A Multiple Optical Tracking Based Approach for Enhancing Hand-Based Interaction in Virtual Reality Simulations

Adam Grant Worrallo

BSc (Hons), MSc

A thesis submitted in partial fulfilment of the

requirements of the University of Wolverhampton

for the degree of Doctor of Philosophy

September 2020

Signature……A G Worrallo……

Date……26/09/2020……

UNIVERSITY OF WOLVERHAMPTON

Friday 25 November 2016

Dear Mr Warrollo

RE: "Comparison of Gesture Based Input Sensors"

Thank you for submitting your ethics application form to the Life Sciences Ethics Committee for internal review.

I am pleased to informed you that your project 'Comparison of Gesture Based Input Sensors' was reviewed by LSEC on Tuesday 22 November 2016.

The LSEC advises that the following amendments are required:

Student:	Adam Worrallo (PG) LSEC/2016/17/FL/59
Supervisor:	Thomas Hartley, Fernando Loizides and Kevan Buckley
Project Title:	Comparison of Gesture Based Input Sensors
Outcome: 22/11/2016	**Approved Subject to Amendments**
Amendments:	**Bill of Rights – 'To receive something in return for your time and energy' and 'To be offered a range of research studies or experience from which to select, if the research is part of fulfilling your educational or employment goals' both statements to be removed.** **Consent Form and Bill of Rights are to be on University headed notepaper.**
25/11/2016	**Amendments Received**
Outcome: 25/11/2016	**Approved**

The Committee wishes you every success in your research project.

Kind regards,

[redacted]

Professor Tracy Warr
Professor of Neuro-oncology
Chair of the Life Sciences Ethics Committee
Faculty of Science and Engineering
University of Wolverhampton

Dean: Professor Nazira Karodia PhD CChem FRSA SFHEA
University of Wolverhampton, Faculty of Science and Engineering, Technology Centre, Alan Turing Building,
City Campus Wulfruna, Wulfruna Street, Wolverhampton WV1 1LY, United Kingdom
T: +44 01902 322129 F: +44 01902 321478 W: www.wlv.ac.uk/FSE

Acknowledgements

Throughout my PhD I have received a great deal of support and advice. I would like to first thank my supervisor, Dr Tom Hartley whose expertise, advice and endless support was invaluable throughout my studies. In addition, I would like to thank my PhD administrator, friends and colleagues at the University of Wolverhampton for their support.

I would also like to thank my entire family for their endless support and encouragement throughout my studies. You were always there for me with a sympathetic ear and made the hardest parts seem that much easier. I would like to say a special thank you to my uncles; Dave and Tim, you lit and nurtured my passion for computer science in my early years. I would not be where I am today without your input.

Finally, I need to say a special thank you to my father, Mark, for always being there for me no matter what. From my first day of school, you have always supported me with my studies, going to the library with me on a Saturday morning and helping with homework in the afternoon. It's been many years since we did homework together, but your support has never wavered. Your endless support during this endeavour has been invaluable. Whether it was offering advice or just a friendly face at the end of a long day, I could not have done this without you.

Abstract

Research exploring natural virtual reality interaction has seen significant success in optical tracker-based approaches, enabling users to freely interact using their hands. Optical based trackers can provide users with real-time, high-fidelity virtual hand representations for natural interaction and an immersive experience. However, work in this area has identified four issues: occlusion, field-of-view, stability and accuracy.

To overcome the four key issues, researchers have investigated approaches such as using multiple sensors. Research has shown multi-sensor-based approaches to be effective in improving recognition accuracy. However, such approaches typically use statically positioned sensors, which introduce body occlusion issues that make tracking hands challenging. Machine learning approaches have also been explored to improve gesture recognition. However, such approaches typically require a pre-set gesture vocabulary limiting user actions with larger vocabularies hindering real-time performance.

This thesis presents an optical hand-based interaction system that comprises two Leap Motion sensors mounted onto a VR headset at different orientations. Novel approaches to the aggregation and validation of sensor data are presented. A machine learning sub-system is developed to validate hand data received by the sensors. Occlusion detection, stability detection, inferred hands and a hand interpolation sub-system are also developed to ensure that valid hand representations are always shown to the user. In addition, a mesh conformation sub-system ensures 3D objects are appropriately held in a user's

virtual hand. The presented system addresses the four key issues of optical sessions to provide a smooth and consistent user experience.

The MOT system is evaluated against traditional interaction approaches; gloves, motion controllers and a single front-facing sensor configuration. The comparative sensor evaluation analysed the validity and availability of tracking data, along with each sensors effect on the MOT system. The results show the MOT provides a more stable experience than the front-facing configuration and produces significantly more valid tracking data. The results also demonstrated the effectiveness of a 45-degree sensor configuration in comparison to a front-facing. Furthermore, the results demonstrated the effectiveness of the MOT systems solutions at handling the four key issues with optical trackers.

Contents

List of Figures

List of Equations

List of Algorithms

List of Tables

List of Abbreviations

2D – Two dimensional

3D – Three dimensional

AABB – Axis-aligned bounding box

ANOVA – Analysis of Variance

API – Application Programming Interface

ATP – Arm Transfer Packet

BTP – Bone Transfer Packet

CAVE – Cave Automatic Virtual Environment

CNN – Convolutional Neural Network

df – degrees of freedom in statistical analysis

DNN – Deep Neural Networks

DoF – Degree of Freedom in 3D manipulation (position & rotation)

FPS – Frames Per Second

FRTP – Frame Transfer Packet

FTP – Finger Transfer Packet

GAN – Generative Adversarial Network

GPU – Graphics Processing Unit

HCI – Human Computer Interaction

HMD – Head mounted display

HTP – Hand Transfer Packet

IP – Internet Protocol address

IR – Infrared

IVR – Immersive Virtual Reality

kNN – K-nearest neighbour

LCD – Liquid Crystal Display

LED – Light Emitting Diode

LI – Leap Interaction Engine

MLP – Multilayer Perceptron

MLTP – MultiLeap Transfer Packet

MOT – Multiple Optical Tracking

NB - Naïve Bayes

NN – Neural Networks

NUI – Natural User Interfaces

PLA - Polylactic acid, or polylactide is a thermoplastic polyester

SDK – Software Development Kit

SGEQ – Standardized Gesture Evaluation Questionnaire

SPES – Spatial Presence Experience Scale

SSQ – Simulator Sickness Questionnaire

SUS – System Usability Scale

SVM – Support Vector Machine

UI – User Interface

USB – Universal Serial Bus

VR – Virtual Reality

VUI – Virtual User Interface

Chapter 1 – Introduction

1.1 – Virtual Reality

Virtual reality (VR) is the use of hardware and software to create interactive three-dimensional computer-generated environments, worlds, simulations or visualisations that immerse a person and enable them to feel as though they have presence within a virtual world. To create immersive experiences computer controlled electronic equipment is used to feed sensory input to users and monitor their actions. A wide variety of technology is used to create virtual reality experiences, including, displays, haptic gloves, sound devices and input devices. Figure 1.1 contains a taxonomy of virtual reality technology within the commercial sector. The taxonomy is broken down into five different categories that make up our real-world senses (El Saddik, et al., 2011; Muhanna, 2015). This approach to organising VR technology was inspired by the "Ultimate Display" concept envisioned by Ivan Sutherland in 1966 (Sutherland, 1968). In the concept, all a user's real-world senses are replicated so that the virtual world appears as real as the physical world.

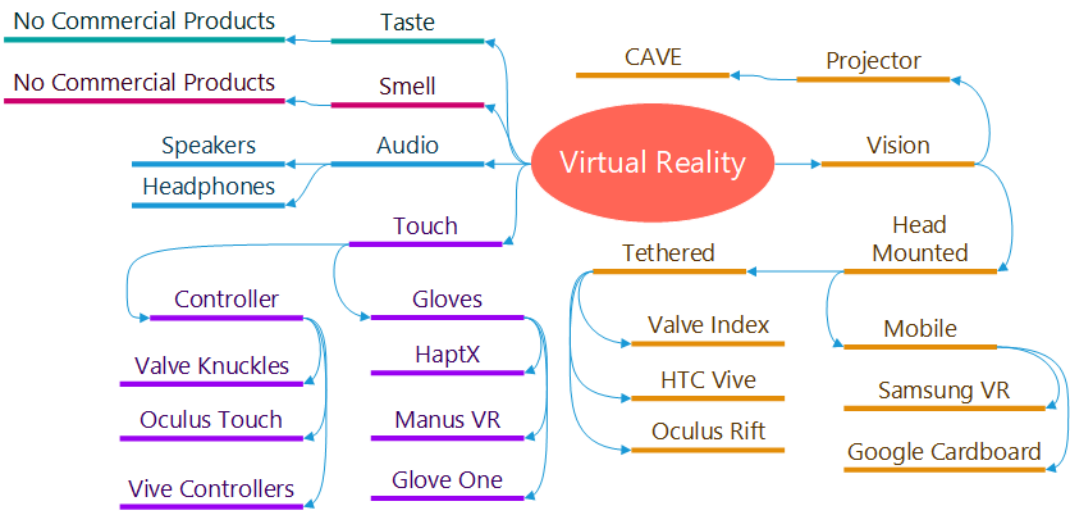


Figure 1.1 - Taxonomy of Commercial Virtual Reality Technology

The vision section of Figure 1.1 (e.g. virtual reality headsets or head mounted displays (HMD)) is the most critical technology needed for virtual reality as sight plays a key role in creating an immersive experience that enables a user to feel part of the virtual environment. Virtual reality headsets are the most prominent consumer display technology. They typically have two small displays with lenses that output three-dimensional computer-generated images. By using a headset, the illusion of depth perception can be created thus a more immersive experience is provided to the user.

Commercial virtual reality headset technology, such as the Oculus Rift S (2019) and Valve Index (2019) has developed significantly over recent years and has now reached a level of high visual immersion, relatively low cost and robust quality. For example, modern displays used in HMD's are capable of frame rates more than 120hz providing a much smoother visual experience. Furthermore, modern HMD's use significantly higher resolutions than their predecessors which in combination with modern GPU technology enables users to experience more realistic simulations that have higher visual fidelity. Most recently, mobile VR headsets such as the Oculus Quest have added inside-out tracking enabling room-scale tracking without the need for tracking base stations. Figure 1.2 illustrates the difference between the two main types of commercial headset tracking. Outside-In tracking uses static sensors for positional tracking. While Inside-Out tracking uses Infrared cameras to track the headsets position relative to a series of points within the real-world.

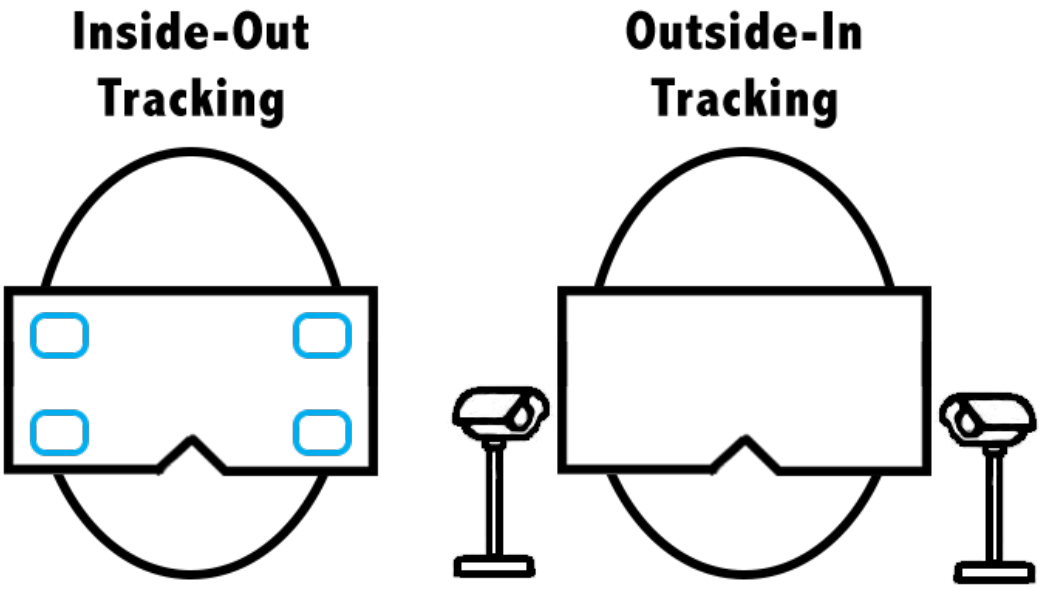


Figure 1.2 - Diagram showing the difference between inside-out and outside-in tracking

The Touch section of Figure 1.1 outlines the current input devices available for commercial virtual reality technology. Current Virtual Reality simulations typically make use of tactile modality devices, such as gamepads, keyboards and mice or motion controllers to enable the user to interact with the virtual world (Lee, 2016; Jarvis, 2016; Buchanan, 2018). Motion controllers are handheld devices that provide users with an interaction mechanism via traditional inputs such as buttons, triggers and joysticks. To provide precise positioning within the 3D virtual environment, motion controllers use HMD tracking technology. In the case of the Oculus Touch controllers, infrared (IR) LEDs are tracked by cameras on the headset to determine the position of the controllers. In the case of the Vive, the base stations emit and sweep IR beams across the physical VR space, one axis at a time. To calculate their position, the headset and controllers are equipped with an array of IR photodiodes that

measure the time between an IR flash and being hit by the laser sweep (Heaney, 2019).

Prominent Virtual Reality headset developers Oculus (2019) and Valve (2019) have created commercial virtual reality motion controllers that incorporate finger tracking technology and traditional game input. The Oculus controllers use buttons to provide the gesture inputs ‘Grab’ and ‘Release’, which in turn toggle virtual hands between an open and fist pose. The Valve Knuckle controllers use a touch-capacitive grip to determine both grip strength and the extension state of fingers by determining which parts of the fingers are in contact with the surface. This provides the virtual hands with additional intermediary animation steps as opposed to only an open and fist pose.

Motion controllers can be considered intuitive because of their use of movement tracking and traditional input technology such as triggers, buttons and trackpads. However, while familiar, the use of traditional inputs limits the user’s level of interaction to the number of buttons/functions the controller can perform. Furthermore, motion controllers can introduce significant learning curves, especially in more complex simulations. Thereby, limiting the level of immersion the user can achieve within the simulation (Gallotti, et al., 2011; Yan & Aimaiti, 2011; Bowman, et al., 2012; Anthes, et al., 2016; Navarro & Sundstedt, 2019).

The limited and un-natural interaction that motion controllers provide is particularly problematic in educational/training applications. The use of HMD tracking technology provides an accurate and stable interaction experience. However, the un-natural interaction prevents the direct transfer of real-world

experience to and from the virtual world. Research such as that of Feng et al (2018) and Krokos et al (2019) has shown the significant impact immersive VR has on the effectiveness of training/education-based simulations. The replication of real-world senses and interactions is a significant factor in a user's sense of immersion and presence as discussed further in section 2.4.

1.2 – Serious Games

Serious games refer to the use of digital games for educational purposes or solving complex problems through collaboration (Freire, et al., 2016). Serious games provide users with an engaging experience that requires underlying knowledge which the user must understand to aid in the completion of the game. Furthermore, serious games provide users the opportunity to practice and hone skills that can then be applied to real-world situations.

Serious games are used in a variety of areas including medical and military applications, training people on procedures and using skilled equipment. Researchers such as Zhang et al (2018) and Ko et al (2019) have shown the effectiveness of serious games in medical training applications, developing simulations for endoscopic and laparoscopic surgery respectively.

The ability to create an immersive virtual environment using virtual reality offers huge potential to the area of serious games. Research has shown that an immersive virtual reality (IVR) experience can have significant impact on the effectiveness of a training scenario and improve memory recall (Krokos, et al., 2019). Furthermore, research has shown IVR experiences can provide a higher degree of engagement in comparison to non-immersive experiences (Gao, et al., 2017; Feng, et al., 2018). Virtual reality enables users to

experience an engaging immersive virtual environment, providing an ideal platform for serious games. Natural interaction has been found to significantly effect a user's sense of immersion, with hand-based interaction found to be particularly beneficial (Han & Kim, 2017; Almeida, et al., 2019). Hand-based interaction enables users to practice and refine real-world hand movements into muscle-memory with the skills transferred into the real-world.

As we can see serious games are effective in creating an engaging training experience, offering benefits such as improved memory recall. Virtual reality can provide significant benefits to serious/games training by providing a 3D virtual world in which the user can better visualize the scenario and take in all the visual information. However, the previously discussed research has shown a significant factor in VR immersion is the ability to interact using hand-based interaction for natural and realistic gestures.

1.3 – Hand Interaction

Researchers are currently exploring a variety of more natural interaction approaches that make use of technology, such as haptic gloves and optical based trackers (Hannema, 2001; Park, et al., 2017; Breslauer, et al., 2019). In particular, work exploring the use of optical based trackers has seen significant success (Chaudhary, et al., 2013; Rautaray & Agrawal, 2015; Anthes, et al., 2016) as this technology frees users hands by allowing them to freely interact with a virtual environment without an interaction device affecting their hands (e.g. holding or wearing).

Optical trackers are camera-based sensors designed to capture user input without direct interaction with an intermediary device. Common approaches

include hand and skeletal tracking with current prominent devices including the Leap Motion and Kinect V2. The Leap Motion is an infrared based stereoscopic camera that uses infrared light to capture a user's hand and finger movements (Wozniak, et al., 2016; Guzsvinecz, et al., 2019). Figure 1.3 shows a Leap Motion sensor.

Figure 1.3 - Leap Motion Sensor (Motion, 2013)

In contrast to controller-based interaction, hand interaction enables users to interact with the virtual environment using their own hands. In VR hand interaction approaches typically use gestures to interact with virtual objects. The gestures used for hand interaction are generally simplified versions of the real-world equivalent. To pick up a virtual object, users typically use the 'grab and release' gesture, making a fist near a virtual object to hold it until they open their hand. One of the main strengths of hand interaction is the intuitive, natural interaction with the ability to leverage real-world experience to interact with the simulation. The intuitive and natural interaction also helps to create a more immersive experience, which as discussed in sections 2.4 and 2.5, is particularly beneficial in educational/training applications.

Hand interaction within a virtual environment typically takes one of two forms; indirect and direct interaction. Indirect interaction uses traditional tactile

devices such as motion controllers and employs either ray-based pointing or proximity-based object interaction. Ray-based interaction uses a laser for object selection while proximity-based interaction selects the object within proximity of the controller.

Optical trackers use a direct interaction approach, tracking the individual fingers of each hand and providing a real-time 3D virtual hand representation (Moehring & Froehlich, 2011; Li & Dai, 2014). Virtual hand representations provide users with a more natural and immersive hand interaction experience, enabling them to perform realistic interactions. Users interact with virtual objects using hand gestures that often mimic real-world actions, typically performing a fist pose for object selection. However, the individual finger tracking capabilities enables users to perform more complex and precise gestures in optimal conditions such as performing telerobotic surgery (Zhou, et al., 2016).

1.4 – Optical Sensor Issues

As previously discussed, optical-based hand trackers can provide users with real-time, high-fidelity virtual hand representations for a natural and immersion experience. However, work in this area has identified four key issues, specifically; occlusion, field-of-view, stability and accuracy. These issues negatively effect the user experience through introducing inconsistency and limiting the sense of naturalism and immersion of the interaction experience.

The single axis view that optical trackers have results in occlusion, in which the sensors view of the user's fingers is obstructed. This is typically caused by the back of the hand during interactions. However, it can also be as the result

of two-handed interactions, with one hand obscuring the sensors view of the other. The obstructed sensor view can result in data instability with virtual hands disappearing as a result. This breaks the user's interaction and sense of immersion, having to wait for the tracking to resume before continuing the interaction. In addition, the disappearing hands impacts the sense of naturalism in the interaction mechanism with hands never disappearing in the real-world.

Research such as that of Marin et al (2014), Clark and Moodley (2016) and McCartney et al (2015) has shown the use of machine learning to reduce occlusion and improve recognition. However, such approaches typically use gesture vocabularies limiting the user interaction. In order to overcome the issue of occlusion in optical-based hand tracking, this research explores the use of multiple sensors to provide an additional view of the user's hands. The data from the sensors is aggregated together to produce optimal tracking data. A novel occlusion detection system is also explored that presents an inferred virtual hand to the user during times of occlusion. This approach was taken to reduce and handle occlusion whilst ensuring users have unrestricted hand interaction.

As previously discussed, optical trackers use a single axis view to track the user's hands assuming they are not occluded. However, the use of camera technology results in the sensors having a limited field-of-view, outside of which the hands cannot be tracked. Furthermore, sensors have an optimal tracking area outside of which the accuracy and stability increasingly degrade as the edges are approached (Colgan, 2015). Figure 1.4 shows the field-of-view of the Leap Motion sensor and the optimal tracking area. The accuracy

and stability within the optimal tracking area can provide an engaging experience. However, ensuring their hands are within this area, heavily restricts user interaction and reduces the natural feeling. Modern VR headsets have a field-of-view significantly larger than the optimal tracking area; specifically, 110-degrees for the HTC Vive and 135-degrees for the Valve index (Valve, 2019; Vive, 2020). The large field-of-view typically results in users attempting to interact outside the optimal area and experiencing issues with the field-of-view and accuracy. The issues reduce the user's sense of immersion and the natural feeling of the interaction.

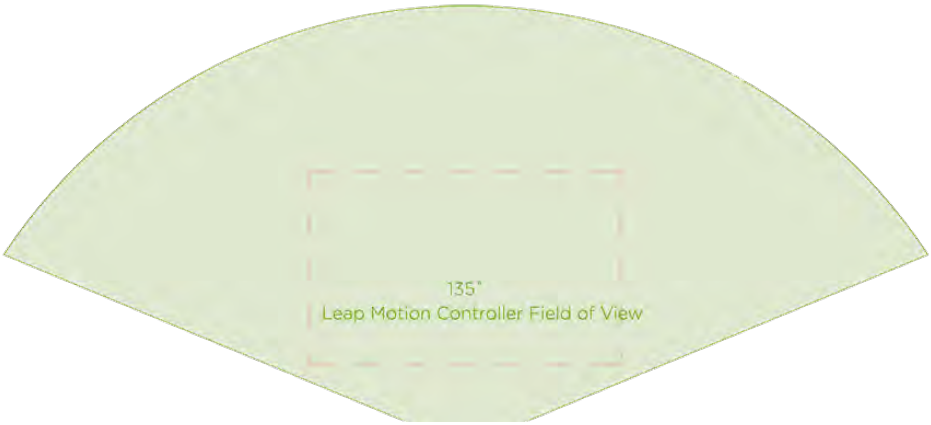


Figure 1.4 - Diagram showing the field of view of the Leap Motion and the optimal tracking area

The single-axis view of optical trackers and the use of camera technology also introduces general issues with accuracy and stability. Optical trackers such as the Leap Motion and Kinect use infrared depth images and thus can be affected by excessive light or reflections. In addition, the limited view also requires hand and finger positions to be calculated from the depth images of a single plane. These factors affect the user experience with precise control

being sometimes difficult and can result in finger twitches and movements without user input.

Research such as that of Marin et al (2014) and Jin et al (2016) have explored the use of multiple sensors to increase the tracking field-of-view. However, approaches have typically used statically positioned sensors such as the Kinect to provide additional data. While effective such solutions suffer from two key issues, with the first being the static position of the sensor. The additional sensors position results in the user's body occluding the sensors view as they move and rotate. The second issue is that such solutions typically use sensors such as the Kinect that are not designed for hand tracking and rely on custom implementations using image processing. In order to overcome the issues of limited field-of-view and instability in optical-based hand tracking, this research explores the use of multiple sensors to increase the field-of-view and provide an additional view of the user's hands. This research uses commercial sensors specifically designed for optical-based hand tracking. Furthermore, the sensors used are attached to the VR HMD to overcome the issues of statically positioned sensors. The data from the two sensors is aggregated together to produce the optimal tracking data. A novel motion-controller based tracking system is also explored that enables hand tracking and interaction outside of the field-of-view of the optical tracking sensors.

In order to overcome the issue of instability in optical-based hand tracking, this research explores the use of previous sensor tracking data to smooth inconsistencies and stabilize tracking data. A novel stability analysis system is also explored that presents an inferred virtual hand to the user during time of tracking instability for a more consistent user experience.

The issues of instability, occlusion and noise can produce in inaccuracies with optical-based hand tracking data, resulting in the virtual hand appearing deformed. Previous research approaches have relied on machine learning and a gesture vocabulary against which tracking data can be evaluated and matched. These approaches can be effective; however, the gesture vocabulary limits hand poses/gestures and as a result, the user interaction experience. In order to overcome the issues with hands deforming and un-natural poses as a result of the aforementioned factors, this research explores the use of machine learning to validate optical-based hand tracking data. A novel hand validation system is also explored that analyses multiple hand features to ensure virtual hand movement is within the natural twenty-seven degrees of freedom with which the human hand is modelled. The novel approach explored can validate tracking data in real-time without the need for a gesture vocabulary, enabling more natural and unrestricted interaction.

As previously discussed, hand-based interaction approaches typically present the user with virtual hand representations and use simplified versions of real-world gestures, most commonly ‘grab and release’. These controls enable users to interact using familiar gestures. However, because of the non-physics-based approach, typically the fingers of the virtual hand pass through the surface of the object. Figure 1.5 shows the two main user experiences when interacting with virtual objects using virtual hand representations. Figure 1.5a shows the virtual hand passing through the surface of the object, obscuring the fingers. Figure 1.5b shows the virtual object appearing to float in front of the virtual hand.

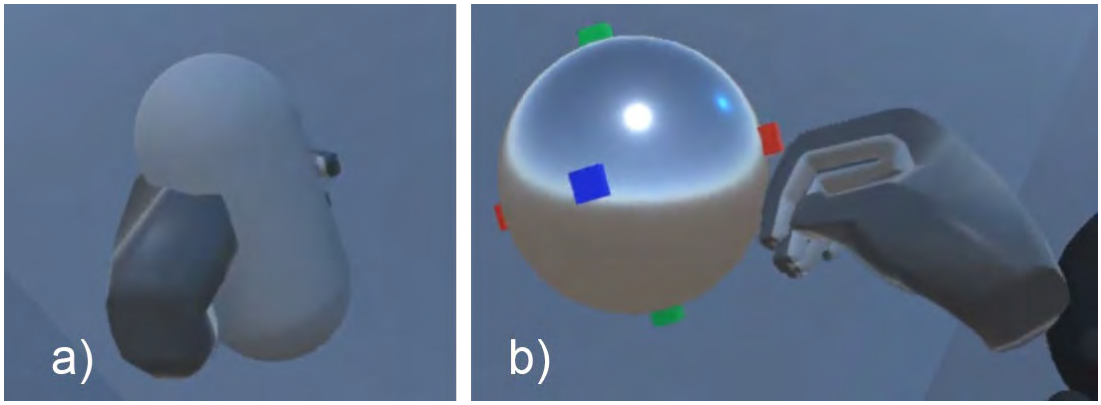


Figure 1.5 - Typical Virtual Hand Object Grasp Issues; a) Hand passes through the surface of the object, b) Object floats in front of the virtual hand (Dittrich, 2017)

This results in an un-natural hand appearance and can obscure view of the hand, making object to object interactions difficult. In addition, the un-natural and un-realistic appearance reduces the user's sense of immersion. In some cases, the virtual object appears to float in front of the virtual hand, also providing an un-natural, un-realistic appearance (Figure 1.5b).

In order to overcome the issue of un-natural hand appearance during object interactions in optical-based hand tracking, this research explores the techniques for creating realistic hand poses for holding objects. Previous research has typically relied on machine learning approaches trained on the virtual objects within the environment. Alternatively, virtual hand representations are frozen upon contact with a virtual object. The presented novel approach does not require any prior knowledge, by analysing virtual objects in real-time. Furthermore, the presented approach does not freeze virtual hands ensuring users can continue to control their virtual hands and interact with the simulation.

Research has shown hand interaction is one of the most intuitive ways for users to interact (Teleb & Chang, 2012; Silva, et al., 2013; Ishiyama & Kurabayashi, 2016). In particular, work exploring the use of optical based trackers has seen significant success (Chaudhary, et al., 2013; Rautaray & Agrawal, 2015; Anthes, et al., 2016) as this technology frees users hands by allowing them to freely interact with a virtual environment without an interaction device affecting their hands. Optical-based hand interaction approaches can provide users with the natural interaction required for an engaging and immersive experience. However, the approach has four key issues; namely, occlusion, field-of-view, accuracy and stability. As discussed above, further research is required to address these issues and provide users with an immersive, natural and consistent interaction experience.

1.5 – Academic Questions, Aims and Objectives of the Research

The main aim of this research is to develop an understanding of how optical based tracking technology can be used to provide users with more natural interaction and an immersive experience. This work explores how hand-based input mechanisms might offer more natural approaches to VR interaction that are more intuitive and immersive than traditional motion controller approaches. To achieve the aim of this research, the following academic questions were proposed:

- How can multi-sensor aggregation improve optical-based hand tracking solutions in virtual reality?
- What is the most effective sensor configuration for optical-based hand interaction in virtual reality?

The specific objectives of this research are:

- To identify and evaluate the relevant scientific literature, methods, tools and technologies to develop a critical understanding of the literature.
- To design and implement a real-time tracking optical-based hand interaction approach using multi-sensor aggregation.
- To evaluate the effect of different sensor positions and orientations on optical tracking data validity and stability.
- To evaluate the developed interaction approach against traditional VR input approaches such as gloves and motion controllers at creating a sense of immersion and naturalism.
- To evaluate the developed optical-based interaction approach against a single sensor configuration.

To evaluate the developed interaction approach educational simulations were developed to provide users with different virtual object interactions. The experimentation was conducted over two phases. In the first phase of experimentation, a Chemistry based simulation was developed. The simulation used three interaction types: object translation & rotation, virtual user interface and two-handed interactions. The second phase of experimentation was split into two parts. The first used a Chemical Engineering based simulation with three interaction types: object translation & rotation tasks, a virtual user interface and precise manipulation of virtual controls. The second part used a proposed Standardized Gesture Evaluation Questionnaire (SGEQ) designed to provide a standard test for the evaluation of optical-based trackers & solutions.

1.6 – Thesis Contributions

This research makes many original contributions to knowledge in the fields of virtual reality, serious games, immersion & presence, human-computer interaction and artificial intelligence. The contributions are as follows:

- The development of an optical tracking-based hand interaction system, known as the MOT system, that has been specifically designed to enhance hand interaction in virtual reality.
- The development of a head-mounted custom 3D bracket for additional sensor positioning.
- The development of a multiple deep neural network-based hand validation system for the validation of hand-tracking data in real-time. This system validates based on the hands natural movement range as opposed to a pre-set gesture vocabulary.
- The development of several hand tracking data analysis systems, designed to detect and handle the issues of occlusion, instability, invalid data and sensor/tracking data synchronization.
- The development of a virtual object analysis system for the creation of realistic hand grasp poses in real-time. The system generates realistic hand poses without the need for a physics engine or prior knowledge/training of the virtual objects within the simulation.
- The design and evaluation of a standardized gesture evaluation test for evaluating optical-based trackers and interaction mechanisms, regarding the four key issues of optical trackers and user experience. The test provides a standard upon which alternative and future optical based interaction approaches can be evaluated and compared.

- The evaluation of the developed interaction approach, in an educational VR context, against haptic gloves, motion controllers and a traditional optical-based approach using a single front-facing sensor.

1.7 – Research Methodology

The research presented was conducted in two design and experimental phases. The presented research methodologies use of two experimental phases, enabled the effectiveness of the presented approach to be evaluated against traditional approaches. Furthermore, the comparison enabled the key features of traditional approaches to be identified and incorporated into the presented hand-based interaction approach. This approach provided an iterative design process, enabling the system design to be revised/extended (Nielsen, 1993). The extended interaction approach was then evaluated against a traditional optical-based approach. The comparison was used to determine the effectiveness of the presented design at addressing the four key issues with optical trackers that were identified in section 1.4 of this chapter.

In the first design phase a hand-based interaction approach using multi-sensor aggregation was developed. In the first experimentation phase the presented approach was evaluated against traditional interaction approaches; specifically, motion controllers and haptic gloves.

The second design phase was planned based on the results of the first phase of experimentation, with additional functionality implemented to address the issues identified during the analysis of the phase one results. This included an occlusion detection system designed to detect and handle hand occlusion to ensure a more consistent experience. In the second experimentation phase, the presented interaction approach was evaluated against a traditional optical-

based hand interaction approach. The second phase of experimentation was conducted in two parts: the first using an education-based VR simulation. The second part used a proposed standardized gesture evaluation test for the evaluation of optical-based hand interaction approaches. The experiment was broken into two parts so the effect of the presented approach could be evaluated in both an immersive environment and in direct gesture performance. Furthermore, the phase two experimental design enabled the effect of user focus and engagement at concealing limitations with hand-based interaction approaches to be evaluated.

In both phases of experimentation, to evaluate the presented hand interaction approach, a mixed methods approach using both qualitative and quantitative methods was used. The qualitative analysis employed semi-structured interviews containing both open and Likert-based questions. The quantitative analysis was used to evaluate user performance and the performance of the presented interaction approach. The use of both qualitative and quantitative analysis allowed a combination of numerical measurement and in-depth exploration of the effect on user experience (Adams, et al., 2008; Assila, et al., 2014; Preece, et al., 2015). The use of both methods of analysis was necessary to address the research questions. Quantitative analysis of optical sensor performance could identify the optimal sensor configuration with regards to factors such as occlusion and hand validity. In addition, it would also evaluate the effect of multi-sensor aggregation on improving optical tracking. However, quantitative analysis might not identify issues with user experience, such as a system may perform well but restrict user movement and negatively affect the user experience. Therefore, both quantitative and

qualitative analysis was needed to identify the approach which improved both optical-based hand tracking and the overall user experience.

The semi-structured interviews conducted were analysed using thematic analysis to identify key themes and emotions of user experiences during experimentation. The interviews also enabled key features and issues/limitations with interaction approaches to be identified. Furthermore, the semi-structured interviews provided the opportunity to engage in a discussion to clarify answers and ascertain more information.

To further evaluate user experience, statistical analysis was performed on the results of the interview questionnaires. A Bivariate Pearson Correlation test was performed to identify correlations between interview questions, enabling key features/functionality to be identified. This test was used to determine whether any correlations found were linear, demonstrating a constant effect of the feature/functionality on user experience. Furthermore, the analysis was used to identify correlations between the four previously discussed issues with optical trackers. Additional statistical tests were performed including Cronbach's Alpha to assess the reliability of the interview questions used. Cohen's Kappa was used to measure inter-rater reliability in the thematic analysis, ensuring the selected themes and sub-themes were agreed. To determine whether the mean difference between interaction conditions was statistically significant, Paired-Sample T-tests were used.

As previously discussed, quantitative analysis was used to evaluate user performance and the performance of the presented interaction approach. In the first phase of experimentation, the time taken to complete individual tasks and the overall simulation was recorded. The results were analysed using

ANOVA tests to determine factors such as the effect of the interaction approach on completion time. The tests also looked the effect of participant prior experience on the performance of the interaction approaches. These tests enabled factors such as ease of learning to be evaluated through looking at the performance of both new and experienced users. The tests also enabled factors such as usability to be analysed and compared with user feedback to identify user preference between difficulty and interaction approach.

