objectInHand;
31       public bool vibration = true;
32
33       private SteamVR_Controller.Device Controller
34       {
35           get { return SteamVR_Controller.Input((int)trackedObj.index); }
36       }
37
38       void Awake()
39       {
40           trackedObj = GetComponent<SteamVR_TrackedObject>();
41       }
42
43
44       public void OnTriggerEnter(Collider other)
45       {
46           SetCollidingObject(other);
47       }
48
49       public void OnTriggerStay(Collider other)
50       {
51           SetCollidingObject(other);
52       }
53
54       public void OnTriggerExit(Collider other)
55       {
56           if (!collidingObject)
57           {
58               return;
59           }
60           collidingObject = null;
61       }
62
63       private void SetCollidingObject(Collider col)
64       {
65       if(vibration  && !objectInHand){
66         Controller.TriggerHapticPulse(600);
```

```csharp
67          }
68              if (collidingObject || !col.GetComponent<Rigidbody>())
69              {
70                  return;
71              }
72              collidingObject = col.gameObject;
73          }

75      void Update()
76      {
77              if (Controller.GetHairTriggerDown())
78              {
79                  if (collidingObject)
80                  {
81                      GrabObject();
82                  }
83              }

85              if (Controller.GetHairTriggerUp())
86              {
87                  if (objectInHand)
88                  {
89                      ReleaseObject();
90                  }
91              }
92      }

94      private void GrabObject()
95      {
96          objectInHand = collidingObject;
97          collidingObject = null;
98          var joint = AddFixedJoint();
99          joint.connectedBody = objectInHand.GetComponent<Rigidbody>();
100             if (objectInHand.GetComponent<FittsLawObject>())
101             {
102                 objectInHand.GetComponent<FittsLawObject>().GraspObject();
103             }
104     }

106     private FixedJoint AddFixedJoint()
107     {
108         FixedJoint fx = gameObject.AddComponent<FixedJoint>();
109         fx.breakForce = 20000;
110         fx.breakTorque = 20000;
111         return fx;
112     }

114     private void ReleaseObject()
115     {
116             if (GetComponent<FixedJoint>())
```

```
117         {
118             GetComponent<FixedJoint>().connectedBody = null;
119             Destroy(GetComponent<FixedJoint>());
120             objectInHand.GetComponent<Rigidbody>().velocity =
                    Controller.velocity;
121             objectInHand.GetComponent<Rigidbody>().angularVelocity =
                    Controller.angularVelocity;
122             if (objectInHand.GetComponent<FittsLawObject>())
123             {
124                 objectInHand.GetComponent<FittsLawObject>().ReleaseObject();
125             }
126         }
127         objectInHand = null;
128     }
129 }
```

On each controller are placed two *Box Colliders*, as shown in Figure 12, which act as triggers. Then, from the script listed above the functions `OnTriggerEnter (Collider)`, `OnTriggerStay (Collider)` and `OnTriggerExit (Collider)` are executed when a *GameObject* with a *Rigidbody* and a *Collider* respectively enters, stays or exits the *Trigger Collider* on the controller. The script keeps track of the objects that are colliding with the controller at any time. If the user presses the trigger button while the controller is colliding with an object, then a joint between it and a controller is created and the object appears as grabbed to the user. The code has been further modified in the Fitts' Law test scene, where the status of the object (i.e. grabbed or released) is updated on the FittsLawObject script attached to that object (see line 100 and 122). The vibrotactile feedback is created with the code at line 65. When the controller collides with an object and it does not have anything attached, the method `SteamVR_Controller.Device.TriggerHapticPulse (ushort durationMicroSec)` is called with a duration value of 600 ms. When the user presses the trigger, the vibration is suppressed as it could result annoying and less realistic.
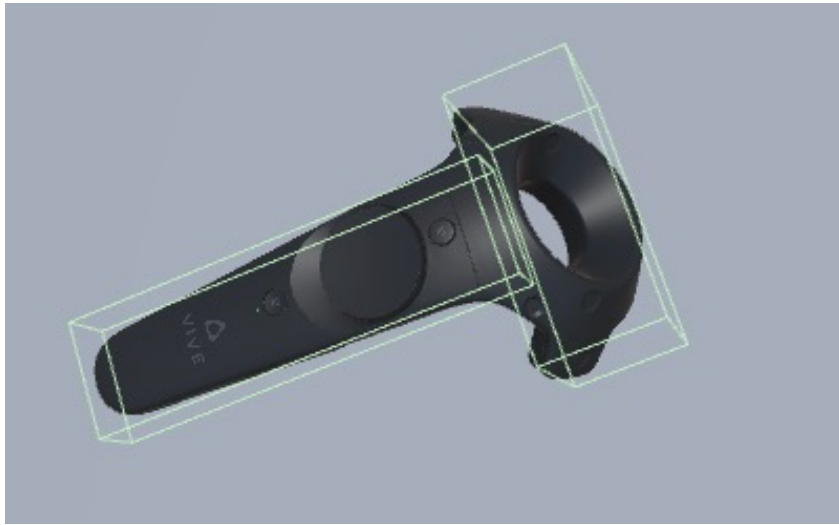
Figure 12: Screenshoot of Unity showing the position of *Box Colliders* on a Vive controller model

4.2.3 <u>Real Objects</u>

The passive haptic feedback on which this research is mainly focused consists in using real physical objects mapped to their virtual counterparts. As discussed earlier in this document (see 3.0.1) there are studies that showed how the use of real-world objects can enhance the realism of a Virtual Reality experience. In this research we wanted to go further along the concept of the so called "Substitutional Reality" [12]. We wanted to investigate the response of users as the physical proxy for a virtual object changes in different tasks of object manipulation. The reason we did that is to try and understand which physical characteristics affect certain parts of an interaction. We therefore mapped to the virtual objects two different sets of physical proxies, the 1:1 replica and some different objects with variant levels of similarity.

Next we will first discuss two ways we could have implemented the tracking of real-world objects and then we will explain in detail our work.

4.2.3.1 HTC Vive Trackers

To implement the tracking of real objects we took into consideration two commercially available systems:

- HTC Vive Trackers

- OptiTrack™ optical tracking

The Vive Tracker is an optional device part of the HTC Vive system. It uses the same tracking technology as the HMD and the two controllers and it connects wirelessly to the PC with a USB dongle (see Figure 13). The cost of each apparatus is around $ 99 [26].

From the point of view of the developer, to associate a virtual model to the tracker is rather straightforward. The SteamVR Unity Plugin manages all the low level part and returns a *GameObject* to which it is possible to attach an object.

Although the use the use of this device would have eased the workload of mapping virtual objects in VR, we decided to opt for another technology for several reasons. First, with its almost 10 cm of diameter and 4 cm of height it is a modestly bulky device to be attached to some small objects. Second, its weight of 89 grams make it less discreet than other technologies. And finally, in our applications we needed to track a reasonable number of objects at the same time (around 10) and we decided that it was not appropriate to invest in a "closed" tracking system (i.e. the HTC Vive) when we had available another solution in our laboratory.

Figure 13: The Vive Tracker with its USB dongle

4.2.3.2   OptiTrack™ optical tracking

The other technology that we took in consideration is positional optical marker-based tracking implemented with OptiTrack cameras. Positional tracking is the detection of objects (called *rigidbodies*) within Euclidean space [27]. It detects the translational movements and the rotation (pitch, jaw and roll) and it is able to provide the exact postion of a rigidbody. An optical tracking system is composed by three main components:

- Tracking devices (such as IR cameras)

- Computer vision algorithms

- Objects to track.

This technology is based on the same concept as stereoscopic human vision. As we can tell the distance of an object using our binocular vision, in a similar way, using an array of cameras, it is possible to retrieve the position in a calibrated space of some markers placed on an object using certain algorithms. There are two different types of optical tracking:

- Outside-in tracking: the sensors are stationary and tracked objects are equipped with markers detectable by the sensors. This the case of the motion capture system that we used in our work, but it is present in other VR devices such as the Oculus Rift and the Playstation VR [27].

- Inside-out tracking: the sensor is placed on the tracked object and the markers are placed in fixed location. This technique is less used, but it is in some ways similar to the one used in the HTC Vive, where the sensors (not cameras) are on the the tracked devices and detect an IR beam.

Optical tracking systems like the OptiTrack offer a high precision in positional detection: with an optimal camera configuration and capture volume size it is possible to achieve a submillimiter accuracy, and, contrary to the HTC Vive tracking system, the position of each tracked device is determined at each instant of time. The main problem associated with this technology is the cost, a single camera costs around $ 2,499 [28], and for a correct tracking we need at least 3 of them, plus the cost of a PoE network switch and a dedicated server to run the software for the tracking. On the other hand, this system uses retro-reflective markers which are lightweight and very small small compared to the the HTC Vive Trackers. The system can track more than 25 rigidbodies at the same time and the cost of retro-reflective markers is rather low, so once

we have the system set up, it is basically costlessly "scalable" in the sense that we can add as many rigidbodies as we want with a little expense. An optical tracking system is open to a large set of applications, not only for object tracking in VR. These systems are used for example also in Movement Science to analyze humans movements, in Computer Animation to capture the motion and apply it to a virtual model and in Robotics to track drones and industrial robots. Ideally, if a similar system was used for physical rehabilitation applications in VR, it would be possible to attach additional markers to the patients' body to record the position of a certain limb and evaluate their movements.

We decided to use this technology as it was already available in our laboratory, and mainly beacuse of its precision and the possible high number of rigidbodies that can be tracked at the same time.

4.2.3.2.1   Implementation

As we just said, an optical tracking system is composed by three main components, in the following section we will discuss how we implemented each component in our system. The schema in Figure 14 shows how the components are connected together. A line in the schema represent a Cat6 Ethernet cable.

Tracking devices

We developed our applications in room 2068 of the ERF building at UIC. The room was equipped with 24 OptiTrack Prime 13W cameras, anyway, for our work we just used 13 of them. The Prime 13W is a camera optimized for low-latency high precision tracking for compact spaces. It has an ultra-wide field of view (82° x 72°) and 10 ultra high power 850 nm IR LEDs with a
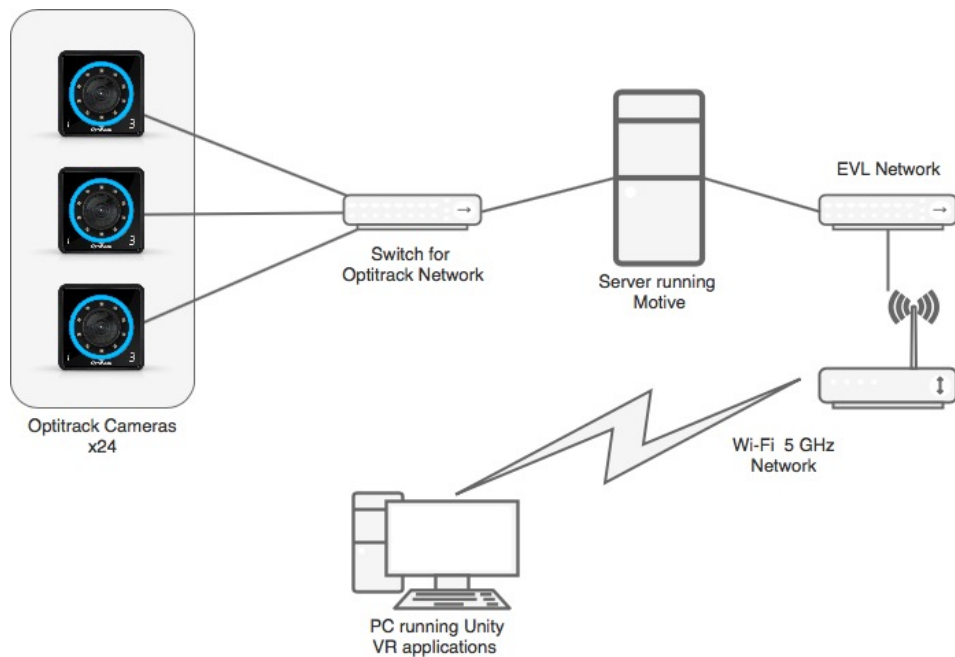
Figure 14: Schema of the connections of the system components

low distortion 3.5mm lens with 850nm band-pass filter. The images are captured at 240 FPS at the resolution of 1.3 Megapixels [28]. The image processing is executed on the camera to reduce the workload of the server, so it recognizes the markers and sends to the server the data of their position.

The cameras are mounted on the ceiling at a heigh of 2.70 meters (to verify). We rearranged the cameras in order to obtain an optimal tracking in the area calibrated for the HTC Vive. Figure 15 shows the arrangement of the cameras and their position with respect to the Vive calibrated area. The cameras have a Ethernet/PoE port and are so powered via a Cat6 Ethernet cable from a PoE switch. Due to particularly high bandwith requirements the OptiTrack
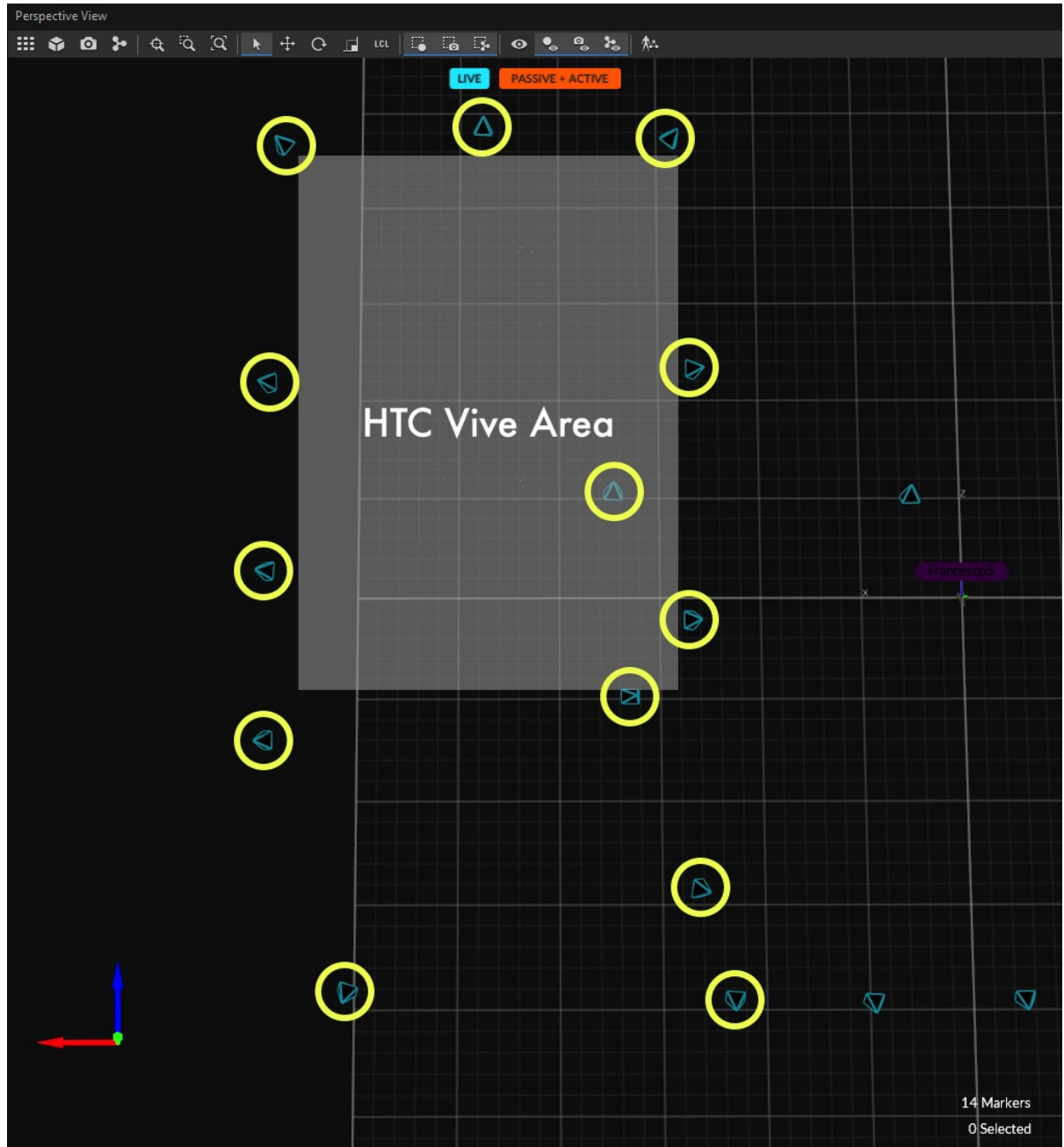
Figure 15: Screenshot of Motive showing the top-view arragement of the cameras (circled) and the HTC Vive area

network is separated fom the LAN. As illustrated in Figure 14, a server is connected to that network with two network interfaces which runs the software for the tracking system.