In addition, a pilot study was conducted in which the hand tracking data produced by each sensor and the presented approach was evaluated with regards to validity. The validation data was analysed using both Chi-Square and Linear regression tests. Statistical tests, such as Linear regression and Chi-Square Test were conducted to identify the effects of the sensors on the MOT system and any relationships between the sensors and MOT system.

The second phase of experimentation also used Chi-Square and Linear regression tests for the analysis of hand validation data. However, ANOVA and Friedman tests were also used to identify any statistically significant difference in the number of valid hands produced and the time in which no hand data was produced. These tests were conducted to evaluate the effectiveness of the sensor configuration and aggregation process. In addition, these tests were used to compare reported user experience with regards to the four key issues with the performance of the sensors and MOT system.

1.8 – Thesis Organization

The remainder of this thesis is organized as follows. Chapter two presents the literature review of virtual reality, hand-based input mechanisms, machine learning in optical tracking, immersion & presence in VR and interactive virtual reality in education. Chapter three presents a detailed design of the various sub-systems that make up the developed Multiple Optical Tracking (MOT) system.

Chapter four details the experimental setup and procedure for the first phase of experimentation. In addition, Chapter four presents the results of the first phase of experimentation, including both quantitative and qualitative analysis. The two analysis types are presented separately with the results of their respective tests detailed and analysed.

Chapter five presents a detailed design of the second design phase of the Multiple Optical Tracking (MOT) system. Furthermore, the chapter includes the designs of the various additional sub-systems added to provide additional functionality to further resolve the four main issues with optical trackers.

Chapter six details the experimental setup, procedure and results for both parts of the second phase of experimentation. In both parts tracking data is analysed for validity and semi-structured interviews are analysed by thematic analysis of participant comments and statistical analysis of questionnaire scores. Chapter seven concludes this thesis through highlighting the original contributions of this research and answering the research questions. Furthermore, Chapter seven makes recommendations for future works in this field.

Chapter 2 – Literature Review

Virtual reality interaction approaches have undergone significant research in recent years (Hannema, 2001; Park, et al., 2017; Breslauer, et al., 2019). Research into optical trackers has seen significant success due their more direct and intuitive interaction leveraging real-world experience (Rautaray & Agrawal, 2015; Anthes, et al., 2016). However, as previously discussed in Chapter 1, section 1.4, optical trackers have four key issues: field-of-view, occlusion, accuracy and stability. In order to investigate this area, current interaction approaches should be established, and their respective hand-based research discussed. Figure 2.1 presents an organizational chart showing the structure of this chapter. The chart shows the topics covered and their corresponding sections along with sub-topics within subsections.

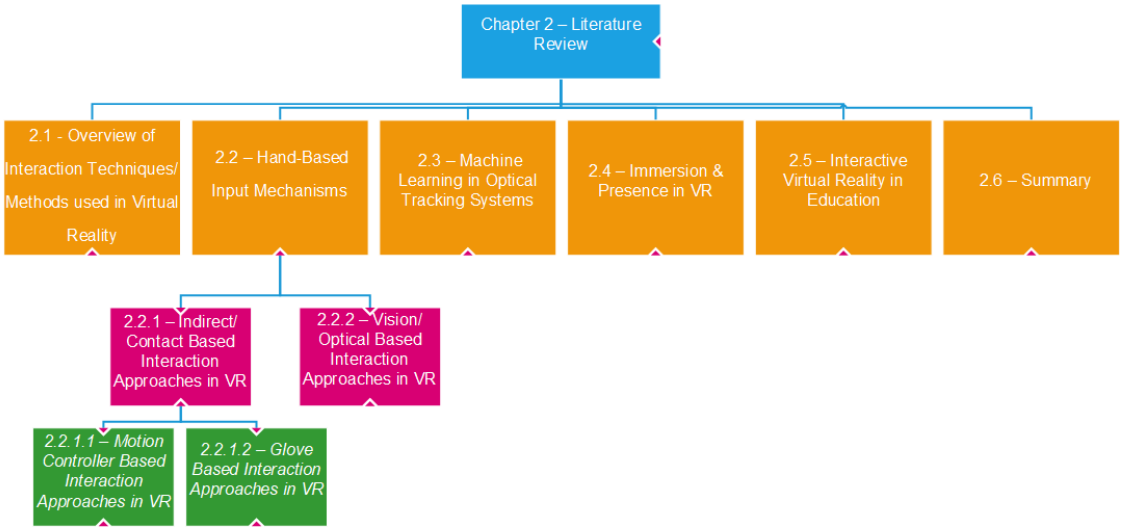


Figure 2.1 - Organizational Chart presenting the structure of the Literature Review

This chapter starts by introducing and discussing virtual reality and the commercial approaches that are currently available such as CAVE's and head-mounted displays. Research in hand-based interaction approaches is then presented in two sub-sections based on the way the user interacts as shown in Figure 2.1. Section 2.2.1 presents indirect/contact-based interaction approaches using technologies such as gloves and motion controllers. The research presented discusses different research approaches to providing more natural and immersive interaction experiences using those technologies. Section 2.2.2 presents the alternative, direct interaction approaches; specifically, optical-based hand interaction approaches. The research presented discusses current research into optical-hand based hand interaction approaches and techniques for addressing the four key issues. In addition to current interaction approaches, research into the effect of virtual hand representations and the generation of realistic hand grasps is presented.

Section 2.3 explores research into the use of different machine learning approaches and multiple sensors at improving optical-based hand interaction. The research presented is discussed with regards to their effects in areas such as real-time performance and user interaction experience.

As previously discussed in Chapter 1, hand-based interaction helps create an immersive user experience. Therefore, section 2.4 introduces the terms immersion and presence and the effect that both VR and interaction approaches have on a user's perceived sense. Before, exploring the use of different VR interaction approaches and their effects on education in section 2.5. Finally, section 2.6 summarizes the research findings presented in this chapter.

2.1 – Overview of Interaction Techniques/Methods used in Virtual Reality

Virtual reality (VR) is an interactive three-dimensional computer-generated environment that is created using hardware and software to immerse a person and enable them to feel as though they have presence within a virtual world. To generate VR experiences computer controlled electronic equipment is used to feed sensory input to users and monitor their actions. The main piece of electronic equipment required for VR is some form of display, that outputs the virtual world to a user based on input it receives from a computer. The two main approaches to VR displays are CAVE's and head mounted displays.

CAVE (Cave Automatic Virtual Environment) use walls and a floor made up of rear projection screens, with an optional ceiling screen, to display the virtual environment. An illustration of this setup can be seen in Figure 2.2 (a). Figure 2.2 (b) illustrates stereoscopic glasses which are used to create 3D views of scenes in CAVE setups (Slater, 2009; Manjrekar, et al., 2014; Pinto, et al., 2015).

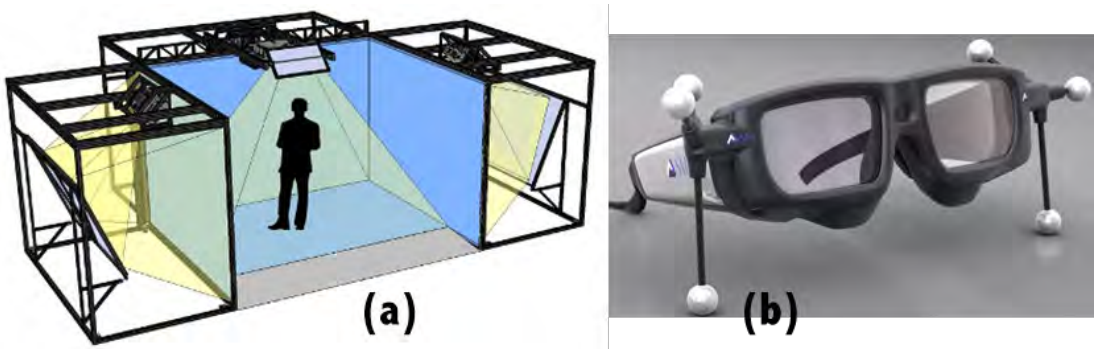


Figure 2.2 – (a) CAVE System (Visbox, 2020), (b) CAVE Goggles (Lang, 2013)

CAVE environments typically use stereoscopic LCD shutter glasses, such as the ones in Figure 2.2 (b), to display a 3D image to a user. Positional markers are used to track the head position and orientation of multiple users. The trackers are also used to calibrate the displays to prevent image distortion and update them based on the user's location within both the CAVE and virtual environment. CAVE's have been shown to be effective is providing a virtual reality experience, particularly for multi-user simulations and social interaction. However, research such as that of Schmidt et al (2018), Ghinea et al (2018) and Nunes et al (2019) suggests that HMD's provide a more immersive experience and stronger sense of presence.

User interaction (e.g. navigation, selection and manipulation of virtual objects) in virtual environments is typically achieved through the use of 6 degree of freedom (DoF) devices such as wands (Chapoulie, et al., 2014). Six degrees of freedom refers to the user being able to both move and rotate on all three axes (X, Y, Z). Wand devices typically use positional markers to track their position and orientation within the CAVE as can be seen in Figure 2.3. User interaction is typically performed using buttons and/or triggers to select and move virtual objects, as well as navigating (walking or flying) the virtual environment (Muhanna, 2015). Figure 2.3 shows a diagram illustrating the features of both a typical wand (a) and motion controller (b), VR interaction device.

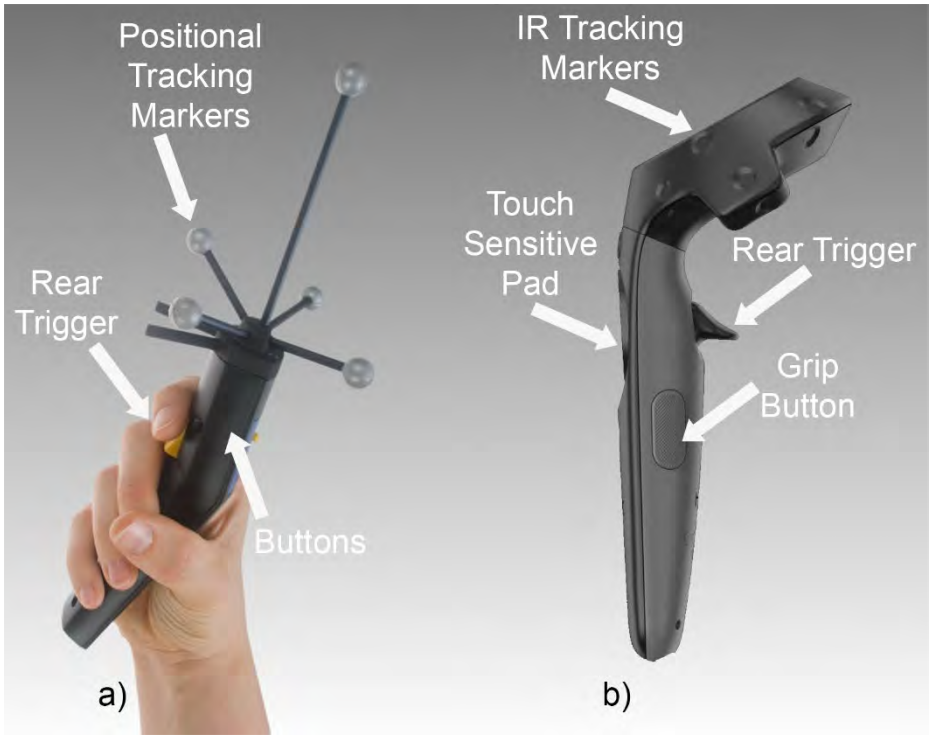


Figure 2.3 - Diagram showing the features of a) Flystick 3 Wand (ART, 2020), b) HTC Vive Motion Controller (Marroquin, 2017)

One of the key goals of an immersive virtual environment is to make the user's experience as realistic as possible, through the use of 3D graphics and 3D manipulation techniques that simulate interactions with the physical world (Bowman, et al., 2012; Chapoulie, et al., 2015). However, the functionality and level of interaction in traditional input devices such as wands and motion controllers are limited by the number of buttons and controls. To provide users with a more realistic and immersive experience, more direct interaction mechanisms are required.

An important aspect of direct interaction is the mechanism for interacting with virtual objects. Chapoulie et al (2014) analyse the feasibility of direct manipulation in a fully immersive space and its effects on presence. The authors present a hand interaction mechanism that combines finger tracking

with a real-time physics engine and a heuristic approach for hand manipulation. The evaluation focuses on simple contact-based interaction, enabling virtual objects to be translated and rotated while held. The system was evaluated against a 6DoF wand and real-world performance. It was hypothesised that the wand would be faster and more precise, but the finger tracking would be more natural and provide a higher sense of presence. The wand interaction used a ray-based 'point-to-select' approach. The results show the virtual tasks took longer than the real-world tasks. The wand and finger-tracking were equivalent in terms of speed despite the two-handed finger tracking having a longer median completion time. The results also show in terms of accuracy, the wand performed better than both the direct interaction and real-world conditions in the one-handed task. Users subjectively considered the wand to be more precise than the direct interaction even if the objective performance did not always confirm it. Furthermore, users reported the wand to be easier to use but rated the sense of presence and naturalism significantly higher for direct interaction compared to the wand. The authors conclude that the speed and accuracy of the finger tracking solution make it a feasible alternative for interaction. The authors concur with Bowman et al (2012) and Hinchet et al (2018), stating that finger tracking is better suited for training since the performance is closer to real-world interactions than the wand.

Zhang et al (2014) analyse the effect of gesture interaction using a combination of a Wand and positional markers for hand tracking. The individual performance of the two techniques was compared with a CAVE used to provide the virtual reality simulation. The developed interaction mechanism

used two different selection techniques; ray-based ‘point-to-select’ and a paint selection with objects painted over selected. The results showed a preference for the gesture interaction, particularly during menu interaction due to simplicity and not requiring memory of the wand button layout. In addition, the results showed an overwhelming preference for gestural interaction due to the naturalness and generally being more engaging.

Chapoulie et al (2014) and Zhang et al (2014) demonstrate the effectiveness of combining the stability and control of wands with more natural direct hand-based interaction. However, the use of a wand as the primary interaction mechanism limits the users sense of immersion with the hand acting as a 3D mouse (Zhang, et al., 2014).

Nan et al (2014) build upon the work of Chapoulie et al (2014) and Zhang et al (2014) comparing the performance of wands against vDesign, a hand-based interaction system using 6DoF tracking. A CAVE was used to provide the virtual reality simulation. The system used positional markers for 6DoF hand tracking, using hand-to-hand and interactable object proximity with a selection gesture to detect interactions. Fifteen participants were split into two groups: novice and expert. Experts were participants who had been using the wand and hand interactions for more than 6-months. In comparison to wand interaction times, the hand menu navigation task saw a 22.5% decrease in time for experts and a 27.2% decrease for novices. Hand object manipulation tasks showed decreases of 24.3% and 19.6% respectively. The authors conclude that wands cannot provide fast and convenient interactions for the manipulation of virtual objects. Furthermore, the authors conclude that their

proposed high-fidelity hand interaction technique can provide faster, more accurate interactions compared to traditional wand interactions.

The previously discussed research in this chapter has shown the effectiveness of wand-based interaction, providing users with an accurate and easy to use interaction mechanism. However, the discussed research has also shown direct hand-based interaction to provide a stronger sense of presence in a virtual environment. One of the key goals of an immersive virtual environment is to make the user's experience as realistic as possible by simulating the physical world. The complexity of hand interaction can result in greater user error and occlusion can be a problem when performing certain gestures/poses. However, research previously discussed has shown that direct hand-based interaction can provide a more natural interaction experience that can be particularly beneficial in training/educational applications.

This section has introduced key interaction techniques/methods used in virtual reality. In the next section, research in hand-based input mechanisms will be analysed and discussed. The section will focus on the two key types of interaction: contact-based and vision-based. The research presented will look at different hand-based interaction approaches including custom controller haptics and the aggregation of multiple optical trackers. The approaches presented will be analysed with regards to the sense of naturalism and immersion they provide.

2.2 – Hand-Based Input Mechanisms

Hand-Based input mechanisms can be split into two categories: Contact Devices and Vision Devices (Rautaray & Agrawal, 2015). These categories classify techniques based on the manner with which the user interacts with them, such as controllers in contact based and the Kinect in vision devices.

The aim of VR interaction is to enable natural and human-to-human like interactions, therefore the incorporation of gestures is an important area of research (Li & Dai, 2014; Perret & Vander Poorten, 2018). Gestures have long been considered an interaction technique that can deliver more natural, creative and intuitive methods for communicating with computers (Chaudhary, et al., 2013; Rautaray & Agrawal, 2015; Park, et al., 2017). For example, gestures can offer a more convenient way to explore three-dimensional (3D) virtual worlds (Trigueiros, et al., 2012). Traditional HCI devices such as keyboard & mouse have grown to be familiar but inherently limit the speed and naturalness with which we can interact (Moehring & Froehlich, 2011).

Gestural interaction typically takes one of two forms: indirect and direct interaction. Indirect interaction involves some form of tactile device such as the motion tracked controllers used with current VR headsets. In Indirect interaction, objects are typically selected by ray-based pointing or proximity-based interaction. In ray-based pointing, the user points at the object they wish to interact with and then presses a button on the controller. In proximity-based interaction the user positions the controller near the selected object and presses a button to “pick it up”. In both techniques, the selected object then ‘snaps’ to the end of the controller and moves with it.

In contrast Direct interaction uses devices such as optical camera-based tracking solutions and haptic gloves to track the user hands and fingers. A 3D representation of the user's hand is then integrated within the virtual environment and continuously updated to match the user's hand (Moehring & Froehlich, 2011; Li & Dai, 2014). The user interacts with virtual objects using hand gestures that often mimic real-world actions. For example, a user could pick-up a cup in the virtual world by performing a grasping gesture. The selected object then moves with the virtual hand representation, maintaining the offset between the hand and objects position.

In the next section, hand-based interaction mechanisms using the contact-based approach will be presented. The research presented is broken down into two further sub-sections, the first discussing motion controller-based interaction approaches. The research will analyse and discuss different controller designs to provide users with greater interaction fidelity and basic haptic interaction. The second sub-section discusses glove-based interaction approaches. The research presented will first discuss the different glove types before analysing and discussing different research approaches such as pattern-based gesture recognition.

2.2.1 – Indirect/Contact Based Interaction Approaches in VR

Contact based interaction involves some form of tactile device such as the motion tracked controllers used with current VR headsets. Alternatively, motion gloves can be used to track a user's fingers using bend sensors in the lining of the glove. Bend sensors are typically sewn into the fingers of a glove to be positioned on the top of the fingers. The recorded resistance of the sensors changes as they bend with these values then used to calculate finger

positions. Recent examples of contact-based devices include: HaptX glove, Manus VR glove, HTC Vive controller, Oculus Touch and Valve Knuckles.

2.2.1.1 – Motion Controller Based Interaction Approaches in VR

Motion Controllers are visual based input devices that are designed to allow the user to perform gestures within the simulation by interacting with a physical device. Recent examples of commercial motion controllers can be seen in Figure 2.4. Motion controllers provide high-precision, low-latency support for six degrees of freedom (6-DoF) manipulation (Clifford, et al., 2017).

Figure 2.4 – (Left) HTC Vive (Vive, 2020) & (Right) Oculus Touch Controllers (Oculus, 2019)

Research suggests that motion controllers provide a more natural form of interaction than traditional games controllers, however, the need for interaction with the physical device restricts the user (Chaudhary, et al., 2013; Clark & Moodley, 2016). While reliable and familiar due to their derivation from gamepads and use of reliable HMD tracking technology, motion controllers

can often be rather restrictive and limit the level of immersion that the user can experience (Chapoulie, 2014; Davis, et al., 2015).

Motion controllers suffer similar issues as traditional game input technology; the user's interaction is limited to the number of buttons on the controller with the addition of basic motion/gesture recognition. The interaction limitations of motions controllers limit a user's sense of presence (Choi, et al., 2016; Pallister, 2017). In addition, motion controllers suffer with poor affordance making interaction difficult and often leading to a sense of restriction/limitation due to the inability to use presumed functionality. Affordance is a property or feature of an object which represents a prompt on what can be done with this object. In short, affordances are cues which give a hint how users may interact with something, no matter physical or digital (Studio, 2018; Kim & Maher, 2019).

Derpanis (2004), Murthy & Jadon (2009), Bowman et al (2012), Choi (2018) and Yang et al (2018) suggest that traditional input mechanisms such as gamepads, keyboards and mice present a bottleneck. In particular, applications that rely on heavy interaction, such as video games and virtual reality simulations, due to the unnaturalness of the interaction. Hyper-natural techniques offer a middle ground between motion controlled based interaction and that of the real-world, using natural movements to give users powerful new abilities or intelligent guidance (Nabioyuni & Bowman, 2015).

Researchers such as Poupyrev et al (1996) and Achibet et al (2015) use a hyper-natural approach known as the 'Go-Go' technique. The 'Go-Go' technique enables users to use natural arm extension to extend a virtual arm

far into the environment allowing objects to be selected at distance. An alternative approach known as ray-casting uses a laser with the object aimed at selected upon click. However, ray-casting can be difficult to select very small objects (Bowman, et al., 2012).

Research has shown that well-designed techniques based on the hyper-natural approach can feel natural and familiar while avoiding some of the unwanted side effects of replicating real-world interactions (Bowman, et al., 2012; Chapoulie, 2014; Davis, et al., 2015).

Clifford et al (2017) present a technique for pseudo-telekinetic object manipulation in VR using slight downward tilt of the head to simulate Jedi concentration. The HTC Vive was used due to the low-latency 6DoF controller and headset tracking enabling smooth interaction and detection of the head-tilt gesture. The head-tilt gesture was used to enable “Jedi” mode thus changing the movement to rate-controlled. A ray-casting based approach was used for object interaction with the rear trigger used for selection. The controllers grip was used for one-to-one rotation control and tapping the trackpad toggled the translation state. In the translation state objects are rotated by rotating the controller, with objects moved closer or further away at a constant rate by holding either down or up on the trackpad, respectively. To evaluate the design, the authors conducted a pilot study in which eight participants were asked to move a series of coloured cubes around, stacking the cubes on top of larger stationary cubes. Participants reported difficulty in maintaining focus on an object whilst performing the “Jedi” gesture. Several participants suggested the use of visual effects such as a coloured glow

around an object when pointed at and then changing glow colour when selected.

One aspect of hyper-natural interaction is a sense of weight, a hyper-natural sense of weight can overcome the challenges of replicating a realistic sense of weight and aid in creating a sense of immersion. Choi et al (2017) detail the design of ‘Grabity’, a wearable haptic device designed to simulate kinaesthetic pad opposition grip forces and weight for grasping virtual objects. An initial quantitative study was conducted to identify the optimal configuration to generate compelling forces and their effect. A subsequent qualitative study was performed to gather user responses to differing levels of virtual forces using a Likert scale questionnaire. The results of the quantitative study indicated significant differences among the different weights (Light, Medium, Heavy). However, the results of the qualitative study showed users found it difficult to tell the difference between two weights and identify the block that weighed the most.

The design presented by Choi et al (2017) has several limitations, the first being that the user is only able to control one finger. In addition, the prismatic joint heavily restricts user dexterity to 1DoF, giving users an un-natural and unrealistic level of control. The results of the quantitative analysis show the design to be ineffective in providing reasonably realistic haptic feedback. Furthermore, the results show the system negatively impacts the user’s sense of realism and immersion. In addition, the virtual representation takes the form of two green balls (thumb and index finger positions), which presents an unnatural and unrealistic view.

Choi et al (2018) show a positive effect of haptic based interaction in creating an engaging experience. The authors present CLAW, a haptic controller that augments handheld controller functionality with force feedback and actuated movement to the index finger. The controller uses an HTC Vive tracker for 6DoF tracking with a linear resonant actuator in the grip and two cables providing power and USB communication. A voice coil actuator is used under the fingertip to render haptic textures of virtual objects. Qualitative analysis was performed using comments made during a freely interactable simulation on a 1-7 Likert scale. Time data was collected during an object relocation task that involved frequent switching between interaction modes. The qualitative analysis showed positive results with grabbing objects found to be the favourite task, with many confirming that the CLAW was a reliable and easy to learn method. The quantitative analysis showed the grabbing task to be considerably slower than the touching task (5.9 seconds compared to 2.9). However, participants highly rated their ability to switch between modes; 5.6 on a 1-7 Likert scale.

The results show the controller to be an effective design. However, a virtual hand representation can present perceived affordances with users expecting unsupported dexterity and control. Furthermore, as with their previous work (Choi, et al., 2017), one finger interaction severely limits the natural and realistic feeling of the interaction. The implementation of support for gesture interaction would aid in creating a more natural experience. However, gestural interaction ranges from basic motion detection to full finger tracking, with greater dextral fidelity having a negative effect on accuracy. Thus, further

research is required to determine the level at which VR gesture interaction is considered natural.

Research such as that by Zhang et al (2014) and Collins & Borowski (2018) has shown that gestural interaction can provide a more natural, engaging and easier form of interaction in virtual environments. However, gestural interaction is typically performed via direct hand-based interaction with full finger dexterity rather than the motion of the hand. Direct hand-based manipulation with a realistic hand representation in virtual environments could be a powerful interaction technique. Hand interaction provides users with a strong sense of naturalism and immersion along with improving the sense of realism.

However, current research also indicates there is a trade-off between natural, realistic interaction and reliability. Results, such as Chapoulie et al (2014), show no statistically significant difference in time between a wand and direct manipulation. Participants feedback indicated the direct manipulation technique provided the greatest sense of presence, because of the more natural interaction. However, the results also showed participants made fewer mistakes when interacting with the wand.

Chapoulie et al (2015) analyse direct finger-based and ray-based object manipulation by decomposing 3D movement into two movement types: rotation & translation and comparing the two against real-world interactions. The authors hypothesized that the direct manipulation approach would feel more natural but would have difficulty with real-time tracking. While the ray-based would have better tracking but feel less natural. A real-world version of

the simulation environment was built with objects permitting identical movement while subject to the same constraints as the virtual objects. To evaluate the interaction mechanisms, sixteen participants performed a series of object manipulation tasks using both virtual conditions and the real-world equivalent. The results showed the virtual conditions to generate higher errors with movement tasks in the range of 10-40%. In addition, the real condition resulted in less errors in all tests and faster completion times in all but one test. The authors suggest the errors could be due to limitations of the virtual environment rather than the nature of the movements. It is also stated that occlusion was found to be a problem in the finger-based condition, particularly with rotation tasks. Participants did not express any clear preference but reported the finger-based manipulation to be closer to real-world object interaction. The results show the wand outperforms the finger-based condition having a lower mean time in all but 3DoF translation and free movement (6DoF) tasks.

Research by Chapoulie et al (2014) and Chapoulie et al (2015) has shown that wands and direct manipulation are equivalent with regards to speed. However, wands generally provide a more consistent experience especially during rotation. Analysis of the results for the 3DoF and 6DoF task shows direct manipulation to either outperform or have no statistically significant difference to the wand. In addition, the 1DoF task results demonstrate the effect of occlusion in optical tracking. The poor performance was due to rotation restrictions resulting in the back of the hand occluding the fingers. Chapoulie et al (2015) suggest the results are inconclusive due to the tracking issues the

direct interaction technique experienced, causing the significant difference in rotational performance.

Virtual hand representations and direct interaction can provide a strong sense of immersion and presence. Research such as that by Bowman et al (2012), Nan et al (2014), Chapoulie et al (2014), Chapoulie et al (2015), Choi et al (2017) and Choi et al (2018) has led to the development of innovative designs to provide a more natural experience using motion controller technology. However, the dependency on interaction with a physical device via traditional inputs such as buttons, restricts the sense of immersion and thus more natural and direct interaction mechanism need to be explored. Table 2.1 presents a comparison of the main research interaction approaches presented in this section, showing both the main and missing features. The table compares the key features of the approaches discussed along with missing/limited features such as finger interaction capabilities and virtual representation. Analysis of Table 2.1 shows that despite the improvements made to provide a more natural experience, they are still limited with issues such as one finger with 1DoF interaction, limiting the user experience.

Table 2.1 - Comparison of the main features of the main presented motion controller-based interaction approaches

Feature	**Authors**			
	Clifford et al (2017)	**Choi et al (2017)**	**Choi et al (2018)**	**Chapoulie et al (2015)**
Tracking DoF	6	6	6	6
Haptics	N/A	Resistive grip forces	Voice coil actuator	N/A
Interaction Approach	Ray-casting	Grab and Release	Grab and Release	Ray-casting
Near Object Interaction	Yes	Yes	Yes	Yes

Far Object Interaction	Yes	No	No	Yes
Finger Interaction	No	1 Finger with 1DoF (bend only)	1 Finger with 1DoF (bend only)	No
Virtual Representation	Controller Model	Two Green balls as finger and thumb position	Virtual hand representation	Wand Model
Average Grabbing Task Time (Seconds)	N/A	N/A	5.9	Hardest Task Wands – 3.5 Hands – 2

2.2.1.2 – Glove Based Interaction Approaches in VR

Data gloves have become one of the innovative input devices that can be used for interacting with computers in virtual and augmented reality applications. Data gloves use embedded sensors; typically bend-sensors, to capture users hand movements. Hand gesture interaction is one of the most intuitive ways for users to interact, resulting in a great deal of research regarding hand tracking approaches, such as glove-based interaction (Teleb & Chang, 2012; Silva, et al., 2013; Ishiyama & Kurabayashi, 2016).

Perret & Vander Poorten (2018) classify gloves under three categories; traditional gloves, thimbles and exoskeletons. Traditional gloves refer to the use of bend sensors attached to either the back of the fingers or within the lining of the glove, to measure the flexion of the fingers (Figure 2.5a). Thimbles refers to a semi-flexible device attached to the fingertip (like an Oximeter) containing sensors, actuators, a power source and a wireless transmission device (Figure 2.5b). An Exoskeleton is defined as an articulated structure the user wears over their hand which transmits forces to the fingers, restricting finger movement to provide the feeling that the user is holding something

(Figure 2.5c). Figure 2.5 presents a diagram showing the main features of the three different haptic glove types presented.

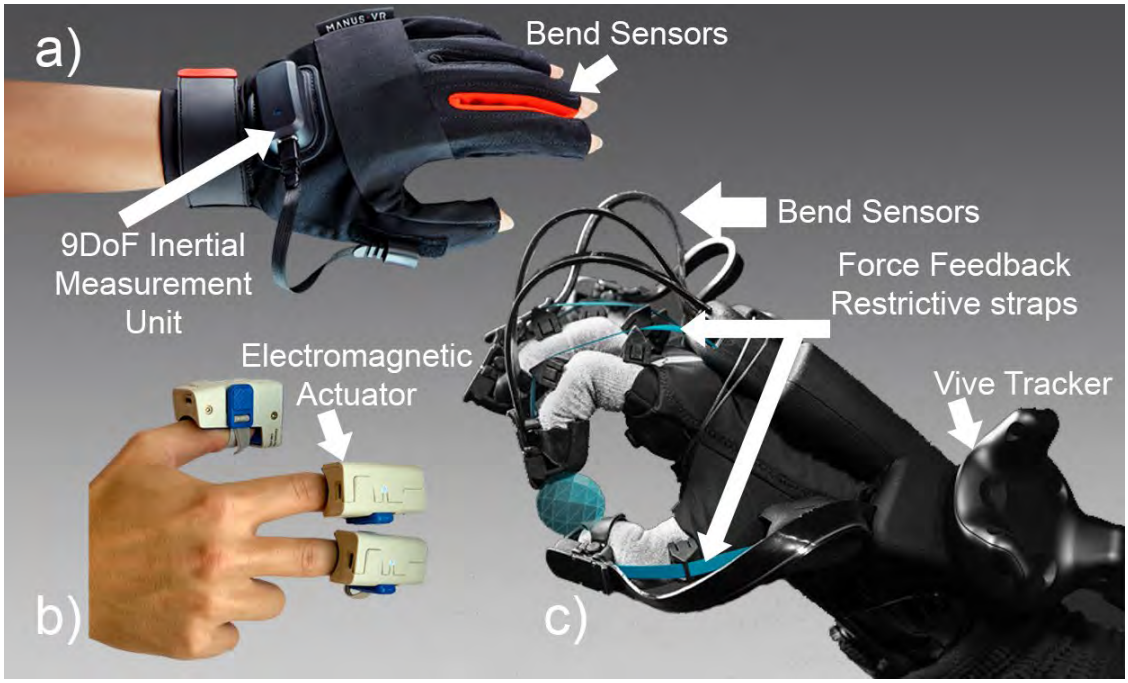


Figure 2.5 - Diagram showing the main features of the three different haptic glove types; a) Traditional Glove (Schenker-Tech, 2020), b) Thimble (GoTouchVR, 2020), c) Exoskeleton (HaptX, 2020)

Ishiyama & Kurabayashi (2016) present an alternative approach, using a marker-based glove interaction mechanism via a single camera setup. The start and end of the metacarpal bones were marked with white circles to identify the palm. The fingers had thick white lines emanating from the end metacarpal bone and spanning the length of the fingers to recognize finger angles and bend states. To evaluate the system, a classification test was performed using 122 hand postures, with the results showing the system can identify ninety-six postures. The results also showed the effects of occlusion with pattern regions occluded during certain postures affecting the accuracy of the system, resulting in false positives and negatives.

Research such as that of Wang & Popovic (2009), Han (2010) and Ishiyama & Kurabayashi (2016) shows the effectiveness of glove-based pattern recognition approaches. However, the dependency upon seeing the glove pattern results in occlusion being a more prominent problem than in alternative optical-based approaches. In addition, pattern-based approaches require a pre-set gesture vocabulary against which patterns can be matched. In comparison to motion controllers, pattern-based vocabulary approaches significantly increase the interaction affordances. However, they do not provide the same freedom of interaction as human hands within the real-world, which are typically modelled with 27 DoF (ElKoura & Singh, 2003).

Alternative approaches using the exoskeletal approach typically do not require a gesture vocabulary, often using bend sensors for finger positioning and straps to provide a resistive force for haptic feedback. A recent example being HaptX (Figure 2.5c), which uses bend sensors and a HTC Vive tracker for tracking, with resistive straps and small arrays of pneumatic actuators to provide haptic feedback (HaptX, 2020).

Jadhav et al (2017) present a wearable soft robotic haptic feedback glove for force feedback in virtual environments. The glove uses McKibben muscles controlled by a custom fluidic control board to provide resistance during object interactions, with a Leap motion for finger tracking. To evaluate the glove design, fifteen people completed a piano-based VR simulation using their index and middle fingers on both hands, both with and without the haptic feedback enabled. The results found all participants reported the haptic experience increased the sense of immersion, with eleven participants stating the haptic glove accurately simulated the pressing of a key. However, some

participants reported experiencing a slight delay in haptic feedback and suggested reducing the size of the control board to increase portability.

Irfan et al (2018) detail the design of a prototype force feedback glove based on the Arduino microprocessor. The glove tracks joint movement using an incremental encoder attached to a motor. The motors provide resistive torque during object interactions, emulating the physical presence of the virtual object. The design follows the exoskeletal approach with motors attached at each finger joint to generate haptic feedback. The prototype developed was only capable of tracking the index finger. To evaluate the design, two test environments were developed with soft (elastic) and hard (non-elastic) objects; soft experiencing a gradient of force, while the hard object provides high torque. The results found participants were able to discern if the virtual object they were interacting with was hard or soft. The authors conclude by stating that while the glove was sufficient for a proof of concept, the motor used proved to be too large and did not provide enough torque for realistic feedback.