The software

Motive is the software developed by OptiTrack for managing the optical tracking system. It comes in two versions: *Motive:Tracker* and *Motive:Body.* The former is used for the tracking of objects only, while the latter has some more advanced functionalities for the tracking and resolution of body skeletons. For our case *Motive:Tracker* was the right tool. For our work, Motive manages four important parts of the system:

- Cameras calibration.

- Rigidbody creation.

- Rigidbody tracking.

- Data streaming.

In order to track the objects all of the cameras have to be calibrated. The calibration calculates camera position and orientation in addition to the lens distortion in the images. The calibration results are used by Motive to create a 3D capture volume within which the objects are tracked. The calibration process consists of three steps:

1. Masking: during this phase it is possible to mask from the detection area some reflections or lights that otherwise would be recognized as markers.

2. Wanding: this phase consists in walking around in the capture volume waving a special wand (CW-500 calibration wand) in order to allow the cameras to collect samples. Once

(a) Person performing the wanding calibration

(b) Tool used to set the ground floor and the origin

Figure 16: Tools used for OptiTrack calibration

enough samples are collected, Motive calculates cameras position, orientation and optical distortion. Figure 16a shows the process of wanding.

3. Ground Plane setting: with this last step is set the origin and the direction of the axis of the tracking area. It is done by placing a triangular calibration tool in the desired origin (Figure 16b). In our case we set the center of the room as origin and we aligned the X and Z axis parallel to the walls.

In Figure 17 we can see how the residual mean error of the triangulation is on the order of submillimiters, this means that the tracking based on this calibration is highly accurate, at least for our use. Once the system is calibrated, Motive is able to do the *Reconstruction*, namely
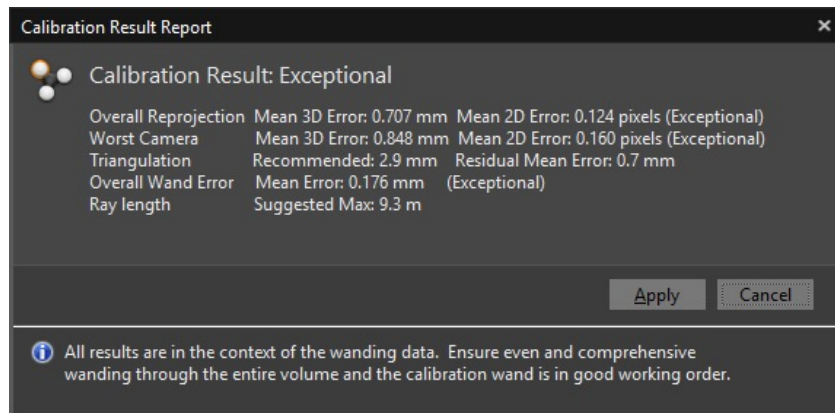
Figure 17: Motive calibration results

the process of deriving 3D points from 2D coordinates aquired by multiple synchronized images. The position of each detected marker is triangulated frame-by-frame [29].

Using the reconstructed markers, it is possible to register rigidbodies into the software. For doing that, the object has to be placed within the calibrated volume and equipped with at least 3 retro-reflective markers (7 is the maximum number supported by Motive). Then, the markers are selected from the interface and their unique spatial relationship is associated with an asset, and then it is possible to assign a name and a streaming ID (this will be discussed in the next paragraph) and other parameters that we are not going to discuss here.

Motive uses proprietary algorithms to solve the rigidbodies present in the capture volume. Using rigidbodies composed of at least three markers, the software can calculate their position with 6 degrees of freedom (6DoF). The maximum number of objects that it can solve for at the same time is 32 according to the documentation[30].
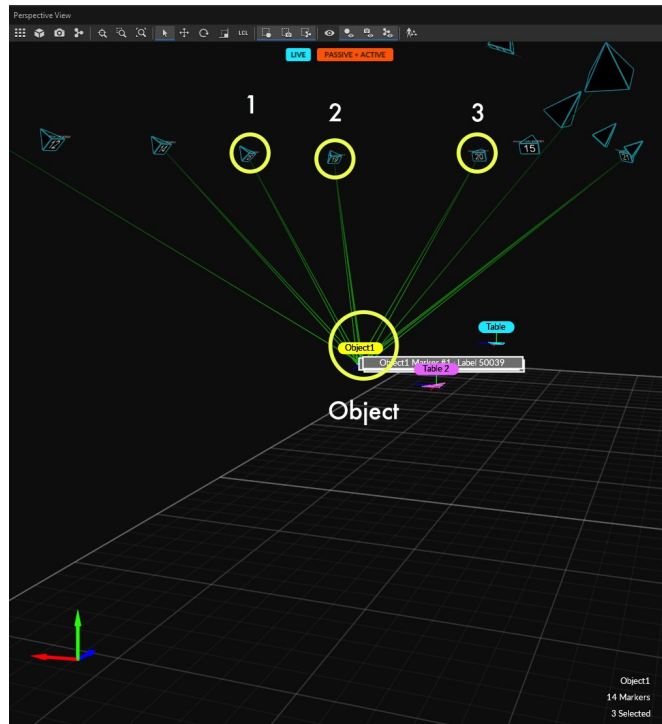
(a) Screenshot of Motive showing a tracked object and the tracking rays from the cameras
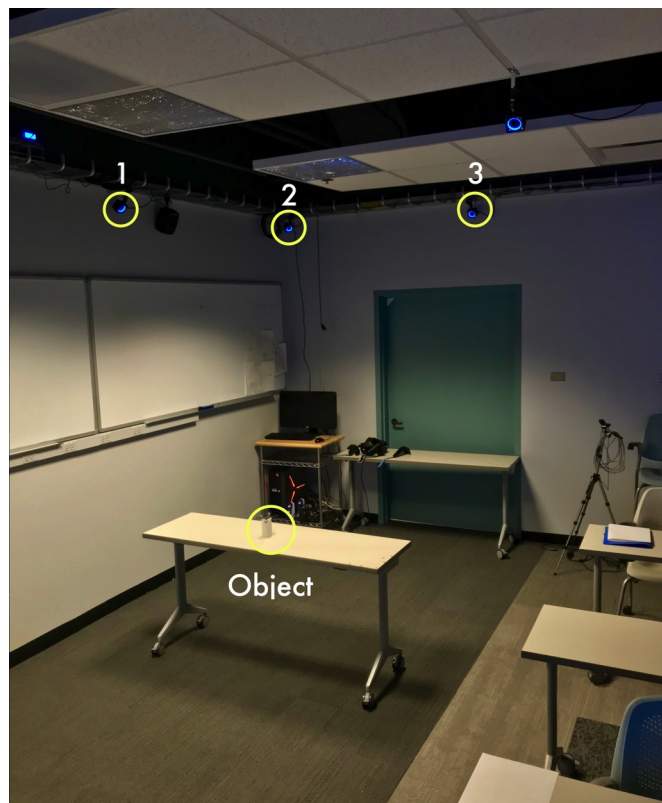
(b) Same object in the real world

Figure 18: A tracked object as seen by Motive and in the real world. Cameras visible in (b) are highlighted in (a)

We set the preference for the tracking algorithm to *Auto-Select*, meaning that the software automatically chooses the algorithm between:

- Marker Based: it uses just the 3D reconstructed coordinates to calculate position and rotation of the rigidbody.

- Ray Based: in addition to the marker based, it uses the geometry of the registered rigidbody and position data that is less accurate (because over the error threshold) when a marker is partially occluded to estimate the position and rotation of the rigidbody.

In Figure 18a shows how an object is seen by the optical tracking system and in Figure 18b shows the object in the real world setting. t Finally, Motive manages the streaming of positional and rotational data of the rigidbodies. Data is streamend using the NatNet protocol on the second network interface, the one connected to the laboratory LAN. The PC running VR applications connects to Motive with Point-To-Point Unicast to retrieve the data.

Latency

The OptiTrack system is optimized to provide a real-time 3D tracking. However, there are some inevitable latencies due to data transmission an processing. Figure 19 shows the latency components of the tracking system. The *System Latency* is the total time elapsed from when the cameras expose to when the data is fully solved and ready to be streamed to the client. The *Software Latency* is the time taken by Motive to process each frame of data and it is contained in the *System Latency*. The *Transmission Latency* is the time difference between when the network package is streamed out by Motive to when it reaches the client application (in Unity) and it depends by the network infrastructure.

Motive automatically calculates the System Latency. During our tests it maintained an average value of 6.5 ms with occasional peaks of 9.0 ms. To obtain a rough estimation of the Transmission Latency we run several ICMP Echo requests (i.e. Ping) from the PC that was running the Unity application to the server running Motive. With 32 bytes packets the Round Trip Time was 1.75 ms with a maximum value of 4 ms. The time needed by the tracking data packets is actually lower than those values as they are transmitted using the UDP protocol which does not require the host to confirm reception of the packet.

With this data we can estimate an average total latency of 8.25 ms and a maximum latency of 13 ms.
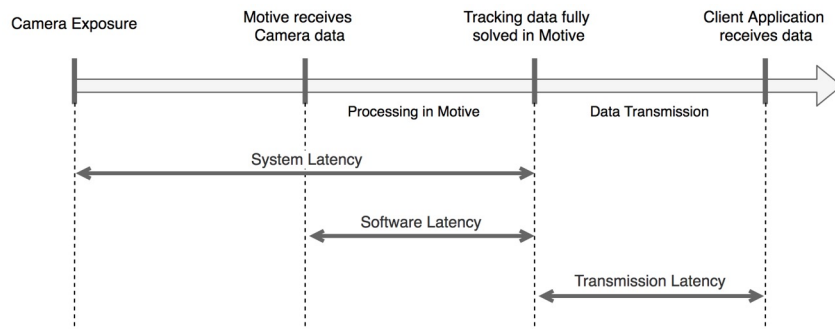

Figure 19: Diagram showing the latency components of of the OptiTrack system

Tracked objects

The last component of the system is the tracked objects. As we said before to achieve a 6DoF tracking, a rigidbody has to be composed of at least three markers. OptiTrack cameras can track any retro-reflective surface able to reflect IR light back to its source and detected by the sensor. There are different types of passive markers. In our implementation we used three of them which are displayed in Figure 20. The most important factor in optical tracking is the marker placement. Each tracked object needs to have an unique marker arrangement and to achieve a 6DoF capture, markers must be placed asymmetrically.

For bigger objects we used a rigidbody marker base (Figure 20b) with which is possible to easily achieve an unique and asymmetrical marker arrangement by mounting spherical markers (at least 3) on the 6 posts creating different combinations. That solution could be bulky for smaller objects and could interfere with interaction. For this reason, we attached some spherical markers (Figure 20a) directly onto those objects using velcro adhesives. Here caution had to be made in order to avoid to create an arrangement that is too similar to rigidbodies already registered. For other objects that were more stationary and presented a more uniform surface (e.g tables) we decided to use flat markers (Figure 20c). This last kind comes in the form of stickers to be attached on the surface, their tracked range of motion is limited when compared to tracking using fully spherical markers. However, we have used them effectively with tables and as additional tracking support for some other objects on which we applied additional tracked markers (see Figure 26b).
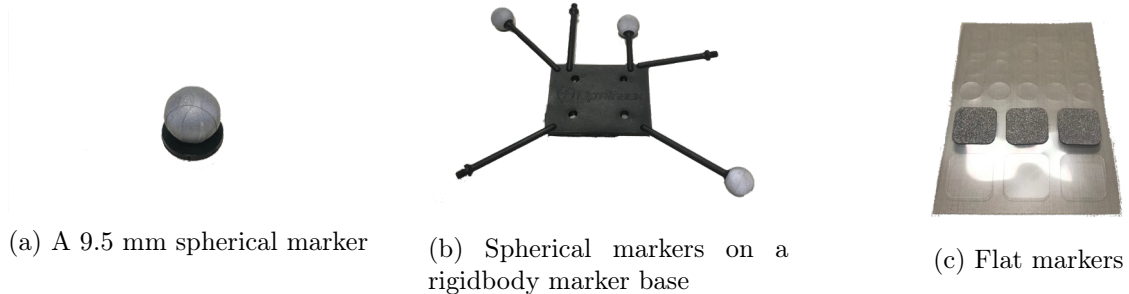
(a) A 9.5 mm spherical marker

(b) Spherical markers on a rigidbody marker base

(c) Flat markers

Figure 20: Different kinds of retro-reflective markers used

Integration with Unity

OptiTrack provides a Unity plugin to connect to Motive and receive tracking data within the application. The components of the plugin that are relevant to our work are two scripts:

- `OptitrackStreamingClient.cs`: it manages the connection to the Motive server. It has to be attached to just one GameObject in the scene. One option that it provides is the rendering of the reconstructed markers in the scene, that has been useful during the calibration and registration of objects in the scene.

- `OptitrackRigidBody.cs`: it retrieves the positional and rotational data of a certain rigidbody (whose Streaming ID is specified in the inspector) and applies it to the GameObject on which it is attached.

Calibration of tracked objects with the HTC Vive

The HTC Vive and OptiTrack are two separate systems, each one is calibrated in a different way and has its own reference space with a different origin. To synchronize the two reference

systems we set the position of the Vive CameraRig to the origin in world space (0, 0, 0). Then, we created a GameObject (named *RealObjects_1:1* in Figure 21) whose children are all the tracked objects. In this way, changing the position of *RealObjects_1:1*, also the position of its children changes.
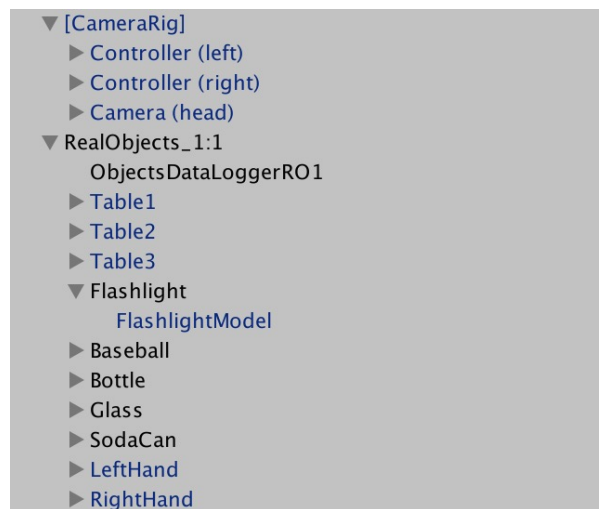
Figure 21: Unity hierarchy of GameObjects for OptiTrack tracked objects

Another problem we had to face was that sometimes we needed to re-calibrate the cameras and doing so the origin of the system could change by several millimeters and it was necessary to recalibrate every scene. So we created a Unity scene just for the calibration which saves the calibrated Transform values of the GameObject parent of the tracked objects in the *PlayerPrefs*,

a file accessible by all the scenes in the Unity project. The code of our simple script is listed here:

```
1  public class OptiTrackObjectsCalibrator : MonoBehaviour {
2    public bool saveCalibration = false;
3    void Update () {
4      if(saveCalibration){
5        saveCalibration = false;
6        PlayerPrefs.SetFloat("C_x", transform.position.x);
7        PlayerPrefs.SetFloat("C_y", transform.position.y);
8        PlayerPrefs.SetFloat("C_z", transform.position.z);
9        PlayerPrefs.SetFloat("C_rx", transform.rotation.eulerAngles.x);
10       PlayerPrefs.SetFloat("C_ry", transform.rotation.eulerAngles.y);
11       PlayerPrefs.SetFloat("C_rz", transform.rotation.eulerAngles.z);
12     }
13   }
14 }
```

Then, on the parent of the tracked objects in the other applications this script is attached to retrieve the calibrated values:

```
1  public class OptiTrackObjectsCalibrationRetrieve : MonoBehaviour {
2    void Start () {
3      Vector3 position = new Vector3();
4      Vector3 rotation = new Vector3();
5      Quaternion rot = new Quaternion();
6
7      position.x = PlayerPrefs.GetFloat("C_x");
8      position.y = PlayerPrefs.GetFloat("C_y");
9      position.z = PlayerPrefs.GetFloat("C_z");
10     rotation.x = PlayerPrefs.GetFloat("C_rx");
11     rotation.y = PlayerPrefs.GetFloat("C_ry");
12     rotation.z = PlayerPrefs.GetFloat("C_rz");
13     rot.eulerAngles = rotation;
14     transform.position = position;
15     transform.rotation = rot;
16   }
17 }
```
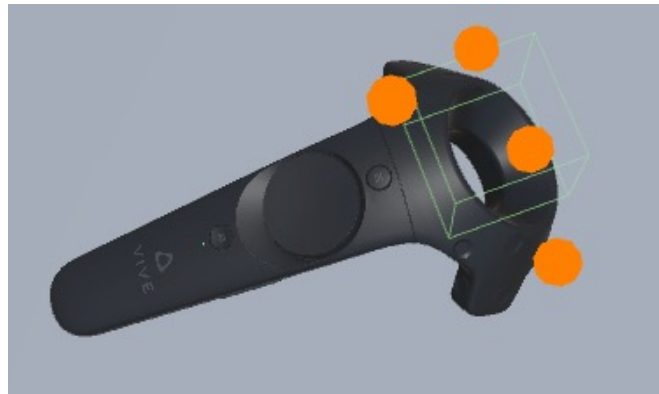
The alignment of the two reference system was done manually. During the calibration of either the HTC Vive or the OptiTrack we aligned the axis parallel to the room walls so that we did not have to adjust the rotation, but the origins were different. We took inspiration from

the work of Roo et al. [31] and we attached four retro-reflective markers on a Vive controller (see Figure 22a) and created a rigidbody in Motive, then we aligned the position of the markers displayed in Unity to their real position on the controller by manually changing the values in the Transform component of the GameObject parent of the tracked objects (Figure 22b).

(a) Vive controller with retro-reflective markers

(b) Markers aligned on the controller model in Unity

Figure 22: Alignment of the OptiTrack and Vive reference systems

4.2.3.3   1:1 Replica

As we have already said in this document, we wanted to investigate the differences in interaction between using as physical proxy the same objects that the user sees in VR and using objects with different characteristics. The first step was to identify the real objects that we wanted to use and create or find online their 3D model. Then, we applied retro-reflective markers on them, created the rigidbodies in Motive and registered them to their virtual counterparts. In this section we will discuss the registration process and the placement of the markers on the objects.

Registration

(a) Markers displayed on the virtual object   (b) Markers on the real object

Figure 23: Markers on the virtual object and markers on the real object

We had to manually register every virtual object to its virtual counterpart. To do that, for each object, we created a Unity GameObject, we attached a `OptitrackRigidBody.cs` script to it with the correct Stream ID set. Then, as child of that GameObject we put the corresponding 3D model (see Figure 21). Similar to what we did in the calibration process just described, we used the position of the markers shown in Unity to align the 3D model to the markers' position on the real object. Figure 23 shows an example of markers aligned on the virtual object and markers placed on the real object.

Fitts' Law application

In the Fitt's Law test application only one object was present (i.e. the cylinder). We used a plastic cylinder of 5.5 cm of diameter and 5 cm of height (Figure 24). We made sure that the virtual cylinder had the same dimensions. The real object was big enough to apply a rigibody marker base on it. We applied the base on the bottom of the object because from a pilot study we had conducted we noticed that the user tends to grab the object mainly from the top and not from the side, in this way the markers would be less invasive during the interaction.

Breakfast Time application

In this application are present three objects. For all of them we used a rigibody marker base as their size allowed it. Figure 25 shows the virtual object on the left and their physical proxy on the right, it is also possible to see the positioning of the markers. For the milk box and the cereal box we placed the base on the top of the object since the common approach is to grab them by the side. Furthermore, by placing it on the top the tracking is better as the line of sight with the cameras is is less likely to be occluded. For the bowl we used a rigidbody
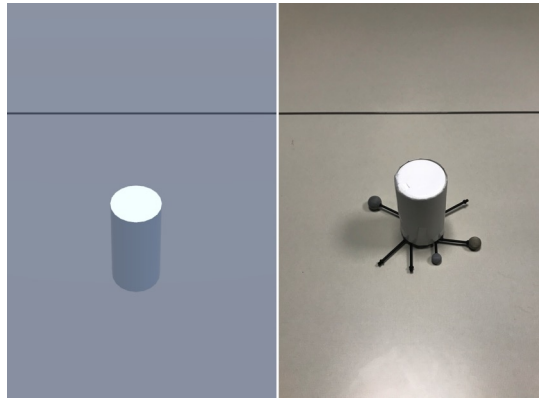
Figure 24: Virtual object (left) and its real-world 1:1 replica (right) for the Fitts' Law test application

base as well on the side, but its position is expected to be more of an interference during the interaction. The milk box was half filled with water and then sealed. The cereal box was half filled with real cereal and then closed.

Colored Objects application

In this last application were present five objects. We used a mixed approach in this case. For the soda can (Figure 26a) and the plastic glass (Figure 26c) we did the same reasoning as the Fitt's Law cylinder and we placed a marker base on the bottom of the objects as users were more likely grab them from the top. For the flashlight (Figure 26e) we attached a marker base on the top, as far as we could from the handle. For the bottle (Figure 26e) we used four spherical markers placed in different positions. The baseball was the most problematic case we had to face (Figure 26b). Our first attempt was to use just flat markers in order to avoid to adding foreign bodies on a such small object. Unfortunately, due to their limeted tracked range
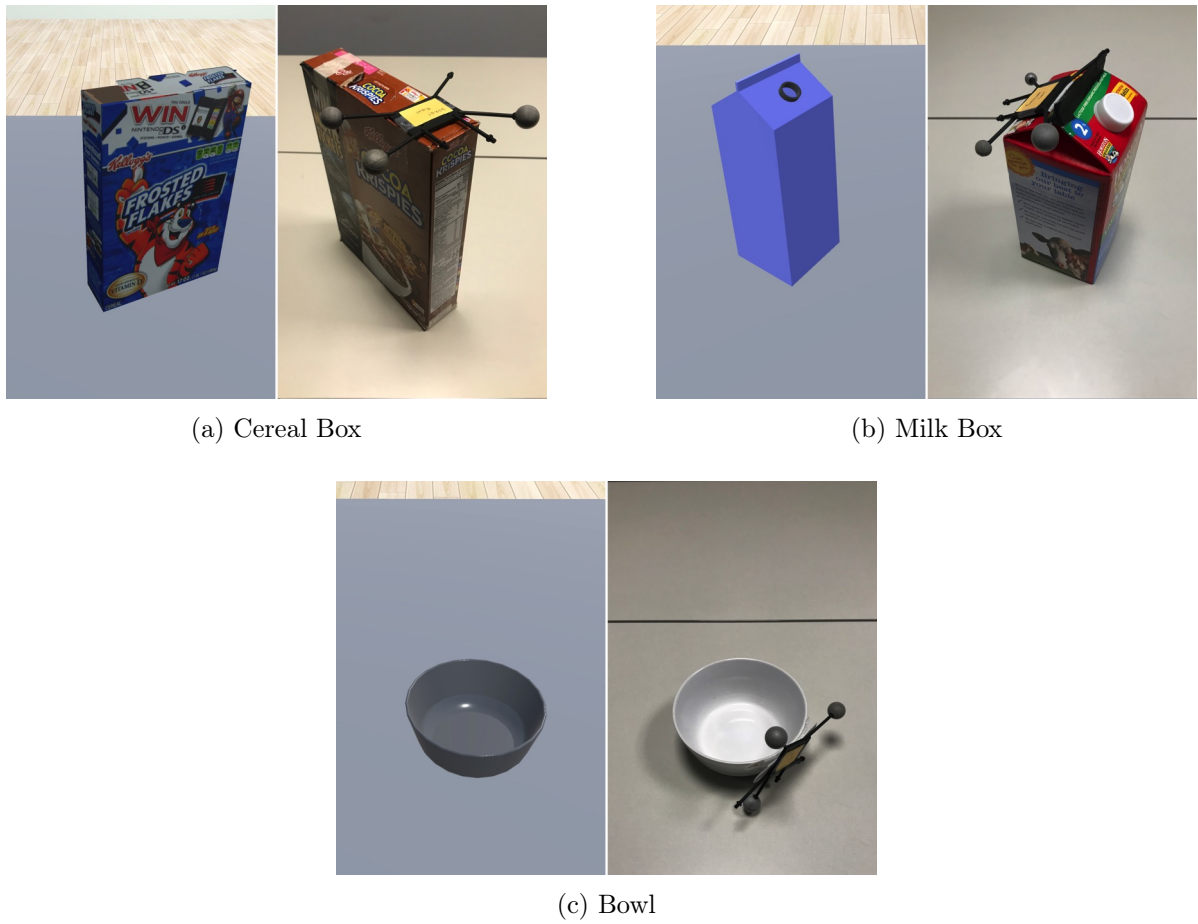
(a) Cereal Box

(b) Milk Box

(c) Bowl

Figure 25: Virtual objects (left) and their real-world 1:1 replica (right) for the cereals application

of motion the tracking was laggy and inaccurate. So we decided to use two spherical markers and three flat ones. Motive can recognize a rigibody with just 3 markers, in this way the two spherical marker were always detected, while at least one of the other three was seen by the cameras depending on the ball orientation.

4.2.3.4 Different Objects

In this second type of interaction using physical proxies in virtual reality we applied in part the concept of "Substitutional Reality" introduced by Simeone et al. in [12] where a substitution process adapts the virtual environment to the physical world. In other words, physical elements in the real world are matched in the virtual environment with different levels of mismatch. In the cited article are present some recommendations to follow when it comes to substituting an object, the most important to us are:

1. Objects that detract users from their "suspension of disbelief"[1] are those with a different shape or temperature.

2. Virtual objects that are smaller than their physical counterparts impact in a negative way in terms of believability, however bigger objects do not cause the same problem.

3. Virtual objects should be "closer to the proxy's physical appearance in the parts users are most likely to contact"[12].

Following those suggestions we identified the objects that could substitute virtual objects in the real world. Here below we will explain our choices for each application.
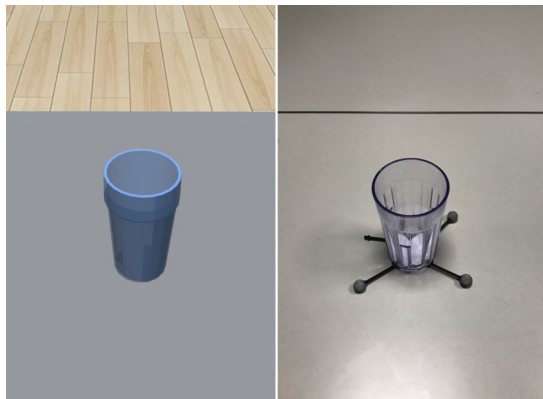
---

[1]Willingness to suspend one's critical faculties and believe something surreal [32]
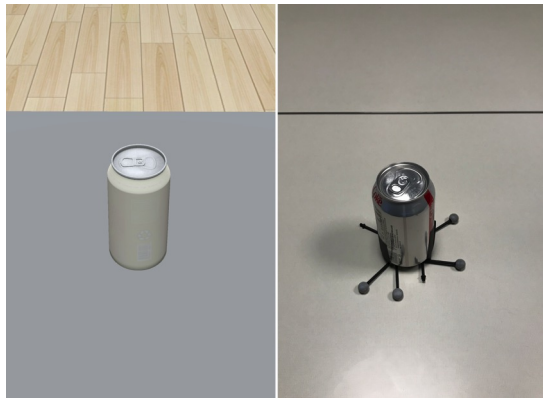
(a) Soda can
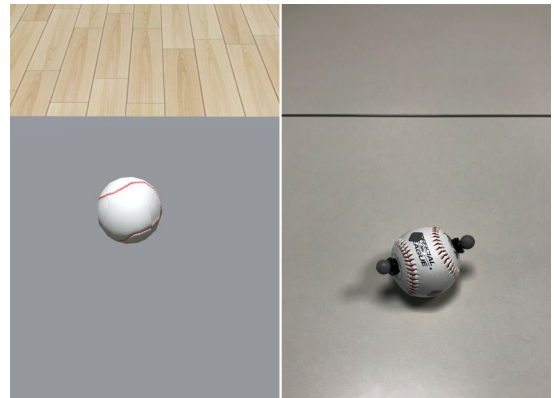
(b) Baseball

(c) Plastic glass

(d) Bottle

(e) Flashlight

Figure 26: Virtual objects (left) and their real-world different replica (right) for the colored objects application

Fitts' Law application

In this application we substituted the cylinder with a rectangular cardboard box (Figure 27). The dimensions of the base of the box were similar to the cylinder (5.5 cm x 2.5 cm) but the height was greater (7 cm). Also the material was different ((*shape, size and material mismatch*)). We placed a rigidbody base on the bottom of the box and registered it to the virtual object by aligning the bottom of the virtual object to the bottom of the tracked object.

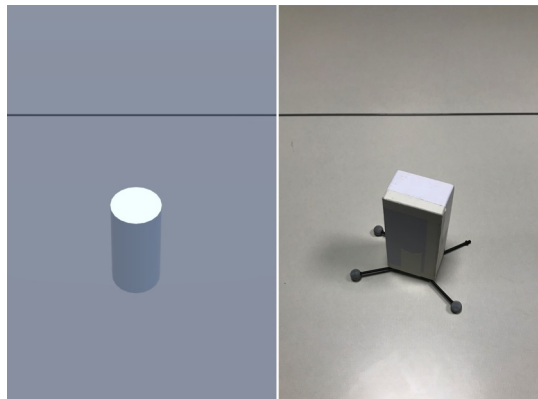Figure 27: Virtual object (left) and its real-world different replica (right) for the Fitts' Law test application

Breakfast Time application

For this application we purposely pushed the subtitution to the limit. The cereal box was replaced by a box significatively smaller (*size mismatch*) than the real one (28a). It was registered to the virtual object such that one side and the bottom were aligned. The milk box

(28b) was replaced by a water bottle of similar size (*weight, shape and material mismatch*). The bowl (28c) was replaced by a cardboard box of the same height and width (*weight, shape and material mismatch*).
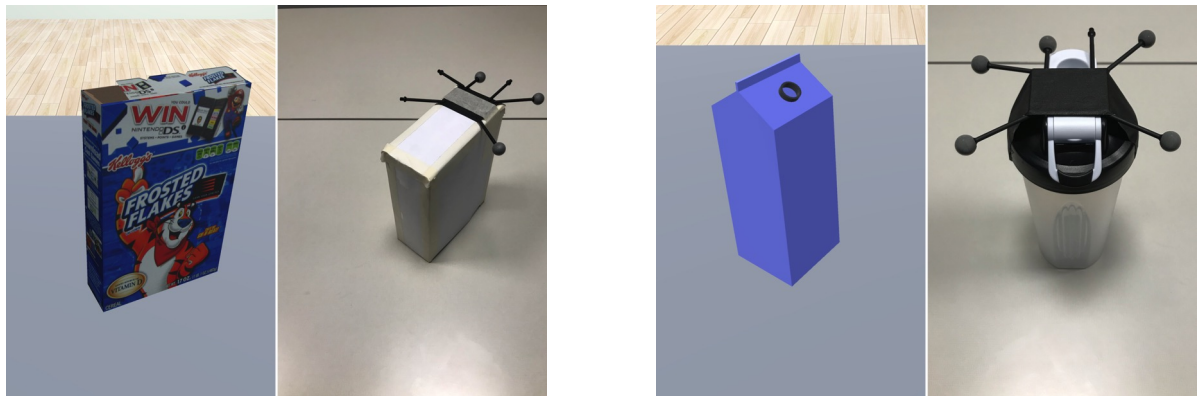
(a) Cereal Box

(b) Milk Box
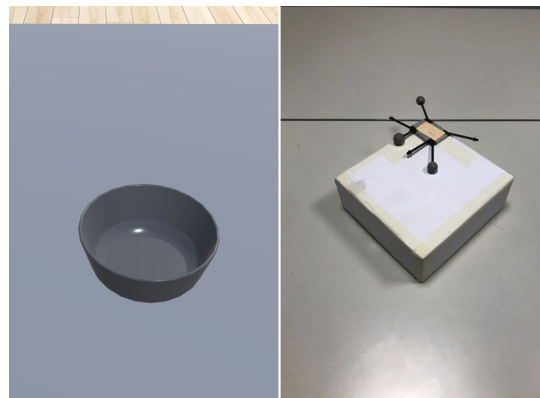
(c) Bowl

Figure 28: Virtual objects (left) and their real-world different replica (right) for the cereals application
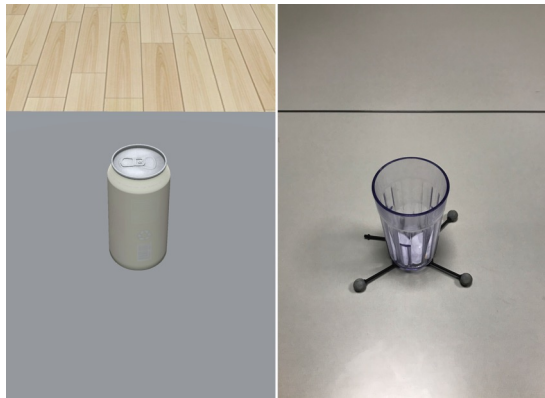
Colored Objects application

In this last application we tried to choose substitution replicas having parts that the user would most likely to contact be closer to the virtual objects. For the soda can (29a) we used the plastic glass as its physical proxy as its size and shape were similar to virtual model (*weight and material mismatch*). For the baseball (29b) we used a slightly smaller plastic ball (*weight and material mismatch*). For the plastic glass (29c) we used the cylinder of the Fitts' Law test which was smaller than the real glass (*weight and size mismatch*). The bottle (29d) was replaced by another empty bottle, the one already used in the "Breakfast Time" application, whose size is bigger than the original (*weight and size mismatch*). Finally, for the flashlight (29e) we used a PlayStation Move controller as the handling points of the two objects were similar (*weight and shape mismatch*)

As it is possible to notice, one of the advantages of using not exact replicas is that it is possible to reuse the objects across different applications.