The design presented by Irfan et al (2018) shows the positive effects haptic feedback can have on a user experience, providing the sense of touch to aid in creating an immersive experience. However, as stated by the authors, the motors used at each joint are not capable of providing realistic haptic feedback. In addition, the inconsistent motion is likely to cause inconsistent wear on the motors resulting in varying sensitivity across fingers and both hands. The authors suggestion of following the more traditional exoskeletal approach and using either wiring or a pneumatic system should aid in creating a more consistent and realistic experience.

Hinchet et al (2018) concur with Irfan et al (2018) and Jadhav et al (2017) on the idea of using larger motors with wiring and the issues experienced with the size of pneumatic braking systems. The authors present a flexible haptic glove that integrates both kinaesthetic and cutaneous feedback. An electrostatic clutch is used to generate up to twenty newtons of holding force on each finger by modulating the electrostatic attraction between two flexible metal strips. For tracking, the glove used an Opti-Track tracking system for finger tracking with an Oculus CV1 headset to display the virtual scene. The system evaluation was split into two components. The first a psychological evaluation measuring the “just noticeable difference” of stiffness, with the second exploring the effects on user immersion. The results of the psychological evaluation found that by varying the input voltage it is possible to render objects with different levels of deformable stiffness. The results of the second test indicate the glove was able to provide a sensation of holding an object. However, grasps with a short distance between the index and thumb do not provide sufficient space to exhaust mechanical slack. Thus, the brakes do not fully engage, resulting in minimal holding force. In a physics playground test, the haptics were particularly effective with virtual fingers naturally conforming around virtual objects but passing through with the haptics disabled.

Glove-based approaches have been shown to be effective in increasing a user’s sense of immersion using approaches such as bend sensors for finger tracking. The previously discussed research has also shown the effectiveness of haptic feedback to further improve a user’s sense of immersion. However, research discussed has shown glove-based approaches reliance upon optical

tracking technology to provide realistic DoF in hand control due to bend sensors 1DoF tracking.

Analysis of the research discussed shows that researchers typically use bend/flex sensors for finger tracking which limits the tracking to bend only. Furthermore, glove-based approaches typically use VR motion controller technology for 6DoF hand tracking, such as the Manus VR and HaptX gloves that use HTC Vive trackers (Schenker-Tech, 2020; HaptX, 2020). To provide more complex hand and finger tracking, optical trackers can be implemented into glove-based interaction approaches. Research such as that of Jadhav et al (2017), Borja et al (2018) and Edwards et al (2019) has shown the effect of combining the haptic feedback of gloves with optical tracking for 27DoF finger movement and tracking. However, as previously discussed optical trackers suffer with issues such as occlusion and tracking instability (Chapoulie, et al., 2015) and therefore cannot provide the stability of bend sensors. To further improve the tracking capabilities and DoF in haptic gloves, optical trackers need to provide more stable and accurate tracking. Optical trackers and research into interaction approaches is discussed further in section 2.2.2.

Table 2.2 shows a comparison of the main glove-based interaction approaches presented in this section. The table compares the main features of the approaches discussed along with missing/limited features such as finger tracking capabilities and virtual representation. Analysis of Table 2.2 shows that while gloves provide a more natural experience than controllers, the DoF is limited with only finger bending tracked. Furthermore, alternative approaches to bend sensors offer little tracking improvement whilst introducing additional weight and potential user discomfort.

Table 2.2 - Comparison of the main features of the main presented haptic glove-based interaction approaches for VR

Feature	Authors			
	Ishiyama & Kurabayashi (2016)	**Jadhav et al (2017)**	**Irfan et al (2018)**	**Hinchet et al (2018)**
Glove Type	Pattern Recognition	Exoskeletal	Exoskeletal	Exoskeletal
Finger Tracking Technology	AR Markers & 1 Camera	Leap Motion finger tracking	incremental encoder attached to a motor	Opti-Track tracking system (optical)
Haptic Feedback	No	Resistive forces	Resistive forces	Resistive forces
Gestures Identifiable	96	Real-time hand control	Real-time hand control	Simple pinch gesture
Performance	Only 50 frames per second	Slight delay reported in haptics	Unrealistic haptic feedback	240Hz Tracking
Recognition Accuracy	99% in all patterns	N/A	N/A	N/A
Finger Tracking Axis	Bend only, no side movement	Both bend and side movement	Bend only, no side movement	Both bend and side movement
27 DoF Finger tracking	No	Yes – Leap Motion	No and only the index finger was tracked	No, only the thumb and index finger were tracked
Hand Representation	N/A	Leap Motion Skeletal Hand	rectangles for bones and a cylinder for the joint	Two spheres (index and thumb)

A comparison of current commercial motion controllers and gloves can be seen in Table 2.3. The key features of the commercial technologies are compared, with missing and limited features presented such as finger tracking range and finger DoF. Analysis of Table 2.3 shows while research has identified more natural techniques using motion controllers and gloves, the interaction experience is still limited to basic hand interaction using flexion extension tracking.

Table 2.3 - Comparison of the main features in current prominent commercial motion controllers and haptic gloves

Feature	HaptX	Manus VR	HTC Vive Controllers	Oculus Touch	Valve Knuckles
Input Type	Glove	Glove	Motion Controllers	Motion Controllers	Motion Controllers
Interaction Type	Indirect	Indirect	Indirect	Indirect	Indirect
Hand/Controller Position & Rotation DoF	6	6	6	6	6
Hand Interaction	Yes	Yes	No	No	No
Haptic Feedback	Fingertip haptics & restrictive feedback	Basic Vibration	Basic Vibration	Basic Vibration	Basic Vibration
Finger Tracking Technology	Bend sensors	Bend sensors	N/A	N/A	Capacitive grip
Finger Tracking Axis	Bend only, no side movement	Bend only, no side movement	N/A	N/A	Bend only, no side movement
27 DoF Finger tracking	No	No	N/A	N/A	No
Virtual Representation	Hand	Hand	Controller	Open or Fist hand pose	Hand
Tool Tracking	No	No	No	No	No
Portability	Tethered to control box	Untethered	Untethered	Untethered	Untethered
Interaction Mechanism	Gestures	Gestures	Button & Trigger Inputs	Button & Trigger Inputs	Gestures
Battery Life (Approximately)	Powered	3-6 hours	6-9 hours	20 hours	7-8 hours
Weight	“a few pounds each” (ICVR-Interactive, 2018)	68.5 grams each	205 grams each	166 grams each	184 grams each

This section has introduced indirect/contract-based interaction techniques/methods used in virtual reality. In the next section, hand-based interaction mechanisms using vision-based approaches are presented. First, the section will introduce optical trackers and the two prominent devices,

before evaluating the accuracy of optical trackers. Furthermore, the research presents an evaluation of optical tracker-based solutions against traditional interaction approaches such as gloves. Finally, the current issues with optical trackers as identified by research are presented along with research conducted to try and address these limitations.

2.2.2 – Vision/Optical Based Interaction Approaches in VR

An alternative to contact-based interaction is vision-based techniques using optical trackers that rely on receiving data from the user in the form of hand and/or body recognition and tracking. In this section research in hand recognition and tracking using optical trackers such as the Leap Motion will be explored.

Vision-based hand interaction research can make use of custom-built tracking systems and hardware, commercial tracking systems or custom tracking systems that integrate commercial tracking technology. There are various forms of commercial optical based trackers, recent examples include the Leap Motion (2013) and Kinect (2014). The Leap Motion controller (Figure 2.6a) is a small USB device, that uses cameras and infrared LEDs to track the user's hands and fingers. In contrast to the Leap Motion, the Kinect (Figure 2.6b) provides full-body skeletal tracking. However, the Kinect does not natively support finger tracking (Qingchao & Jiangang, 2017), therefore it is typically not used as the main sensor for hand tracking solutions.

Figure 2.6 - Two prominent optical tracking sensors; a) Leap Motion (Motion, 2013), b) Kinect (Microsoft, 2014)

Marin et al (2014) and Weichert et al (2013) show the Leap motion is suitable for hand-based interaction in VR simulations, with a high level of accuracy and precision. Valentini & Pezzuti (2017) further analyse the Leap Motion controller for interactive VR applications, with respect to the accuracy and precision of the sensor. A specially designed test rig was used using human participants in a real-world context, assessing the accuracy of all five fingers on the right hand. The results showed robust and stable hand tracking with fingertip errors within 4-5 millimetres. The authors conclude the leap motion is suitable as an interaction control mechanism for object manipulation within virtual environments. The authors found the vertical distance had no statistically significant effect on tracking but that it does improve slightly at closer proximity. The results of Valentini & Pezzuti (2017) show the Leap Motion sensor is suitable for hand based interaction in VR simulations.

Research such as that by Pinto et al (2015) and Figueiredo et al (2018) shows user preference for gesture based interaction in virtual environments. Pinto et al (2015) present a study comparing gesture-based interaction and positional tracking with a conventional controller-based ray-pointing system. The results showed the differences in navigation times are negligible. However, controller interaction was much faster than gesture based, 16.6 compared to 30.03 seconds. Questionnaire results indicated participants preferred the gesture based over the controller-based method despite the interaction time being slower. The results also showed the effect of combining optical trackers; specifically, two Kinects, to increase the field-of-view and improve tracking, maintaining immersion.

Figueiredo et al (2018) build upon the work on Pinto et al (2015) through the evaluation of wand and hand based interaction using a HTC Vive controller and Leap Motion. The two interaction mechanisms were evaluated through five different scenarios exploring both near and far object interaction. Near object interactions were performed using the rear trigger in the case of the Vive controller with the Leap Motion using natural grasp gestures. Far object interactions used a ray-based approach with rays emanating from the end of the controller and the base of the index finger, respectively. To select the far object currently aimed at, the near object controls were used. The results showed similar performance for the Leap motion and Vive controllers in grab and release task with average times of 3.31 and 3.28. Furthermore, the results showed similar System Usability Scale (SUS) scores with 88 and 93, respectively. Qualitative results show most users reported the Leap Motion as

the preferred when interacting with near objects since the interaction is more natural and does not require any information or training to be used.

The previously discussed research such as that of Pinto et al (2015) and Figueiredo et al (2018) has shown the effectiveness of optical-based tracking systems in recognizing hand gestures for more natural and direct interaction with VR simulations. However, a prominent issue with optical-tracking solutions is occlusion in which the sensors view of a tracked hand is impeded or blocked entirely. This results in either a decrease in tracking accuracy or tracking to be lost (Qingchao & Jiangang, 2017).

Shao (2016) suggests that tracking issues with the Leap Motion are down to three key factors; self-occlusion, distal phalanges, and detection region. Self-occlusion occurs during VR interactions, when a user's hands often move into various orientations in which the Leap Motion loses sight of the fingers. The author suggests that the Leap Motion is unable to accurately detect the distal phalanges of the middle and 'pinky' fingers and so its tracking data should be avoided. Thirdly it is suggested that the tracking data becomes unstable when hands are near the detection region boundaries. However, it is worth noting that the SDK version used was v3.1.2 and the current version number is Orion v4.0.0, with the latest version having several tracking improvements (Motion, 2013). Clark & Moodley (2016) concur with Shao (2016) that occlusion is due to the sensor losing sight of the fingers and expands suggesting that the occlusion problem is due to the sensor being mounted on the front of the user's HMD. Thus, the sensor can only view the back of the user's hand and therefore when the fingers are bent over, they are occluded by the palm.

To overcome the issues of occlusion, researchers have investigated multi-sensor data aggregation with previous research such as that of Marin et al (2014) showing the effectiveness of sensor aggregation. Jin et al (2016) build upon the work of Marin et al (2014), analysing the effect of multi-sensor data aggregation on reducing occlusion within optical tracking data. Two Leap Motion sensors were used. The first positioned face-up and the second to the side at an angle of 120 degrees to the bottom sensor. The results show an improvement in the recognition accuracy with a simple grasping gesture (loosely closed fist), showing 63.3% on the bottom sensor, 86.7% on the side and 90% when combined. Furthermore, the results show the combination of two Leap sensors does improve the tracking accuracy; however, the maximum difference between the side only and combined conditions was only 10%.

The results show the use of multiple sensors to be effective at improve tracking accuracy and reducing occlusion. However, the sensors were mounted on tripods which in a VR simulation with 360-degree rotation, would heavily restrict the effectiveness of the system. The mounting of multiple tracking sensors to a user's HMD could improve the accuracy of tracking data (Clark & Moodley, 2016; Caserman, et al., 2019; Kiselev, et al., 2019) through the alignment of the sensors and users gaze direction. The previously discussed research such as Shao (2016) shows that optical tracking sensors, such as the Leap Motion are highly accurate but suffer from issues such as occlusion. However, these issues can be mitigated using multi-sensor data aggregation. In addition, multi-sensor aggregation with complex software to evaluate tracking data and adjust for noise/errors could further improve tracking reliability and the sense of immersion. Table 2.4 presents a comparison of the

main optical-based interaction approaches presented, identifying the main features and limitations. The table compares factors such as sensor used, degrees of freedom and the presence of issues such as occlusion and tracking instability. Analysis of Table 2.4 shows the Leap Motion to provide the most accurate hand-tracking of the two prominent commercial sensors. The table also shows sensor positioning and data aggregation to be effective approaches in improving tracking accuracy.

Table 2.4 - Comparison of the main Vision/Optical based interaction approaches presented

Feature	**Authors**			
	Valentini & Pezzuti (2017)	**Pinto et al (2015)**	**Figueiredo et al (2018)**	**Jin et al (2016)**
Sensor/s	Leap Motion	2xKinect	Leap Motion	2xLeap Motion
Hand Interaction	Yes	Yes	Yes	Robotic Arm
Designed for Hand Tracking	Yes	No	Yes	Yes
Virtual Representation	N/A	None	Virtual Hand	N/A
27 DoF Finger tracking	Yes	No	Yes	Yes
Accuracy	4-5 millimetres	N/A	Grab Task errors Leap-5.28% Vive-1.5%	Fist Recognition = 63.3%(face-up) 86.7%(120º), 90%(combined)
Interaction Mechanism	N/A	Gestures	Ray-casting & pinch gesture	Gestures
User Performance	N/A	Controller-16.6 (sec) System-30.3 (sec)	Leap-3.31(sec) Vive-3.28(sec)	N/A
Usability	N/A	Interaction Preference Kinect-6 Controller-3 Indifferent-2	SUS Scores Leap - 88 Vive - 93	N/A
Realtime Tracking	Yes	No	Yes	Yes

Occlusion Issues	N/A	Not reported	Not reported	No
Tracking Instability	N/A	N/A	Reported	No

The previously discussed research such as Figueiredo et al (2018) and Pinto et al (2015) has shown, hand based gestural interaction provides a natural form of user interaction over more traditional interaction mechanisms. However, as part of the replication of real-world interactions, the hand-object interaction behaviour should be natural and virtual hand representations appear realistic. Figure 2.7 shows the standard virtual hand representations available with the Leap Motion; a) Capsule/Skeletal, b) Robotic and c) Realistic Hand. These represent the typical virtual hand representations used in optical-based hand interaction approaches.

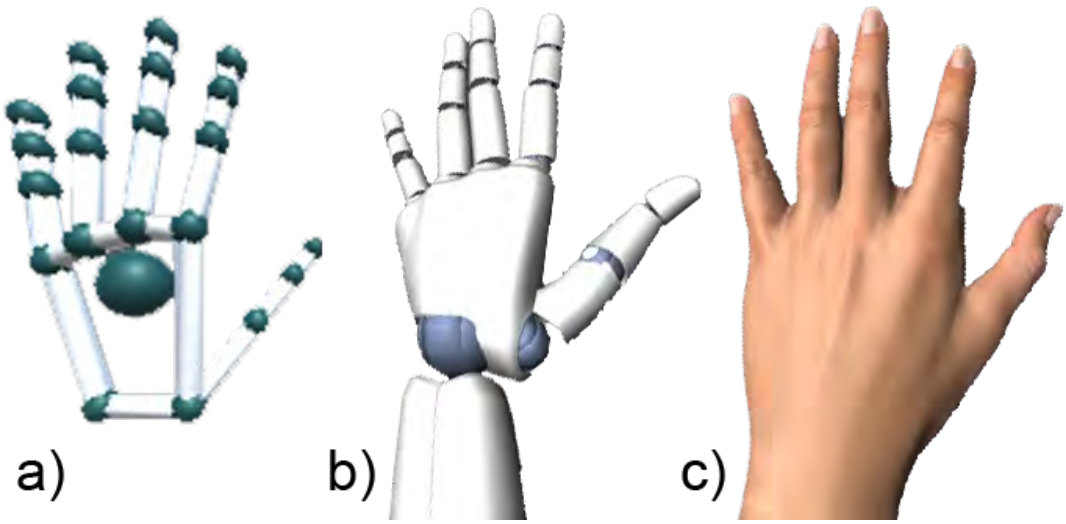


Figure 2.7 – Standard virtual hand representations available in the Leap Motion Unity Core Assets; a) Capsule Hand, b) Robotic Hand, c) Realistic Hand (Motion, 2013)

Argelaguet et al (2016) use vision-based hand tracking devices to explore the effects of different virtual hand representations on a user's sense of embodiment. The main experiment was a pick-and-place task in which participants had to move a virtual cube to an indicated position whilst avoiding an obstacle (brick, fire or barbed wire). An additional task was performed in which participants had to place their virtual hand near a spinning table saw blade. Three different virtual hand representations were used. Firstly, an abstract hand represented by a sphere that changed colour during grasps. Secondly, a simplified robotic hand that used open and close animations. Thirdly, a realistic virtual hand controlled via Leap Motion tracking data. Qualitative analysis showed participants found the realistic hand to be more difficult to control as they had anticipated greater precision. All three representations elicited strong feelings of control with the realistic hand providing the greatest sense of ownership. Quantitative analysis showed a decrease in performance as the realism of the representation increased, which supports user's comments regarding finding the realistic virtual hand more difficult to control. The authors suggest that one of the reasons for the decrease in performance could be due to the Leap Motion tracking system, with a few participants experiencing an issue where the right hand was mistaken for their left. The authors note that the reaction of participants was generally to shake their hand as if they were trying to shake it back.

Lougiakis et al (2020) extend the work of Argelaguet et al (2016) through exploring the effects of different virtual hand representations on user interaction and sense of embodiment when using motion controllers. Participants were tasked with moving a cube to and from specific positions

using three different hand representations. The system used a sphere, 3D model of the Vive controller and a realistic looking human hand for virtual hand representations. Participants completed the task twice for each representation; both with and without obstacles including a brick wall and barbed wire. The questionnaire results show no significant differences between representations with regards to a user's sense of control. However, task duration was significantly worse with the sphere and the hand was considered the favourite, providing the strongest sense of ownership.

The previously discussed research such as Argelaguet et al (2016) has shown that a realistic looking hand model can significantly improve a user's sense of ownership. Schwind et al (2017) investigate the effect of gender of the perception of virtual hand representations. Both quantitative and qualitative results show that women perceive lower levels of presence while using male hand representations. However, the results show that men are unaffected by the gender appearance of a virtual hand but experience lower levels of presence when using non-human hand representations. Schwind et al (2017) have shown the effect of virtual hand appearance on the sense of ownership. However, research such as Lin et al (2019) has shown virtual hand size relative to the users hands, has no significant impact on the sense of ownership. Table 2.5 presents a comparison of the main research discussed on virtual hand representations in optical-based interaction approaches. Analysis of Table 2.5 shows that realistic virtual hands can increase task completion time compared to simple ball based representations, as more of the virtual object is obscured when interacting. However, the table also shows user preference for the more realistic representation.

Table 2.5 - Comparison of the main virtual hand representation approaches presented

Feature	Authors		
	Argelaguet et al (2016)	**Lougiakis et al (2020)**	**Schwind et al (2017)**
Sensor/s	Leap Motion	Vive controllers	Leap Motion
Hand Interaction	Yes	No	Yes
Interaction Mechanism	Gestures	Rear trigger	Gesture
Virtual Hands Appearance	Sphere, robotic hand, realistic hand	Sphere, controller, hands wearing gloves	Skeletal, abstract, cartoon, male, female, male/female blend
Accuracy	Placement precision (Lower=better) Realistic-3.52cm Robotic-2.7cm Sphere-2.7cm	Hand had the shortest collision time per Wilcoxon test results	N/A
User Preference	Realistic	Hands wearing gloves	Women – Female hand Men – human hand
User Performance (Seconds)	Realistic - 3.21 sec Sphere - 2.46 sec	Sphere-3.3 Controller-3.07 Hand-3.12	N/A
Usability (Likert)	Control Realistic-4.49, Robotic-5.36, Sphere-5.70 Ownership Realistic-5.55, Robotic-4.52, Sphere-3.97	Ownership Sphere – 2.05 Controller – 2.28 Hand – 3.26	N/A
Occlusion Issues	Yes	No	None reported
Tracking Instability	Yes	No	Yes

The previously discussed research and that of Lin & Jorg (2016) and Lin et al (2019) has shown the effect that virtual hand representations can have on creating a natural and immersive experience. However, research discussed has also shown that optical based trackers are not yet capable of providing consistent and reliable tracking. Thus, further research is required to evaluate the effectiveness of virtual hand representations for object interaction and manipulation within virtual environments. Additionally, further research is required to overcome the limitations of current systems in providing a natural interaction experience.

One of the main challenges with natural hand-based interaction is visual realism, with many VR applications using simple interaction methods such as penetration selection or gesture-based grasping. While these techniques are effective, they do not result in natural or realistic interactions. Furthermore, such methods typically result in the virtual fingers passing through the surface of the object or the virtual object appears to float in front of the user's virtual fist (Tian, et al., 2018). Research such as that of Zhao et al (2013), Kim & Park (2015) and Holl et al (2018) investigated physics-based approaches to interacting with virtual objects. Holl et al (2018) present a technique based on the Coulomb friction model to produce realistic grasps without requiring pre-recorded data. A pilot study was conducted with 12 participants, who experienced three conditions; purely kinematic, Leap Motion interaction engine and the authors proposed system. Participants were asked to move a series of virtual objects to specific randomized target locations. Results of post-experience interviews showed users preference for the proposed

technique, stating its naturalness as the reason. However, the results also indicated participants found the technique the most difficult to learn.

While effective at creating a realistic view, the user's virtual fingers were individually frozen once they collided with a virtual object to prevent them from passing through the object. However, this results in the hand distorting if the user moves one of their fingers away from the virtual object. In the experiment scenario this was not an issue. However, in simulations where users interact with a held virtual object, such as pressing a button on a torch or the end of a pen, the thumb movement away from the object's surface will cause the hand to deform. Thus, breaking any sense of immersion and the naturalism of the interaction. Furthermore, the freezing of the fingers upon contact can make two handed interactions difficult and in some cases impossible (Oprea, et al., 2019).

Nasim and Kim (2018) use the Coulomb friction model and present an interactive grasping algorithm to provide physically realistic interactions in virtual environments. They used a Leap Motion for tracking data and the Coulomb friction model to ensure friction contacts and thus compute a stable virtual grasp. The system split the visuals and physics interaction with the user controlling a Kinematic hand used to update the physics hand. The friction model was used to ensure the physics hand did not penetrate the objects surface. The visual representation was then updated using the physics hand. The system was evaluated against the Leap Motion Interaction Engines pinch gesture, by twelve participants. Qualitative analysis showed that users generally preferred the proposed system. Quantitative analysis showed user performance improving with subsequent attempts. The authors conclude that

while the system worked well it does have limitations such as its dependency on discrete collision detection. The physics system in the game engine can fail to detect collision event resulting in the virtual hand penetrating the virtual object. The use of mesh-to-mesh collision analysis could improve hand penetration situations. However, the complex physics calculations would significantly increase the required computations and thus negatively affect performance. In comparison to Holl et al (2018), the system presented by Nasim and Kim (2018) provides a more natural interaction experience as users are able to move their fingers away from the virtual object without deforming the hand. The separation of physics and visuals enables the user to move fingers without deformation while holding a virtual object, providing a greater sense of realism than the system presented by Holl et al (2018). However, the dependency on the physics system limits the realism of the interactions possible, due to more realistic physics calculations significantly increasing the performance cost.

Tian et al (2018) present a real-time virtual grasping algorithm to model interactions with virtual objects using machine learning and particle swarm optimization. A Leap Motion was used to generate hand tracking data to compute the nearest grasp posture from the stored samples, during interactions. Similar to Nasim & Kim (2018), the authors used a visible virtual hand that was manipulated to grasp the object and an auxiliary hand that was determined by the Leap Motion tracker. In addition, users were able to manipulate held objects with hand deformation. Qualitative analysis found participants preferred the authors system over pinch grasping or raycast picking, particularly when interacting with more complex objects. One of the

limitations of the system is that it does not consider the user's intent. Therefore, the pose selected may significantly differ from the users intended hand pose and thus appear jarring, reducing the sense of immersion and naturalism consequently.

Table 2.6 presents a comparison of the main research approaches presented on calculating natural hand grasp poses for virtual objects. The table compares the different approaches main features along with limitations and missing functionality. Analysis of Table 2.6 shows user preference for realistic hand grasps during object interaction. However, the table also shows that the current approaches do not consider user intent which can result in jarring hand changes. Furthermore, current approaches can be computational expensive, particularly when using machine learning.

Table 2.6 - Comparison of the main virtual hand object grasping approaches presented

Feature	**Authors**		
	Holl et al (2018)	**Nasim and Kim (2018)**	**Tian et al (2018)**
Technique	Coulomb friction model	Coulomb friction model and assistive grasp VS Leap Interaction (LI)	Particle swarm optimization
Pre-recorded Training Data	No	No	Yes
Machine Learning Use	No	No	Yes
Physics Based	Yes	Yes	No
Game Engine Collision Detection	Yes	Yes	Yes
Performance	0.6ms for 6 phalanges, 1.5ms for 32	Average Total (sec) LI - 25.39 Coulomb - 25.47 Assistive - 14.41	15.4ms for a grasp Average Times 63.4 - raycast 46.8 – pinch 53.8 - proposed

Considers User Intent	No	No	No
Freezes Hand Interaction	Yes	No	No
User Evaluation	Naturalness System-4 Leap Interaction-2.5	Naturalness LI – 19% Coulomb – 43% Assistive – 38%	Simple Objects 70.5% vs pinch 82.1% vs raycast Complex Objects 80.6% vs pinch 91.3% vs raycast

This section has introduced vision/optical-based interaction techniques/methods used in virtual reality. In the next section, research into the use of machine leaning to improve optical tracker-based solutions is presented. Furthermore, the research presented attempts to address the four key issues with optical trackers, previously presented.

2.3 – Machine Learning in Optical Tracking Systems

The previously discussed research on optical-based hand interaction such as Pinto et al (2015) has shown significant user preference. Furthermore, research discussed such as Valentini & Pezzuti (2017) has shown the accuracy of optical trackers such as the Leap Motion to be ideal for hand interaction mechanisms. However, research has also shown that optical trackers suffer with issues such as occlusion (Shao, 2016) and can experience issues with tracking accuracy. There has been some research that has explored the use of multiple sensors through approaches such as data aggregation and machine learning for greater depth and more accurate data (Clark & Moodley, 2016; Kumar, et al., 2017; Bachmann, et al., 2018; Caserman, et al., 2019; Kiselev, et al., 2019). However, the resulting user experience was restricted to a limited field-of-view and a pre-set gesture

vocabulary as opposed to unrestricted and natural, direct hand-based interaction.

Marin et al (2014) analysed the performance of both a statically positioned Leap Motion and Kinect at detecting a series of gestures and the effect of combining the two devices on recognition accuracy. A dataset containing ten different gestures, performed ten times by fourteen different people, for a total of 1400 data samples was obtained. Fingertip distances provided the greatest accuracy (76%) compared to fingertip angles (74.2%) and elevations (73%). However, the system confused two mirroring gestures (palm facing with index extended and palm facing with pinky extended). To improve the systems accuracy the three feature descriptors were combined, resulting in 81% accuracy. Data analysis shows the Leap Motion provides a higher level but more limited data description, while the Kinect provides a full depth map. The combination of the two sensors resulted in an accuracy of 91.28%. Occasionally the Leap Motion data is not completely reliable as some fingers can be occluded (Shao, 2016). To overcome this issue the authors proposed a Support Vector Machine (SVM) classification algorithm to improve recognition. The accuracy of the Kinect system at recognizing gestures was greater than the Leap Motion (89.71% to 80.86%), showing the Kinect is a feasible option for gesture recognition. However, the Kinect must be statically positioned, therefore the user's body can occlude their hands. Nevertheless, these works show that multi-sensor aggregation can improve accuracy by reducing body occlusion.

Clark and Moodley (2016) evaluated a hand gesture recognition system for VR applications using the Leap Motion mounted on the front of a Head-

Mounted Display (HMD). The authors evaluated the system with regards to latency, accuracy, ease of expansion, ease of use and comfort. The system used a feature vector of five normalized tip-to-palm distances and a k-nearest neighbour (kNN) classifier to classify hand gestures. The results showed an average classification time of 0.057ms with an accuracy of 82.5% on four distinct gestures. The work expands upon that of Marin et al (2014) and their use of normalized finger distances (fingertip to palm). A preliminary user study showed the gestures; fist, point, open hand and ok were correctly classified 100%, 60%, 80% and 90% respectively, giving an average accuracy of 82.5%. This work positioned the Leap Motion sensor on the front of the HMD to provide a better view of the user's hands. While an improvement, this approach can still result in the user self-occluding, with the back of their hand blocking the sensors view of their fingers. The authors suggest that augmenting the data with an additional Leap Motion sensor on a different axis, could improve accuracy and limit this issue (Clark & Moodley, 2016; Placidi, et al., 2017; Zeutzheim, 2019; Zou, et al., 2019).

Machine learning techniques such as that seen in Clark & Moodley (2016) can be highly effective in static gesture recognition. The kNN system presented is lightweight and fast. However, as the authors suggest, the system would struggle to handle more complex gestures or a large gesture vocabulary as it must iterate through every dataset entry. There has been significant research in applying different machine learning techniques to real-time gesture recognition. Research has evaluated techniques such as neural networks (NN) and support vector machines (SVM), with each showing their own strengths and weaknesses (Trigueiros, et al., 2012).

One of the most important factors in using machine learning for gesture recognition is the simplification of the algorithm and processing time (Choi, et al., 2001). This is necessary to ensure that gestures can be recognized in real-time and the system does not introduce lag that would negatively affect a user's sense of immersion.

Trigueiros et al (2012) present a comparative study of four classification algorithms for static hand gesture classification using two different hand feature data sets. Their approach was to identify hand pixels in each frame, extract features and use those features to recognize a specific hand gesture. The techniques compared were k-Nearest-Neighbour (kNN), Naïve Bayes (NB), Neural Networks (NN) and Support Vector Machines (SVM). The features for the training sets were mean and variance of grey pixel values, blob area, perimeter and number of convexity defects, hand orientation, orientation histogram and radial signature. The results concluded that neural networks provided the best performance for gesture recognition. The results also concluded that the first datasets attributes (hand angle, mean and variance of the segmented hand grey image, area and perimeter of hand blob and the number of convexity defects) resulted in better hand gesture identification. The authors use of a low-resolution camera makes the results particularly applicable to devices such as the Kinect which has been shown to use low resolution (640x480) images in its colour and depth maps (Ren, et al., 2013). The addition of a dedicated hand tracking sensor such as the Leap Motion, could make identifying hand orientation easier along with aiding in improving the overall tracking.

McCartney et al (2015) investigate the effectiveness of the Leap Motion controller for hand gesture recognition, through the training of a 3D recognition model based on convolutional neural networks. The system was trained using nine-thousand six-hundred gesture instances resulting in an accuracy of 92.4%. The training data consisted of several palm features including width, position, normal, pitch, roll and yaw. In addition, fifteen distinctive features from each finger were used including position, length, width and direction. In total 116 features for each frame of the recording were captured, with a gesture typically lasting around 100 to 200 frames. The authors concluded that despite its good performance, one of the limitations of this model is that it cannot provide online recognition of gestures in real time. The authors also concluded that the Leap Motion controller is a promising device for enabling gesture interaction.

Whilst this system offers a high recognition rate, it is significantly limited by the lack of real-time recognition which is critical for immersion and usability in a VR simulation. Any form of latency also called ‘lag’, between the participant’s action and the associated applications response could cause a conflict in the users brain (Halarnkar, et al., 2012). Gesture interaction requires real-time accuracy to detect and track small precise movements, particularly in the case of educational applications such as those within the medical profession, so precise movements can be practised.

Li and Dai (2014) present a hand gesture recognition system that uses a Bayesian neural network to translate hand gestures to corresponding commands. The authors used a Cyberglove featuring twenty-two sensors to get hand joint angles for their recognition system. Two datasets consisting of

320 hand gestures each, were created by three people performing gestures from sixteen categories. The first dataset was used to train the system and the second to test it. To test the generalization capability of the neural network, two additional datasets were created. The first consisting of 320 gestures with the testing dataset containing 960 gestures from three people (320 each). The results showed a 99.06% recognition rate for the first experiment, with a 95.6% recognition rate for the generalization test. The authors conclude that vision-based recognition is an alternative to the glove-based systems but that continuous gesture recognition is more complicated and requires further research.

Diliberti et al (2019) concur with Li and Dai (2014) on the use of joint angles, proposing a gesture recognition system using joint angles in a convolutional neural network. The authors used custom motion capture gloves to obtain the gesture datasets for both training and real-time recognition. Data augmentation was used to prevent overfitting by modifying factors such as starting hand pose prior to the gesture and by modifying the recorded bone angles +/- 20%. To evaluate the proposed system, fifteen participants randomly performed each of the twenty-three gestures, twenty times. The systems performance was compared against previous research approaches including that of Alavi et al (2016), Luzhnica et al (2016) and Yusnita et al (2017). The results show the authors method to outperform the other methods in at least one area such as number of inputs, vocabulary size or recognition accuracy. The authors conclude that while the system works well, the system could be improved by setting limits on predicted values to prevent impossible, un-natural hand positions being produced.

Research such as that of Li and Dai (2014), McCartney et al (2015) and Diliberti et al (2019) has shown hand features such as bone/joint angles are effective in hand gesture classification. However, traditional approaches such as the use of gesture vocabularies results in systems either being unable to provide real-time recognition or impossible hand poses being produced. Furthermore, the use of a gesture vocabulary as opposed to freely interacting with a virtual hand restricts the user's actions to a series of poses that must be memorized. The ability to freely interact with a virtual environment is a particularly important factor in providing users with a sense of presence and an immersive experience.

This section has introduced machine learning techniques/approaches for improving optical-based hand interaction in virtual reality. In the next section, research on the sense of immersion and presence within a VR simulation are presented. First, the concepts of immersion and presence are introduced with the definitions clarified. Research is then presented evaluating the effects of hand-based interaction mechanisms on a user's sense of immersion and presence.