(a) Soda can
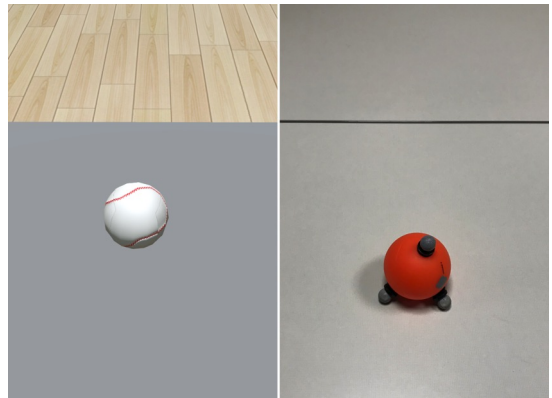
(b) Baseball

(c) Plastic glass

(d) Bottle

(e) Bottle

Figure 29: Virtual objects (left) and their real-world different replica (right) for the colored objects application

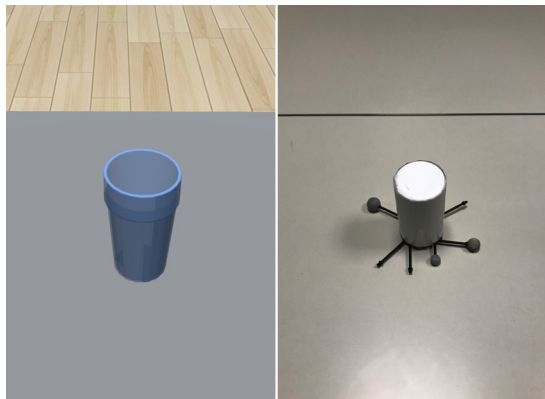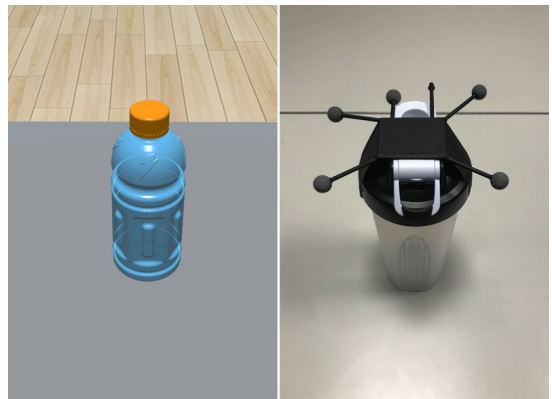### 4.2.3.5 Tables

The physical objects were placed on tables. For our applications we used three tracked tables: two of them were used for the Fitts' Law test application, just one for Breakfast Time, and all three of them for Colored Objects.

For the real tables we used three *Allsteel© Aware* line (30a). On each of them we placed six flat retro-reflective markers in the top left corner, making sure to always create a unique asymmetrical pattern. Figure 31 shows an example of markers arrangement. These real tables have the dimension of 150 cm W x 45 cm D x 70cm H. On the manufacturer's website we found the the 3D model of the table and we imported it into our project (30b). We scaled it to make sure that the dimensions in the virtual environment were the same as in the real world. That model was used in all the test applications for the 1:1 Replica and also in the Colored Objects for the Different Objects interaction method. For the Fitts' Law and Breakfast Time applications with Different Objects we decided to use different models for the table (shown in Figure 30c and Figure 30d).

During the experiment we arranged the tables in the calibrated area respecting the same arrangement that we used in the interaction methods without physical proxies. Figure 32 shows the placement with respect to the Vive area and the OptiTrack cameras. We placed the tables in a way to have a good number of cameras facing the user so that she could not occlude the line of sight.

(a) Real table

(b) Virtual model of the table

(c) Virtual model used in Fitts' Law test application (Different Objects)

(d) Virtual model used in Breakfast Time test application (Different Objects)

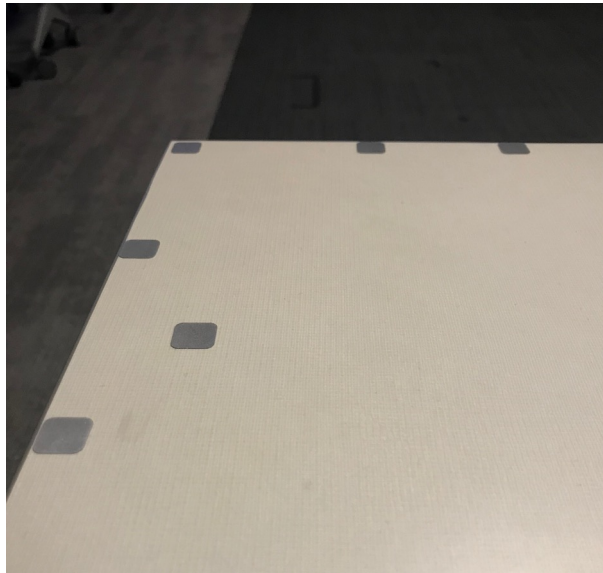Figure 30: Real table (a) and virtual table models used in the test applications

Figure 31: Flat markers arrangement on a table
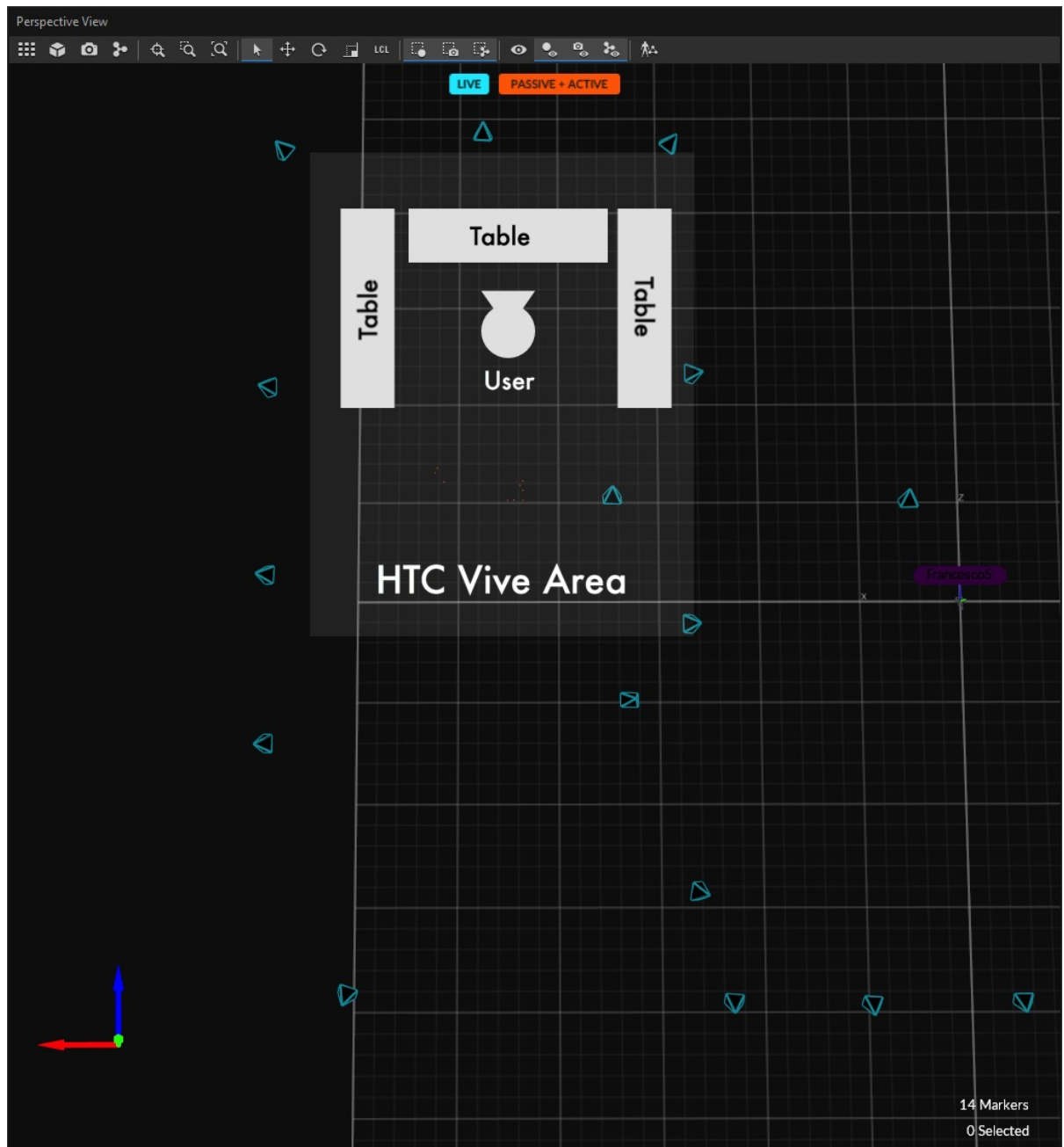
Figure 32: Illustration showing the placement of tables in the calibrated area.

4.2.3.6   <u>Hands tracking</u>

To interact with the real objects in virtual reality, users need to see the position of their hands. Originally we wanted to use the Leap Motion for hand tracking as seeing the fingers moving would have increased the level of immersion in the virtual environment. Unfortunately, we experienced some issues with the Leap Motion tracking while the OptiTrack cameras were active. That was because the two systems use the same IR wavelength (850 nm). We had to find an alternative. We discarded the idea of using tracking gloves as those would have interfered with the users' sense of touch. Our solution consisted in using two rigidbody marker bases and attach them to the users' hands with a band of fabric (Figure 33). We associated those rigidbodies to two hands models in our applications. The main drawback of this implementation is that users cannot see their real fingers position but they can use the models as a reference. Especially when users are holding an object in their hands, the models would represent a wrong pose. We tried to minimize this issue by augmenting the transparency of the models' material while they were colliding with an object.

Figure 33: User wearing trackers for the hands

CHAPTER 5

USER STUDY

In this chapter we will illustrate and discuss a user study that has been conducted to test the different interaction methods explained in the previous chapter with some assessment applications we developed.

5.1   Goals

The goal of this user study was to observe which interaction method gives the best results in terms of realism, level of immersion, enjoyment and the precision and quality of movement in VR. Furthermore, we wanted to understand which are the most important factors that influence the correct execution of manual tasks in VR.

5.2   Hypotheses

We hypothesized that a passive feedback system where the physical objects are 1:1 replicas of the virtual objects should give better results in terms of realism. However, using physically different real objects could lead to interesting findings as, for example that differently weighted object could ease the execution of a task. So, in this case if the perception of realism is similar to the one with the 1:1 replica, this passive feedback system could be the best.

5.3   Apparatus

The study was performed in the UIC ERF room 2068 (called "Continuum"). We used the applications described in chapter 4 for our trials. The applications were executed on an

Alienware Aurora PC (CPU: *Intel Core i5-7400 @3.00GHz*, GPU: *NVIDIA GeForce GTX 1080 Ti*) with Windows 10 as operative system. We ran the applications directly in the Unity editor (*Unity 2017.3.0f3*). We did not build executables because sometimes it was necessary to modify some parameters at run-time.

To the PC was connected a HTC Vive head mounted display equipped with the Vive Deluxe Audio Strap[1] for a better comfort for the user and to reproduce sounds. In front of the HMD we attached a Leap Motion device and connected it on the USB port of the Vive. The HTC Vive lighthouses (for the tracking) were placed on the ceiling at 2.7 m of height. We connected the two devices together with the provided sync cable (we did not use the default optical synchronization because we noticed that the IR light emitted by the OptiTrack cameras was occasionally causing interference). The calibrated resulting area measured 3.3m x 3.3m. In the same space was installed the OptiTrack system as already described in 4.2.3.2.

The study foresaw that the same tasks done in VR were executed in the real world without any kind of visual augmentation. In this case we used the same set of objects that we used in the 1:1 replica interaction method and we gave to the users a pair of tracked eye-glasses (without lenses) to track their head. For the "Breakfast Time" and "Colored Objects" applications we did not have to modify anything, the only difference was that the user did not have to wear the HMD. For the "Fitts' Law Test" application we had to find a way to display the targets on the table. Our solution was to use a short range projector facing upwards placed underneath a

---

[1]https://www.vive.com/eu/vive-deluxe-audio-strap/

plexiglass sheet resting between two tables (see Figure 34). The projector was shooting on a film placed over the plexiglass the targets of the application. The projected image was calibrated so that the targets had the right dimensions. We put some reflective markers on the plexiglass in order to have a reference of its position and being able to use the tracked cylinder of the 1:1 replica interaction method for the target acquisition task.
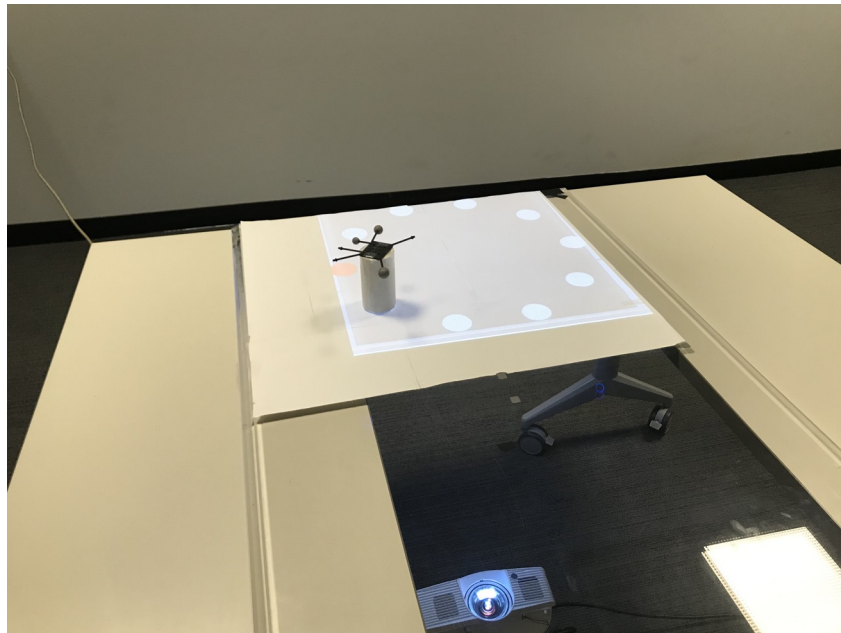
Figure 34: Apparatus used for the Fitts' Law test application in the real world

Waiting room scene

Since we needed time to set the environment before each trial, we created an additional VR application in which the participant would wait. From the experimenter point of view, this application had also the function of a laucher of the test applications with a choosen interaction method (see Figure 35).
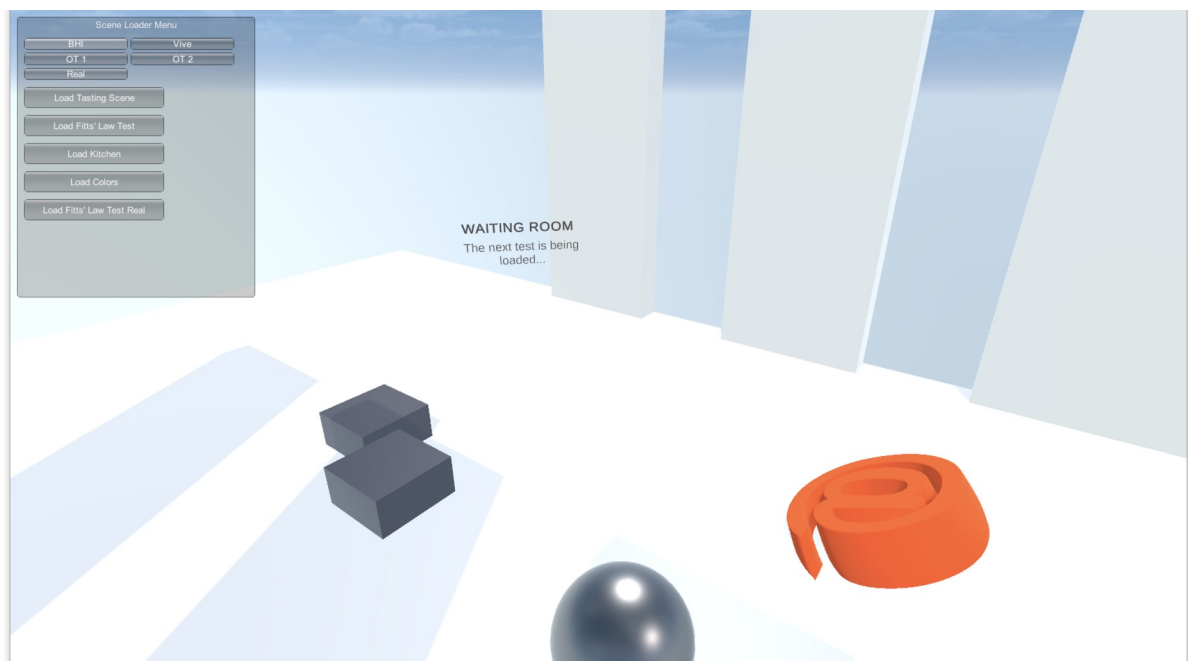
Figure 35: Screenshot of the virtual waiting room. On the top-left it is possible to see the GUI used by the experimenter to select an interaction method and launch a test application

5.4 Participants

Fourteen participants were recruited for this study (11 male, 3 female) aged 18–34. We asked them to rate their experience with head mounted displays, videogames and technology such as smartphones and PCs on a scale from 1 (never used/used once) to 5 (used daily/used daily for many hours). Our sample constituted of monthly gmae players (M = 3.07, SD = 1.44) with low experience with virtual reality (M = 2.36, SD = 1.39), but intensive user of smartphones (M = 4.79, SD = 0.43) and PC (M = 4.86,SD = 0.36). To participate in this study, a subject had be at least 18 years old with no disability that would have hampered the use of the devices been tested. No reimbursement or compensation were given to participants.

5.5 Procedures

Each study was done individually, so only one person at a time did the experiment. Each session lasted approximately 90 minutes. Study procedures were explained to the participants at the beginning of the session. The procedures that the participants were asked to complete are presented in the following subsections:

5.5.1 Pre-study Demographic Questionnaire

During this part the participant had to complete a questionnaire (see Appendix B) regarding gender, age and experience level with the technologies used in the study. This data is used by the investigator to better understand possible anomalies in the results of the study.

5.5.2 Introductory Phase

During this part of approximately 5 minutes, the participant was briefed on the experiment, equipped with the VR HMD and VR controllers and introduced on the operations involved.

5.5.3   Training Phase

Figure 36: Scene create for the training

During the training phase, the participant was free to familiarize himself or herself with the 3 interaction methods. The interaction methods trained are BHI, HTC Vive Controllers and real objects 1:1 replica (the real objects in a different replica were not trained because the interaction works in the same way as the 1:1 replicas, and in this way the results of the study were biased by already having been in contact with those objects). An additional Unity scene (see Figure 36) was created for this purpose where the participant could grab and move objects around. In

this scene was present a countdown timer indicating the time left and a panel with the written instructions to use the current interaction method, the participant was instructed also orally. The participant spent 5 minutes on each interaction method. This phase was necessary as some of the interaction methods (especially BHI and HTC Vive controllers) are not immediate to use for people who did not have experience with those. In this way, biases due to the novelty of the interaction method were minimized.

5.5.4   Assessment/Evaluation Phase

The assessment phase is composed of five parts, one for each interaction method, plus one control trial where the participant is in "full reality" (i.e. not wearing the VR HMD). The order of testing the interaction methods was not the same for every participant. The order of execution was randomized, so that we tried to have an equal number of participants that used a certain interaction method first, the same interaction method second, and so forth. The execution of each part followed the same script.

First, the space in the room used for the trial was cleared and the participant wore the HMD. The starting scene was a virtual waiting room, an application that we created from which the investigator was able to launch the various assessment applications and to select the interaction method to test. While the subject was in the waiting room, the apparatus necessary for the test was placed in the area. For the trial involving the use of HTC Vive controllers, they were given to the subject. Whereas, for the trials acting to assess the interaction methods that relied on the use of real physical objects, those were placed in the area.

Once the setup was completed, the investigator launched the first assessment application. At the completion of the tasks of that application the subject was returned to the waiting room and the area was set for the following test.

After all the three assessment tests were completed, the participant was asked to remove the HMD and to complete the *NASA TLX* questionnaire, and experience evaluation questionnaire and to participate in a brief discussion with the investigator regarding the interaction method just tested. Then, the assessment phase continued with the remaining interaction methods. The total duration of this part was around 45-60 minutes.

5.5.5   Post-Experiment Unscripted Interview Phase

During this part, the experimenter had an unscripted discussion with the participant regarding the experiment, the various types of interaction, the level of immersion, enjoyment, realness felt during the evaluation phase and the system in general. During the discussion, the experimenter asked the participant for general thoughts and feedback about the system, what they liked and did not like. In general, the discussion's aim was to gain insights into how the participants used the system and what can be improved in the future. The participant's answers were logged by the experimenter.

5.6   Data Gathering

In this user study we collected data in several ways. First, there is the data coming from the questionnaires (NASA TLX and experience evaluation). Then, there are the logs of the Unity test applications. In every test application the positonal and rotational data of the participant's head and hands were logged in a CSV file at a rate of 20 times per second. In the same way,

data on every virtual object the user interacted with was logged 20 times per second. For the Fitts' Law test application we logged the time for each target acquisition from the moment that the object leaves the start target to when it is released on the destination target together with the target widht and distance. In the "Breakfast Time" application we additionally recorded the time elapsed achiving the task completion. Additionally, we have the notes from the interviews done after the test of an interaction method and the final interview. Finally, every study was video and audio recorded and we used the records as support of our analysis of the other data.

5.7 Results and Discussion

In this section we will analyze the collected data and we will discuss the results.

5.7.1 Evaluation Questionnaires

In the evaluation questionnaire were asked five questions respectively on the realism, immersion, enjoyment, similarity and ease of use on a scale from 1 to 5:

1. *How realistic was the experience overall?* (1: Not at all, 5: A lot).

2. *How immersed were you in the virtual environment while interacting with the virtual objects?* (1: Not at all, 5: A lot).

3. *Did you enjoy interacting with the virtual objects?* (1: Not at all, 5: A lot).

4. *Did you think that the physical objects that you were touching were the same you were seeing?* (1: For nothing, 5: Same objects).

5. *How easy was it to interact with the virtual environment?* (1: Not at all, 5: Really easy).

In Figure 37 we created a boxplot with the answers to the questions of this questionnaire. BHI stands for Bare Hands Interaction, OT1 and OT2 are respectively the 1:1 replica and different objects and Vive stands for Vive Controllers. The mean and standard deviation calculations for each question and for each interaction method are also shown in Table I. To evaluate the statistical significance of the results we executed the T-test on each pair of interaction methods for each question. The significance threshold was set at 0.05.
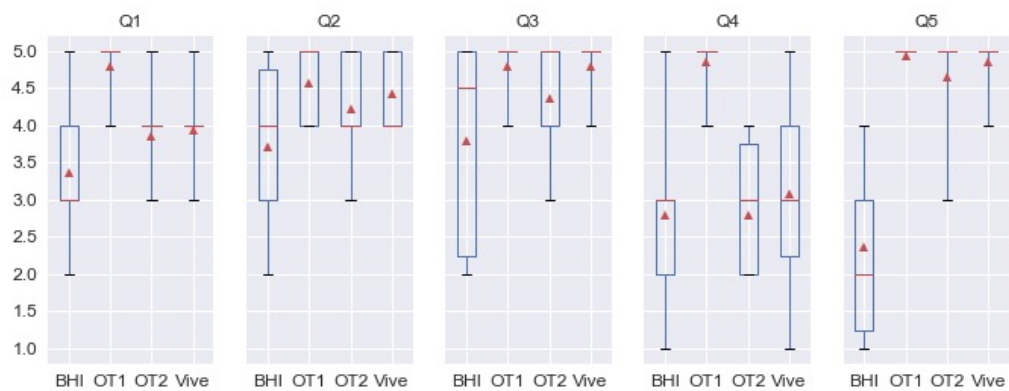

Figure 37: Boxplot of the answers to the evaluation questionnaire grouped by question. Triangles represent mean values and lines median values. Whiskers show max and min values

Realism

For question 1 we have really uniform results with the exception of BHI. As we expected the 1:1 replica gave the best result in terms of realism and we found significancy in the results

TABLE I: RESULTS OF THE EXPERIENCE QUESTIONNAIRE

Interaction Method		Q1	Q2	Q3	Q4	Q5
BHI	Mean	3.35	3.71	2.78	2.78	2.36
	SD	0.93	1.07	1.48	1.25	2.36
1:1 Replica	Mean	4.78	4.57	4.78	4.85	4.93
	SD	0.42	0.65	0.58	0.36	0.27
Different Objects	Mean	3.86	4.21	4.36	2.79	4.64
	SD	1.03	0.70	0.84	0.97	0.74
Vive Controllers	Mean	3.93	4.43	4.79	3.07	4.86
	SD	0.61	0.51	0.43	1.27	0.36

($p<0.001$) compared to the others interaction methods. The Different Objects and the Vive controllers have similar results and their difference is not significant ($p>0.8$), while BHI is behind but not too distant (again, not significant $p>0.2$), some people indeed appreciated the realism given the by the finger tracking.

Immersion

The levels of immersion is comparable for the 1:1 replica, different objects and Vive controllers and it is high. While for BHI it is slightly lower but with results less uniform. The difference was significant only between BHI and Vive controllers ($p=0.035$) and 1:1 replica ($p=0.0026$), in the other comparisons p-values were greater than 0.05.

Enjoyment

In terms of enjoyment, Vive controllers and 1:1 replica have the exact same results and they are the most enjoyable. They are followed by the different objects, which people found annoying in that what they were seeing was different than what they were touching, but the difference

is not statically significant (p=0.082). BHI gave conflicting results, but on average the level of enjoyment is lower than the one with the other three interaction methods (significant, Vive Controllers: p=0.033; 1:1 Replica: p=0.024).

Similarity

As expected, when we asked the subjects to evaluate the level of similarity between the objects that they were touching and the ones that they were seeing, the 1:1 replica got the highest values (p<0.001). The interaction method with different objects received lower values, meaning that people recognized that the objects were not the same. For BHI we were surprised that users were answering this question as if they were really having an object in their hands. When we asked for clarification some of them replied that after a while they had the illusion of touching an object for real. Finally, also for the Vive Controllers, we found rather high levels of perceived similarity. That could be justified in another way. Users were seeing the controller models in their hands and so the actual grabbed object was the controller, but what is more interesting is that more than one person defined the interaction as if he was picking up object using a pair of thongs. The differences between BHI, Vive Controllers and Different Objects are not statistically significant (BHI-Vive: p=0.43; BHI-DifferentObj: p=1.00; Vive-DifferentObj:p=0.54).

Ease of use

For this last question, all the subjects confirmed unanimously that the 1:1 replica was the easiest interaction method to use. Similar results were obtained by the different objects and the

Vive controllers while BHI was the hardest to use. Significance in the results was found only between BHI and the others interaction methods (p<0.001).

5.7.2 Workload Results

To evaluate the workload we used a standardized self-evaluation tool, the NASA Task Load Index (TLX) scale is widely used in human factors research. The questionnaire consist in 6 questions to be answered on a scale from 1 to 21: "

1. **Mental Demand:** How mentally demanding was the task? (1: Very Low, 21: Very High)

2. **Physical Demand:** How physically demanding was the task? (1: Very Low, 21: Very High)

3. **Temporal Demand:** How hurried or rushed was the pace of the task? (1: Very Low, 21: Very High)

4. **Performance:** How successful were you in accomplishing what you were asked to do? (1: Perfect, 21: Failure)

5. **Effort** How hard did you have to work to accomplish your level of performance? (1: Very Low, 21: Very High)

6. **Frustration** How insecure, discouraged, irritated, stressed, and annoyed were you? (1: Very Low, 21: Very High)

"

The original TLX is composed of two parts, the questions listed above are the first part, while the second part would be some pairwise questions on the perceived importance of each

factor to calculate a weighted workload score. This part is often skipped by researchers and the single questions of the first part are analyzed separately. According to [33] this approach might increase the experimental validity. We decided to analyze our data in that way. Similarly to what we did with the experience questionnaires results, we elaborated the answers with the Python data analysis library Pandas. To evaluate the statistical significance of the results we executed the T-test on each pair of interaction methods for each question. The significance threshold was set at 0.05. In Figure 38 we reported a boxplot with the answers grouped by question and interaction method. The mean and standard deviation values are also reported in Table II.

TABLE II: RESULTS OF THE NASA TLX

Interaction Method		TLX.1	TLX.2	TLX.3	TLX.4	TLX.5	TLX.6
BHI	Mean	7.28	8.78	5.93	9.21	13.07	11.07
	SD	5.72	25.74	5.78	4.54	4.45	5.98
1:1 Replica	Mean	2.57	4.36	3.50	1.71	2.07	1.64
	SD	2.20	4.20	3.52	1.59	1.33	1.15
Different Objects	Mean	3.36	4.07	3.71	2.64	4.07	3.86
	SD	2.17	3.52	3.62	1.39	2.97	3.87
Vive Controllers	Mean	2.64	3.50	3.07	2.21	2.78	1.93
	SD	2.56	2.95	3.14	1.52	2.12	1.93

For every category, lower values indicate a low workload while higher values indicate an high workload. It is possible to notice that for each question the mean values of the BHI are
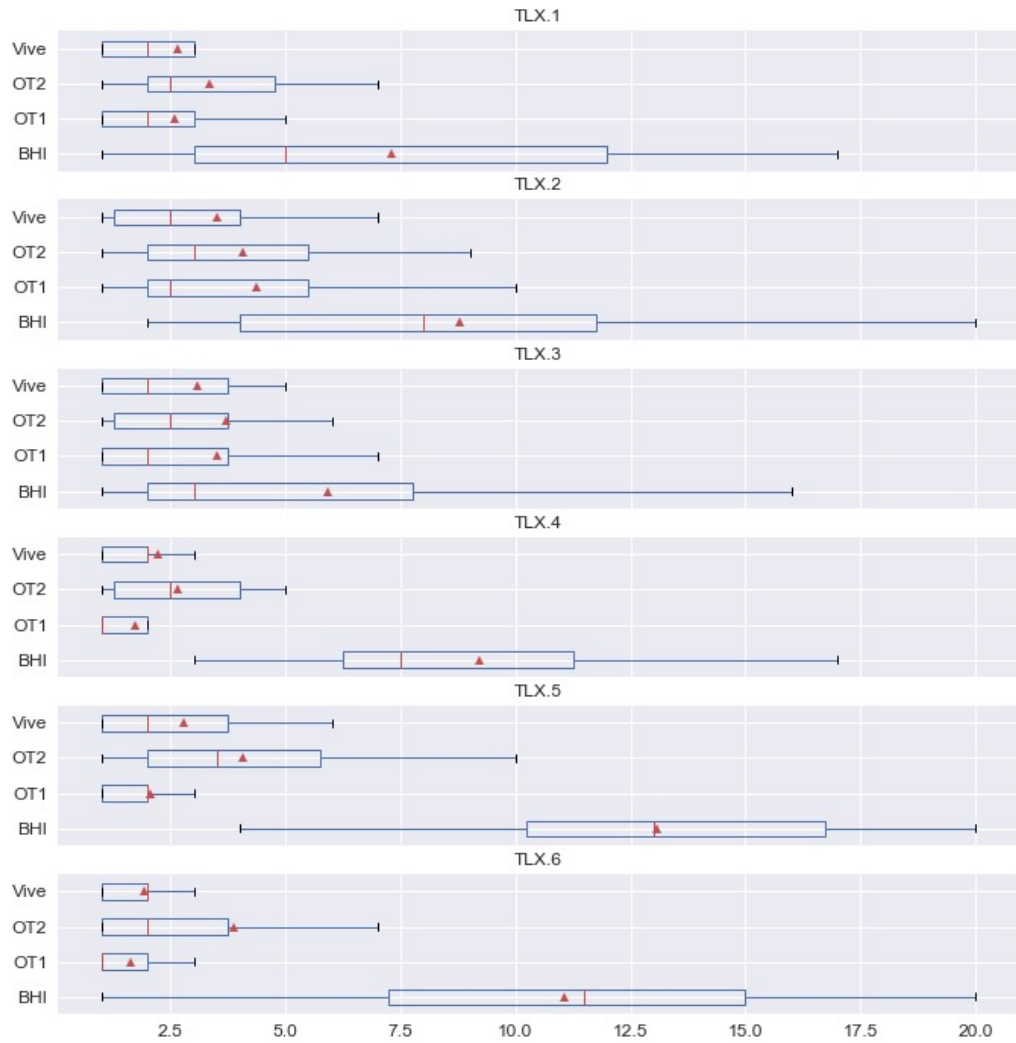
Figure 38: Boxplot of the answers to the NASA TLX grouped by question. Triangles represent mean values and lines median values.