2.4 – Immersion & Presence in VR

In a virtual reality environment, a user experiences immersion when they feel a part of the virtual world rather than the real world. An effective virtual reality experience allows users to become unaware of their surroundings and focus on their presence within the virtual world (Strickland, 2007; Furht, 2008; Colagrossi, 2019; Alfaro, et al., 2019).

The terms Immersion and Presence represent distinct concepts but are often confused or interchangeably used. Immersion enables a user to experience the virtual world as though it were real, using their five senses. In addition, immersion is the extent of engagement experienced. Immersion is a term used to describe the technology that can give rise to presence. Presence, on the other hand refers to a phenomenon where users act and feel as though they are in the virtual world (Meehan, et al., 2002; Slater, 2003; Kim, et al., 2017; Park, et al., 2019).

Immersion is objective and measurable; one system can have a higher level of immersion than another. Presence on the other hand is a psychological, perceptual, and cognitive consequence of immersion. Presence is an individual and context-based user response related to the experience of “being there” (Bowman & McMahan, 2007). The distinction between immersion and presence, with immersion considered a combination of many components rather than a single construct, enables applications to leverage the effects of strong immersion without the need for presence.

Park et al (2019) present a collaboration-based interaction mechanism to provide an improved sense of presence and immersion for users, using either hands or feet to interact with the virtual environment. Hand interaction was implemented using the HTC Vive controllers with the rear trigger used to open and close the virtual hand for object interaction. Foot interaction used the position and orientation data from the Vive controllers to control the virtual foot representation. The experimental application was divided into two tasks. The first task involved users collaborating using their hands and the second task focused on users using their feet. In the simulation, the first user must throw a

dart towards a specified number with the other user adjusting the position of the dart board to ensure the number is hit. This task was completed using both hand and foot control. Twenty-four participants were recruited and divided into groups of two, with participants taking turns to play each role. The results showed role to have no statistically significant effect on presence with participants experiencing a strong sense of immersion in both. In addition, the user collaboration was found to significantly increase the sense of presence compared to solo playthrough. Park et al (2019) show the effect that innovative interaction mechanisms designed around the use of traditional motion controllers can have on user experience and presence. However, it should be noted that a significant factor in the results was the collaborative element.

The previously discussed research has shown that more direct, natural interaction such as optical-based hand interaction can enhance a sense of immersion and presence (Argelaguet, et al., 2016; Lin & Jörg, 2016; Schwind, et al., 2017). Natural interaction allows users to transfer existing skills and expertise from real to virtual environments (Chapoulie, et al., 2015; Kim, et al., 2018).

Han & Kim (2017) evaluate a gaze-based hand interaction mechanism against a traditional gaze-based interaction mechanism, analysing the effects on user immersion, interest, VR sickness and dizziness. A Leap Motion sensor was used to provide the hand tracking data to detect the control gestures. Participants were divided into four groups: Gaze then Hand, Hand then Gaze, Gaze only and Hand only. Participants were asked to evaluate their interaction experiences on a scale of one to five. The results showed 80% of participants who experienced the gaze condition first preferred the hand interaction, with

83.34% preferring the hand interaction when experienced first. Statistical analysis showed statistically higher scores for satisfaction and immersion in the hand interaction condition compared to the gaze condition. Analysis of user experience showed that 45.24% of participants (81.28% of all respondents) stated the hand condition improved their immersion and helped them to accurately control the virtual objects.

User's react strongly when first experiencing immersive VR; seeing the stereoscopic graphics pop out of the screen and being able to interact with the simulation by performing hand gestures. Han & Kim (2017) have shown the effectiveness of more natural, direct interaction in creating an immersive user experience. Natural User Interfaces (NUI) are interfaces in which user interaction is natural, common and familiar (Alfaro, et al., 2019). Approaches such as hand interaction provide users with real-world familiarity and contribute to a strong sense of immersion and presence. However, while effective at enabling purely hand based interaction, optical tracking systems cannot provide tactile feedback, a key feature in replicating a user's real-world senses.

Kim et al (2017) present a hand-based interaction mechanism with a low-cost portable haptic feedback system. A Leap Motion sensor was used to provide the hand tracking with an Arduino and vibration motor providing the haptic feedback. The proposed system was compared against a standard Leap Motion system to determine the effectiveness of the haptic system in providing a stronger sense of presence to users. Twenty-one participants were recruited for experimentation in which participants experienced two virtual scenes; the first providing users with vibration feedback when a virtual object is touched.

The second scene used a defined gesture, triggering a heat sensor to accumulate heat, with a virtual bullet fired once the participant could feel the heat, simulating the heat of gun. The results show the proposed system provided greater overall satisfaction, with the system found to significantly enhance presence during object interactions. A Wilcoxon test showed the proposed interaction mechanism to provide statistically significantly higher presence in VR.

The previously discussed research such as Han & Kim (2017) has shown the importance of immersion for a strong sense of presence within a virtual world. Furthermore, a key aspect of a user's sense of immersion is the replication of real-world senses and interactions. The results of studies such as those by Han & Kim (2017) and Schwind et al (2017) have shown the impact of natural hand-based interaction and the importance of a realistic virtual hand representation. Researchers such as Feng et al (2018) and Krokos et al (2019) have shown that immersive virtual reality experiences can have significant impact on the effectiveness of training/education-based simulations. In addition, immersive VR training simulations have been found to improve memory recall in comparison to non-immersive VR simulations.

This section has introduced the concepts of immersion and presence, along with the effects caused by using vision/optical-based interaction techniques/methods in virtual reality. In the next section, existing works in the field of interactive educational VR are presented. The research presented is evaluated with regards to the interaction mechanism used and their effect on both user experience and training/educational aspects. Furthermore, the

limitations of current interaction approaches, with regards to interactive educational applications are discussed.

2.5 – Interactive Virtual Reality in Education

Immersive Virtual Reality applications can been used in a variety of real-world applications; such as, museum exhibitions (Kyriakou & Hermon, 2019), construction, medical rehabilitation and training (Postolache, et al., 2017; Pulijala, et al., 2018; Fahmi, et al., 2018; Moore, et al., 2020; Pinter, et al., 2020). Virtual reality is used as a learning medium because its immersive nature enhances learning by enabling users to explore and immerse themselves in a studied environment, such as ancient cities or looking into the human body (Madathil, et al., 2017). Research such as that of Stone (2000) and Lee et al (2010) has shown clear benefits of VR with reduced learning time and improved learning outcomes. Research has also shown natural VR controls further increase the interactivity and the sense of immersion as a result (Fahmi, et al., 2020).

Moore et al (2020) present the formative evaluation of a VR simulation designed for training troubleshooting skills during robotic surgeries. To interact with the simulation users, use the HTC Vive controllers following the simple virtual hand technique. The technique directly maps the users hand motion to the virtual hand (LaViola Jr, et al., 2017). In the case of this simulation, a model of the Vive controller was used as the virtual hand. As the simulation was designed for training, visual cues were used; specifically, green semi-transparent animations of the Vive controller performing the required movement. After completing the simulation, participants were asked to complete a series of questionnaires; specifically, Simulator Sickness

Questionnaire (SSQ), Spatial Presence Experience Scale (SPES) and System Usability Scale (SUS). Participants were also asked to complete a questionnaire about the level of interaction fidelity experienced in the simulation and a knowledge test. Twenty participants took part in the study. The results of the SUS show the controller-based interaction to reach an above average score, with users generally rating the interaction fidelity as acceptable. However, the results also show the use of virtual controller representations to be problematic with regards to interaction cues. The poor feedback in terms of handedness the controllers provide resulted in participants attempting to interact with the incorrect hand and dropping a previously held virtual object breaking sterility.

Research such as Moore et al (2020) shows motion controllers can support an interactive training VR simulation. However, the indirect interaction between the user and the simulation restricts the sense of immersion and by extension the learning effect (Patel, et al., 2006; Zhang, et al., 2019). Fahmi et al (2020) conducted a comparative study of user experience between motion controllers (Vive controllers), Leap Motion sensor and haptic gloves (Senso Glove) in an anatomy based educational simulation. Users were asked to complete the simulation with each control mechanism and complete a Likert-based questionnaire post each experience. The study focused on the level of acceptability, user satisfaction, ease of learning, the suitability of movement, suitability of display, and haptic feedback. The simulation was designed to enable participants to pick up anatomical objects with the name displayed. The three interaction mechanisms used the traditional ‘grab and release’ interaction approach with controller models and virtual hand representations

used, respectively. Twenty participants took part in the study. The results showed participants considered all three mechanisms to be useful but the Vive controllers scored the highest in both user satisfaction and ease of use.

While the results of studies such as Moore et al (2020) and Fahmi et al (2020) have shown motion controllers to enable sufficient fidelity for basic interactions. The previously discussed research in this chapter such as Pinto et al (2015), Argelaguet et al (2016) and Figueiredo et al (2018) has shown that optical trackers provide a more direct and natural interaction enhancing a user's sense of presence and immersion (Lin & Jörg, 2016; Schwind, et al., 2017; Fröhlich, et al., 2018). Research such as that of Fahmi et al (2018) has shown the effectiveness of optical trackers at providing direct interaction with virtual controls in a simulation designed for training on the operation of excavators. Furthermore, direct natural hand interaction better enables the transfer of existing skills and expertise from the real-world to virtual environments (Bowman, et al., 2012; Chapoulie, et al., 2015; Han & Kim, 2017; Kim, et al., 2018; Almeida, et al., 2019; Zhang, et al., 2019).

The ability to interact directly with the simulation to practice routine procedures and hand movements is particularly important in areas such as medicine. The capabilities of modern optical tracking sensors such as the Leap Motion to track tools held (Wozniak, et al., 2016) is particularly beneficial in fields that use tools as an extension of the hand. Lahanas et al (2017) investigate the potential of a VR simulation for the assessment of basic laparoscopic surgery skills using the Leap Motion sensor. A simple interface was used to simulate the real-world instruments with the Leap Motion sensor providing positional and rotational tracking of the instrument tips. The system evaluation consisted

of three tasks: camera navigation, instrument navigation and bimanual operation. The camera task required participants to search the environment and position the camera near a virtual gallstone. The instrument navigation task required participants to touch lit buttons within five seconds of activation without touching the board on which they are scattered. The bimanual operation task required participants to hold one instrument steady within a specified small area. This enabled the other instrument to pick up and drop a coloured ball into the corresponding pot. Forty-nine participants were recruited: twenty-eight experts (>100 surgeries) and twenty-one novices (<10 surgeries). Participants were assessed with regards to time, path length, and two task specific errors. The realism and training ability of the scenario was also evaluated via five-point Likert questionnaire. The results showed experts to significantly outperform novices in all tasks. However, results of the questionnaire indicate participants in both groups saw the value of the system as a training tool.

The past few years have also seen an increase in the use of virtual reality in museum environments in an attempt to embrace modern technology and adapt to the challenges of the digital era (Shehade & Stylianou-Lambert, 2020). Virtual reality provides museum visitors with an educational and entertaining experience, enhancing their overall experience by facilitating their interactions with the exhibitions (Lee, et al., 2019).

Kyriakou & Hermon (2019) integrate direct natural Interaction and augmented reality applied in a cultural heritage museum, to overcome the problems of inaccessibility and non-interaction with the museum artifacts. The authors used a generic HMD to house a mobile phone that provided the visuals and a

Leap Motion to enable hand interaction. Object interactions were performed using the traditional ‘grab and release’ gesture with the virtual object moving and rotating with the virtual object while held. To evaluate the effect of the system the authors demonstrated the system at three different museums, recruiting a total of fifty participants. Participants were asked to interact with the six virtual objects, both moving and rotating them one at a time. At the end of the experiment, participants were asked to complete a questionnaire. The results showed participants of all ages to enjoy the experience along with having felt a strong sense of immersion and presence.

Augmented sandboxes have been used as educational tools to create, explore and understand complex topics. However, current solutions lack interactive capabilities (Stabbert, et al., 2017; Millar, et al., 2018; Alexandrovsky, et al., 2019; Gabele, et al., 2019; Muender, et al., 2019). Frohlich et al (2018) present a VR sandbox system consisting of an actual sandbox, triple Kinect depth sensing, HMD and hand tracking via a Leap Motion. The depth images from the three Kinect sensors are combined to generate a point cloud and modify the terrain of the virtual environment. To ensure a real-time experience the terrain is represented in a block-based style on a 98x59 mesh with 32 height levels providing a compromise between detail and latency. The total latency from sand forming to the virtual environment being updated was 1.5 to 2 seconds. Users can decorate the virtual environment, changing the environment lighting and adding details such as water and trees. Nine participants were recruited to gather feedback from both a technical perspective and application areas such as education. Participant feedback showed the experience to be engaging and immersive with a strong sense of

presence, particularly in the first-person view exploring the virtual world. The free-hand interaction was positively received with natural gestures used for object interaction such as pouring. However, participants also reported issues with the Leap motion including losing tracking, occlusion, orientation issues and field-of-view limitations.

Qingchao & Jiangang (2017) investigate the cause of occlusion in hand based interaction using a Leap Motion in an astronaut training simulation. Initially the authors experimented with the Leap Motion under various occlusion cases and evaluated the tracking accuracy. The Leap Motion was then implemented into a training simulation in which users must interact with virtual objects and move them around the environment. To interact with virtual objects, the 'grab and release' mechanic was used with an object in contact with a virtual hand, held upon making a fist. Based on observations during experimentation, the authors suggest that during occlusion, the similarity between the current and previous tracking data is such that the previous un-occluded data can be used. Secondly, when a finger is occluded, the Leap Motion cannot determine the occluded finger and thus the tracking data cannot be trusted. Finally, during hand-to-hand interactions the Leap Motion is likely to lose tracking due to occlusion. The authors conclude that self-occlusion is a significant factor in training applications and thus gesture vocabularies should be restricted to gestures that avoid or reduce occlusion.

Despite the possible benefits that VR can offer, it is still not widely used in education. Furthermore, the potential as an assistive technology for alternative communication is recognized but has not been fully exploited (Zilak, et al., 2018). Zilak et al (2018) present the development of an elementary

mathematical virtual classroom using the Oculus Rift and Leap Motion. User evaluation of the prototype was conducted to assess user satisfaction and acceptance of the technology. The simulation required participants to press virtual buttons represented as domino's that either show the specified numbers or match the given domino tile pattern. To evaluate the system, thirty students were recruited. Analysis of user performance showed no significant correlation between simulation time and the number of incorrect answers. The authors conclude that user performance indicates individual levels of immersion, individual interaction approach and feelings about the technology. Questionnaire results show generally positive feedback from users without disabilities thus the authors conclude that VR is appropriate for education and as an alternative communication approach. Analysis of questionnaire responses and simulation time shows that participants who reported high scores and long playtime, experience a strong sense of immersion, and thus prolonged the interaction. The results also show factors such as immersion, virtual hand behaviour and familiarity with VR are key areas that need to be considered in applications for users with disabilities.

The previously discussed research such as Zilak et al (2018) and Kyriakou & Hermon (2019) has shown the effectiveness of virtual reality in education applications. Immersive VR provides users with an engaging experience in which they can gain understanding of complex concepts. The discussed research has also shown the effect of direct natural hand-based interaction in providing a strong sense of presence and immersion. Furthermore, direct hand interaction enables the direct transfer of skills and experience, to and from the virtual and real world. However, research such as that of Qingchao & Jiangang

(2017) and Frohlich et al (2018) has shown the limitations and issues of current optical trackers. Current optical trackers provide an effective solution to interaction in simulations that do not require consistent precise interaction such as the museum application of Kyriakou & Hermon (2019). The current issues with areas such field-of-view and occlusion limit the areas in which the technology can be adopted. Thus, despite the potential that has been demonstrated by research such as Lahanas et al (2017), Pulijala et al (2018) and Pinter et al (2020), medical training research applications typically use motion controllers for the stability and tracking precision.

2.6 – Summary

Research has shown wands and motion controllers provide faster, more accurate, stable and consistent interactions in both CAVE and HMD based virtual environments. However, research such as that of Zhang et al (2014), Nan et al (2014) and Chapoulie et al (2014) has shown a preference for direct hand-based interaction with participants stating it provided a more natural and realistic experience. Hybrid approaches merging motion controllers with basic finger tracking and haptics such as Choi et al (2018) have shown positive results in users able to perceive differences in haptic feedback force. However, such approaches are typically limited to one or two finger interaction and can encumber users with large motors and actuators mounted on the hand or arm.

Glove-based approaches can provide more direct interaction than motion controllers, typically using either pattern-based recognition or bend sensors to track hand and finger positioning. Pattern-based recognition can detect the positioning of fingers on multiple axes but can be significantly affected by

occlusion. In contrast, bend sensors are only able to track fingers on a single axis but are not subject to occlusion.

Research has shown optical-based trackers to provide the most immersive and natural hand interaction experience. Results from participant feedback has shown an overwhelming preference for direct hand interaction, transferring skills back and forth between the virtual and real world. The previously discussed research in section 2.5 has shown hand interaction is particularly important in educational applications in areas such as medicine.

However, as the previously discussed research has also shown, optical trackers suffer with four key issues; stability, occlusion, accuracy and field-of-view which limit the interaction experience. Research such as Trigueiros et al (2012) and McCartney et al (2015) have shown neural networks to be an effective approach for improving optical trackers. However, research has also shown that using a gesture vocabulary for recognition can impact real-time performance. Furthermore, such approaches are typically used for static gesture recognition as opposed to direct real-time hand tracking and interaction. Thus, the ability to recognize/validate hand tracking data in real-time is key in the machine learning approach designed.

The use of multiple sensors has been shown to improve both accuracy and field of view as hypothesized by Shao (2016). Furthermore, studies by Marin et al (2014), Pinto et al (2015) and Jin et al (2016) have shown multi-sensor solutions to reduce the effect of occlusion in tracking data. However, while the solutions presented have been shown to reduce the effects of the four key issues, studies have traditionally used statically positioned sensors. Typically

using only Kinect sensors or used a Kinect to provide the additional tracking data. The Kinect has been shown to be an effective optical tracker; however, it is designed for skeletal tracking while statically positioned. Finger tracking is typically implemented using computer vision libraries such as OpenCV, but its static position often results in both body and hand occlusion. Furthermore, multiple sensor solutions have not been evaluated in a HMD based VR simulation context.

Research discussed such as Zhao et al (2013), Kim & Park (2015) and Holl et al (2018) has shown the effect that realistic, physics-based hand representations can have on a user's sense of immersion and the natural feeling of the interaction. However, research has also shown that calculating positions or using predefined poses can appear jarring to the user and negatively affect immersion. In addition, feedback indicates physics-based approaches are more difficult to learn than pinch grasping or ray-casting. Furthermore, physics-based approaches are reliant upon consistent and highly accurate tracking data to compute the required friction forces. The limitations of requiring prior knowledge in the case of machine learning and the inconsistency in the computationally expensive physics-based approaches needs to be overcome. Based on research it can be hypothesized that a ray-casting based approach, such as that by Holl et al (2018) that manipulates user hand positioning for a realistic grasp would provide the optimal performance and accuracy. Furthermore, a ray-casting approach would enable the calculation of surface contain points without the need for a computationally expensive physics system.

The research presented here has shown that using multiple sensors can improve accuracy and field-of-view. However, current research has typically focused on static sensors such as the Kinect. In addition, multi-sensor solutions have not been applied to a HMD context. Furthermore, the four key issues with optical trackers are unresolved, resulting in users still experiencing issues such as hands disappearing, occlusion and the general stability issues present in optical tracking. Based on research it can hypothesized that the use of multiple sensors with complex software to evaluate and process tracking data in real-time, adjusting for noise/errors could provide a more natural and immersive experience. The next Chapter presents the phase one design of the Multiple Optical Tracking (MOT) system and its various sub-systems.

Chapter 3 – Design Phase One

In this Chapter the phase one design of the Multiple Optical Tracking (MOT) system and its sub-systems is presented. The MOT system is a hand-based interaction system for VR that uses optical based tracking sensors to integrate users' hands into a virtual environment. The MOT system is designed to provide more natural and intuitive ways to interact with a virtual environment and can be applied to any interactive application that requires hand-based gesture interaction, such as using virtual equipment in training. Appendix 3 contains screenshots of the implementation code.

The MOT system comprises two Leap Motion sensors mounted onto a VR headset at two different orientations. One Leap is positioned on the front of the headset facing forward and the other Leap is mounted on a custom 3D printed bracket and positioned to face downwards at a 45-degree angle. The design of the custom bracket is discussed in section 3.1. The second sensor was added to provide an additional view of the user's hands to reduce the effect of self-occlusion, whilst not limiting the detection space. Two Leap Motion sensors were used to provide optimal coverage whilst not adding too much weight to the HMD or causing discomfort to the user. Figure 3.1 illustrates the Leap Motion setup for the MOT system, using two sensors: one mounted front-facing with the second using a custom 3D printed bracket.

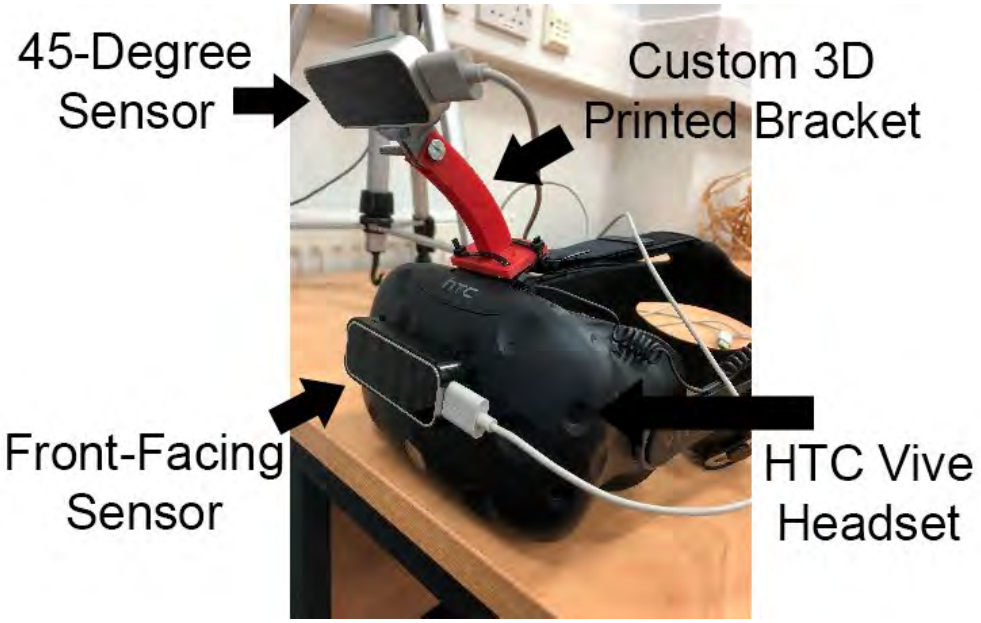


Figure 3.1 - The HTC Vive headset with two Leap Motion sensors attached using a designed custom bracket

Figure 3.2 provides an overview of the MOT hand-based interaction system. Each stage of the MOT system and it's respective sub-stages shown in Figure 3.2 is discussed in the following appropriately named sub-sections within this Chapter.

Figure 3.2 shows the VR HMD on the left of the diagram collecting hand data via two Leap Motion sensors. The front mounted sensor data is fed directly into the data processor. While data from the additional sensor is sent over a local network into the data processor using a generic hand data structure format. The process shown in Figure 3.2 runs at approximately 90 times a second to ensure the virtual representations are updated in real-time.

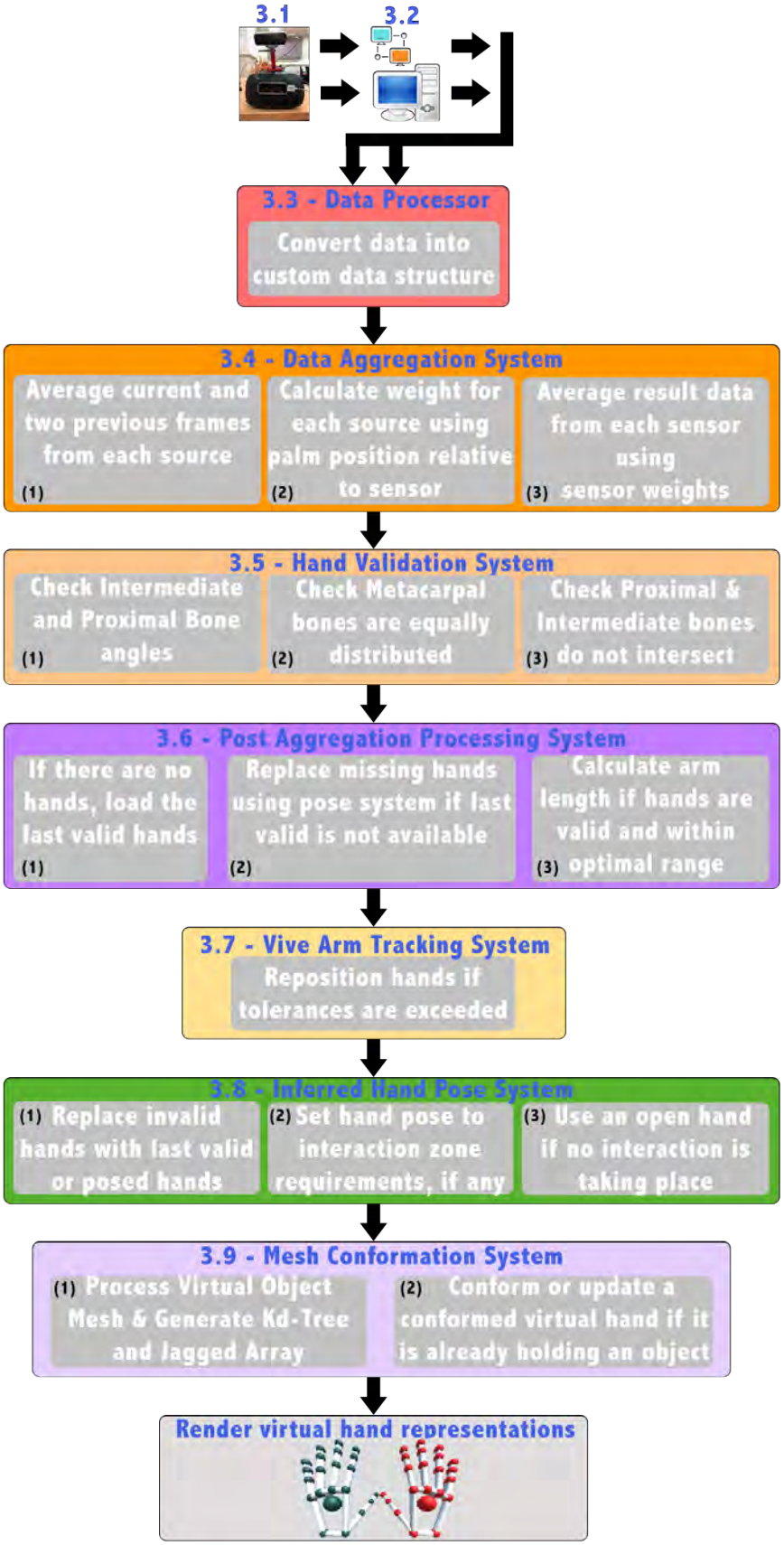


Figure 3.2 - Overview of the Multiple Optical Tracking (MOT) Systems components

As previously discussed, the remainder of this Chapter outlines the design of the various stages/subsystems in the MOT system shown in Figure 3.2, in the following subsections.

3.1 – Custom Bracket & External Sensor Configuration

In this section, the design of the custom bracket and external sensor configuration as shown in Figure 3.3 from the main overview diagram in Figure 3.2 will be detailed.

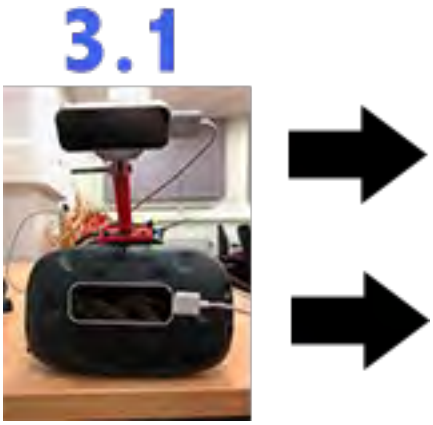


Figure 3.3 - Custom Bracket & External Sensor Stage from the MOT System Overview Diagram

For the additional sensor to maintain a relatively consistent stream of tracking data (assuming the hands are within range) irrespective of the user's gaze direction and HMD rotation, the sensor would need to be mounted to the HMD. A statically positioned sensor would provide additional tracking data; however, it would suffer with significant occlusion issues due to the user's body and arms as they rotate.

An additional sensor needs to have a clear view of the user's hands and must be able to rotate with the user to prevent body occlusion. However, the sensor

should not restrict the user's interaction space, such as mounting the sensor to the bottom of the headset. During the development of the bracket, several different mounting points were explored including face-up on the bottom of the HMD and face-up chest mounted. However, these mounting points either interfered with the user experience or would have a restricted field-of-view. In the case of the bottom mounted HMD bracket, it would restrict how far down a user's head could tilt. In the case of the chest mounted sensor, consistent positioning would be difficult, and it could potentially be uncomfortable. Furthermore, the user's upper arms would frequently occlude the sensors view of the hands.

To provide an additional view of the user's hands without interfering with the user's interaction space, the sensor needed to be mounted on-top of the HMD angled down. This provides an overlap with the front-facing sensor and avoids 'dead space' where the users hands cannot be tracked by either sensor.

During developmental testing, several different sensor angles were evaluated to analyse the effect on field-of-view and accuracy. An angle of 45-degrees was identified as the optimal angle, providing as close to a "top-down" view as possible while keeping the detection range as large as possible. The additional sensor was mounted at a 45-degree angle to provide an additional view of the user's hands to reduce the effect of self-occlusion, whilst not limiting the detection space. The sensor angle used was determined through analysing the weighting of incoming data relative to the external sensors optimal tracking position. This enabled the optimal angle for a smooth transition between the two sensors, while maintaining a strong weighting and ensuring no "dead space", to be determined.

The combination of a longer bracket arm and orienting the sensor to 90-degrees would provide a top-down view, however the longer arm would increase the weight of the HMD. In addition, the short distance between the sensors surface and the minimum tracking distance would result in a very small tracking space minimizing the effect of the additional sensor. Figure 3.4 shows a diagram visualizing the viewing angles of the sensors used by the MOT system and the external sensor if positioned at 90-degrees. Note, the length of the viewing angles in Figure 3.4 is not representative of the Leap Motion sensor range.

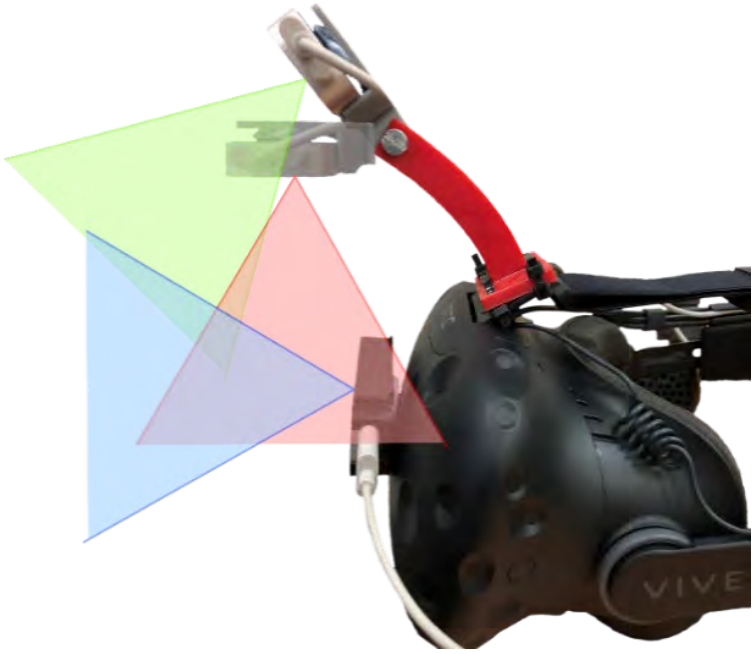


Figure 3.4 - Diagram of sensor view angles. 45-degrees provides the best multiple sensor coverage; blue) front-facing, red) face-down, green) 45-degree sensor

To support the mounting of an additional sensor, a custom 3D printed mount was designed that can be attached to a head-mounted display, providing an additional view of the user's hands. The bracket uses a Go-Pro (GoPro, 2020)

inspired joint for the Leap mounting plate that allows the angle of the sensor to be adjusted. Figure 3.5 shows the final design 3D printed and assembled. The four holes in the bottom plate are used to attach the bracket to the headset using four cable ties. All the parts were 3D printed using PLA and assembled using superglue to attach the base and a bolt & nut to attach the head mount. Figure 3.6 shows a technical drawing of the final bracket design. Appendix 1 and Appendix 2 contain larger and colour versions of Figure 3.6.

Figure 3.5 – Final design for the bracket

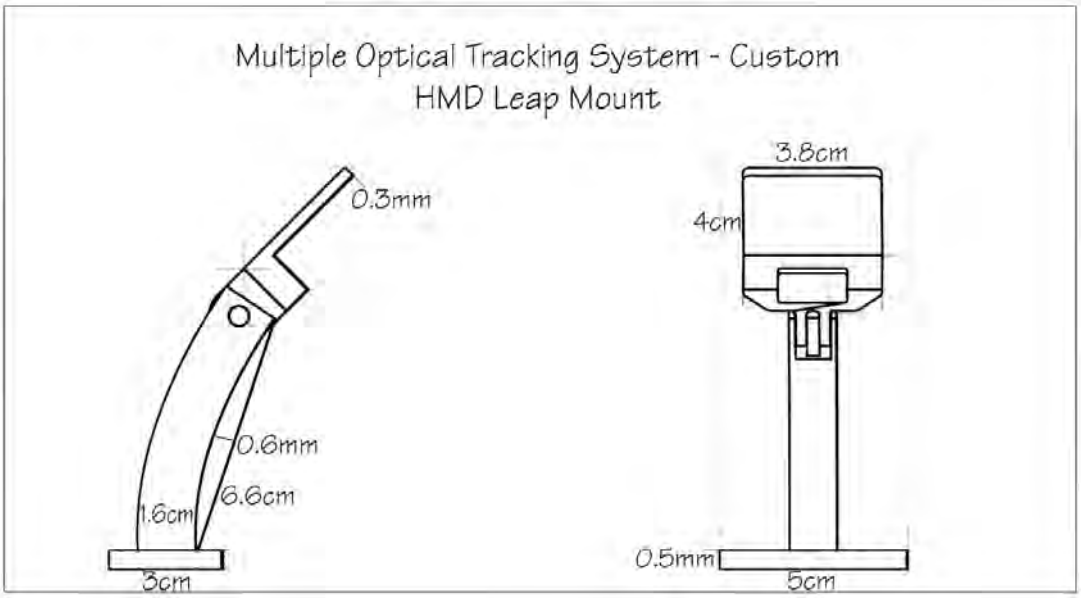


Figure 3.6 - Blueprints for the custom bracket

3.2 – External Data Packet Handler

In this section, the design of the external data packet handler as shown in Figure 3.7 from the main overview diagram in Figure 3.2 will be detailed.

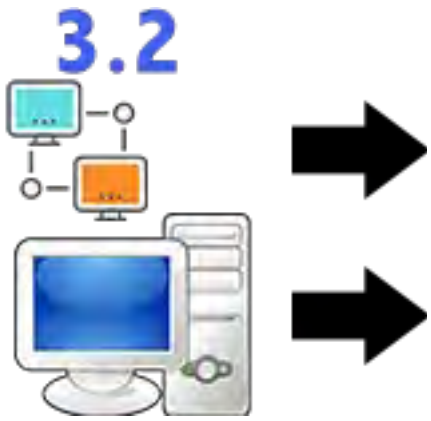


Figure 3.7 - External Data Packet Handler Stage from the MOT System Overview Diagram

Due to the Leap Motion SDK (v4.0.0+52173) only being able to support a single sensor, an external Win32 C# client application was developed to send

the tracking data over the local network. A custom serialization library known as the ‘MultiLeapTransferPacket’ was developed to serialize optical tracking data as serialization had not been implemented into the Leap SDK. The library was created in C# as this is the programming language supported by the Unity3D game engine and Leap SDK, enabling easy implementation of the compiled library. Figure 3.8 shows the structure of the custom ‘MultiLeapTransferPacket’ (MLTP) library developed.

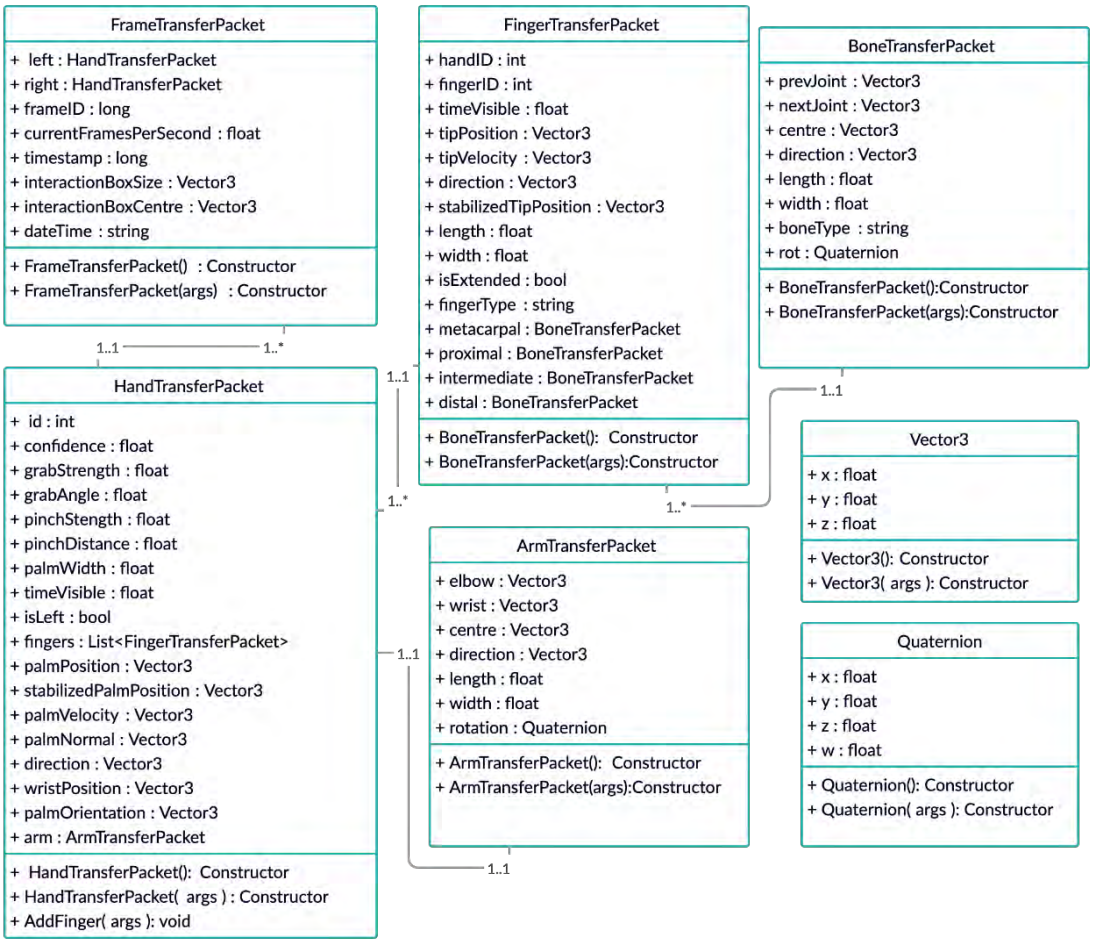


Figure 3.8 – Class Diagram of the custom library developed known as the MultiLeapTransferPacket library

The custom library was designed to contain all the necessary tracking information and serialize the data so it can be sent over the local network. As can be seen in Figure 3.8 the library contains a series of classes for data type such as Vector3 and Quaternion and parts of the hand.

The BoneTransferPacket (BTP) class contains data about bones such as the start and end positions along with rotation. A BTP is created for each bone in the virtual hand with each assigned to the corresponding finger. The position and rotation of each bone is stored so it can be precisely positioned when rendered via the MOT system. Furthermore, this enables poses captured via the sensor to be perfectly recreated.

The FingerTransferPacket (FTP) is comprised of four BTP, one for each bone, enabling the finger to be reconstructed and factors such as extension state determined. Reconstructing factors such as extension state ensures that any gestures or poses captured by the external sensor are detected by the MOT system when the virtual hand is rebuilt. This enables the user to interact with virtual objects when the hand is only captured by the external sensor. The ArmTransferPacket (ATP) provides information required for a virtual arm representation and control of the hand such as direction and rotation.

The HandTransferPacket (HTP) contains a list of FingerTransferPackets, each representing a finger of the virtual hand. As previously discussed, this ensures the hand can be accurately rebuilt within the MOT system. The HandTransferPacket also provides valuable information such as the position and rotation of the hand. This is used to accurately reposition and rotate the virtual hand data to align with the data of the front-facing sensor. The

FrameTransferPacket (FRTP) contains a HandTransferPacket for both hands and information such as the timestamp of the frame to determine the age.

The library enables the tracking data to be sent to the MOT system and be rebuilt for use in the MOT system via the data processor detailed in section 3.3. The use of a custom library for transferring external tracking data enables alternative optical trackers to be used as an external source in the MOT system without modification. The process of receiving the data and processing the data using the library is presented in this section.

The data received on the host machine is deserialized into a FrameTransferPacket object and added to a queue for processing by the “MultiLeap_DataHandler. Each of the HandTransferPackets are processed individually, iterating over the FingerTransferPackets, creating Leap.Bone objects from the BoneTransferPackets. A Leap.Finger object is created once all the corresponding bones have been processed. Once all the finger objects have been created, a hand object is constructed using the finger objects and additional data stored within the HandTransferPacket, such as palm position and rotation, before being added to an empty frame object.

Once all the hands have been processed, the generated frame object is transformed to account for the orientation of the head-mounted display. The transformation process orients the hand data to align with the rotation of the HMD. If the virtual hand data from the external sensor was not positioned and rotated based on the position and rotation of the HMD. The virtual hands would move and rotate with the HMD as with the data from the front-facing sensor. However, while rotating the HMD, the hands would rotate on a differing axis

and thus would appear to spin in place. The resulting frame object is then sent to the data processor with the front-facing sensor data for processing into the MOT system.

3.3 – Data Processor

In this section, the design of the data processor stage as shown in Figure 3.9 from the main overview diagram in Figure 3.2 will be detailed.

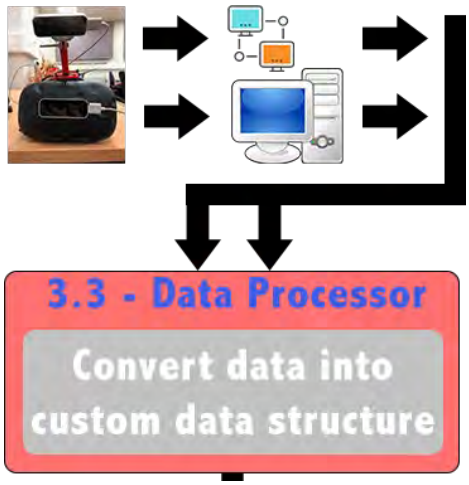


Figure 3.9 - Data Processor Stage from the MOT System Overview Diagram

To separate the Leap Motion tracking data from the overall system, a conversion feature was developed that converts sensor specific tracking data into a custom data structure. The design of the data structure is based on the Leap Motion API using objects such as Bone and Finger. The data structure also provides additional functionality to make accessing and manipulating tracking data easier. This custom data structure forms the input to the MOT system.

The Data Processor receives Leap.Frame objects directly from the front-facing sensor and external data via the External Packet Handler. The frame objects received are converted into the MOT systems custom data structure; specifically, a HandData object. A HandData object is used to store Leap.Hand objects and provide additional/ease of access functionality such as getting a specific hand (left or right).

The resulting HandData objects are stored in a dictionary containing a list of HandData objects, using the source as the key. For the front-facing sensor the key “localhost” is used with the IP address of the external client used for external data. The dictionary stores the current HandData object and those for two previous tracking frames to provide a tracking history for the Data Aggregation system detailed in section 3.4.

This pre-processing enables the system to work with any hand tracking sensor, the hand data converter would just require an update to support the sensors data format.

3.4 – Data Aggregation System

In this section the design of the data aggregation system (Figure 3.10) and the algorithms used to analyse the tracking data from multiple sources are detailed. Figure 3.10 shows the Data Aggregation System stage from the MOT System Overview Diagram shown in Figure 3.2.

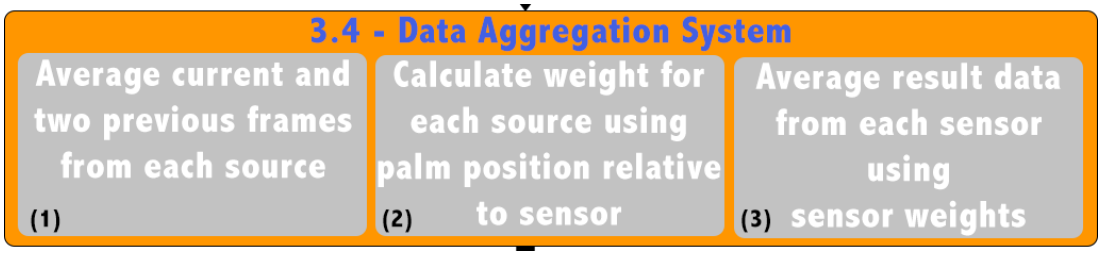


Figure 3.10 - Data Aggregation System from the MOT System Overview Diagram

The Data Aggregation System is designed to aggregate hand tracking data received from the two sensors to improve the accuracy of the virtual hand representation and reduce the effects of self-occlusion. This is the first key stage in the MOT system, taking place directly after the data processor has converted the input data to the systems custom structure.

The data aggregation system starts by processing both the current and previous two frames for each of the data sources to smooth out any glitches or noise in the tracking data. During developmental testing, several frame history sizes were evaluated including storing one, four and six previous frames. However, it was found that averaging more than two frames would result in a user feeling that their hand movements were in slow-motion or lagging and that the virtual hands did not align with their own. This was due to the averaging of more frames smoothing out too much information from a user's hand movements. Users preferred virtual hands that felt responsive and closely aligned to their real-world actions. Furthermore, the use of only a single previous frame did not provide sufficient data to have any significant impact on accuracy and noise.

The average hand position and rotation, across all three frames for each sensor is calculated using a set of weights (0.5, 0.3, 0.2). The weights were chosen so that more recent frames were given greater impact over previous frames. Based on experience of the Leap Motion, a range of weightings were explored, whilst ensuring the most recent frame had the greatest impact. However, it was found that the weighting above provided the optimal configuration. These weights were selected to give the most recent data greater significance over older frames. While using the two previous frames to reduce the effects of any noise or errors in the tracking data. The weights ensure the most recent data significantly contributes whilst ensuring the previous frames have enough weight to affect the recent data.

To aggregate the frame histories for each sensor, each frame in each sensor's history is iterated over. The first step in the aggregation is calculating the weight of the current tracking data. This calculation uses the hands proximity to the sensor to rate how visible the hands are and as a result how confident the system is in the hand data. To rate the visibility of the hand data the system calculates its distance from an origin that represents the optimal hand position for hand tracking. For each frame of each sensor, a weight in the range of -1 to 1 is calculated based on the hands position relative to the defined origin, with greater proximity to the origin position scoring higher. Algorithm 3.1 shows pseudocode for the calculation used to determine the weighting of a sensors tracking data using a hand position. The position on all three axes is evaluated against a pre-set range and mapped to the range of minus one to one. Figure 0.1 shows the code implementation.

Algorithm 3.1 - Pseudocode for calculating a Sensors weight using a hand position

```
Calculate Sensor Weight using Hand Position
clampedX = (1f - (-1f))/(MaxX - MinX) * (HandXpos - MaxX) +1f;
clampedY = (1f - (-1f))/(MaxY - MinY) * (HandYpos - MaxY) +1f;
clampedZ = (1f - (-1f))/(MaxZ - MinZ) * (HandZpos - MaxZ) +1f;
totalClamp = clampedX + clampedY + clampedX;
weight = (1f - 0f) / (3f - (-3f)) * (totalClamp - 3f) + 1f;
weight = ((weight - 0f) / (1f - 0f)) * (1f - (-1f)) + (-1f);
weight = 1f - Abs(weight);
weight = Round weight to two decimal places;
```

Once the weight of the current tracking data has been calculated, the data from the 45-degree sensor is transformed to align with the position and rotation of the local sensor data. The data from the two sensors is then used to perform the aggregation process. To aggregate the tracking data, each bone in each finger of each hand is iterated over. The value of each property is multiplied by the current frames weight and added to the HandData object. Table 3.1 shows a sample of the properties aggregated. Figure 3.8 shows all the properties/data for each of the objects that a tracked hand is comprised of.

Table 3.1 - Sample of the data aggregated from the Leap Motion sensors

Bone	**Finger**	**Hand**
nextPosition (end)	tipPosition	palmPosition
prevPosition (start)	direction	palmNormal
centre	length	width
direction	metacarpal	isLeft
length	isExtended	palmOrientation

Equation 3.1 shows the equation used to perform a weighted average of the bone vectors and quaternions along with length and width. In the equation $\boldsymbol{B^1}$, $\mathbf{B^2}$ and $\boldsymbol{B^3}$ represent the values (vectors, floats or quaternions) with $\boldsymbol{W^1}$, $\mathbf{W^2}$ and $\boldsymbol{W^3}$ representing the weights of the three corresponding frames.

Equation 3.1 - Equation used to perform a weighted average of three Vectors, Floats or Quaternions

$$A = (B^1 * W^1) + (B^2 * W^2) + (B^3 * W^3)$$

Figure 3.11 visualizes the frame history averaging process with three frames (blue, brown and green) averaged to produce a resulting hand (red).

Figure 3.11 - Diagram showing the effect of frame history averaging

Algorithm 3.2 shows pseudocode that performs the weighted averaging of a frame history. If the current frame is the first in the history the current hand object is assigned property values with subsequent frames adding to the properties to perform the weighted average. Figure 0.2 shows the code implementation of Algorithm 3.2.

Algorithm 3.2 - Pseudocode for averaging hands using the frame history for a given source

```
Average the Frame History of a Sensor
HD = Create Empty HandData Object to store result;
foreach Frame in Sensor History do
    Calculate Sensor Weight for this Frame;
    Transform the data to align with front-facing if it's external data;
    foreach Hand in frame do
        foreach Finger in Hand do
            foreach Bone in Finger do
                if Current Frame is the first frame then
                    Set Length, Width and Type of current bone in HD
                      object using data from the current bone;
                    Set Direction, Next Joint, Centre, Previous Joint and
                      Rotation in HD Object using the data from the
                      current bone * current frame weight;
                else
                    Add Direction, Next Joint, Centre, Previous Joint and
                      Rotation to the current bone in HD Object using
                      the data from the current bone * current frame weight;
                end if
            end
            if Current Frame is the first frame then
                Set Direction of current Finger in HD object using
                  direction from current Finger * current frame weight;
                Set Type of current Finger in HD object using
                  current Finger Type;
            else
                Add Direction to the current Finger in HD object using
                  Direction from current Finger * current frame weight;
            end if
        end
        if Current Frame is the first frame then
            Set IsLeft of current Hand in HD object using
              IsLeft from current Hand;
            Set PalmNormal, PalmPosition and Rotation of current Hand
              in HD object using data from
              current Hand * current frame weight;
        else
            Add PalmNormal, PalmPosition and Rotation of current Hand
              to HD object using data from
              current Hand * current frame weight;
        end if
    end
end
```

The weight calculation and averaging/adding process detailed is performed for each frame in each sensors frame history to calculate the resulting hand data for each sensor. Once the hand data for each sensor has been averaged using their respective frame histories. The average weight for each sensor is calculated using the weights calculated for each frame in their respective histories.

The final step is to aggregate the resulting tracking data from each sensor to produce the final tracking data. To determine the weighting of each sensor with regards to the final data, the average weights from all sensors is totalled. A sensors contribution is then calculated by mapping the sensors average weight from the combined total to between zero and one. Equation 3.2 shows the mapping equation, where ***O*** represents the result, ***A*** the input value, B^{min} is the minimum value of the input range, B^{max} is the maximum value of the input range, C^{min} is the minimum of the output range and C^{max} is the maximum of the output range.

Equation 3.2 - Equation used to convert a number from one range to another range

$$O = \left(\frac{A - B^{min}}{B^{max} - B^{min}}\right) * \left(C^{max} - C^{min}\right) + C^{min}$$

For example, if the user's hand were vertically in-between the two sensors, the weighting would be 0.5 and 0.5 approximately. Figure 0.3 shows the code used to calculate the overall weight for each sensor.

Once the overall sensor weights have been calculated, the averaged frame hand data for each of the sensors is averaged using the sensors overall weight to calculate the final hand data. The sensors are aggregated using the same approach as used in Algorithm 3.2. The first sensor is assigned and then updated for subsequent sensors, with the corresponding sensor weights used in both cases. This process results in the hand data that is closest to its sensors optimal hand position contributing more to the final aggregated hand data. This process enables the system to seamlessly transition to the sensor with the optimal view of the user's hands.

3.5 – Hand Validation System

Optical trackers can partially lose tracking or experience noisy data, resulting in invalid hand representations being presented to the user. The hand validation system (Figure 3.12) was designed to analyse and validate hand data. Hand data classified as valid would then be presented to the user with invalid data being replaced by an inferred hand, using the inferred hands system detailed in section 3.8. Figure 3.12 shows the Hand Validation System stage from the MOT System Overview Diagram shown in Figure 3.2. This step takes place immediately after the tracking data has been aggregated so invalid data can be flagged and ignored through the remainder of the system.

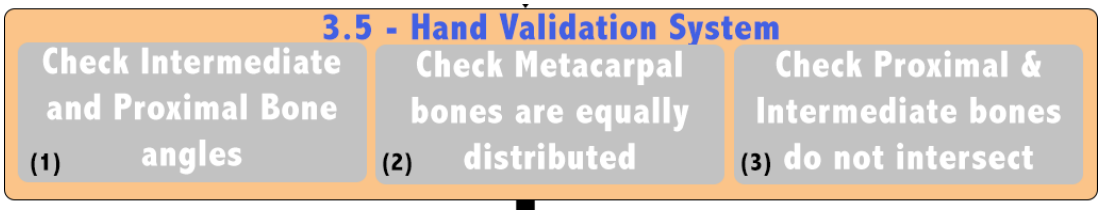


Figure 3.12 - Hand Validation System from the MOT System Overview Diagram

The hand validation system was designed to evaluate hand data, processing each finger to ensure the angles of the intermediate and proximal bones do not exceed set natural movement thresholds. In addition, the system ensures natural finger spacing across the hand and that fingers do not un-naturally intersect. The hand validation system processes each finger of the given hand, validating the hand through a series of tests.

The first test evaluates the bend angles of both the proximal and intermediate bones (Figure 3.13). The angles are calculated using the direction of the bone relative to the palmar axis i.e. the direction the palm is facing as shown in Figure 3.14, to ensure consistent measuring. Figure 3.14 shows a virtual hand representation rendered using red lines connecting bone positions, with blue, white and yellow vectors representing the radial, distal and palmar axis, respectively.

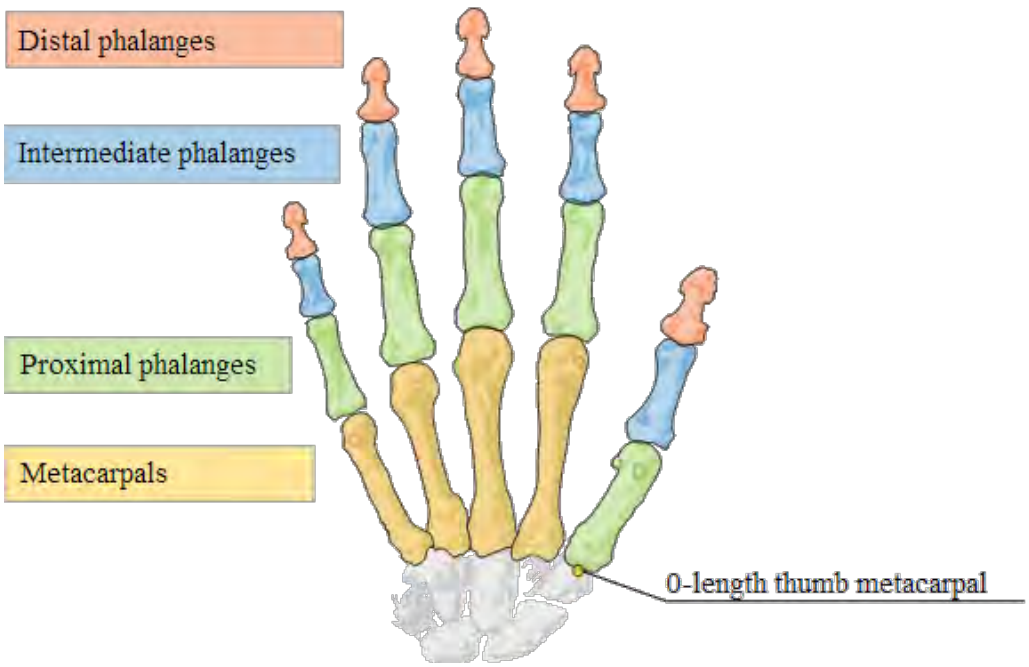


Figure 3.13 - Diagram showing the names of the different bone types in the human hand (Horowitz, 2014)

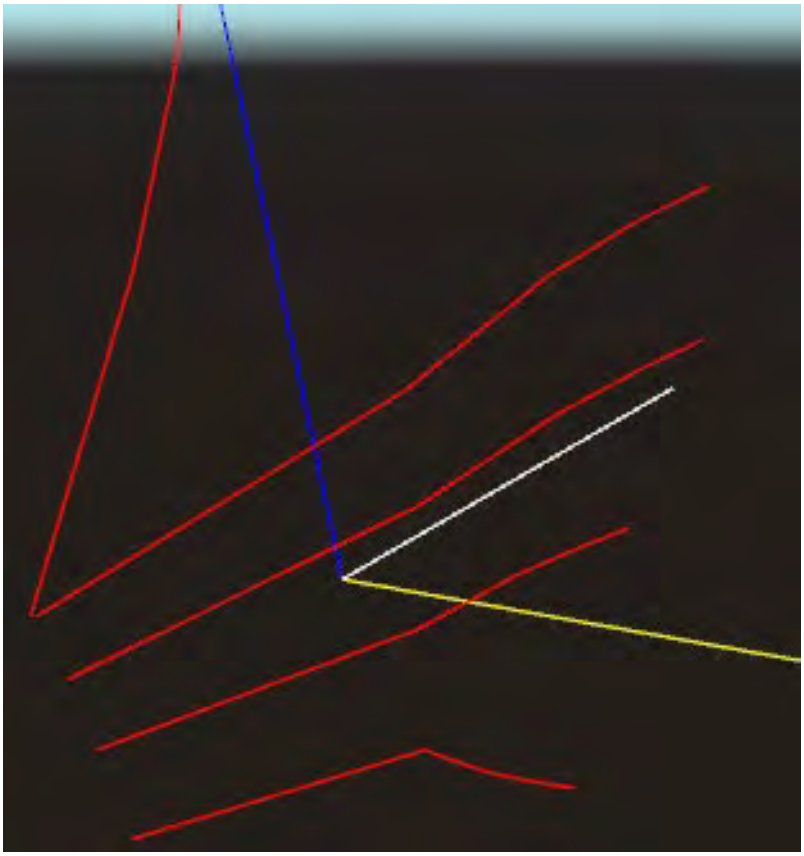


Figure 3.14 - Three main axes (yellow = palmar axis, white = distal axis, blue = radial axis) of a virtual hand rendered using red lines for each bone

The angles calculated are compared against pre-set thresholds with the hand classified as invalid if they are exceeded. The angles ranges used for the bones are 60-180 degrees for the proximal bone and 70-180 for the intermediate bone. The angle ranges used were determined during developmental testing. During development, various hand poses were performed and tracked by the Leap Motion to identify the maximum angles the hand could naturally reach. If all proximal and intermediate bones are within the required angles, the test passes and the next test is performed.

The next test analyses the spacing between the metacarpal bones ensuring that the spacing between each bone is within a ten percent tolerance of the

average across the hand. This test is used to ensure the fingers are relatively evenly spaced with tracking data such as that of partially occluded hands occasionally resulting in virtual fingers overlapping each other. Each finger is processed, calculating the spacing between the metacarpal bone of the current and next finger. The spacing between the metacarpal bone of each finger is totalled and divided to calculate the average. The spacing between each finger is then compared against the calculated average. If the spacing exceeds a ten percent tolerance, the hand is classified as invalid. Figure 0.4 shows the code for the test to ensure the fingers of the given hand are equally spaced.

The final test analyses finger positioning to determine if any of the intermediate or proximal bones are intersecting. For example, if one of the fingers is bent too far resulting in the finger pointing through another. An intersection is considered to have occurred when a bones centre position is within a set radius of another bones centre and is within the length of the other bone. To identify intersections between fingers, both the intermediate and proximal bones of each finger are evaluated against the corresponding bone in all other fingers. To detect whether two bones are intersecting, the centre positions of the two bones are evaluated to ensure a bones centre position is not within a given radius of the other. The radius of the virtual bone representation is used as the evaluation radius. Equation 3.3 shows the equation used to calculate the square distance between the two centre positions. The equation result is then compared against the bone radius to the power of two. If the square distance is less, the bones are too close.

Equation 3.3 - Equation used to calculate the square distance between two Vector3 positions

$$D = (A.x - B.x)^2 + (A.y - B.y)^2 + (A.z - B.z)^2$$

Figure 3.15 visualises two example radius tests. Figure 3.15a shows two fingers side-by-side with both centre positions outside the radius of the other. Figure 3.15b shows a finger (purple) bent too far to one side and thus its centre position is within the radius of the other and therefore intersecting.

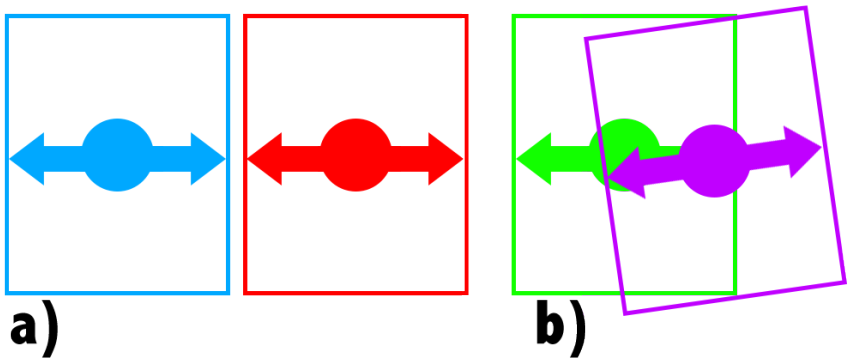


Figure 3.15 - Diagram illustrating the radius intersection test for hand validation

The second part of the intersection test determines whether the centre position of a bone is within the length of the other bone. The centre position is considered within a bone length if its position on the Z axis is within the minimum and maximum range. Figure 3.16 shows an example of a bone centre position within the length of another. In the figure, the purple bones centre position (purple ball) is within the length of the green bone (green box) and thus the finger is intersecting. The Z axis position of the purple ball is within the minimum (blue dot) and maximum (orange dot) bounds.

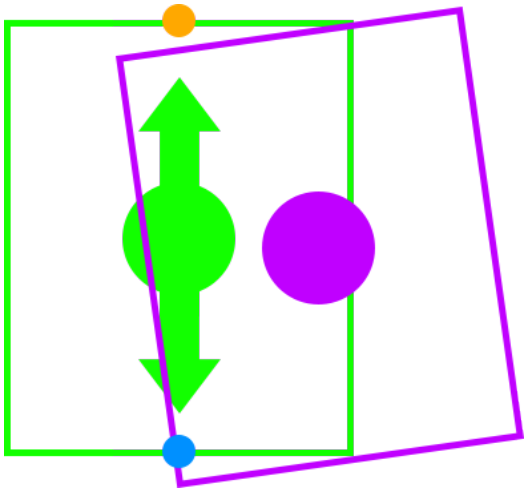


Figure 3.16 - Diagram illustrating the bone length intersection test for hand validation

To calculate the minimum and maximum bounds, the bones normalized direction vector multiplied by half the bone length, is applied to the centre position, negatively and positively, respectively. Figure 0.5 shows the code to determine whether two centre positions are within the length and radius of each other. Figure 0.6 shows the entire function for determining whether two fingers are intersecting, checking each bone against the same bone in all other digits. If both the length and radius test report intersections, the finger is classified as intersecting with another and thus the hand is invalid.

If the three tests pass, the hands are classified as valid. However, if any of the tests fail, the hands are classified as invalid. If an invalid hand is detected, the validation system will report the hand as invalid which will trigger the post aggregation processing system and inferred hand pose system to replace it.

This ensures that the user has a virtual hand representation with which they can interact and do not experience deformed virtual hands. This also ensures that the user has a consistent experience and any object interactions can

continue without interruption. In the rare event that a hand is incorrectly classified as invalid, the hand will be replaced with an inferred hand pose. As a result of the simulations high framerate and continuous validation, the frame upon which the inferred hand is based is typically only a few milliseconds old. From a user perspective this results in the hand pose presented typically matching the pose the user was performing. Thus, minimizing any impact on the user experience.

3.6 – Post Aggregation Processing System

As can be seen from Figure 3.2 the “Data Aggregation System” combines the data from multiple sensors using a weighting system to produce optimal hand tracking data, that minimizes the effect of sensor occlusion and noise. The Post-Aggregation Processing system (Figure 3.17) is designed to identify any missing hands such as those not within range of the sensors. Figure 3.17 shows the Post Aggregation Processing System stage from the MOT System Overview Diagram shown in Figure 3.2. This stage of the process is designed to analyse the current hand data and ensure that the user will have two virtual hand representations. The system checks for any missing hands and triggers the inferred pose system to provide a virtual hand, if any are detected. The system is also responsible for maintaining consistent arm length.

Figure 3.17 - Post Aggregation Processing System from the MOT System Overview Diagram

In optimal conditions, both hands would be within range and valid, thus the system would simply use the aggregated tracking data. If the tracking data for a hand is missing, the hand is created using the last valid hand data, should any be available. However, if previously valid data is not available, an inferred hand pose is used. Once the missing hand has been created, it is controlled and updated using both the Vive Arm Tracking System and Inferred Hand Pose System as detailed in sections 3.7 and 3.8, respectively. Once any missing hands have been handled, valid hands within optimal range of the sensor are used to update the offset to the Vive controller (arm length).

3.7 – Vive Arm Tracking System

Optical based sensors can lose tracking, this results in a reduction in both immersion and functionality for a user. This is typically due to the user being unable to interact with the environment and often get unnatural virtual representations of tracked elements such as their hands. The MOT system uses multiple optical sensors to reduce tracking loses. In addition, the proposed system can integrate sensor data from other sources to support hand tracking. For example, trackers attached to a user's arms can provide additional positional and rotational data for the virtual arm.

The Vive tracking system (Figure 3.18) was developed to incorporate additional positional and rotational tracking data when optical tracking data is not available, such as hands outside the visible range or the data is considered poor. Figure 3.18 shows the Vive Arm Tracking System stage from the MOT System Overview Diagram shown in Figure 3.2. This stage of the MOT system is designed to ensure consistent arm length and relative hand positioning.

Figure 3.18 - Vive Arm Tracking System from the MOT System Overview Diagram

The system was developed in conjunction with the inferred hand pose system detailed in section 3.8. The Vive hand tracking system can provide additional tracking data to the data aggregator. The system can also solely control the position and rotation of the hands in conjunction with live hand data or be used to control inferred hands without any tracking data from the Leap sensors. The system can use either trackers or controllers, attached to the upper arm via a strap as illustrated in Figure 3.19.

To ensure the relative positioning of the hand is maintained, the users forearm length is calculated using the tracked position of the Vive trackers and the hand via the Leap Motion sensors. This offset is stored and used for positioning inferred hands to ensure consistent relative positioning between the users real and virtual hands. If a virtual hand is positioned outside a set tolerance of the calculated arm length, the hand is automatically repositioned to ensure consistency. The Vive Tracking system provides the user with a real-time tracked virtual hand position. Figure 3.19 shows how a user's arm length is measured using the Leap Motion hand and Vive Tracker positions.

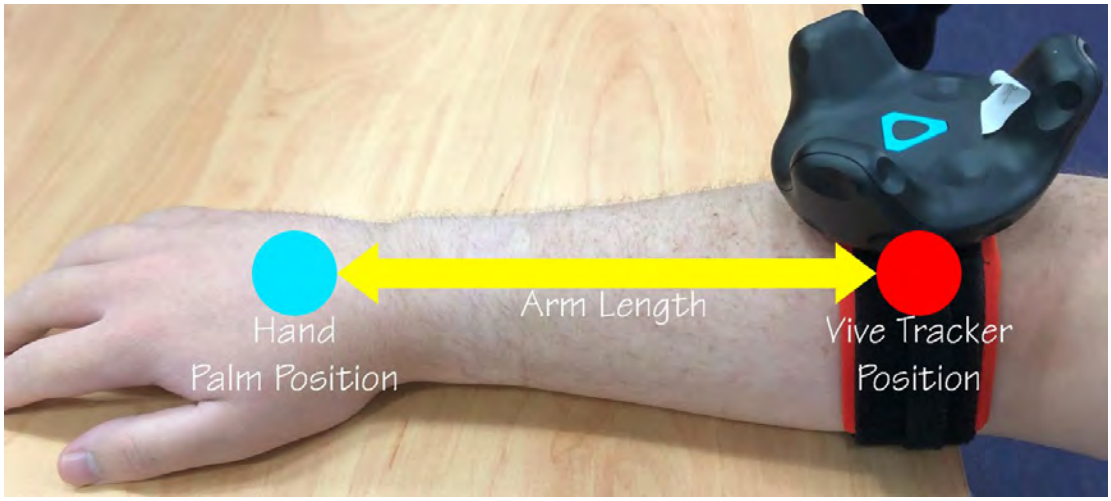


Figure 3.19 – HTC Vive Tracker mounted onto a user's arm using the Manus Vive Tracker arm mounts

3.8 – Inferred Hand Pose System

The inferred hand pose system (Figure 3.20) was designed to provide the user with an inferred virtual hand pose if the optical based tracking data becomes unusable or the sensors lose tracking of the user's hands. In the case of an invalid data, the hand is replaced with an inferred pose. The hand pose used is based on the last valid hand and the user's current actions, with an open hand pose used by default. The system uses the Vive Arm Tracking System detailed in section 3.7 to determine the position and orientation of an inferred hand.