always the the highest, but also their standard deviation is rather high, meaning that there were conflicting opinions. For BHI we registered significantly high (Different Objects: p=0.002; other interaction methods: p<0.001) levels of frustration (question 6), this was due mainly to the fact the the tracking of the Leap Motion device was not always stable and users were occasionally loosing the virtual object in their hands. Additionally, the fact of not having touch feedback increased their frustration as users did not have an immediate signal telling them that an object was really grabbed or release. These motivations are valid also for the high level of effort that resulted in question 5 for BHI (significant, p<0.001). Mental demand (question 1) results are not particularly relevant for this study. The physical demand (question 2) highlights how using Vive controllers was the least physically demanding interaction method on average, but not significantly better than 1:1 Replica and Different Objects (respectively p=0.24 and p=0.41). This is motivated by the fact that users in that case did not have to grab and move a physical object, but just to approach a virtual one with the controller already in their hand and press a button. For the 1:1 replica and different objects results are comparable. In terms of performance (question 4), subjects on average felt that they did better using the 1:1 replica with results similar to the ones with the controllers and Different Objects (difference not significant, respectively p=0.38 and p=0.097), while using BHI they significatively performed worse (p<0.001).

Overall, we can say that the best results in terms of workload are those of the 1:1 replica. We noticed that Vive controllers performed really well and our users appreciated the ease of use of the interaction method. Using different objects as physical proxy did not to prove to be significantly worse than the 1:1 replicas (significance was found only for question 5, effort, with

p=0.02) while the absence of a touch feedback of the BHI caused the users to perceive a greater workload in the execution of the tasks compared to the other methods (p values constantly p<0.05 in every category).

5.7.3   Data Log

In this section we will discuss on the data analysis of the logs of our test applications. The analysis was done in Python using Pandas to manage the large dataset and Matplotlib, Seaborn, and plotly for the visualization part. To evaluate the statistical significance of the results we executed the T-test on each pair of interaction methods for each data category analyzed. The significance threshold was set at 0.05.

5.7.3.1   Fitts' Law test

As we previously noted in this document, we decided to use a different approach in the analysis of the Fitts' law test. The Shannon's formuation of the law [34] applies to 2D pointing tasks and although there are some examples in the related literature to extend the law to three-dimensional pointing tasks (see the work of Murata & Iwase [35] and Cha & Myung [36]) those are just experimental methods and the objective of our work was not to create a prediction model for the used interaction methods. Since to all the subjects were presented with the same combinations of target widths and distances for each interaction method (even if in a randomized order), we conducted an analysis first on the mean time for a target acquisition and then on the precision.

TABLE III: MEAN TIME VALUES FOR TARGET ACQUISITION

Interaction Method	Mean Time (seconds)
BHI	2.271
Vive Controllers	1.132
1:1 Replica	1.001
Different Objects	1.050
Real World	0.957

Time

For this first analysis, we merged all the logs from the 14 studies and then ran some descriptive statistics functions on it. In Figure 39 we presented a boxplot of the dataset grouped by interaction method. The mean values for each interaction method are presented also in Table III.

We presented also the data of the same trials executed in the real world without a HMD for comparison. As expected, the best results are those in the real world, but are really close to the ones in VR with passive haptics (1:1 Replica and Different Objects). The tasks required on average 0.175 seconds more using the Vive controllers while with the BHI results are significantly worse (p< 1.6 e-111). During the post-experiment interviews it came out that indeed that users were having troubles in releasing the object on the target because of the absence of any kind of feedback. In addition, using that interaction method the object was dropped more often with respect to the others due to the occasional loss of tracking of the Leap Motion device. The differences of the mean values resulted statistically significant according to the T-test executed
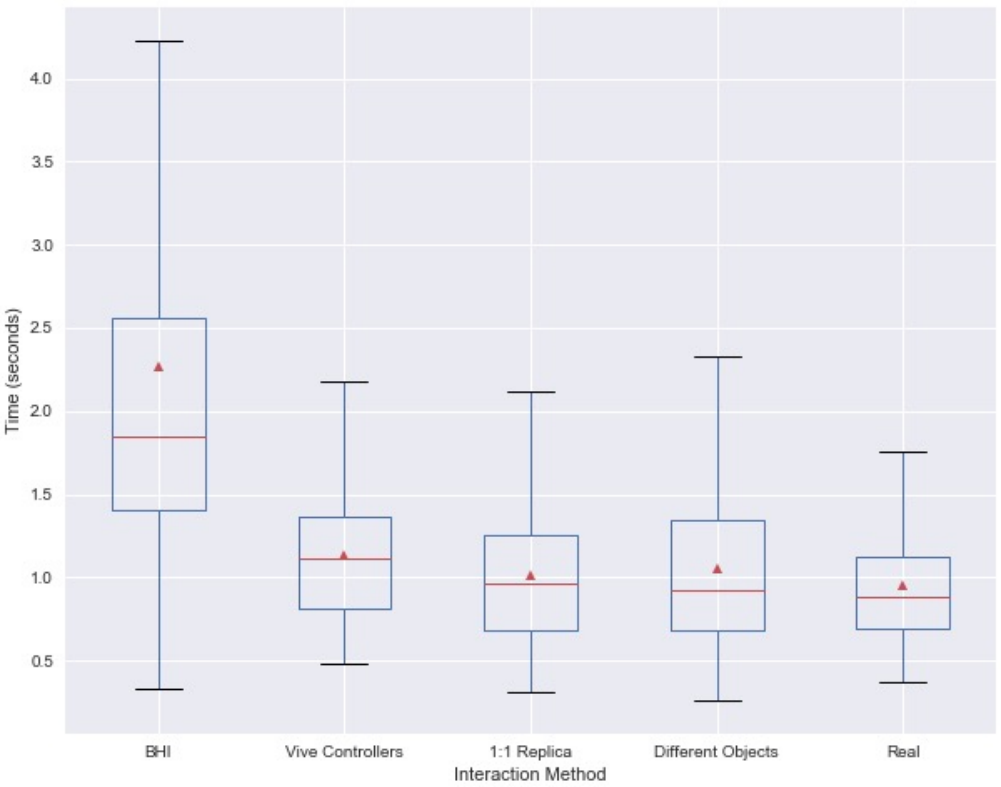
Figure 39: Boxplot of the time for a target acquisition grouped by interaction method. Triangles represent mean values and lines median values.

on each single pair of combination of interaction methods, p values are always lower than 0.005 (p<0.005).

Precision

In this test were presented nine combinations of target arrangement, as already described in 4.1, to the research subjects. The different targets' widths and distances were:

- Widths (diameter): *5.5 cm, 8.0 cm, 12.0 cm*

- Distances: *32.0 cm, 45.0 cm, 50.0 cm*

We used the logs of the object positional data to retrieve its position at the time when a target is acquired. This information was used to create a visualization of the point of contact of the object with the target to have a qualitative idea of the precision using a certain interaction method. We are reporting 3 of the 9 cases tested for each interaction method for just one study in Figure 40, Figure 41, Figure 42, Figure 43 and Figure 44. In those plots the red circles are the targets and the blue dots are the points of release of the object.

That visualization does not make possible a comparison of the interaction methods across all the studies, so we calculated the distance between the contact point and the center of the target for each target acquisition and we grouped the results by interaction method. The elaboration is shown with a boxplot in Figure 45. In Table IV are reported the mean values of the distances.

It is possible to notice that we have the maximum accuracy in the case of full reality, but the mean values are not too distant from the ones of the Vive Controllers and 1:1 Replica (which are almost identical, their difference is not significant, p=0.53). BHI showed also in this case

TABLE IV: MEAN DISTANCE VALUES BETWEEN THE TARGET CENTER AND THE OBJECT

Interaction Method	Mean Distance (cm)
BHI	2.48
Vive Controllers	1.84
1:1 Replica	1.88
Different Objects	2.16
Real World	1.16

the worst results in the study. It is surprising that the Different Objects method gave a mean result different than 1:1 Replica (significant, p<0.001). We hypothesize that the variation is due to the fact that since the shape and size of the real object differed from the virtual object's the user calibrated the movement on the object she was actually holding, not the one she was seeing.

The analysis just presented gives important data on the accuracy of the interaction whose statistical significance is confirmed in all the cases (p values always <0.001) except between Vive Controllers and 1:1 Replica. However, we did not take in consideration the fact that the target width changes in the 9 test cases. So we did a further analysis, this time comparing the distance from the center of the target in relation to its width. We used the formula in Equation 5.1 (where Tr is the target radius and D is the object distance from the center of the target) to calculate the percentual distance from the center related to the target size. The results are shown in Figure 46 and the mean values in Table V

TABLE V: MEAN DISTANCE BETWEEN THE TARGET CENTER AND THE OBJECT RELATED TO THE TARGET WIDTH

| Interaction Method	Distance (ratio)
BHI	0.633
Vive Controllers	0.460
1:1 Replica	0.470
Different Objects	0.547
Real World	0.285

$$Dist' = 1 - \frac{Tr - D}{Tr} \tag{5.1}$$

Even with this further analysis the result are in line with the previous ones, so we do not have any other comments to make. Also here significance was found in the comparison of all the interaction methods (p values always <0.001) with the exception between Vive Controllers and 1:1 Replica (p=0.53).
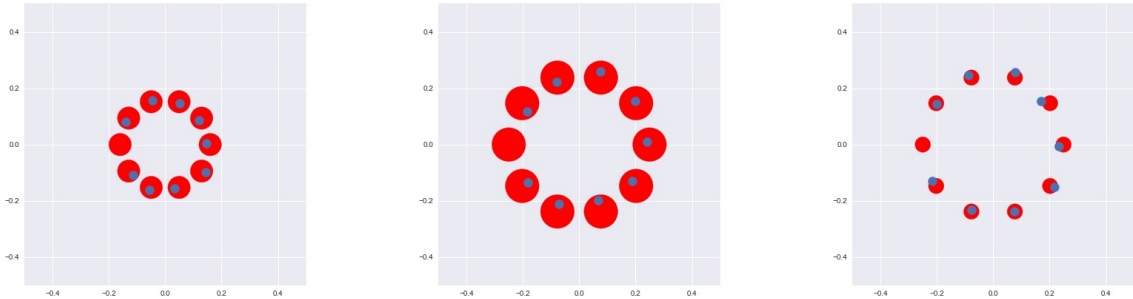
(a) W = 8,0 cm, D = 32.0 cm   (b) W = 12,0 cm, D = 50.0 cm   (c) W = 5,5 cm, D = 50.0 cm

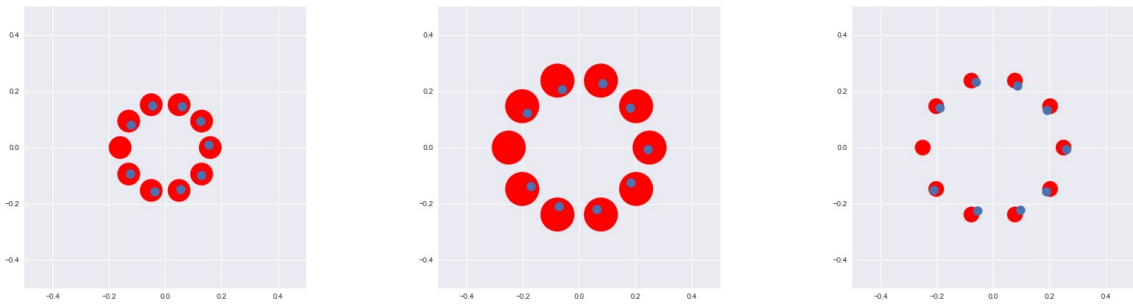Figure 40: Visualization of points of contact between the object and the targets for BHI
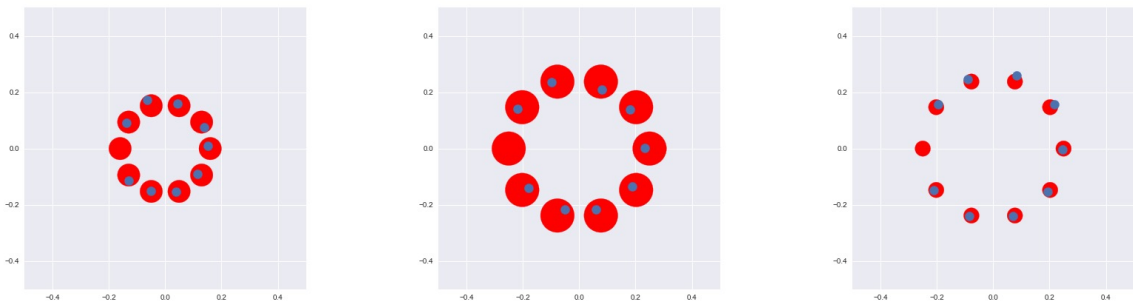
(a) W = 8,0 cm, D = 32.0 cm   (b) W = 12,0 cm, D = 50.0 cm   (c) W = 5,5 cm, D = 50.0 cm

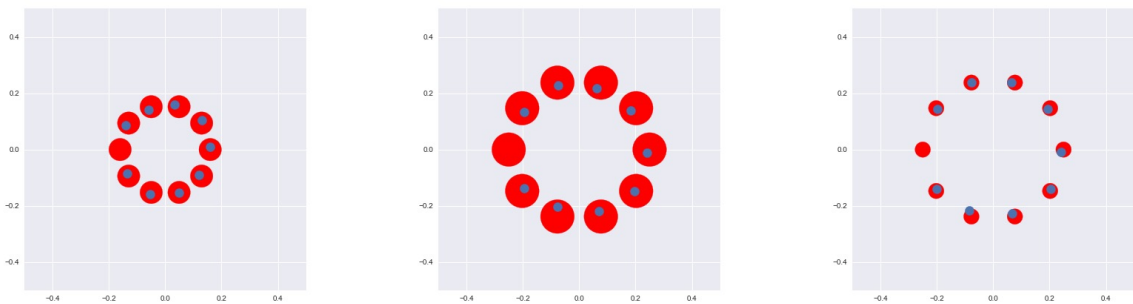Figure 41: Visualization of points of contact between the object and the targets for 1:1 Replica
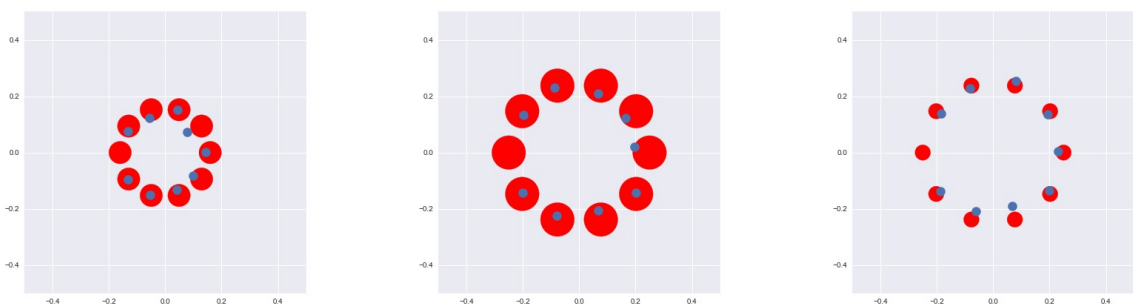
(a) W = 8,0 cm, D = 32.0 cm   (b) W = 12,0 cm, D = 50.0 cm   (c) W = 5,5 cm, D = 50.0 cm

Figure 42: Visualization of points of contact between the object and the targets for Different Objects

(a) W = 8,0 cm, D = 32.0 cm   (b) W = 12,0 cm, D = 50.0 cm   (c) W = 5,5 cm, D = 50.0 cm

Figure 43: Visualization of points of contact between the object and the targets for Vive Controllers

(a) W = 8,0 cm, D = 32.0 cm   (b) W = 12,0 cm, D = 50.0 cm   (c) W = 5,5 cm, D = 50.0 cm

Figure 44: Visualization of points of contact between the object and the targets for Real World
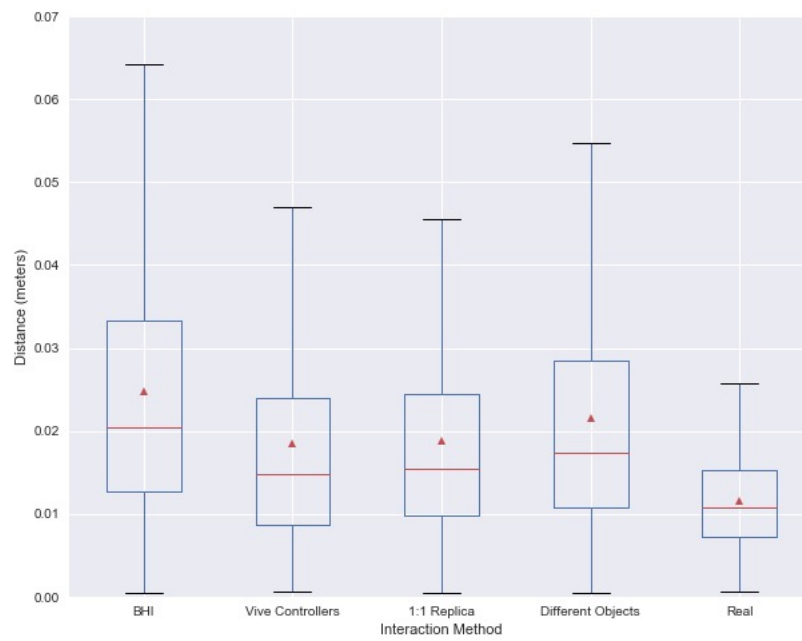
Figure 45: Boxplot of the distances between the contact point and the center of the target grouped by interaction method. Triangles represent mean values and lines median values.