Figure 3.20 shows the Inferred Hand Pose System stage from the MOT System Overview Diagram shown in Figure 3.2. This stage of the MOT system is designed to ensure that two virtual hand representations are presented to the user. The system replaces any invalid hands with an inferred pose based on previous frames. The system can also provide a specific hand pose if an interaction requires it. This stage takes place directly before the mesh

conformation stage detailed in section 3.9 to ensure the system will have a virtual hand to conform if necessary.

Figure 3.20 - Inferred Hand Pose System from the MOT System Overview Diagram

One of the main issues with optical based hand tracking is dealing with loss of tracking whilst ensuring the user has a smooth experience. By default, when the Leap Motion loses tracking the virtual hand disappears and any objects held are released. The Inferred Hand Pose system was designed to seamlessly interpolate between the last valid hand data and an inferred hand pose to ensure a smooth experience for the user.

Hand data objects contain a series of bones each of which has start, end and centre positions along with a direction from its start to end position. Figure 3.21 shows a diagram of a virtual hand representation with the start (yellow), end (red) and centre (green) positions of all the index finger bones displayed with the bone directions indicated by a blue arrow. In a hand data object, the start position of a bone is the same as the end of the previous bone, if available. Figure 3.13 shows the names of the different bones within the human hand that are used in the hand tracking data.

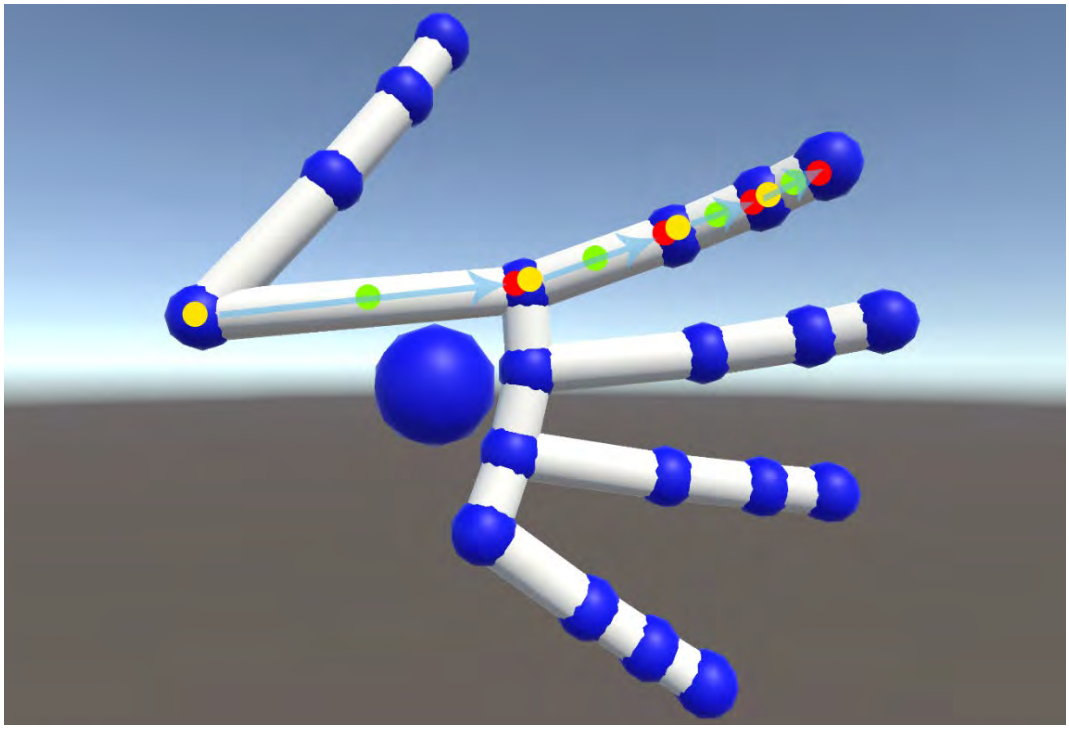


Figure 3.21 - Diagram showing the direction (blue) and start, end and centre positions for hand bones; start (yellow), centre (green), end (red)

The Inferred hand pose systems interpolator processes each finger, interpolating the start and end positions of each bone using the equation shown in Equation 3.4.

Equation 3.4 - Equation used for Linear Interpolation of Bone positions

$$V = V^1 + (V^2 - V^1) * T$$

Equation 3.4 shows the linear interpolation equation used, where V^1 represents the current vector, V^2 is the target vector and T is time. Once the bones start and end positions have been updated, the bones direction and centre vectors are recalculated using the updated points. Finally, when all the fingers have been processed, the vectors for the hand and arm, such as palm position and the arm direction are interpolated.

In addition to interpolating between two hands, the Inferred Hand Pose System can update a hands position and rotation using data from three different sources: live hand data, stored hand data and arm position data from the Vive Arm Tracking System. This aids in reducing the amount of processing required during each frame update. Once the interpolation from either the live or last known hand to an inferred pose has been completed, the system can simply update the position of the inferred hand rather than having to continuously interpolate. Furthermore, interpolation over long periods can give a feeling of 'lag' or delay due to the position and pose changes being based on the interpolation speed rather live data, such as motion trackers. Therefore, the interpolation was designed to be fast to avoid the user feeling any sense of delay. The interpolation speed used the simulations delta time multiplied by a factor of twenty. This was due to the delta time being very small as it represents the time between frames and thus the interpolation would be very slow at that speed. The increase by a factor of twenty was chosen based on developmental testing using different factors including five, ten and thirty. A factor of twenty was selected as it ensures the transition animation is as short as possible, so as not introduce a sense of lag. Whilst still providing the user with a long enough animation so the transition does not appear to jump or 'jerk', thus ensuring a smooth user experience.

The inferred virtual hand provided allows users to pick up virtual objects as normal and drop them using a flick/throw gesture. The Inferred Hand Pose system uses the engines physics systems overlap sphere test to determine whether the virtual hand is within proximity of an interactable object. The overlap sphere test creates an invisible sphere with a given radius and returns

everything it intersects with. Figure 3.22 shows a visualization of the overlap sphere test with the red cube (left) considered within proximity while the blue cube (right) would be ignored.

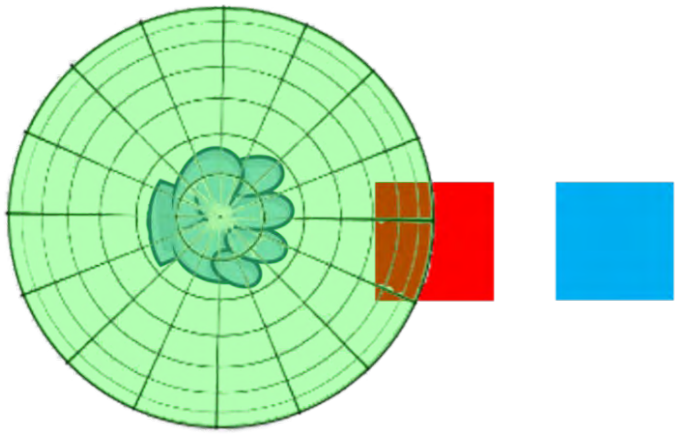


Figure 3.22 - Diagram of the "OverlapSphere" test built into the Unity physics system (not to scale)

If a hand is within range and not holding a virtual object, the system begins interpolating from the current hand pose to an open hand pose in preparation for the mesh conformation system detailed in section 3.9. The interpolation process also sets a flag to interpolate to the live tracking data when tracking resumes to ensure a smooth experience. Algorithm 3.3 shows a pseudocode implementation of the Inferred Hand Pose Systems interpolation. The interpolation positions are calculated using the Linear interpolation equation shown in Equation 3.4.

Algorithm 3.3 - Pseudocode of the Inferred Hand Interpolation

```
Inferred Hand Interpolation
Input:  I : Current Hand Pose, P : Target Hand Pose
Output:  H
H = Create empty Hand object;
foreach Finger in Hand do
    foreach Bone in Finger do
        Set the end position of the current bone in H to the result of the
          interpolation between the end positions of the
          current bone in I and P;
        Set the start position of the current bone in H to the result of the
          interpolation between the start positions of the
          current bone in I and P;
        Calculate the new bone direction and centre in H using the
          calculated start and end positions;
    end
    Set the direction of the current finger in H to the result of the
      interpolation between the direction of the current finger in I and P;
end
Set the palm position, wrist position, arm centre, arm direction, arm end
  and arm start of H to the results of the interpolation between the
  corresponding properties of I and P;
```

If the hand is holding a virtual object but has not been conformed to the virtual object, the mesh conformation system is used as detailed in section 3.9 with the virtual hand state changed to closed. If the virtual hand has already been conformed around the object, the position and orientation of the virtual hand are updated. To update the virtual hand, either the live tracking data or the Vive trackers position will be used, depending upon tracking data availability.

Once the Inferred Hand Pose System has finished processing, the MOT systems hand data is updated to the resulting hand pose. In the case of virtual object interactions, the MOT system hand tracking data is updated to store a pose different to that which is rendered. The hand tracking data is updated to a fist pose so the virtual hand continues holding the virtual object.

3.9 – Mesh Conformation System

The mesh conformation system (Figure 3.23) was designed to manipulate the users virtual hand representation into a natural grasp pose as shown in Figure 3.24. Figure 3.23 shows the Mesh Conformation System stage from the MOT System Overview Diagram shown in Figure 3.2. This final stage before rendering is designed to provide the user with a realistic hand grasp if they are interacting with a virtual object. This is the final stage of the MOT system cycle to ensure that whether live tracking data or an inferred hand is being used, a realistic grasp will be provided to the user if required.

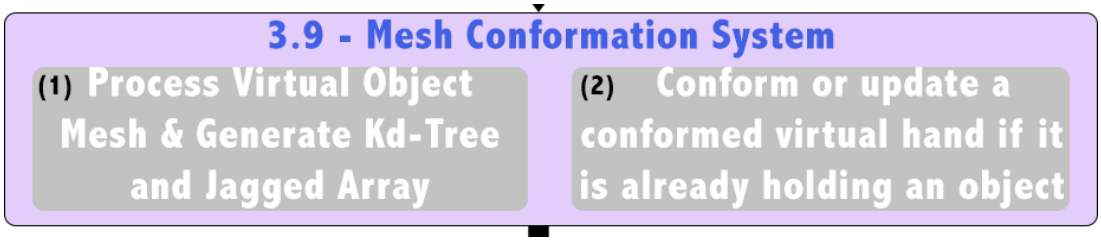


Figure 3.23 - Mesh Conformation System from the MOT System Overview Diagram

Previous approaches as detailed in Chapter 2 have relied upon physics-based interaction. Unfortunately, these approaches can be computationally expensive. Alternative approaches have explored machine learning techniques. However, such approaches require extensive training on both hand data and each virtual object within the simulation. Furthermore, machine learning approaches typically limit user grasps to a pre-set grasp. This can result in the user attempting to grab the object, with the hand jumping to a different position and hand pose to grasp. The sudden change of the virtual

hand pose and position can reduce a user's sense of immersion and embodiment with their virtual hand control being overridden.

This system was designed to dynamically manipulate the positioning of the virtual hands fingers to provide a realistic grasp without any prior knowledge of virtual objects. One of main benefits of the system designed is that it can calculate realistic grasps in real-time and continuously update/adjust them according to user movements, every frame. Furthermore, the systems approach and real-time performance enables it to instantly conform inferred hands ensuring the users grasp is maintained if tracking data is lost.

As the system can calculate the hand grasps using the mesh of the virtual object, it enables users to grasp objects created or modified within the simulation. For example, the system could be used in a sculpting simulation to enable users to realistically grasp any clay sculpture. From a user experience perspective, the system design does not freeze the user's virtual hand, so the sense of embodiment and control is not broken. In contrast to previously discussed research approaches which result in deformed hands if a user resumes control of the hand. The system enables users to continue interacting with the virtual object, such as pressing the button on a torch while holding it without deforming. Furthermore, in contrast to previously discussed approaches, the system does not snap the virtual hand to the optimal position and angle relative to the virtual object. The system performs minor corrections to the objects position and orientation to ensure optimal conformation, minimizing any effects on user experience.

Figure 3.24 - Virtual Hand Representation showing fingers conformed around a virtual object

To create the realistic grasp poses, the virtual fingers are repositioned to wrap around the surface of the virtual object as shown in Figure 3.24. The surface positions are calculated using the vertices of the object mesh. As multiple points are needed, a Kd-Tree (Friedman, et al., 1976; Bentley, 1980) is used to organize the vertices of the virtual object, minimizing computational expense and processing time.

The Kd-Tree used nearest-neighbour searching to find the closest vertex to a given position. This enabled the system to quickly find the closest position on the objects surface for each bone in a virtual hand representation. A brute-force approach checking every vertex for each bone would have taken significant computational expense and time. The main reason being a result

of the large number of bones and potentially large number of vertices on a virtual object.

The Kd-Tree was used in combination with a jagged array to find the closest vertex position to a given bone position. The jagged array uses one top level entry per vertex and each of the separate sub-arrays is a list of all the triangles that use that vertex. A jagged array or array-of-arrays enabled the triangles for each vertex to be stored and quickly accessed for analysis. Once the closest vertex position has been found by the Kd-Tree, all the triangles in which the vertex is used are processed to find the position closest to the target.

As the virtual hand representation may need to be repositioned or re-conformed around the virtual object each frame. The mesh of the virtual object is analysed during the initial pickup event, with the Kd-Tree and Jagged array generated. The initial creation of the Kd-Tree and Jagged array requires the entire mesh to be processed. In the case of the Jagged array, it requires the two iterations with the array structures created in the first and assigned in the second. However, once the Kd-Tree and Jagged array have been created, the necessary vertex and triangle data can quickly be accessed, ensuring minimal computational expense in subsequent frames. The Kd-Tree and Jagged array persist until the object has been released. As the data stored is not specific to one instance of an object. The Kd-Tree and Jagged array can be used to conform the other hand to the same type of object without having to process that object.

Once the data structures have been created, the virtual object is repositioned and rotated to align with the virtual hand representation. This provides the optimal conditions for the system to manipulate the virtual fingers around the object.

To orientate the object, the angle between the virtual object's upwards direction and the virtual hands radial axis (Figure 3.14) is calculated. Once the angle has been calculated, the object is orientated to have a rotational offset of either 90 or -90 degrees to the hand's radial axis. The closest rotational offset is selected to minimize the visual change the user experiences. Figure 3.25 shows the two possible object orientations once it has been adjusted.

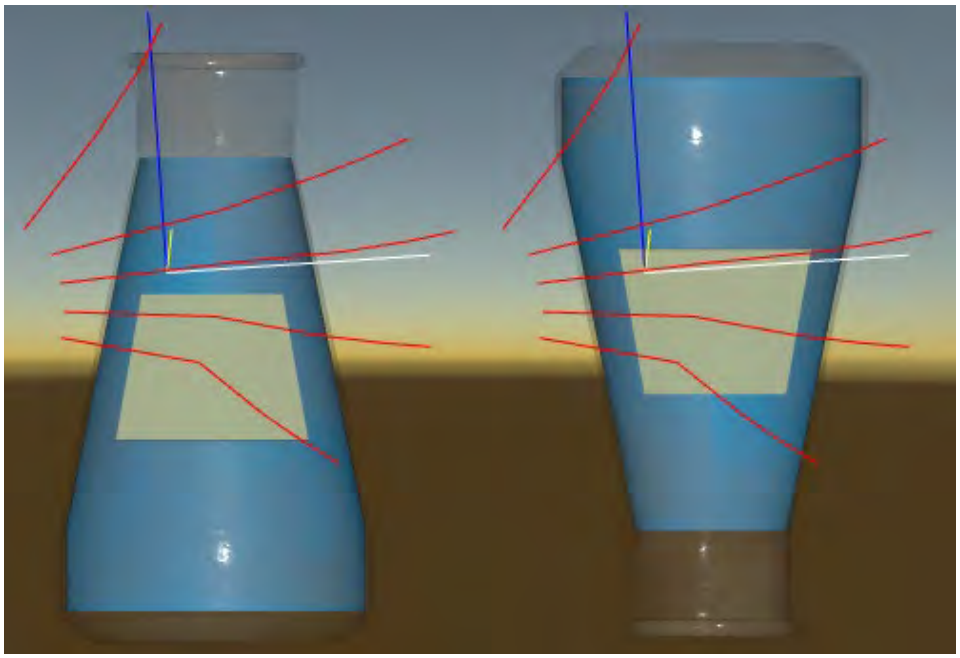


Figure 3.25 - Two orientation options for virtual object conformation

Once the object has been orientated to align with the virtual hand representation, it is repositioned so the palm rests of the surface of the virtual object.

To reposition virtual object in front of the palm, the Kd-tree is used to find the nearest point on the mesh surface to the palm position. The radius of the virtual object is then calculated by projecting a vector between the surface and centre positions onto a plane, using the virtual objects up vector as the plane normal. Figure 3.26 shows a Vector (red arrow) projected onto a plane (green arrow) using the normal direction of the plane (blue arrow).

Projecting the calculated vector onto a plane “flattens” it out so the direction of the vector is from the surface position to a position at the same height in the centre of the object. Figure 3.27 shows the vector projection in the context of a virtual object with the objects up vector (blue arrow), surface to origin vector (red arrow) and projected vector (green arrow) displayed. In Figure 3.27 the “length” of the vector illustrated by the green arrow would be the objects radius at that vertical position.

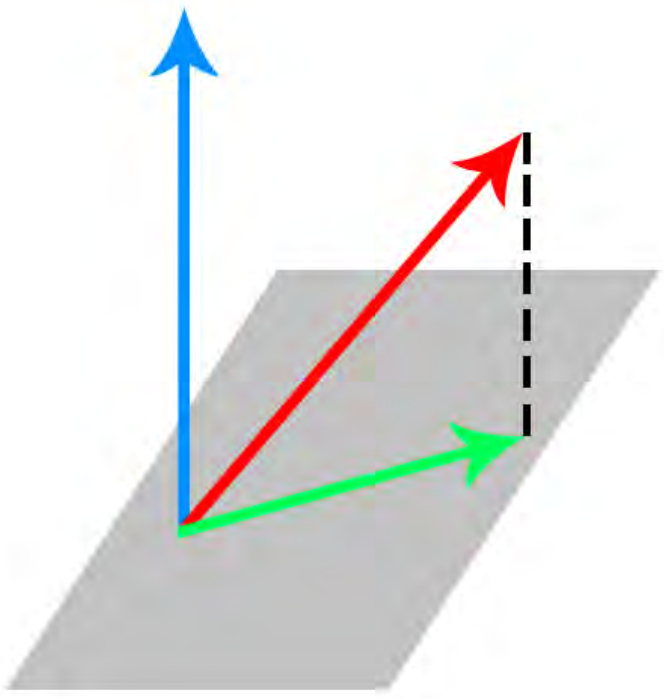


Figure 3.26 - Diagram showing a Vector projected onto a plane (Blue = plane normal, red = source vector, green = projected vector)

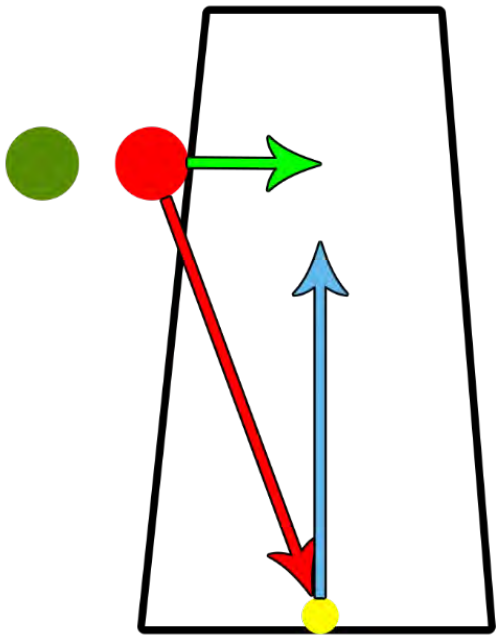


Figure 3.27 - Diagram showing the vectors and projection result used for radius calculation in the mesh conformation system

Once the virtual objects radius has been calculated, a vertical offset vector is calculated. The offset is calculated to ensure that the virtual hand is vertically in the centre of the object once it has been repositioned. The vertical offset is calculated by first determining half the height of the object using the extents property from its collider via the physics engine. The calculated height is then multiplied by the negative of the object's upwards vector. The result is the vector necessary to move the object to be vertically centred with the virtual hand. Figure 3.28 illustrates how the vertical offset is calculated. In the diagram the yellow dot represents the objects origin and the blue arrow represents the objects up vector. The green box represents the box collider, with the orange arrow representing the half height reported by the 'extents' property of the physics collider.

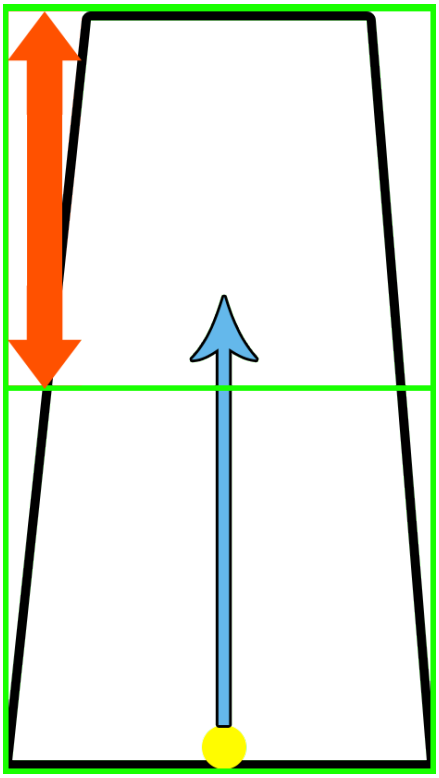


Figure 3.28 - Diagram showing the calculation of objects vertical offset; yellow) object origin, blue arrow) up vector, green box) physics collider and orange arrow) physics extents height

Once the vertical offset has been calculated, it is used with the object radius, radius of the virtual fingers, hand position and the palmar axis to calculate the new position of the virtual object. Equation 3.5 shows the equation for calculating the new position where, $\boldsymbol{H^p}$ represents the middle of the hand, $\boldsymbol{H^n}$ is the palm normal, ***R*** is the virtual object radius ***O*** is the previously calculated vertical offset. The middle of the hand $\boldsymbol{H^p}$ is approximated using the end of the middle fingers metacarpal bone.

Equation 3.5 - Equation used for calculate the optimal position of a virtual object for conforming

$$P = H^p + (H^n * R) + O$$

The resulting position places the object so the virtual hands palm rests on the surface in line with the vertical centre of the object. With the virtual object repositioned in-front of the palm and orientated to align with the virtual hand, the process of conforming the virtual hand around the surface of the virtual objects mesh begins. Algorithm 3.4 shows pseudocode of the mesh conformation system processing a virtual hand and repositioning the bones to wrap around the virtual object.

Algorithm 3.4 - Pseudocode implementation of the Mesh Conformation System

```
Mesh Conformation System
Input:  I : Current Hand
Output:  H
Kd = Create KD-Tree from object mesh vertices;
Vt = Create Jagged Array from object mesh triangles;
H = Create new Hand object cloning I;

Calculate closest point for hand wrist position and update wrist
 position in H;
Calculate closest point for hand palm position and update palm
 position in H;
foreach Finger f in Hand do
    Calculate the closest point for the end of the metacarpal bone and
     update H;
    foreach Bone b in Finger do
        prevBone = previous bone;
        startPoint = End of Previous Bone + b.direction * b.length;
        closestPoint = Calculate closest point to startPoint on mesh
         surface;
        newDistance = Calculate distance between closestPoint and
         prevBone;
        direction = Calculate vector from closestPoint to the end of
         prevBone;
        lengthDiff = b.length - newDistance;
        closestPoint += (direction * lengthDiff);

        Set end of current bone to closestPoint;
        Set the start of the current bone to the end of prevBone;
        Recalculate the centre, direction and length of current bone
         using new start and end positions;

    end
end
Recalculate the palm normal;
```

To conform the hand to the virtual mesh, the hand, wrist and bones are repositioned to the nearest vertex points using the Kd-Tree. The system iterates over each bone of each finger, calculating the bones new starting position to ensure consistent hand size. To calculate the new start position, the system uses the end position of the previous bone with the direction and length of the current bone. This accounts for the change in position of the previous bone as a result of being conformed to ensure consistent hand size. The calculated start position is then used with the Kd-Tree to find the closest position on the surface of the objects mesh.

Once the surface position has been calculated, the current bone length is compared against the distance between the end of the previous bone and the calculated surface position. The difference in bone length is calculated and the surface position is updated using the Δ length to ensure consistent bone length. The start and end positions of the bone are then updated to use the end of the previous bone and calculated position, respectively. Once the bone has been repositioned the bone direction, length and centre position are recalculated using the new start and end positions. Once all the fingers have been processed the palm normal is re-calculated. Figure 0.7 shows the code used to calculate the objects surface position and offset it using the bones direction and length to maintain hand size.

The Mesh Conformation System results in the fingers of the virtual hand representation appearing to be wrapped around the virtual object in a realistic and natural pose as can been seen in Figure 3.24.

3.10 – Summary

This Chapter presented the phase one design of the Multiple Optical Tracking (MOT) system and the sub-systems of which it is comprised. This Chapter has shown the MOT systems novel approach to the aggregation of optical tracking data from multiple sensors. Furthermore, this Chapter has presented the design of novel systems designed to improve the user experience using techniques including replacing missing hands with inferred poses and generating realistic object grasps. Finally, this chapter has shown a novel approach to provide realistic hand grasps for virtual object interactions without training or prior knowledge of the virtual environment. The designs presented have been discussed in detail with justification for the selected methods used. In addition, details of developmental testing have been presented as part of the discussion behind decisions such as the selected frame weighting. The next Chapter presents the phase one experimental procedure and analysis of the results.

Chapter 4 – Experimentation Phase One

In the first phase of experimentation, the MOT hand-based interaction system was applied to a virtual reality Chemistry simulation and compared to other VR interaction types; namely, Vive controller and Manus VR gloves. In addition, the MOT system was compared to a single head mounted Leap Motion Sensor approach, to determine if the MOT system increases the number of valid hands presented to a user. A larger number of valid hands should result in a more immersive user experience because there are fewer accuracy and stability issues observable to the user. The designs of the Vive condition, Manus condition and the simulation can be found in Appendix 4,5 and 6, respectively.

The three interaction conditions (MOT system, Manus Gloves and Vive Controller) were evaluated using semi-structured interviews, questionnaire style questions and performance logs. The results from the experimentation produced both qualitative and quantitative data that were analysed and evaluated using appropriate techniques.

Cohen's Kappa was used to evaluate selected thematic themes and coding performed by two raters to ensure the data was analysed in a similar manner. Cronbach's alpha was used to determine the reliability of the Likert scale-based interview questions (Statistics, 2018). In addition, a bivariate Pearson Correlation was performed to identify any statistically significant linear relationships between the questions (Statistics, 2018). A statistically significant relationship is one that is large enough to be unlikely to have occurred in the sample if there is no relationship in the population.

The performance logs are analysed using one-way repeated measures ANOVA (Analysis of Variance) tests with Post hoc comparisons using the Bonferroni test to compare the effect of each condition on total completion time and task completion time. In addition, the performance logs are analysed using two-way mixed ANOVAs to analyse the effect of condition order and user experience on condition performance.

The initial hypothesis was that the usability of an interaction technique would affect the user's engagement and therefore the sense of immersion. As an extension it was hypothesized the MOT system would provide the users with the most natural, realistic and engaging interaction as a result of the direct hand-based manipulation. Additionally, the Vive controllers would provide the easiest interaction due to their simplistic control schemes, use of HMD tracking technology and similarity with traditional inputs such as games controllers.

4.1 – Experimental Setup & Procedure

The Chemistry simulation is a seated VR experience that requires users to interact with objects and equipment at a table. During the experiment, participants were seated in a standard swivelling office chair facing the interviewer. The simulation takes participants through the process of completing a chemistry experiment using virtual equipment such as test tubes, beakers, centrifuge and pipette dropper.

A Chemistry-based simulation theme was selected for development as previously discussed research has shown natural VR interaction to be effective in education/serious games. In addition, science-based simulations offer a great test bed for VR simulations, presenting users with the opportunity

explore new environments and interact with complex and expensive equipment. Furthermore, the various virtual equipment types provide different object interaction types, to evaluate a full range of movements and hand gestures. This enabled a scenario to be developed that used different interaction types and complexities to test the limitations of the evaluated interaction approaches.

From an interaction perspective, the simulation presents the user with three different task types: translation/rotation, two handed interaction and a virtual user interface (UI). The translation/rotation tasks require the user to manipulate virtual objects such as pouring beakers into a large mixing beaker. The simulation also features two handed tasks such as filling up a test tube using the pipette dropper. Finally, the simulation features virtual UI; specifically, a simple touch screen interface for configuring the centrifuge. The user interacts with the virtual UI by tapping the buttons using their index finger. Figure 4.1 illustrates the VR Chemistry simulation experience desk and some virtual objects that users can manipulate. Figure 4.2 shows the virtual UI used to configure the centrifuge. Figure 4.3 shows the pipette dropper used with the simulation, with a large button on the end that can be tapped to change modes in the hand-based interaction conditions. Table 4.1 shows the details of the four object task types in the simulation, including average time taken for the motion controller (C), Haptic Glove (G) and MOT System (M).

Table 4.1 - Table showing the Simulations setup with regards to user tasks

	Beaker/Test Tube Tasks	**Pipette Dropper Tasks**	**Centrifuge Task**	**Bunsen Burner Task**
Task Type	Translation & Rotation	Translation & Rotation with Two-Handed Interaction	Virtual UI	Translation & Rotation
Number of Actions	4xBeakers 2xTest Tubes	4xSuction 4xRelease	Once	Once
Difficulty (MOT)	Grab and Release Gesture	Grab and Release, Tapping, Precise positioning	Tapping	Grab and Release Gesture
Difficulty (Glove)	Grab and Release Gesture	Grab and Release, Tapping, Precise positioning	Tapping	Grab and Release Gesture
Difficulty (Controller)	Trigger Press	Trigger Press, Trackpad Click	Tapping	Trigger Press
Average Time Taken (seconds)	C – 6.85 G – 6.85 M – 9.95	Suction/Release C – 12.39/12.63 G – 9.42/9.96 M – 13.44/16.55	C – 14.36 G – 16.43 M – 17.85	C – 2.21 G – 2.99 M – 3.51

To minimize the risk of users experiencing simulator sickness and its potential impact on the results, the simulation is designed as a seated experience with the user moving their hand (rotation and translation) to view the virtual environment.

Figure 4.1 - Render of the Chemistry Fun simulation

Figure 4.2 – Screenshot of the simulation showing the Centrifuge Configuration UI

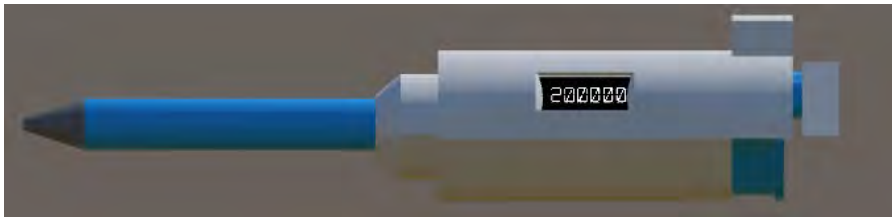


Figure 4.3 – Dropper used in the simulation

During the simulation users are presented with instructions via a virtual clipboard positioned on the back wall, with clarification provided by the instructor if necessary. An audible success/tick sound effect is used to signal a task had been completed. In addition, the text of the current task is changed to green.

The participants completed the scenario three times, once for each type of interaction; Vive controller, Manus VR Gloves and MOT system. The order of the interaction techniques was randomly determined for each participant to minimize the learning effect that each subsequent playthrough would have on the results.

Twenty adults (one female) participated in the study. The average participant age was 28. All participants had perfect or corrected to, vision. Participants were pre-screened into one of four categories depending on their experience, namely, Gamer and VR Gamer (3), Gamer (13), VR Gamer (1) or None (3). Due to missing data in the performance logs, one participant had to be removed from analysis.

Before participants started the simulation, they were shown a screenshot of the virtual environment, with the experimental procedure and control schemes explained. The controls were detailed with the help of handouts, showing the controller buttons along with their corresponding action and in the case of the hand-based techniques, the gesture vocabulary and the corresponding actions. Once the experimental procedure had been explained, participants were asked to make themselves comfortable and begin the first simulation. After completing the first simulation, participants were asked a series of

questions in a semi-structured interview. Once the first set of questions had been completed, participants experienced the remaining two versions un-interrupted before being asked another series of questions to ascertain their experiences of the three. Table 4.2 shows the interview questions asked during the semi-structure interviews. The questions were designed to evaluate the usability, realism of the interaction conditions and the effect on user experience. The questionnaire contains questions based on the System Usability Scale (SUS) of Brooke (1996) and the work of Witmer & Singer (1998).

During the interviews, the questions were clarified to participants if they were unsure of the context. In the questionnaire, engagement is defined as the feeling of wanting to continue the use of the interaction condition. The immersion question focused on the simulation, object behaviour and the replication of the real-world senses per the definition discussed in Chapter 2.4 (Meehan, et al., 2002; Slater, 2003; Kim, et al., 2017; Park, et al., 2019). The questionnaire used both immersion and engagement questions to elicit greater detail on the user experience for each condition, to identify key strengths of each. It was anticipated, the simplistic design and ease of the controller condition would provide a strong sense of engagement, despite the poor sense of immersion. In contrast, it was anticipated that the hand-based conditions would provide a strong sense of immersion whilst potentially being un-engaging as a result of any potential frustration due to any potential tracking and/or interaction issues. The use of both questions enabled the two hand-based conditions to be evaluated to determine user preference with regards to tracking stability and hand tracking DoF.

The question regarding usability was designed to assess the approach with regards to user goals, tasks and requirements as per Quesenbery (2001). The question was designed to determine whether a user felt restricted as to how they could interact, and they were able to do everything they wanted. The question regarding ease was designed to assess how difficult the interaction approaches were to use, how easy was it to complete a desired action. The question regarding resemblance to real life was designed to compare the interaction approaches to that of the real-world. This enabled the effectiveness of the interaction approaches at transferring real-world skills and experience to be evaluated. The questionnaire did not use questions regarding presence as the "sense of being there" for two reasons. Firstly, it was anticipated that due to unfamiliarity with the simulation, clarification on instructions would have to be given. Secondly, as part of the detailed analysis of user experience, clarification and discussion of comments made by participants during the simulation would take place. Both cases would significantly reduce the user's sense of presence as talking with the interviewer would remind them of the existence of the real world.