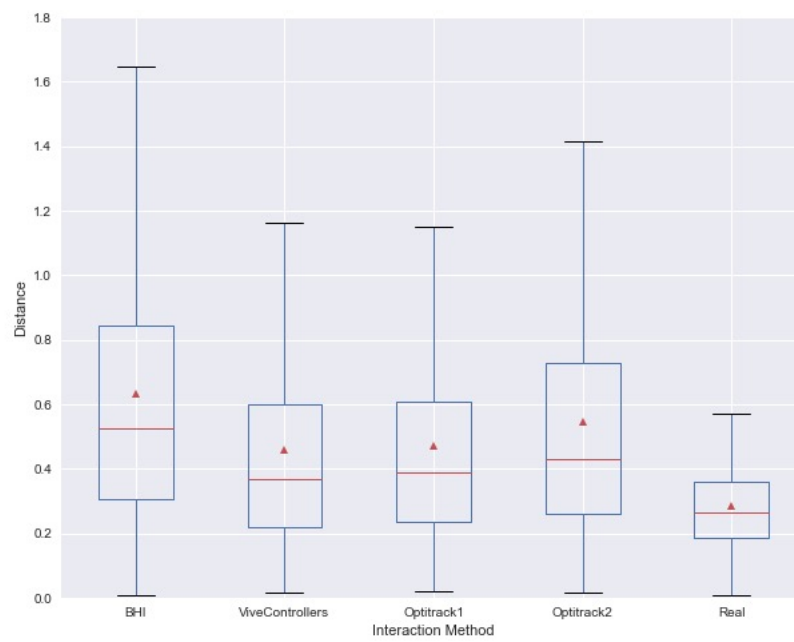
Figure 46: Boxplot of the distances between the contact point and the center of the target in relation to the target width grouped by interaction method. Triangles represent mean values and lines median values.

5.7.3.2 <u>Colored Objects</u>

For this test application we decided to conduct an analysis just on the speed of movement. We used the data collected on the position of the 5 objects present in the scene. Users were requested to move the object from a point to another and then back to the starting point. The position of the object was logged only when they were grabbed, at a rate of 20 times per second. We then calculated the mean velocity of each object. We merged the data from the 14 studies and grouped it by interaction method. Results are shown below in Table VI and in Figure 47. If we look a the Table VI by rows we can notice that for BHI and Vive Controllers the mean movement velocity is similar for all the objects while for the other two interaction methods and in the real world it varies from object to object. This could be explained by the fact that in BHI and Vive Controllers the actual weight of the objects to move was always the same (i.e. nothing or the controller) whereas in the interactions that used a physical proxy the velocity was probably adjusted according to the objects weight. It could be considered a good indicator of the realism of the interaction. In addition, we have to justify the low values for the baseball in 1:1 Replica and Different Objects. We noticed that the tracking of the physical balls was afflicted by some issues due to the user occluding the markers while grabbing them. This resulted in some lag in the displaying the model of the ball in the virtual environment in the correct position. As a result (which was confirmed by the video recordings) users slowed down their movement. For all the objects except the flashlight users could reach the objects destination by just moving their upper limbs, while for the latter they had also to make some steps towards the destination. For this reason the mean velocity of movement of this object

TABLE VI: MEAN MOVEMENT VELOCITIES FOR THE OBJECTS

| Interaction Method	Mean Velocity (m/s)				
	Soda Can	Baseball	Plastic Glass	Bottle	Flashlight
BHI	0.599	0.664	0.582	0.537	0.617
Vive Controllers	0.854	0.901	0.978	0.787	1.155
1:1 Replica	0.707	0.274	0.641	0.557	0.994
Different Objects	0.666	0.341	0.662	0.787	0.955
Real World	0.930	0.507	1.008	0.829	1.410

higher than other objects (around or higher than 1 m/s). Finally, the average objects' velocity in BHI is always lower than in other interaction methods. One possible interpretation of this result is that subjects were more careful in the movements as they were worried about dropping the object since the only feedback of having it in their hands was the visual one.
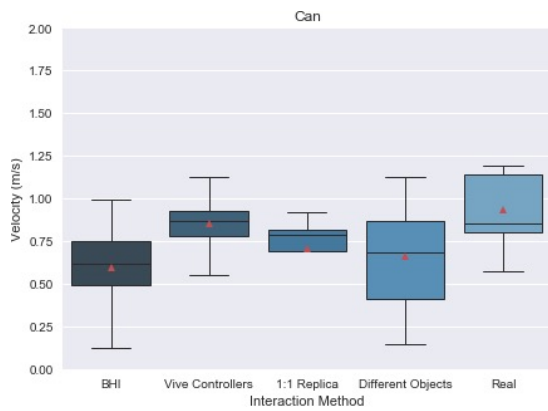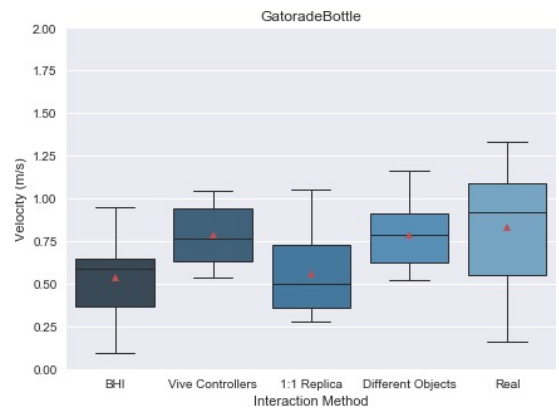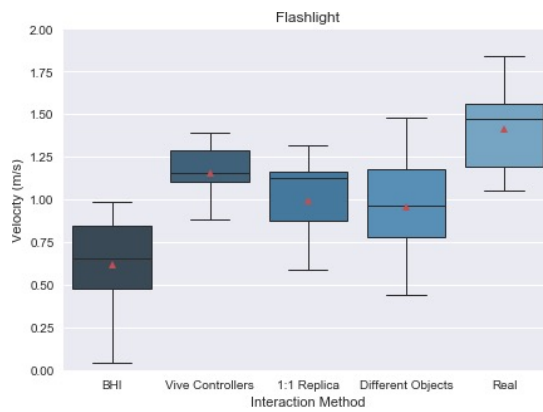
(a) Soda can
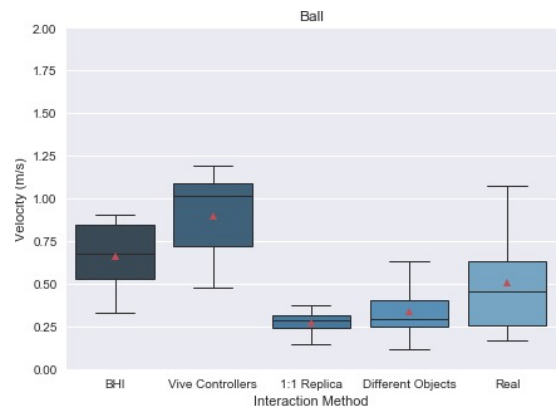
(b) Baseball
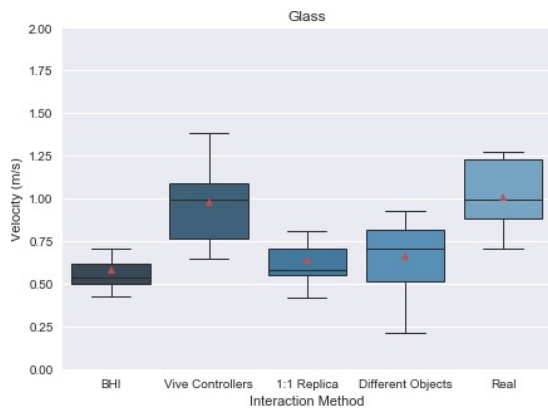
(c) Plastic glass

(d) Bottle

(e) Flashlight

Figure 47

5.7.3.3 Breakfast Time

For this test we just conducted a study on the time spent on completing the task. Users were instructed to grab the bowl and bring it in the middle of a table, then pour milk and cereals into that bowl. The amount of the two ingredients was limited and we instructed our subjects to pour them until they were out. When both ingredients were all in the bowl the task was considered achieved. We started the timer from the moment that the bowl is moved and we stopped it at the task completion. Figure 48 reports a boxplot of the results for all the participants.

For the analysis we included only the attempts where the task was completed. During the experiment if one of the ingredient was spilled, we restarted the test and flagged that attempt as incomplete. While we did not have to restart the test for the interaction methods that involved the use of real objects, for BHI and Vive Controllers that happened. Before achiving the task using BHI the average required number of task was $M = 2.93, SD = 3.95$ while for Vive Controllers was lower:$M = 0.22, SD = 0.58$.

If we look a the plot in Figure 48, we observe that on average users performed better in the real world (significant difference for BHI: p=0.021; 1:1 Replica: p=0.0018; Different Objects: p=0.028, but not for Vive Controllers p=0.52). Using Vive controllers the task was completed faster than using real objects in VR, this could be justified by the absence of a real physical counterpart for as we have explained earlier. It is interesting how with 1:1 Replica and Different Objects (respectively Optitrack1 and Optitrack2 in the plot) users spent more time to complete the task. For the former we tried to find an explanation in the video recordings. It happened in

some cases that the subject continued to pour an ingredient (the first one she choose) even if in the virtual environment it was already all poured. That is because the ingredients were really present in the boxes but those were closed. As a result, the weight of the contents was felt and that prevailed over their sight. For the case of Different Objects the only thing we noticed was that users were spending some time trying to figure out what object was really in their hands since we pushed the substitution to the limit in this application. The difference between BHI, 1:1 Replica and Different Objects is not statistically significant (1:1 Replica: p=0.98; Different Objects p=0.53) while the one between BHI and Vive Controllers is significant (p=0.014).
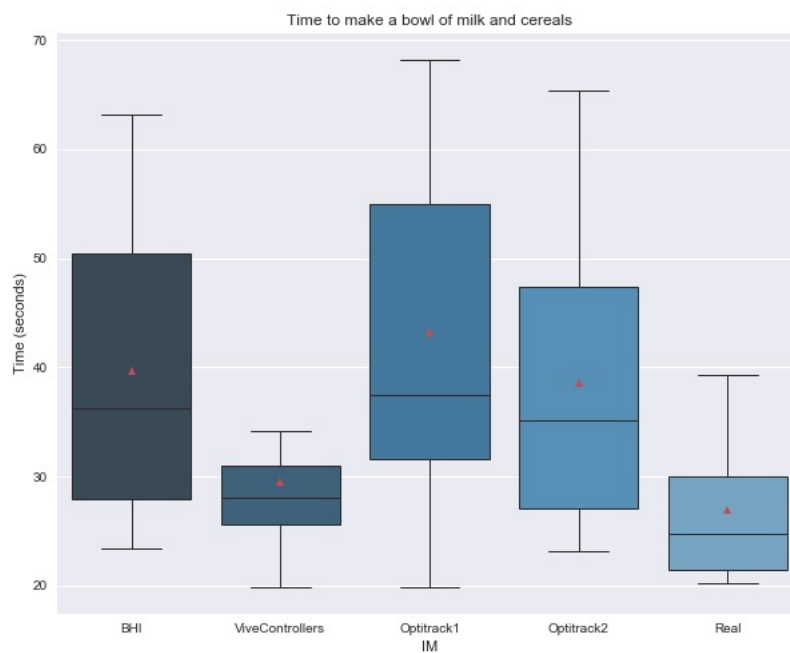
Figure 48: Boxplot of the time spent on the cereal bowl task grouped by interaction method. Triangles represent mean values and lines median values.

5.7.4   Interviews

At the end of the testing of each interaction method and at the end of the experiment we had some brief discussions with the research subjects. In this section we will summarize what emerged.

5.7.4.1   Bare Hands Interaction

Overall this method was the least enjoyed of the ones tested. People reported occasional issues with the tracking of the hands which were suddenly disappearing. The hardest test was the Fitts' Law application where they have to repetitively grab and release an object. What was most annoying is the fact that there is not any feedback telling them that the object was actually grabbed or released, they had to try to move their hands to test the object's response. Another observation was that the movements to do to achieve a particular result were sometimes unnatural and the grab gesture had to be exaggerated as a fully closed fist. On the other hand the immersivity given by the virtual hands reflecting the movements of their real hands was appreciated . Two subjects said it was their favorite interaction method.

5.7.4.2   Vive Controllers

Vive controlles obtained an unexpected success. Five people out of fourteen said that is was their favorite interaction method. The justifications were that the tracking of the controllers was impeccable and and having all the objects only virtual made them feel that they had better control of the situation. Moreover, they said that they were feeling as though they were in a game because there were no consequences for their actions. The interaction was really easy, one person said: *"If I had to describe this interaction with a word, that would be 'click' "* . However,

other people liked it less, because it was not so realistic, one said *"It was like picking objects up using a pair of thongs"*. Concerning the vibrotactive feedback that we tried to emulate, none of the subject connected it to a sort of physicity of the obect. The vibration was more interpreted as a signal saying "ok, now you can press the button to grab the object".

5.7.4.3 1:1 Replica

Overall this method was the most liked, seven people said that it was their favourite. They appreciated the correspondence between touch and sight. The tracking of the object was said to be really accurate. One of the most frequent complaints was about the markers on the objects: sometimes their location interfered with the first grab approach. In general, the fact that they saw a static model of their hands instead of one with moving fingers was not considered important to the end of the interaction.

5.7.4.4 Different Objects

None of the subjects said that this was their favorite interaction method. In the final rankings it was always behind the 1:1 Replica. However we received some conflicting feedback. A minority of the subjects found the discrepancy between touch and sight really annoying, especially when there was a shape mismatch. Others said that it depended on the application. For the Fitts' Law test, for example, they said they were surprised at first to touch a different object but during the execution of the task they did not care. During the cereal test is when the difference was felt the most. In that case the bowl and the cereal box were both physically represented by two carton boxes, and this created confusion with their perceptions. In the Colored Objects application, where we tried to keep the same affordances in the physical objects, the confusion was mitigated

and two subjects reported that they did not notice that the objcets were different. Once they

realized that they were dealing with different objects their performance was not affected.

CHAPTER 6

CONCLUSION

In this thesis we wanted to contribute to the literature on passive haptic systems for virtual reality applications. We proposed an implementation of a system which uses professional optical tracking to track physical objects and integrate them into a virtual environment. The system is used on two different sets of real objects. The first is composed of the same exact replica of the objects present in the virtual environment, the second is composed by objects with several levels of physical characteristics mismatch. The reason we used also different objects is that we wanted to investigate the level of importance given to the details of a physical proxy in a passive haptic feedback system, and how they influence the execution of a task.

We created a test environment to analyze the factors of the object manipulation task execution in virtual reality, such as the accuracy, velocity, time to execution, level of realism and immersion. The test environment was used to compare our passive haptic feedback system to two state-of-art input devices: Leap Motion and Vive Controller. In addition, we compared it to the execution of the same task in the real world.

We formulated a user study in which we asked to fourteen participants to execute three object manipulation tasks with the four different interaction methods. We gathered data through application logs, video recordings, and questionnaires. The results from our analysis show different findings depending on the characteristics of the task.

We can say that enhancing a virtual environment with physical objects increases the level of perceived realism and immersion of the experience. When we analyzed the characteristics of the execution of a movement, we noticed that those done in a passive haptic environment are more similar to those done in the real world than the same actions done with without any touch feedback or using the controllers. In this document we tried to describe the human behaviour during the use of each interaction method to explain the results. Our data highlights the importance of having touch feedback as the tests we conducted using Bare Hands Interaction are the worst over almost all the, categories. We noticed the controllers performed rather well in terms of accuracy, but using them users tend to solve tasks with an approach different than the one used in the real world. The simulation in this way becomes more like a videogame. Since one of the objectives of this research was to find a good realistic interaction methods to be used in applications for physical rehabilitation, we can suggest that the adoption of passive haptics appears to be a better solution.

Concerning the comparison between passive haptics with a 1:1 Replica and Different Objects, we can recommend using as physical proxies objects as similar as possible to the virtual ones or at least with similar shape and affordances in the parts users get in contact with. We noticed that the weight of the object not only does not interfere with the performance (also because we can just estimate the weight of a virtual object), but in some cases it is possible that the performance is improved. For example, in physical rehabilitation a light object could be used to reduce the fatigue for the patient.

For a repeated target acquisition task (like the one we did) where the object does not present any special physical characteristic the discrepancy between touch and sight seems to become of secondary importance, while for tasks which focus more on the object (like making a bowl of milk and cereal) it is important that the physical proxy is close to the virtual one.

Another suggestion that we can make for future work is to try to find a way to display in the virtual environment where the retro-reflective markers are on the object. In this way the user will already know which are the areas to avoid contact with, and the markers would become less invasive. Even just placing a small virtual cube in the corresponding locations could work, but a better fine way would be to incorporate the markers into the design of the virtual objects.

Following these indications it will possible to develop further virtual reality applications which make use of passive haptics to enhance a virtual environment where real objects are a set of 1:1 replicas and different objects according to the type of task. In this way the design and development process will be smoother and it will be possible to use a limited set of tracked object across multiple applications.

Further work should be done to make more consistent our findings. First, more people should be involved to gathrt more data on which to conduct the analysis. Further applications should be developed to collect different kind of data. In general, VR has started spreading in these years and in the HCI field a lot of research work has started to appear on passive haptics. We hope that in future studies our conclusion can be merged, confirmed or also denied by their conclusions.

APPENDICES

Appendix A

FITT'S LAW MANAGER

Here is listed part of the code that we developed for the Fitt's law application:

```csharp
1  using System.Collections;
2  using System.Collections.Generic;
3  using System.IO;
4  using UnityEngine;
5
6  public class FittsLawManager : MonoBehaviour
7  {
8  public GameObject target;
9  public int nTargets;
10 public FittsLawObject fittsLawObject;
11 public AudioSource correctSound;
12 public float[] distances = new float[] { 0.1f, 0.15f, 0.16f };
13 public float[] widths = new float[] { 0.05f, 0.02f };
14 public string subjectId;
15 public string trialId;
16
17 private int[] setsDone;
18 private int current;
19 private int targetCount;
20 private int setsCount = 0;
21 private int nSet;
22 private List<GameObject> targets;
23 private bool targetHit = false;
24 private string iM;
25 private StreamWriter outStream;
26 private float currentDistance;
27 private float currentTargetRadius;
28 private bool firstTarget;
29 private float startTime;
30 private float endTime;
31 private bool objectLeftTarget = false;
32
33 void Start()
34 {
35   subjectId = PlayerPrefs.GetInt("SubjectId").ToString();
36   trialId = PlayerPrefs.GetInt("TrialId").ToString();
37   iM = PlayerPrefs.GetString("IM");
38   CreateFile();
```

Appendix A (continued)

```csharp
39    WriteFirstLine();
40    nSet = distances.Length * widths.Length;
41    setsDone = new int[nSet];
42    System.Array.Clear(setsDone, 0, setsDone.Length);
43  }
44
45  void Update()
46  {
47    if (targetHit && fittsLawObject.isReleased)
48    {
49      NextTarget();
50    }
51  }
52
53  private void OnGUI()
54  {
55    if (GUI.Button(new Rect(20, 80, 200, 30), "Start Test"))
56        {
57      NextSet();
58        }
59  }
60
61  private void OnDestroy()
62  {
63    outStream.Close();
64  }
65
66  private void CreateFile()
67  {
68    string filePath = Application.dataPath + "/CSV/" + "Fitts_" + subjectId +
          "_" + trialId + "_" + iM + "_[" + System.DateTime.Now.DayOfYear + "_"
          + System.DateTime.Now.Hour + "x" + System.DateTime.Now.Minute +
          "].csv";
69    outStream = System.IO.File.CreateText(filePath);
70  }
71
72  private void WriteFirstLine()
73  {
74    string[] rowDataTemp = new string[4];
75    rowDataTemp[0] = "Width";
76    rowDataTemp[1] = "Distance";
77    rowDataTemp[2] = "Time";
78    rowDataTemp[3] = "TimeStamp";
79    int length = rowDataTemp.Length;
80    string delimiter = ",";
81    string output = string.Join(delimiter, rowDataTemp);
82    outStream.WriteLine(output);
83  }
84  private void WriteLog(float time)
85  {
```

Appendix A (continued)

```
 86    string delimiter = ",";
 87    string[] rowDataTemp = new string[4];
 88    rowDataTemp[0] = currentTargetRadius.ToString();
 89    rowDataTemp[1] = currentDistance.ToString();
 90    rowDataTemp[2] = time.ToString();
 91    rowDataTemp[3] = endTime.ToString();
 92    string output = string.Join(delimiter, rowDataTemp);
 93    outStream.WriteLine(output);
 94  }
 95
 96  private void NextSet()
 97  {
 98    firstTarget = true;
 99    setsCount++;
100    if (setsCount > nSet)
101      return;
102    int set;
103    do
104    {
105      set = Random.Range(0, nSet);
106    }
107    while (setsDone[set] != 0);
108    setsDone[set] = 1;
109    currentDistance = distances[set / widths.Length];
110    currentTargetRadius = widths[set % widths.Length];
111    GenerateTargets(currentDistance, currentTargetRadius);
112    current = 0;
113    targetCount = 0;
114    ActivateTarget(current);
115  }
116  private void GenerateTargets(float distance, float targetRadius)
117  {
118    targets = new List<GameObject>();
119
120    for (int i = 0; i < nTargets; i++)
121    {
122      float originX = transform.position.x;
123      float originZ = transform.position.z;
124      Vector3 tmpPosition = new Vector3(originX + Mathf.Cos(2 * Mathf.PI /
          nTargets * i) * distance, transform.position.y + 0.01f, originZ +
          Mathf.Sin(2 * Mathf.PI / nTargets * i) * distance);
125      GameObject t = (GameObject)Instantiate(target, tmpPosition,
          Quaternion.identity);
126      t.transform.localScale = new Vector3(targetRadius * 2,
          t.transform.localScale.y, targetRadius * 2);
127      t.GetComponent<FittsLawTarget>().DisableTarget();
128      t.GetComponent<FittsLawTarget>().SetManager(this);
129      targets.Add(t);
130    }
131
132  }
```

Appendix A (continued)

```csharp
133
134  private void DeleteAllTargets()
135  {
136    foreach (GameObject t in targets)
137    {
138      Destroy(t);
139    }
140  }
141
142  private void ActivateTarget(int index)
143  {
144    targets[index].GetComponent<FittsLawTarget>().ActivateTarget();
145  }
146
147  private void DisableTarget(int index)
148  {
149    targets[index].GetComponent<FittsLawTarget>().DisableTarget();
150  }
151
152  private void NextTarget()
153  {
154    if(firstTarget){
155      startTime = Time.time;
156      firstTarget = false;
157    }else{
158      endTime = Time.time;
159      WriteLog(endTime - startTime);
160      startTime = Time.time;
161    }
162    correctSound.Play();
163    targetCount++;
164    if (targetCount == nTargets)
165    {
166      targetCount = 0;
167      DeleteAllTargets();
168      NextSet();
169      targetHit = false;
170      return;
171    }
172    int next;
173    if (nTargets % 2 == 0)
174    {
175      if (current < (nTargets / 2))
176      {
177        next = (current + (nTargets / 2) + 1) % nTargets;
178      }
179      else
180      {
181        next = (current + (nTargets / 2)) % nTargets;
182      }
183    }
```

Appendix A (continued)

```
184    else
185    {
186      next = (current + (nTargets / 2) + 1) % nTargets;
187    }
188    DisableTarget(current);
189    ActivateTarget(next);
190    current = next;
191    targetHit = false;
192    objectLeftTarget = false;
193  }
194
195  public void TargetHit()
196  {
197    targetHit = true;
198  }
199
200  public void TargetExit(){
201    if(!objectLeftTarget){
202      startTime = Time.time;
203          objectLeftTarget = true;
204    }
205  }
206  }
```

Appendix B

DEMOGRAPHIC QUESTIONNAIRE

Subject ID: _____

<div align="center">

Demographic Questionnaire

"A Study on Passive Haptics for Virtual Reality Experiences"
University of Illinois at Chicago
Department of Computer Science

</div>

1. What is your age?
   a. 18 – 24 years old
   b. 25 – 34 years old
   c. 35 – 44 years old
   d. 45 – 54 years old
   e. 55 – 64 years old
   f. 65 – 74 years old
   g. 75 or older

2. What is your gender?

   _____

3. What is your experience level with VR HMDs (like the one you will be wearing today)?

1	2	3	4	5
Never used before	Used once or twice	Used Monthly	Used Weekly	Used Daily

4. How often do you play videogames?

1	2	3	4	5
Never	Played once or twice	Play Monthly	Play Weekly	Play Daily

5. What is your experience level with smartphones?

1	2	3	4	5
Never used before	Used once or twice	Used weekly or just for calls	Used daily	Used daily more than 4 apps

6. What is your experience level with PCs?

1	2	3	4	5
Used once or twice	Used Monthly	Used weekly or just for work	Used daily	Used daily for many hours

Figure 49: Demographic questionnaire

Appendix C

EVALUATION QUESTIONNAIRE

Subject ID: _____
Trial #: ____

Final Questionnaire

"A Study on Passive Haptics for Virtual Reality Experiences"
University of Illinois at Chicago
Department of Computer Science

1. Realism
   How realistic was the experience overall?

   Not at all   A lot

1	2	3	4	5

2. Immersion
   How immersed were you in the virtual environment while interacting with the virtual objects?

   Not at all   A lot

1	2	3	4	5

3. Enjoyment
   Did you enjoy interacting with the virtual objects?

   Not at all   A lot

1	2	3	4	5

4. Similarity
   Did you think that the physical objects that you were touching were the same you were seeing?

   For nothing   Same objects

1	2	3	4	5

5. Ease to use
   How easy was it to interact with the virtual environment?

   Not at all   Really easy

1	2	3	4	5

1

Figure 50: Evaluation questionnaire

Appendix D

NASA TLX QUESTIONNAIRE

The NASA TLX questionnaire presented to research subjects at the end of each trial.

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Subject code	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Figure 51: NASA TLX questionnaire

Appendix E

IEEE MATERIAL PERMISSION

Figure 52: Permission grant to use IEEE material

CITED LITERATURE

1. Rizzo, A. S. and Kim, G. J.: A swot analysis of the field of virtual reality rehabilitation and therapy. Presence: Teleoperators and Virtual Environments, 14(2):119–146, October 2005.

2. Burke, J. W., McNeill, M. D. J., Charles, D. K., Morrow, P. J., Crosbie, J. H., and Mc-Donough, S. M.: Optimising engagement for stroke rehabilitation using serious games. The Visual Computer, 25(12):1085, Aug 2009.

3. Milgram, P. and Kishino, F.: A taxonomy of mixed reality visual displays. IEICE Transactions on Information Systems, 77(12):1321–1329, December 1994.

4. Society, V. R.: What is virtual reality? `https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html`, 2018. [Online; accessed 11/28/18].

5. Society, V. R.: History of virtual reality. `https://www.vrs.org.uk/virtual-reality/history.html`, 2018. [Online; accessed 11/28/18].

6. Ware, C., Arthur, K., and S. Booth, K.: Fish tank virtual reality, 01 1993.

7. Howls, T. G.: All you need to know about steamvr tracking. `https://skarredghost.com/2017/06/07/need-know-steamvr-tracking-2-0-will-foundation-vive-2/`, 2018. [Online; accessed 11/28/18].

8. Gallace, A. and Spence, C.: In touch with the future: The sense of touch from cognitive neuroscience to virtual reality.. Oxford University Press, 2014.

9. English, O. D. .: haptic | definition of haptic in english by oxford. `https://en.oxforddictionaries.com/definition/haptic`, 2018. [Online; accessed 11/28/18].

10. Yu, N., Wang, K., Li, Y., Xu, C., and Liu, J.: A haptic shared control approach to teleoperation of mobile robots, 08 2015.

CITED LITERATURE (continued)

11. Hoffman, H. G.: Physically touching virtual objects using tactile augmentation enhances the realism of virtual environments. Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No.98CB36180), pages 59–63, March 1998.

12. Simeone, A. L., Velloso, E., and Gellersen, H.: Substitutional reality: Using the physical environment to design virtual reality experiences. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15, pages 3307–3316, New York, NY, USA, 2015. ACM.

13. Pan, M. K. X. J. and Niemeyer, G.: Catching a real ball in virtual reality. In 2017 IEEE Virtual Reality (VR), pages 269–270, March 2017.

14. Shapira, L., Amores, J., and Benavides, X.: Tactilevr: Integrating physical toys into learn and play virtual reality experiences. In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 100–106, Sept 2016.

15. Yoshimoto, R. and Sasakura, M.: Using real objects for interaction in virtual reality. In 2017 21st International Conference Information Visualisation (IV), pages 440–443, July 2017.

16. Kenyon, R. V. and Afenya, M. B.: Training in virtual and real environments. Annals of Biomedical Engineering, 23(4):445, Jul 1995.

17. Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A.: Surround-screen projection-based virtual reality: The design and implementation of the cave. In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, pages 135–142, New York, NY, USA, 1993. ACM.

18. Soukoreff, R. W. and MacKenzie, I. S.: Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in hci. Int. J. Hum.-Comput. Stud., 61(6):751–789, December 2004.

19. Fitts, P. M.: The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental PSychology, 74:381–391, 1954.

20. Motion, L.: How does the leap motion controller work? `http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/`, 2018. [Online; accessed 11/23/18].

CITED LITERATURE (continued)

21. Motion, L.: Unity modules: Getting started - the basic components of interaction. `https://leapmotion.github.io/UnityModules/interaction-engine.html#ie-in-depth`, 2018. [Online; accessed 11/23/18].

22. Software, V.: Steamvr unity plugin. `https://github.com/ValveSoftware/steamvr_unity_plugin`, 2018. [Online; accessed 11/17/18].

23. Unity: Steamvr unity plugin. `https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647`, 2018. [Online; accessed 11/17/18].

24. Rietzler, M., Geiselhart, F., Frommel, J., and Rukzio, E.: Conveying the perception of kinesthetic feedback in virtual reality using state-of-the-art hardware. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18, pages 460:1–460:13, New York, NY, USA, 2018. ACM.

25. de Kerckhove, E. V.: Htc vive tutorial for unity. `https://www.raywenderlich.com/792-htc-vive-tutorial-for-unity`, 2016. [Online; accessed 11/17/18].

26. Corporation, H.: Htc vive tracker. `https://www.vive.com/us/vive-tracker/`, 2018. [Online; accessed 11/17/18].

27. Wikipedia: Positional tracking - wikipedia. `https://en.wikipedia.org/wiki/Positional_tracking`, 2018. [Online; accessed 11/20/18].

28. OptiTrack: Optitrack - prime 13w. `https://optitrack.com/products/prime-13w/`, 2018. [Online; accessed 11/20/18].

29. OptiTrack: Motive api: Function reference - naturalpoint product documentation ver 2.0. `https://v20.wiki.optitrack.com/index.php?title=Reconstruction_and_2D_Mode#Marker_Rays`, 2018. [Online; accessed 11/21/18].

30. OptiTrack: Motive:tracker - motion capture and 6 dof object. `https://optitrack.com/products/motive/tracker/features-specs.html`, 2018. [Online; accessed 11/21/18].

31. Roo, J. S., Basset, J., Cinquin, P.-A., and Hachet, M.: Understanding users' capability to transfer information between mixed and virtual reality: Position estimation across modalities and perspectives. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18, pages 363:1–363:12, New York, NY, USA, 2018. ACM.

CITED LITERATURE (continued)

32. Wikipedia: Suspension of disbelief - wikipedia. `https://en.wikipedia.org/wiki/Suspension_of_disbelief`, 2018. [Online; accessed 11/24/18].

33. Bustamante, E. A. and Spain, R. D.: Measurement invariance of the nasa tlx. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 52(19):1522–1526, 2008.

34. MacKenzie, I. S. and Buxton, W.: Extending fitts' law to two-dimensional tasks. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '92, pages 219–226, New York, NY, USA, 1992. ACM.

35. Murata, A. and Iwase, H.: Extending fitts' law to a three-dimensional pointing task. Human Movement Science, 20(6):791–805, 2001.

36. Cha, Y. and Myung, R.: Extended fitts' law for 3d pointing tasks using 3d target arrangements. International Journal of Industrial Ergonomics, 43(4):350 – 355, 2013.

37. J. Gibson, J.: The theory of affordances chapt. The Ecological Approach to Visual Perception, 8., 01 1977.

VITA

NAME	Francesco Mantovani

EDUCATION

Master of Science in " Computer Science ", University of Illinois at Chicago, May 2019, USA

Specialization Degree in " Computer Engineering ", Apr 2019, Polytechnic of Turin, Italy

Bachelor's Degree in Computer Engineering, Oct 2016, University of Modena and Reggio Emilia, Italy

LANGUAGE SKILLS

Italian	Native speaker
English	Full working proficiency

A.Y. 2017/18 One Year of study abroad in Chicago, Illinois

A.Y. 2016/17. Lessons and exams attended exclusively in English

SCHOLARSHIPS

Summer 2018	Research Assistantship (RA) position (20 hours/week) with full tuition waiver plus monthly stipend
Spring 2018	Research Assistantship (RA) position (20 hours/week) with full tuition waiver plus monthly stipend
Spring 2018	Italian scholarship for final project (thesis) at UIC