Table 4.2 - Interview Questions asked during the semi-structured interviews

Semi-structured Interview Questions	**Likert-based**
On the following scale, how did you find the usability of the controls?	Yes
On the following scale, how easy did you find the controls to use?	Yes
On the following scale, how did you find the engagement of the controls?	Yes
On the following scale, how would you rate the simulations level of immersion?	Yes
On the following scale, how closely did you find the interaction resembles real life?	Yes
How did you find the scenarios? a) Did you find them easy or difficulty to go through? Please explain.	No

I. User interface/interacting with objects. Please explain b) Did they feel realistic? I. 3D environment. Please explain II. User interaction. Please explain Did you find any of the input approaches you used in the scenarios to? a) Be more natural. Please explain. b) Be more realistic or immersive. Please explain c) Improve your engagement with the scenario. Please explain d) Work better for you in any other way. Please explain	
How did you find each of the following interactions in the scenario and which did you prefer? Please explain a) Did you find it easy or difficult? Please explain. b) Did these feel natural? Please explain I. Picking up and pouring a beaker II. Using the pipette dropper III. Moving the test tubes IV. Using the Bunsen burner V. Configuring the centrifuge	No
Would you choose to use any of the control schemes again in the same scenario? Please explain	No
In what type of scenario did you think the control schemes would be best suited (e.g. Medical training simulation, first person shooter, science based educational game)? Please explain.	No
Did interacting with the simulation via hand gestures and no tactile interaction affect your feeling on the following: a) Naturalism. Please explain b) Immersion. Please explain	No
What changes would you suggest we make to future input approaches and training scenarios? Please explain	No

The entire experimental procedure was recorded (audio & video) for each participant to capture comments and reactions along with answers for transcription. A data logger ran in the background of the simulation to capture task completion times for each of the conditions. Participants were informed that they could be as vocal as they wish on comments, criticisms or feelings experienced during the simulations.

4.2 – Semi-structured Interview Analysis

In this section the results of the Qualitative analysis are detailed, analysing the results of the Likert-based questionnaires along with thematic analysis of both responses given to questions and comments made during the experimentation process.

4.2.1 – Likert Questionnaire Analysis

The semi-structured interviews were divided into two sections, one discussion took place post first experience and the second once all conditions had been completed. The interviews comprised of open questions and questionnaire styled questions using a Likert scale 1-5 (1. Strongly disagree, 5. Strongly agree). Cronbach's alpha was used to determine the reliability of the questions (Statistics, 2018), with the questionnaire reaching an acceptable reliability, $\alpha = 0.728$.

Figure 4.4 shows the mean Likert scores for questions 3 to 7 of the questionnaire, with the first two being age and experience. The results show that the MOT system was the most engaging (Q5), scoring 6.38% higher than the Vive controllers (5.0 compared to 4.7) with the Manus gloves scoring the least (4.438). The results also show the MOT system had the highest score for usability (Q3), scoring 16.87% higher than the Manus and 11.33% higher than the Vive controllers. In addition, the MOT system provided the greatest sense of immersion, scoring 8.76% higher than the Manus and 25.53% higher than the Vive conditions. The Manus was the most realistic (Q7) scoring 4.8% higher than the MOT system (3.75 compared to 3.57) with the Vive scoring the lowest (3.40). Even though some participants described the MOT system as glitchy and documented tracking issues, many described the MOT system as

“natural”. Furthermore, the results show it was found to be more realistic than the Vive controllers with a 5% decrease in comparison to that of the Manus.

These results support the research previously discussed in Chapter 2, showing the direct, natural hand-based interaction approach to provide the most engaging and immersive experience. The results also support the notion of direct skill transference with hand interaction, as the MOT system showed the greatest usability. The small percentage difference between the Manus and MOT conditions with regards to realism shows the effect of the reported tracking issues. The Manus conditions use of motion controller tracking and bend sensors provides a more stable experience, thereby having greater resemblance to the stability of real-world hands.

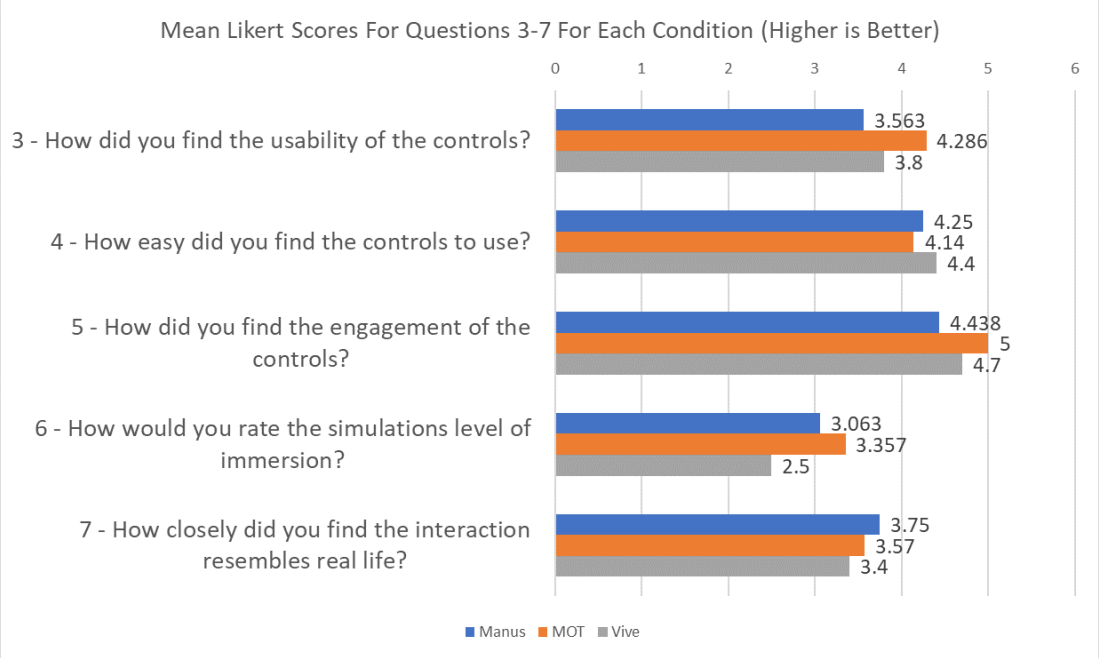


Figure 4.4 - Mean Likert scores from interview questions

A bivariate Pearson Correlation demonstrated strong correlations between questions, with questions three and five ($r(19)=.564$, $p < .01$), four and seven ($r(19)=.633$, $p<.01$) and five and seven ($r(19)=.574$, $p < .01$) having positive correlations. A strong correlation between (Q4) and (Q7) was found, with the Vive scoring 3.4% higher than the Manus. In addition, the MOT system was found to be the most difficult to use. However, (Q7) shows the Manus to be considered the most realistic, scoring 4.8% higher than the MOT system.

While there is a correlation between difficulty (Q4) and resemblance to real life (Q7), the easiest technique is not necessarily the most realistic. Research has shown the limitations of controller-based interaction techniques (Derpanis, 2004; Murthy & Jadon, 2009; Bowman, et al., 2012; Chaudhary, et al., 2013; Clark & Moodley, 2016). Controller based techniques are generally considered to be the easiest due to their use of buttons and triggers. However, they typically suffer with poor ergonomics, limited interaction and have been found to be the least realistic. The Likert score results support the hypothesis that hand-based approaches provide a much more natural feeling. Thus, striking a greater resemblance to real world interaction, from which the experience can be directly translated to the simulation.

The correlation statistics also showed a strong correlation between Q5 and Q7, with the Manus providing the greatest sense of realism, scoring 4.8% higher than the MOT system (3.75 compared to 3.57) and the Vive being the lowest (3.40). In contrast, Q5 showed the MOT system to provide the greatest sense of engagement, scoring 6.38% higher than the Vive. These results indicate that the Manus condition provided the greatest sense of realism likely due to its support for 360-degree tracking. The Manus gloves use of bend

sensors enables a user to continue controlling their virtual hand representations outside of the MOT systems tracking range, providing a more realistic control range. However, the results also show the MOT system provides a more freeing experience thus enabling easier simulation engagement.

4.2.2 - Thematic Analysis

Interview results and observations were transcribed by the main investigator and analysed in line with the six phases of Thematic Analysis as proposed by Braun and Clarke (2006). Four main themes and seventeen subthemes were identified. Figure 4.5 presents the themes that emerged from the thematic analysis of the interviews. It also shows an overview of the coding structure and relative code distribution within each theme. The values in Figure 4.5 show the percentage subtheme coverage within a theme. These values were generated by NVivo 11, software used as part of the presented thematic analysis. The percentages show naturalism and hands were common themes in participant feedback.

To validate the themes, an inter-rater reliability test was conducted using Cohen's Kappa with the result being a score of 0.81, indicating near-perfect agreement (Stephanie, 2014). As per the guideline suggested by Lombard et al (2002), a sample size of three (15%) was randomly selected for the inter-rater reliability testing.

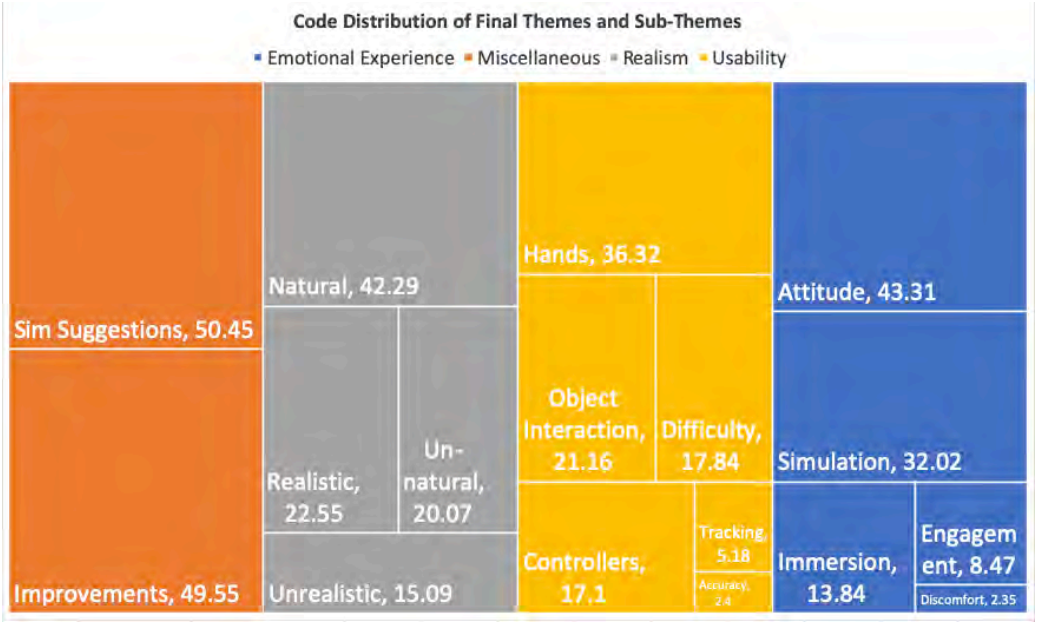


Figure 4.5 - Code distribution of the final themes and subthemes

In Figure 4.5, emotional experience focuses on the participants psychological feelings towards the experience. Miscellaneous explores participants comments on improvements and future experiences. Realism summarizes participants thoughts on the degree to which the simulation and interaction techniques replicate real-world interactions. Usability focuses on participants thoughts on the interaction process in areas such as the ease and way in which they could with objects. The coding distribution of the themes indicates that participants main topic of discussion included feelings of the hands and natural interaction. In the following subsections the main themes that emerged from the interviews are analysed.

4.2.2.1 - Emotional Experience

Analysis of the interviews shows that participants generally considered the MOT system to be the most natural of the interaction conditions. A few participants made minor comments regarding the weight of the HMD. This

could be due to the additional weight of the leap sensors and bracket; however, none of the participants explicitly stated this or indicated that the bracket had any effect on the VR experience. The controllers were generally considered to be the most stable and accurate, “they felt snappier, like more responsive”. In contrast, some participants stated that they did not like holding the controllers, due to discomfort.

One of the most common topics discussed about the MOT system was regarding tracking/accuracy issues that effected their sense of immersion, “you’d have hold of it and then it would flip or rotate or whatever”. However, many participants experienced no tracking issues and even those who did (during stable periods), commented heavily on the natural feeling of interacting with hands. They liked not having to use or hold anything, “it’s a world above using the controllers”, “less thinking about how to use the controller and more about gripping things and moving things and putting things down and picking them up”. Of the hand-based approaches, the Manus gloves received the most positive comments, “I think the gloves were probably the most erm like the closest to real world gestures”. Participants generally considered the Manus gloves to be more stable than the MOT system and found the option to use both hands beneficial. However, the Manus gloves lack sensitivity, with many commenting that the left hand required them to clench more than the right hand. This issue is discussed in more detail in section 4.2.2.2.

Overall, the participants found the Manus and MOT systems to be engaging and immersive with the consensus being that the MOT was the most natural feeling. However, the benefit of natural hand gestures was sometimes overshadowed for some users by tracking issues. This led to some users

preferring to use either the Manus or Vive controllers depending on the type of simulation. Participants suggested that in simulations such as first-person shooters they would opt to use the Vive controllers due to its trigger. Participants also suggested that in simulations that require more consistent but less natural hand interaction such as boxing and flower picking simulations, the Manus condition would be the optimal approach. Many participants stated that should the MOT system match the tracking accuracy of the Manus system; they would opt to use that instead.

4.2.2.2 – Usability

The participants almost unanimously agreed that the Vive Controllers were accurate with strong tracking. In addition, some participants felt that the ease of the Vive controllers allowed them to focus more on the simulation. However, some participants found the hand-based conditions easier as it didn't require them to learn any controls, "I think I'd prefer the tracker [MOT system] just because it felt the most normal to use my hands…". In contrast to the Vive controllers, the Manus and MOT system were generally reported as having issues with tracking and accuracy, which for some users made the experience difficult. This shows that in optimal conditions the MOT system is more intuitive than the Vive controllers, however, the tracking issues that it can experience, significantly reduce the ease with which the user can interact.

The Manus was generally considered to be the easiest of the three conditions but had some limitations. Some participants stated that they did not like having to wear gloves as they found them to be restrictive. The participants comments show that the Manus gloves provide more stable tracking. However, the results also show the effect that bend sensors used in VR gloves can have on

consistency across hands. While they enable a high degree of dexterity, the gloves dependence upon independent resistance sensors introduces an element of discrepancy, as each sensor will reside within minor tolerances and will experience unique wear patterns.

In addition to the discrepancy between hands, participants commented on the grip required to grab an object in comparison to the MOT system, “although these grip, they don’t, you’ve got to do that to grip [fully clenched, tight fist] whereas before [MOT] it was like that [curled fingers, not fully clenched fist], in terms of you were actually getting hold of something”. In contrast the MOT system was reported as providing greater levels of dexterity and enabled more comfortable grasping gestures to be performed, without the restrictive feeling of the gloves.

4.2.2.3 – Realism

The participants agreed that the MOT system provided the most natural and realistic experience, with the Vive controllers providing the least. Some participants described the Vive controller as easier but too ‘gamey’ and thus felt unrealistic. In general, participants stated that interaction with the Vive controllers was unnatural. However, some felt that once they were holding an object, the sense of weight helped create a natural and realistic feeling, “the only weight I had was with the controllers, now that felt more realistic”. The results indicate that in the Vive condition, the sense of weight and high degree of accuracy contributed significantly to the sense of realism, partially counteracting the unnatural nature of the interaction and low sense of realism. The results also indicate that in more natural and realistic interactions such as

the Manus and MOT system, a lack of a sense of weight did not reduce the overall sense of realism or naturalism.

The hand-based approaches; MOT system and Manus, received positive feedback “obviously the one’s where you don’t have the controller are more natural”, with the MOT system considered to be the most natural of the three conditions, “the most natural one was the first one [MOT]”. Like the comments on weight with the Vive, some participants commented that they liked being able to pick up objects naturally with their hand, but once they were holding the object, the lack of weight had a small negative impact on their sense of realism. The Manus was natural and realistic “it did feel realistic, especially with the gloves”, but to a lesser extent than the MOT system due to the need to wear a glove rather than interact with a bare hand. However, considering the tracking issues experienced with the MOT system, many felt that the Manus provided the most realistic experience. The results show the Manus would be considered to provide the most stable experience; however, the MOT system provides users with the greatest sense of naturalism and realism.

4.3 – Tracking Data Validity

In this section the results of a pilot study are detailed, analysing log files generated during the study. The logs are analysed to determine the effect of the MOT system on tracking data validity in comparison to that of a single front-facing Leap Motion sensor.

Leap Motion hand data logs for the MOT system were generated during a pilot study of 5 participant playthroughs of the MOT system. The logs recorded user hand data that was later analysed to determine if a valid left or right hand was

detected by the front-facing or 45-degree angle Leap Motions. In addition, the validity of the final aggregated hand generated by the MOT system was ascertained. The hand data logs were analysed using a deep neural network-based validation system.

4.3.1 – Data Validation Process

Four deep neural networks were trained using 8500 unique samples each, to determine if a valid hand was visible to a sensor of the MOT system. Each network was trained on a different feature of the hands; specifically; upper bone angles (fingers bent backwards), finger spacing, bone length and lower bone angles (fingers bent forwards). Due to the number of possible poses and orientations, a single network trained on a single hand feature would not have provide a detailed enough analysis to validate a hand. The feature “upper bone angles” refers to the angle to which a bone is bending backwards, relative to the previous bone. The feature “finger spacing” refers to the spacing between metacarpal bones. The third feature “bone length” refers to the length of each bone. The “lower bone angles” feature refers to the angle to which a bone is bend forwards (natural bend direction), relative to the previous bone. Table 4.3 shows the structure of the four deep neural networks used to validate tracking data and the feature validated.

Table 4.3 - Structure of the Four Deep Neural Networks

Feature	Network 1	Network 2	Network 3	Network 4
Hand Feature	Upper bone angles	Finger spacing	Bone length	Lower bone angles
Layer Structure and Node Sizes	Flatten, Dense (15), Dense (7), Dense (2)	Flatten, Dense (4), Dense (4), Dense (2)	Flatten, Dense (20), Dense (10), Dense (5), Dense (2)	Flatten, Dense (9), Dense (4), Dense (2)
Activation Functions	Relu, Relu, softmax	Relu, Relu, softmax	Relu, Relu, Relu, softmax	Relu, Relu, softmax
Dropout Between Layers	0.2	0.2	0.2	0.2
Optimizer	adam	adam	adam	adam
Loss	Categorical hinge	Categorical hinge	Categorical hinge	Categorical hinge
Metrics	Categorical accuracy	Categorical accuracy	Categorical accuracy	Categorical accuracy
Validation Split	0.3	0.3	0.3	0.3

During each participants simulation experience, hand data from each sensor and the MOT system was collected for every frame. This data was then loaded into the four networks to determine if a valid hand was detected. Figure 4.6 presents a sample of the data used within one of the networks, specifically the bone length network. The data represents the length of each bone, starting with the thumb and moving through each finger. The first column represents the metacarpal bone of the thumb which is not modelled in the Leap API and thus it consistently provides 0.5 as the length.

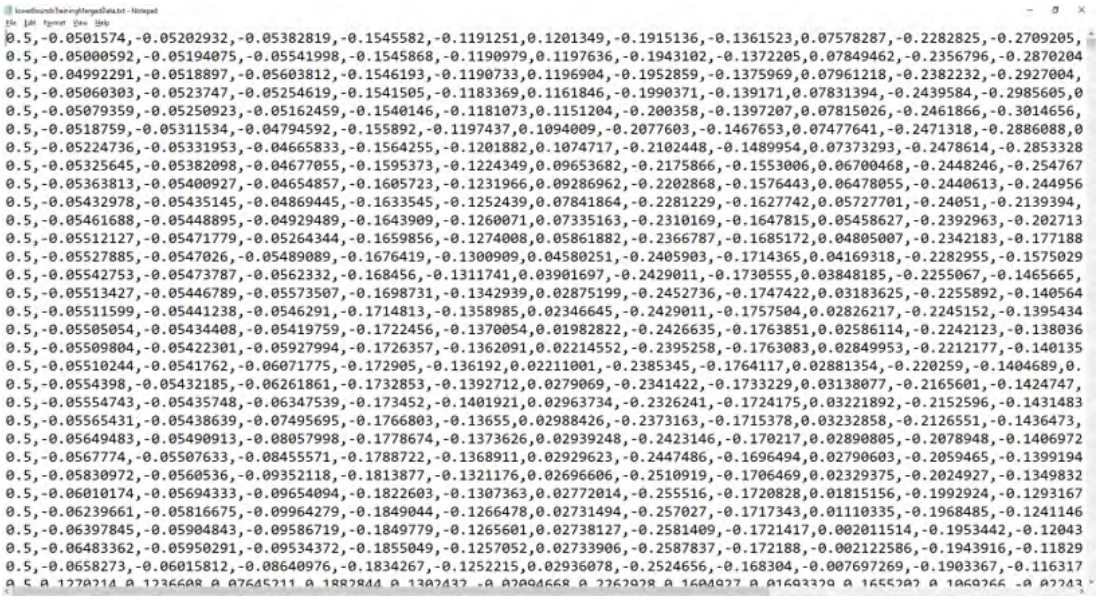


Figure 4.6 - Sample of the hand data used with the validation system

Each line in the log file produced represented a single frame of the simulation. The log file produced by the networks was analysed to determine the percentage of the total frames that were classified as valid for each sensor and the MOT system.

4.3.2 – Sensor Tracking Data Validity

The hand recognition logs were analysed to determine the number of valid left and right hands from each Leap Motion sensor and the MOT system during the simulation. Figure 4.7 shows the average percentage of tracking data in which both the left and right hand were valid for each sensor and the MOT system. Figure 4.8 shows the average percentage of tracking frames in which the left and right hands were valid for each sensor and the MOT system. The aggregated hands are the valid hands output by the MOT system. The front-facing and 45-degree angle hands are the valid hands generated by the corresponding Leap Motion sensors. In both Figure 4.7 and Figure 4.8, higher

bars indicate a greater percentage of tracking frames were valid, indicating better performance. The results presented in Figure 4.7 show the MOT system to have generated more than double the number of frames in which both hands were valid compared to the front-facing sensor. In addition, the graph shows the MOT system produced more valid hand data than the 45-degree sensor but to less of an extent than seen in comparison to the front-facing.

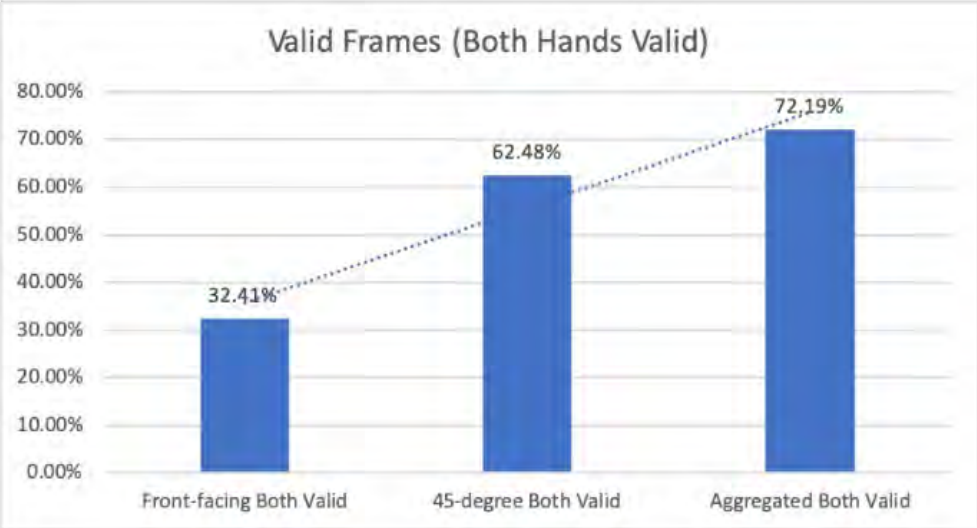


Figure 4.7 - Average number of left and right hands for each sensor and the MOT system

Figure 4.8 shows the front-facing Leap Motion recognized the least number of valid left and right hands on average. In addition, the results show a second Leap Motion positioned on a custom bracket at 45-degrees can identify more valid hands than a single front-facing Leap-Motion. In addition, the graph shows the MOT system produced more valid hand data than the 45-degree sensor but to less of an extent than seen in comparison to the front-facing.

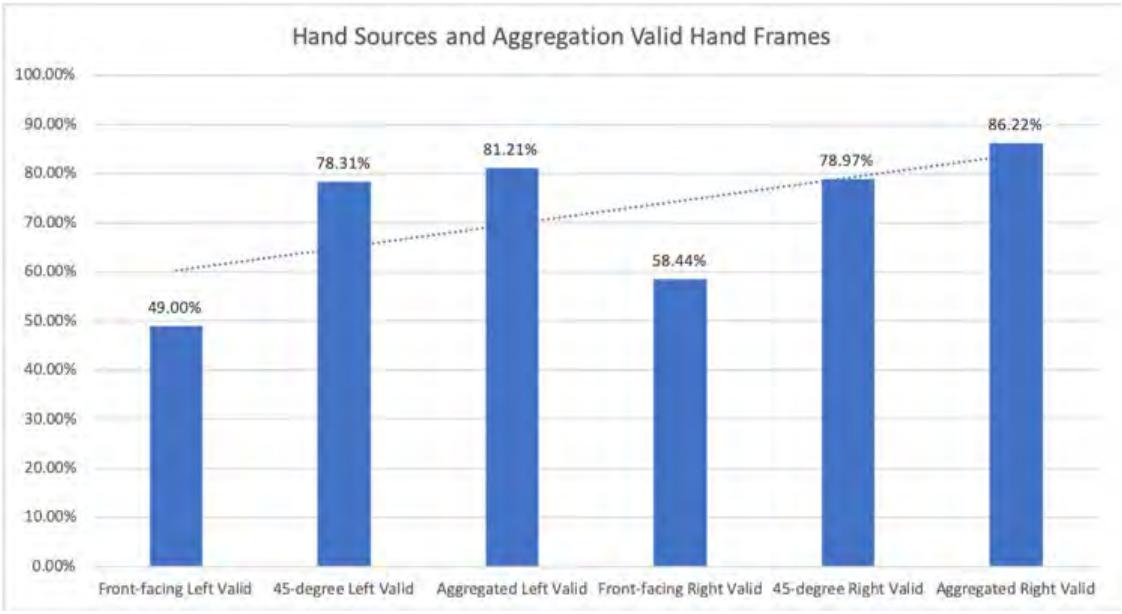


Figure 4.8 - Average number of valid left, right and aggregated hands (MOT system) from each sensor and the MOT system

The results support the previously discussed research in Chapter 2, showing the effects of factors such as self-occlusion on tracking data. The 45-degree sensor positioning provides a clearer view of the user's fingers with the poses in which the back of the hand occludes the fingers substantially reduced. Visual analysis of Figure 4.8 shows the 45-degree sensor produces substantially more valid frames than the front-facing sensor, 78.31% compared to 49% in the case of the left hand. Visual analysis of the graph shows the aggregation process further increases the number of valid frames produced, but to less of an extent than that of the front-facing to 45-degree sensor. This results in the gradual improvement observed in Figure 4.8 for both the left and right hand.

The smaller increase in validity between the 45-degree sensor and MOT system is as a result of the aggregation process. During the simulation, there will have been update cycles in which the front-facing sensor had a valid frame while the 45-degree sensor had either an invalid frame or hands were not being tracked. In such occasions and vice-versa, the aggregation process will have enabled the MOT system to produce a valid tracking frame. The result being the smaller increase seen in comparison to that between the front-facing and 45-degree sensor.

Analysis of the aforementioned smaller percentage difference between the 45-degree sensor and MOT system validity, supports the notion that the 45-degree sensor provides a clearer view of the user's fingers. The validity results suggest the 45-degree sensor contributed significantly more valid frames to the MOT system validity than the front-facing sensor. This hypothesis will be tested with a linear regression test in section 4.3.4.

Analysis of the results for two handed validity shown in Figure 4.7 shows the MOT system to produce more valid frames in which both hands are valid. These results further support the notion that the 45-degree sensor produces more valid hands due to reduce self-occlusion. However, the results also suggest the MOT system aids in improving tracking field-of-view with both hands detected in significantly more frames than a front-facing sensor. The smaller field-of-view of the front-facing sensor will have made detecting both hands difficult. The smaller field-of-view will have a smaller optimal detection space in comparison to the MOT system. This will likely have made positioning both hands within the detection space sufficiently for valid classification unlikely, particularly during object interactions.

4.3.3 – Chi-Square Test

A Chi-Square test was conducted to analyse hand validity and determine the significance between the sensors and MOT system on hand validity produced. The test would enable the 45-degree sensor positioning to be evaluated to determine whether it had increased hand validity as previously hypothesized. Furthermore, the test would evaluate the effectiveness of the aggregation process at further improving hand validity. Table 4.4 shows the details of the Chi-Square test including the null and alternative hypotheses.

Table 4.4 - Details of the Chi-Square test conducted

Null Hypothesis	Assumes that there is no association between the two variables.
Alternative Hypothesis	Assumes that there is an association between the two variables.
Outcome	Null Hypothesis Rejected, Alternative Accepted

To determine the significance between the counts of valid and null hands observed by a Leap Motion sensor (front-facing or 45-degree angle) and the MOT system, multiple Chi-Square tests were run. The tests were run for both left and right hands and compared the front-facing Leap Motion with the MOT system and the 45-degree angle Leap Motion with the MOT system. The p-value of the tests from the front-facing Leap Motion and the MOT system were all significant at $p < .05$. The p-value of the tests from the 45-degree angle Leap Motion sensor and the MOT system were all significant at $p < .05$, except for one that was not significant. The results show that the MOT system was able to produce significantly more valid hands than the front-facing Leap Motion sensor. The MOT system was also able to produce significantly more valid hands in most cases than the 45-degree angle Leap Motion sensor. The

chi-square statistics for the 45-degree angle Leap Motion sensor and the MOT system tests indicate that the data has greater similarity than the front-facing Leap Motion and MOT system. This also shows that the 45-degree angle Leap Motion sensor recognizes more valid hands than the front-facing Leap Motion. This would indicate that a 45-degree angle Leap Motion sensor as presented in this work is better positioned to recognize user hands in VR than a Leap Motion mounted on the front of a headset.

The results show that the MOT system can produce more valid hands during the simulation than either the front-facing or 45-degree angle Leap Motions on their own. This is because the MOT system aggregates all the available data from both sensors to produce the best hands possible. The MOT system allows a front-facing and bracket mounted Leap Motion to be used together to produce better results. A hand-tracking system cannot automatically swap between Leap Motion sensors during a simulation because the virtual world hand position generated by each sensor is not the same. Consequently, if the system were to just swap between sensors, a user would observe a significant jump in their hand positions, reducing immersion and usability. Therefore, the MOT system is needed to aggregate observed hand data and blend between sensor hand positions.

The results in Figure 4.8 indicate that the MOT system offers a small improvement in individual valid hand recognition over a bracket mounted Leap Motion. However, as shown in Figure 4.7 this improvement increases when the number of times that two valid hands were generated by a sensor or the MOT system is considered. The results show that the MOT system was able

to generate two valid hands for a frame more often than a bracket mounted Leap Motion.

The results from the hand logs show that a bracket mounted Leap Motion offers better positioning for hand tracking than traditional front-facing sensors. The MOT system allows multiple sensor data to be combined to produce more valid hand representations than a single sensor. It facilitates more occurrences of two valid hands being presented to a user. This should improve immersion and usability during VR hand-based interaction.

4.3.4 – Linear Regression Test

A Linear regression test was performed to assess the linear relationship between the sensors and MOT hand validity. The test was used to determine the statistical significance of any relationship and how much variation in the MOT systems hand validity is explained by a sensor's validity. Table 4.5 shows the details of the conducted linear regression test including the null and alternative hypotheses.

Table 4.5 - Details of the Linear Regression test conducted

Null Hypothesis	Assumes that there is no relationship between the two variables. The null hypothesis states that the coefficient of the slope is equal to zero.
Alternative Hypothesis	Assumes that there is a relationship between the two variables. If there is a significant linear relationship between the independent variable X and the dependent variable Y, the coefficient of the slope will not equal zero.
Outcome	Null Hypothesis Rejected, Alternative Accepted

Table 4.6 shows the results of the linear regression test, indicating the 45-degree sensor had a large effect size according to (Cohen, 2013) with the front-facing having no statistically significant effect.

The results from the Chi-Square tests indicate a statistically significant difference between the validity of the front-facing sensor and the MOT system. However, the linear regression test shows that while statistically different, the front-facing sensor had an insignificant effect on the validity of the MOT system. The results show the 45-degree sensor was much more effective at producing valid hand data, having a statistically significant effect on the MOT system. The results support the results of the previous tests indicating that a 45-degree sensor offers a better view of the user's hands.

Table 4.6 - Results of Linear Regression test on pilot study data

Task	Regression Equation	ANOVA	R^2 (%)	Adjusted R^2 (%)	Increase in MOT Validity
Front-facing Left Valid -> MOT Left Valid	MOT Left Validity = 60.470 + (0.448 * front-facing left validity)	F(1,3)=2.046, p=0.248	40.5	20.7	0.448% (95% CI, -0.549% to 1.445%)
Front-facing Right Valid -> MOT Right Valid	MOT Right Validity = 72.946 + (0.252 * front-facing right validity)	F(1,3)=1.520, p=0.305	33.6	11.5	0.252% (95% CI, -0.399% to 0.903%)
45-degree Left Valid -> MOT Left Valid	MOT Left Validity = 8.509 + (0.930 * 45-degree left validity)	F(1,3)=329.365, p <.0005	99.1	98.8	0.930% (95% CI, 0.767% to 1.093%)
45-degree Right Valid -> MOT Right Valid	MOT Right Validity = -26.001 + (1.414 * 45-degree right validity)	F(1,3)=34.796, p=.01	92.1	89.4	1.414% (95% CI, 0.651% to 2.177%)

The results of the Linear regression test concur with the results of the Chi-Square tests and visual analysis of Figure 4.7 and Figure 4.8. The large effect size shows a 45-degree sensor provides a better view of the user's hands, reducing the effect of self-occlusion. This results in more valid data being produced; thus, the sensor contributes more to the MOT system validity. Furthermore, the results explain the small percentage difference in valid

frames produced seen in Figure 4.8 between the 45-degree sensor and MOT system.

4.4 – User Performance

In this section the results of the Quantitative analysis are detailed, analysing log files generated during participant experiences to determine the effect of the conditions and the order experienced on task completion times.

During participant experiences a log file was generated that stored the time in seconds taken to complete each task and the simulation overall. The timer would begin upon user interaction with an object required for the current task. The timer would stop once the clipboard had marked the task as completed. Figure 4.9 shows an extract of a log file generated for a participant.

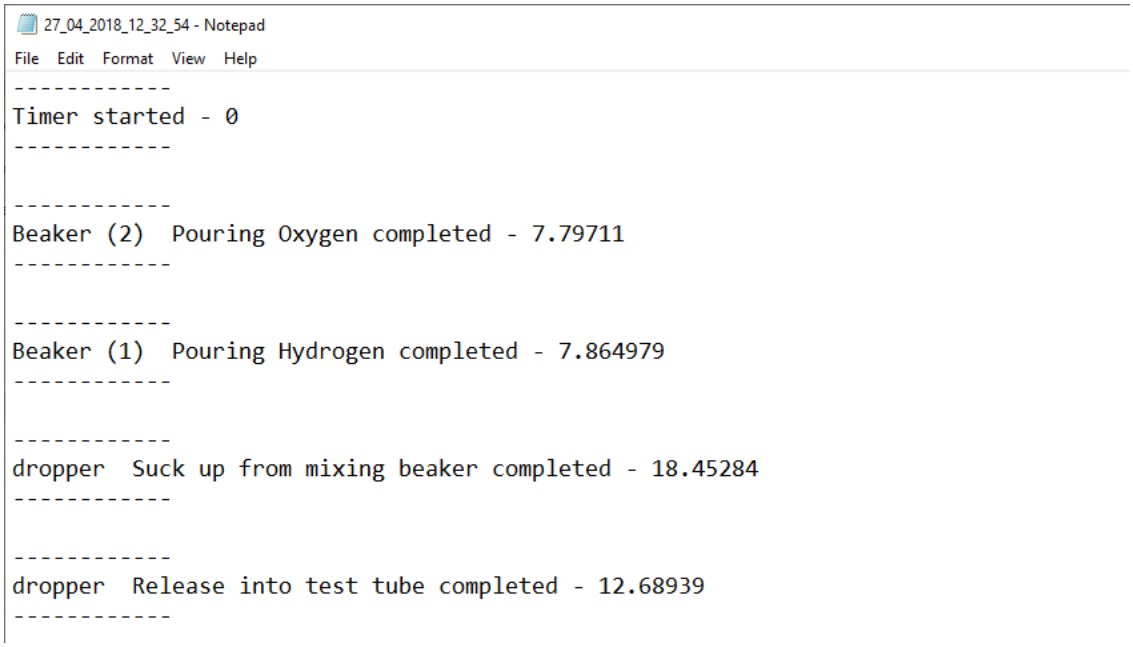


Figure 4.9 - Example of performance data logged during the simulation

Figure 4.10 shows the average task completion times for all key tasks in the simulation for each of the three interaction conditions. Table 4.7 presents a description of the tasks featured within the Chemistry-based simulation used for testing.

Table 4.7 - Description of the Chemistry simulation tasks

Task	Description
Pouring Beaker	This task required participants to pick up the beaker specified, move it to above the large mixing beaker and angle the beaker to pour it. This task was completed for each of the four beakers, with the large mixing beaker in the middle of the table as shown in Figure 4.1.
Dropper Suction	This task required participants to pick up the pipette dropper, change it to suction mode using the approach for the given interaction condition. The tip of the pipette dropper would then be positioned into either the large mixing beaker or a test tube. The participant would have to hold the pipette dropper in place until all the fluid had been collected.
Dropper Release	This task required participants to pick up the pipette dropper, change it to release mode using the approach for the given interaction condition. The tip of the pipette dropper would then be positioned into either the large mixing beaker or a test tube. The participant would have to hold the pipette dropper in place until all the fluid had been released.
Centrifuge Configuration	This task required the user to tap on large virtual UI buttons to decrease a pre-set time from 32 to 20 seconds. Figure 4.2 shows the virtual UI and time buttons used.
Bunsen Loading	This task required participants to pick up and move a test tube to the bunsen burner and hold it steady above the flame.

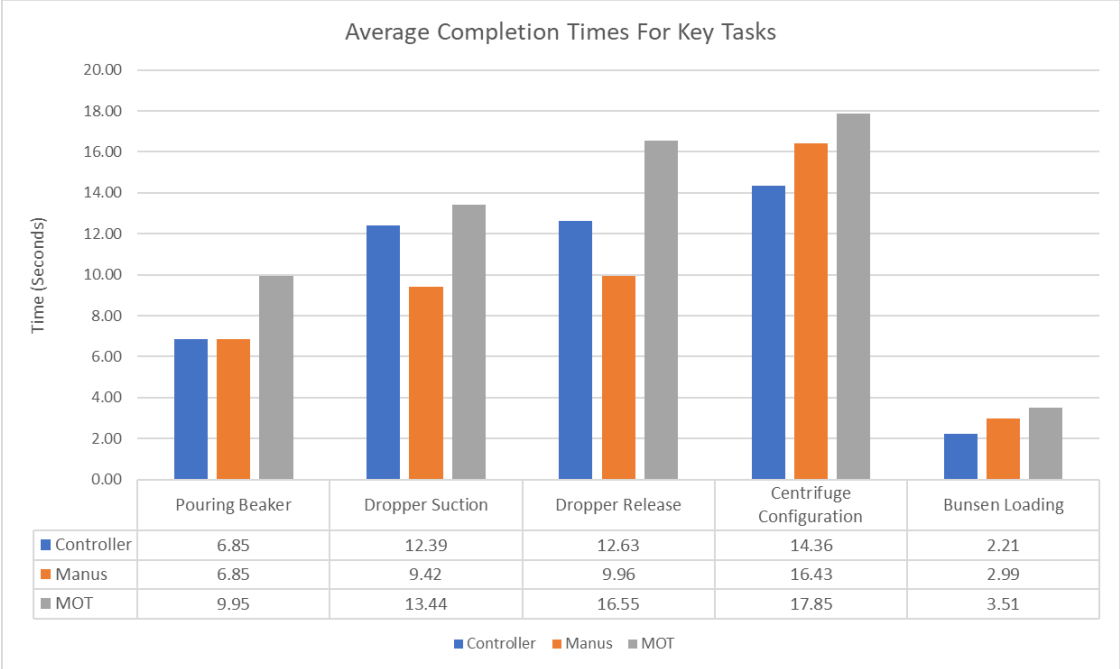


Figure 4.10 - Average Task Completion Times

The results show the MOT system to have the longest completion time in all key tasks. The results of the thematic analysis suggest that this is likely due to several factors. Firstly, some participants reported issues with tracking accuracy and occasionally objects moving or rotating as a result. This will have delayed the user resulting in increased task completion times. Analysis of the times for translation/rotation tasks such as the beaker, bunsen and centrifuge show the controller to have the fastest completion times. Participant feedback indicates the controllers were found to be the easiest due to the simple trigger-based interaction. The simplicity of the task and binary nature of the trigger will have made object selection and movement quicker to detect and complete than hand gestures.

Secondly, analysis of the dropper related tasks shows the MOT system to perform the slowest. The pipette dropper required users to switch modes by tapping the end in the case of the hand-based conditions and clicking a pad for the controller. The hand conditions enabled participants to hold the device in one hand and tap the end with the other. In the case of the MOT system this would typically result in occlusion due to one hand covering the other and thus the virtual object would be released. As a result, participants would generally release the dropper, tap the end with a finger and then pick it up again, using the same hand. This in combination with the tracking issues previously discussed will have accounted for the increased duration compared to the Manus condition. In the case of the controller, participant feedback indicates some participants had difficulty pressing the pad whilst holding the object due to factors such as hand size and controller positioning within the hand. The difficulty with the pad in the controller condition will have increased the task time resulting in the second fastest time, despite the simplistic interaction. The two-handed interaction capabilities and the difficulties experienced in the MOT and controller conditions, show why the Manus condition performed the quickest.

A Bivariate Pearson Correlation test was performed to identify any statistically significant relationships between the condition order and simulation completion times, to detect any effects of learning. The results indicated no statistically significant correlation; therefore, it is concluded that the randomization was sufficient to minimize learning to none to insignificant levels.

Performance logs of task completion times generated during the simulations were analysed to determine the average completion times for each task on each condition. The grand mean for simulation duration was 164.42 seconds, the Manus condition was the fastest with a mean completion time of 136.32 seconds and the MOT taking the longest at 196.79 seconds. The Vive Controllers had a mean of 160.14 seconds. The average completion times in comparison to the Vive controllers show a 22.89% increase in the case of the MOT system with a 14.87% reduction for the Manus. Normality tests were conducted using the Shapiro-Wilk's test and by analysing skewness and kurtosis. Task durations were normally distributed for all techniques, as assessed by Shapiro-Wilk's test ($p > .05$). Task durations were normally distributed for all techniques with Z-scores for both Skewness and Kurtosis.

4.4.1 - Total Completion Time Condition Comparison

A one-way within subjects (or repeated measures) ANOVA was conducted to compare the effect of interaction technique on total simulation completion time. The test was used to determine whether there were any statistically significant differences in the mean completion times of the three interaction conditions. Thus, determining whether the issues reported by participants and identified in the thematic analysis had any significant impact on performance or just user experience. Table 4.8 shows the details of the one-way within subject's ANOVA including the null and alternative hypotheses.

Table 4.8 - Details of the One-Way Within Subjects ANOVA test conducted

Null Hypothesis	The null hypothesis states that the related population means are equal.
Alternative Hypothesis	The alternative hypothesis states that the related population means are not equal.
Outcome	Null Hypothesis Rejected, Alternative Accepted

There were no outliers and the data normally distributed, as assessed by Skewness and Kurtosis Z-scores and Shapiro-Wilk test ($p > .05$), respectively. The assumption of sphericity was not violated as assessed by Mauchly's test of sphericity, $\chi2(2) = 3.846$, $p = .146$.

The interaction technique elicited statistically significant performance differences, $F(2,36)=35.195$, $p <.0005$ and Wilks' Lambda = 0.270, $F(2,17)=22.937$,$p<.0005$, with Manus performance 136.32 ±19.58, MOT system 196.79 ±31.44 and Vive 160.14 ±21.39. The mean completion times and standard deviation show that the MOT system took the longest on average and experienced the greatest variance. In contrast the Manus took the least amount of time and had the lowest standard deviation, showing it provided the most consistent experience. The degrees of freedom (df) for between groups was 2 and within groups was 36 with a F value of 35.195. Post hoc analysis with a Bonferroni adjustment revealed that user performance statistically significantly decreased from the Manus to MOT condition (60.47 (95% CI, 37.54 to 83.41) seconds, $p < .0005$), and from the Manus to Vive technique (23.82 (95% CI, 8.06 to 39.57) seconds, $p = .003$). The results also showed a statistically significant decrease from the Vive to MOT condition (36.66 (95% CI, 18.56 to 54.76) seconds, $p < .0005$).

The results suggest that the glove-based interaction is more effective than the controller-based interaction. The results also support the findings of the research presented in Chapter 2, such as that of Pinto et al (2015) and Figueiredo et al (2018), showing the negative effects of occlusion and instability on tracking. Analysis of the performance data in comparison with the findings of the thematic analysis shows the effects of issues participants

reported, such as occlusion during two-handed interactions, with the MOT system having the longest overall completion time.

The results show the glove condition provided the fastest simulation completion time. It can be hypothesized that the gloves use of both motion controller tracking with basic hand interaction enabled easy user interaction with precise stable tracking. The use of bend sensors will have enabled limited but consistent gesture recognition with motion tracking providing precise positioning of the virtual objects. The thematic analysis and Likert results show the MOT system was found to be the most natural and engaging experience. Therefore, as the two hand conditions share the same interaction approach with the MOT providing greater degrees-of-freedom. It can be hypothesized that the MOT system would outperform the Manus condition if the tracking stability could be improved. Furthermore, a solution for handling occlusion would enable two handed interactions with the dropper, further improving user performance.

4.4.2 - Condition Order and Experience

A two-way mixed ANOVA test was conducted to analyse the effects of condition ordering and participant experience on performance. The tests were conducted to ensure the condition order randomization was sufficient to minimize any effects of learning. Furthermore, the test was conducted to ensure that the detailed explanation of the virtual controls prior to experimentation was sufficient to minimize the potentially advantageous effects of any prior experience. Table 4.9 shows the details of the two-way mixed ANOVA's including the null and alternative hypotheses.

Table 4.9 - Details of the Two-Way Mixed ANOVA tests conducted

Null Hypothesis	The null hypothesis states that the variances of the differences between the categories of the within-subject factors are equal.
Alternative Hypothesis	The alternative hypothesis states that the variances of the differences between the categories of the within-subject factors are not equal.
Outcome	Null Hypothesis Accepted, No statistically significant effects

There were no outliers and the data normally distributed as assessed by Skewness and Kurtosis Z-scores and Shapiro-Wilk test ($p > .05$), respectively. The assumptions of sphericity were not violated. The results shown in Table 4.10 indicate both version order and prior experience had no statistically significant effect on condition performance. In addition, the results suggest that order randomization used was effective in minimizing the learning effect on subsequent playthroughs. Furthermore, the results suggest that the designs of the interaction conditions enabled both new and experienced users to efficiently interact with the virtual environment.

Table 4.10 - Results from Two-way Mixed ANOVA on Condition Ordering and Participant Experience

Version Order On Condition Performance	
Mauchly's test of sphericity	$\chi2(2)=4.833$, $p=.089$
Condition Order and Total Completion Times	$F(10,26)=1.644$, $p=.149$
Between Order Groups	$F(5,13)=.377$, $p=.856$
Experience On Condition Performance	
Mauchly's test of sphericity	$\chi2(2)=2.584$, $p=.275$
Experience and Total Completion Times	$F(6,30)=1.436$, $p=.234$
Between Experience Groups	$F(3,15)=.892$, $p=.468$

4.4.3 – Experience on Performance

A two-way mixed ANOVA was conducted to compare the effect of participant experience on total simulation completion time for each condition. The test was conducted to ensure that the detailed explanation of the virtual controls prior to experimentation was sufficient to minimize the potentially advantageous effects of any prior experience. Table 4.11 shows the details of the two-way mixed ANOVA's including the null and alternative hypotheses.

Table 4.11 - Details of the Two-Way Mixed ANOVA test conducted

Null Hypothesis	The null hypothesis states that the variances of the differences between the categories of the within-subject factors are equal.
Alternative Hypothesis	The alternative hypothesis states that the variances of the differences between the categories of the within-subject factors are not equal.
Outcome	Null Hypothesis Accepted, No statistically significant effects

There were no outliers and the data normally distributed as assessed by Skewness & Kurtosis Z-scores and Shapiro-Wilk test ($p > .05$), respectively. There was homogeneity of variances, as assessed by Levene's test of homogeneity of variance ($p > .05$). The assumption of sphericity was not violated as assessed by Mauchly's test of sphericity, $\chi2(2)=2.584$, $p=.275$. There was no statistically significant effect on performance by experience, $F(6,30)=1.436$, $p=.234$. The interaction technique elicited statistically significant performance differences. The main effect of experience showed no statistically significant difference in mean completion times between experience groups $F(3,15)=.892$, $p=.468$. These results suggest that the three conditions trialled enabled both new and experienced users to efficiently interact with the simulation.

4.5 – Summary

The results of the questionnaire show the MOT system outperforms the Vive controllers and Manus gloves in several areas; such as providing the greatest sense of engagement, immersion and usability. The experimentation results concur with that of the previously discussed research such as that of Pinto et al (2015), showing user preference for direct hand-based interaction as it provides a more natural and immersive experience. The user performance evaluation of the three conditions shows further agreement with that of Pinto et al (2015), with the MOT system shown to have the longest completion time. The questionnaire results show the MOT system had the lowest score in terms of ease of use and the second highest in resemblance to real life. Some participants reported issues with object rotation, due to thumb occlusion and the Leap API having difficulty determining the hands orientation as a result. These results concur with those previously discussed of Chapoulie et al (2015) that found occlusion to be a problem with hand-based interaction, particularly with rotation tasks. Therefore, further work is required to improve the tracking accuracy of the system and the way in which users interact with virtual objects; specifically, user's comments regarding having to over-rotate.

The results of the thematic analysis show users have a strong preference for the MOT system, with many commenting on its natural feeling and freeing interaction. The absence of an indirect interface such as gloves or controllers was found to aid in creating a more natural and immersive experience. Participants commented that it meant less thinking about how to use the controls and more about doing it. Analysis of the Quantitative and Qualitative results show agreement between participants comments on the interaction

and sense of naturalism using the MOT system and the Likert scores. The Likert results show the MOT system to have the highest score for both usability and immersion. The high level of fidelity and dexterity the MOT system provides was found to increase the naturalness of the controls. The MOT system enables users to interact with more natural and subtle gestures than those often required in glove-based interaction. The strong user preference for the MOT system shows users prefer a more natural and immersive experience. The MOT system enables more natural and immersive interactions, significantly improving the level of engagement and the ease with which they can interact with virtual objects.

Analysis of the Quantitative and Qualitative results show agreement in the performance of the three interaction approaches. The thematic analysis and user performance tests show the impact of the tracking issues reported, with the MOT system having the slowest time in translation/rotation tasks. Some participants reported tracking issues in the MOT system condition, resulting in changes in virtual object rotation and difficulty interacting as a result. This is evident in the performance data with the MOT system having the longest completion time in all tasks and the overall simulation on average.

The user performance results also show the effect of the reported feedback with regards to the dropper tasks. The controller condition had longer task times compared to the gloves, while matching or outperforming in other tasks. Participant feedback found a key strength of the gloves to be the support for two handed interactions. This reportedly made the dropper tasks much easier to complete compared to the other conditions. Analysis of the Quantitative and Qualitative results show agreement, with the glove having the shortest

completion times for the two-handed interaction tasks. In the case of the MOT condition, attempts at performing the same two-handed interaction resulted in occlusion, thus participants took to a single-handed interaction approach. Analysis of the performance data concurs with participant feedback, showing the gloves to only outperform both the MOT and controller conditions in the dropper related tasks. The results of the user performance analysis generally concur with previously discussed research such as that by Chapoulie et al (2014) and Chapoulie et al (2015) which showed that wands/controllers and direct manipulation are equivalent with regards to speed, with wands generally more consistent especially during rotation. This is particularly evident when comparing the controller and glove times for the beaker pouring and bunsen loading tasks, where the times are either identical or slightly decreased in the case of the controller. The dropper task times provide contrasting results to the conclusion of Chapoulie et al (2014) and Chapoulie et al (2015), however, this is likely due to the difficulty participants reported using the touch pad to toggle modes.

Experimental results show that optical based trackers are not yet able to match the performance of traditional interaction techniques. However, the study shows that designs such as the MOT system provide a much more stable and immersive experience for the user in comparison to a single front-facing sensor. The results of the tracking validity analysis show the MOT systems data aggregation approach significantly improves the validity of tracking data. The combing of sensor data through an aggregator enables improved tracking of users' hands and more valid hands being presented to the user. These results help to answer one of the research questions, showing multi-sensor

aggregation improves optical-based hand tracking by improving the number of valid frames presented to the user, providing a more consistent experience. These findings also expand on that of Jin et al (2016) who demonstrated, in a non-VR context, the tracking improvements of using an additional tripod mounted leap sensor in controlling a robotic arm. The results presented help in answering one of the research questions, showing a custom mounted 45-degree angled sensor to be the most effective sensor configuration for optical-based hand interaction. These results also concur with previously discussed research such as that of Jin et al (2016), Marin et al (2014), Clark and Moodley (2016) and Zou, et al (2019), showing the use of multiple sensors to improve hand tracking. The MOT systems design narrows the performance difference between optical-based hand interaction and traditional interaction approaches.

The MOT system presented in this phase of the work offers a natural hand-based interaction mechanism that can be applied to any interactive application, such as VR, augmented reality and 3D/2D desktop applications. The presented interaction approach has been evaluated against traditional glove and motion controller-based interaction approaches. The results show that the MOT system provides a strong sense of immersion, usability and engagement. In addition, the results strongly support current research, which suggests that despite the performance decrease, users prefer gesture-based interaction. Furthermore, the results support the hypothesis of Clark & Moodley (2016) showing that the use of an additional leap sensor improves accuracy and limits occlusion.

Finally, the results also show the clear benefits, in a VR context, of mounting an additional sensor on a custom bracket and using software to aggregate multiple sources of data. The results show that a bracket mounted Leap sensor offers an improved field-of-view over a front mounted sensor. This results in improved hand tracking for the user. The results of the tracking data validity analysis, comparing the performance of the two sensors, concurs with previously discussed research such as Shao (2016). The configuration of the 45-degree sensor reduces the effects of self-occlusion by proving more of a top-down view. This enables the user's fingers to be tracked while the front-facing sensor is occluded by the back of the hand. The tracking validity analysis of the two sensors also helps to answer one of the research questions, showing the 45-degree sensor to be a much more effective configuration. The next Chapter presents the phase two design of the Multiple Optical Tracking (MOT) system, specifically the additional systems implemented based on the results detailed in this Chapter.

Chapter 5 – Design Phase Two

In this Chapter the design of the Multiple Optical Tracking (MOT) System is extended based on the results found in phase one. In addition, the software and development tools used are detailed. The phase two design incorporates a Deep Neural Network based hand validation system designed to provide more versatile and accurate detection of invalid hand data. The validation system is used throughout the MOT system to ensure invalid hands are not presented to the user.

The phase two design also includes an occlusion detection system designed to overcome hand to hand occlusion, such as during two-handed interactions. The system works with the inferred hand system to ensure a stable virtual hand representation is presented when occlusion is detected. To smooth the transition between sensors, live and motion controller-based tracking, along with general tracking, an interpolation system was designed along with a stability detection system. The interpolation system is designed to monitor tracking data and smooth out any perceivable ‘jumps’ as well as smoothing the transition between inferred and live tracking data. The stability detection system was designed to ensure the stability of hand tracking data.

The final additional features were designed to further improve the consistency and stability of the system to create a smoother user experience. General improvements made because of the first phase of experimentation can be found in Appendix 7 and 10. Appendix 8 presents a debugging tool created to aid development.

Figure 5.1 shows an overview of the extended MOT system showing the deep neural network-based validation, occlusion detection, stability detection and interpolation components that have been integrated into the system. Appendix 9 contains screenshots of the code implementations.

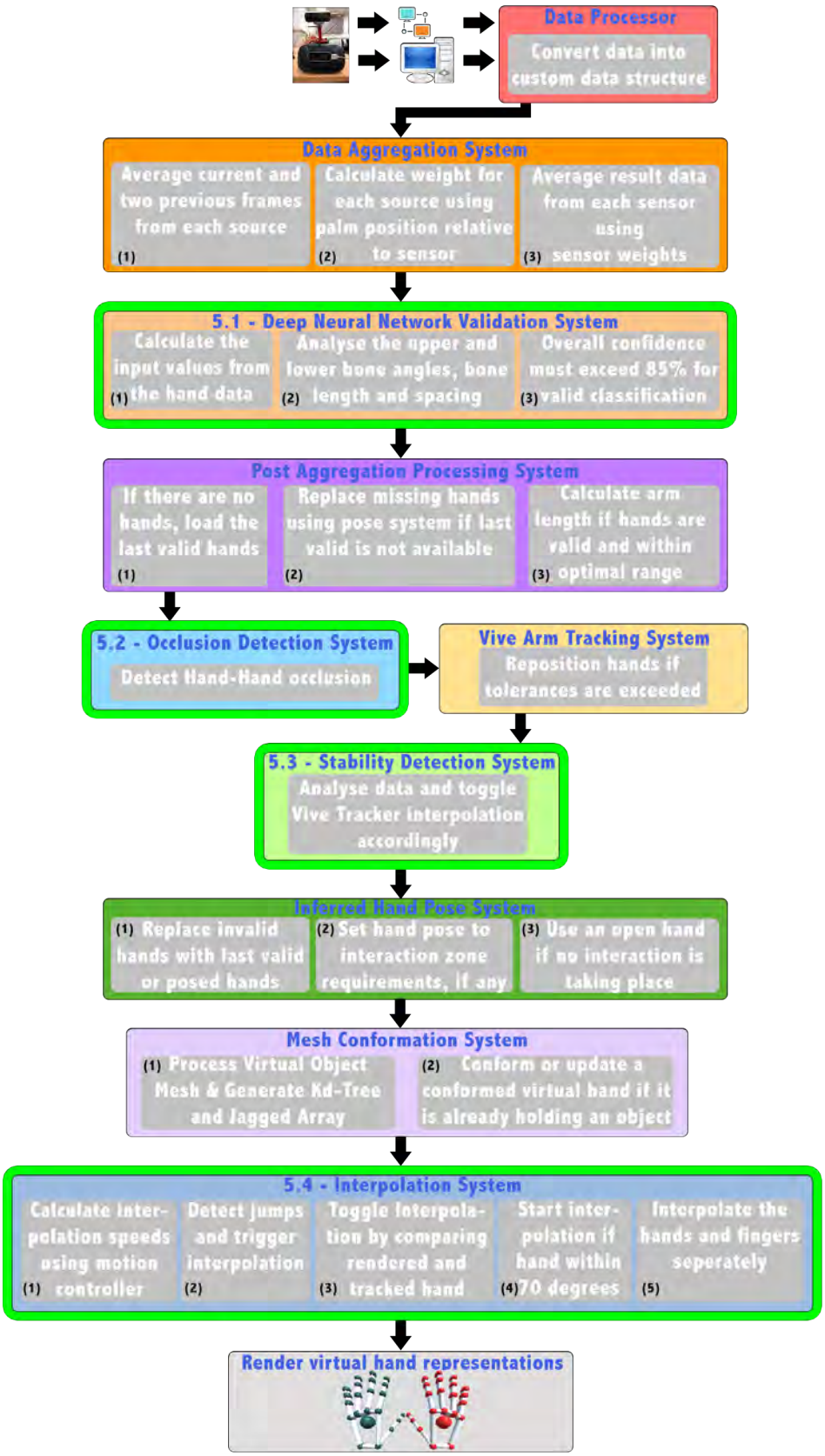


Figure 5.1 - Overview of the extended Multiple Optical Tracking (MOT) Systems components (Additional components have a green outline)

5.1 – Deep Neural Network Validation System

The previous validation system design used a series of tests and rules, analysing the angle of the proximal and intermediate bones relative to the palm normal. In addition, the system checked for intersections between the proximal and intermediate bones and ensured equal metacarpal spacing. However, during phase one experimentation, the performance of the system was found to need improvement. During the simulation, some participants reported briefly seeing slightly deformed hands. The design approach previously taken used strict thresholds for bone angles and two pass/fail tests for intersection. Due to the large number of hand poses the system could be required to validate, a more intelligent and versatile system was needed.

Based on research into hand gesture classification, this extended system followed a machine learning approach using Deep Neural Networks designed in Python using TensorFlow and Keras, to validate hand data. The networks are designed using the multilayer perceptron (MLP) model as opposed to alternatives such as convolutional neural networks (CNN's). An alternative approach, convolutional neural networks are typically used for image analysis such as classification. Previously discussed research such as that of McCartney et al (2015) has shown that CNN's can be used for gesture recognition by generating images from gesture movements. However, because of the additional computation, such approaches are not capable of online real-time recognition. The developed validation system needed to validate both hands in real-time, ensuring natural hand representations are presented irrespective of orientation.

Previous machine learning approaches have relied upon a pre-set gesture vocabulary for verification. The MOT system is designed to enable natural and realistic interactions, enabling real-world skills to be practiced. Therefore, a hand feature-based movement range analysis approach was taken because the validation system needs to be capable of distinguishing between a gesture/pose that a user could perform and a deformed hand. The validation system presented is designed to learn the natural movement range of the human hand. The system uses four deep neural networks, with each network focusing on a specific feature of the human hand. The validation system was designed to learn natural movement ranges rather than specific hand poses/tracking data. During development, the performance of a single deep neural network was evaluated. However, due to the large number of potential hand poses and thus feature ranges, the accuracy of the network was poor. The network was retrained to focus on a specific hand feature; specifically, upper bone angles and re-evaluated. The focusing of the network on a single feature significantly improved performance. However, through the evaluation of deformed/un-natural hand poses additional key hand features were identified.

5.1.1 – Network Designs

The validation system used four deep neural networks; each trained on a specific hand feature; upper bone angles (fingers bent backwards), finger spacing, bone length and lower bone angles (fingers bent forwards). The validation system is used to validate hand data at several stages during the MOT systems update cycle. The stages at which the validation system is used

to validate data include, prior to aggregation to ensure only valid data is used and post aggregation to ensure the result is valid.

Due to the number of possible poses and orientations, a single network trained on a single hand feature would not have provided a detailed enough analysis to validate a hand. Figure 5.2 shows the hand features that the networks were trained on. The four networks were developed using TensorFlow with the Keras API to provide higher level functionality enabling easier and faster development of the networks. Dropout layers were used between layers, setting a fraction of the input units to zero to help prevent overfitting. Each of the networks used Dense layers as part of the MLP approach. Dense layers feed all outputs from the previous layer to all neurons within the layer, with each neuron providing one output to the next layer. Flatten layers were used to flatten the input to a one dimensional vector (TutorialsPoint, 2020; Keras, 2020). Table 5.1 shows the structure of each of the four deep neural networks.

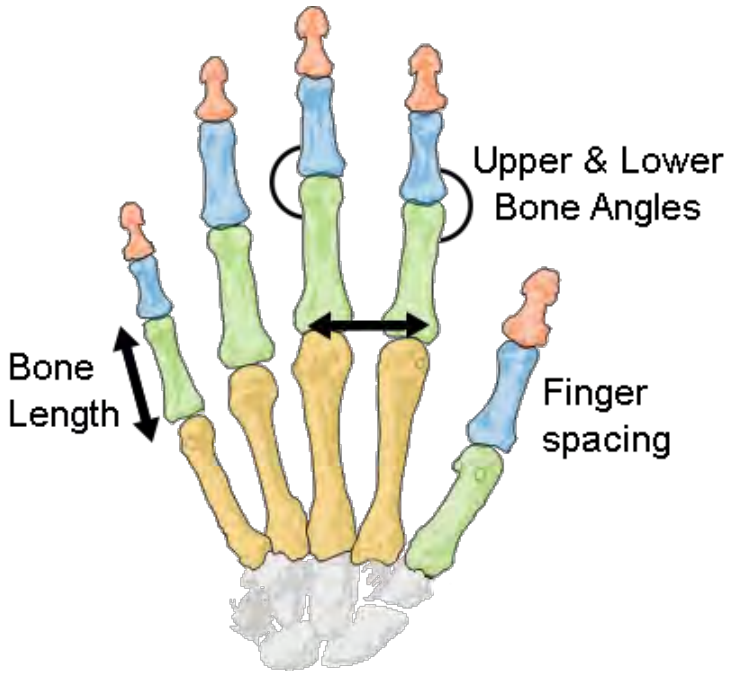


Figure 5.2 - Hand Features Used in Deep Neural Networks

Table 5.1 - Structure of the Four Deep Neural Networks used for hand validation

Feature	Network 1	Network 2	Network 3	Network 4
Hand Feature	Upper bone angles	Finger spacing	Bone length	Lower bone angles
Overall Weighting	0.3	0.3	0.3	0.1
Layer Structure and Node Sizes	Flatten (15), Dense (15), Dense (7), Dense (2)	Flatten (4), Dense (4), Dense (4), Dense (2)	Flatten (20), Dense (20), Dense (10), Dense (5), Dense (2)	Flatten (15), Dense (9), Dense (4), Dense (2)
Activation Functions	Relu, Relu, softmax	Relu, Relu, softmax	Relu, Relu, Relu, softmax	Relu, Relu, softmax
Dropout Between Layers	0.2	0.2	0.2	0.2
Optimizer	adam	adam	adam	adam
Loss	Categorical hinge	Categorical hinge	Categorical hinge	Categorical hinge
Metrics	Categorical accuracy	Categorical accuracy	Categorical accuracy	Categorical accuracy
Validation Split	0.3	0.3	0.3	0.3

The four deep neural networks are used in combination using a weighted average to calculate the overall validation confidence. The weighting of each network in the overall confidence is shown in Table 5.1. The weighting of the networks was determined during developmental testing by continuously performing the same series of gestures whilst monitoring both individual network confidences and the overall confidence. The networks were then presented with the gesture vocabulary mixed with an invalid dataset, with the resulting confidences monitored. Once the data had been validated, the network confidences were evaluated to identify the impact of each feature in distinguishing between invalid and valid poses. The network weightings were then adjusted and re-tested. The network weightings used were selected to

provide the validation system with the highest recognition accuracy whilst ensuring versatility due to the large number of potential hand poses the network may not have previously been trained on.

Each network provides a confidence level based on their feature, with a minimum overall confidence of 85% required to be classified as valid. This ensures only valid frames are presented to the user, providing a consistent experience. A range of confidence values were explored to determine the optimal confidence ensuring high accuracy whilst also being versatile. A minimum confidence of 85% was selected due to the large number of possible hand poses that would require validation. The minimum confidence used needed to account for the required versatility whilst still ensuring validity.

As previously discussed, the validation system was designed to learn natural movement ranges rather than specific hand poses/tracking data. Thus, requiring a higher overall accuracy would require the tracking data to have greater similarity to a training sample, introducing elements of a gesture vocabulary. Research has shown the limitations a gesture vocabulary imposes and thus the system needed to be versatile whilst still able to validate tracking data. Figure 5.3 shows a visualization of the upper bone angles deep neural network with 15 input nodes, 15 nodes in the first hidden layer, 7 nodes in the second hidden layer and 2 output nodes.

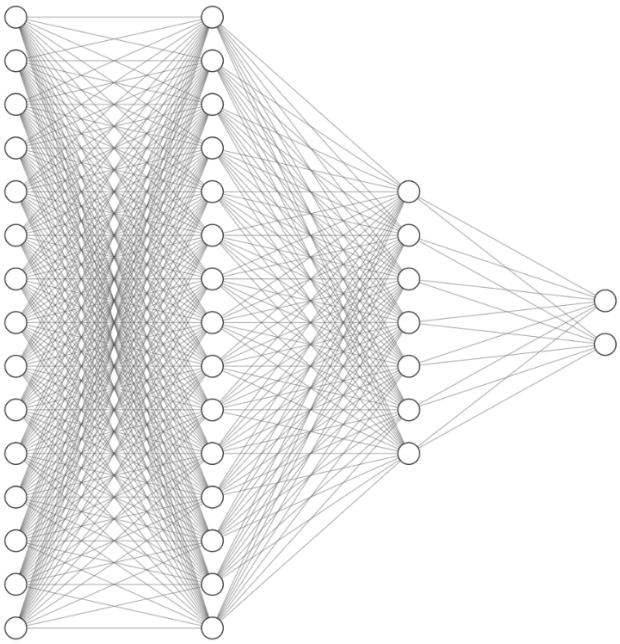


Figure 5.3 – Visualization of the layer structure of the Upper Bone Angles Deep Neural Network

The number of input nodes to a network is determined by the hand feature being validated. The inputs for each network are comprised of the corresponding feature data for each bone of each finger starting with the thumb. In the case of the upper bone angles network shown in Figure 5.3, the fifteen input nodes, are the angles between the proximal, intermediate and distal bones of each finger. In the case of the bone length network the inputs are the length of the metacarpal, proximal, intermediate and distal bones. This would then be followed by the length of the bones in the index finger, in the same order. This process is repeated for all the fingers of the hand, to provide all twenty inputs.

All four of the networks have two output nodes representing the two classifications: valid and invalid. The networks give each node a weight to indicate classification confidence, with the total of the two nodes adding up to

one. Each network uses multiple hidden layers and thus are classified as Deep Neural Networks (Ranjan, 2019; Nicholson, 2020). The layer sizes of each network follow the ‘rule of thumb’ for multilayer perceptron networks, with layers halving in size (Ranjan, 2019).

Each of the deep neural networks was trained using over 8,000 unique data samples consisting of half valid and half invalid hand data. The training data for each network was generated using a combination of live capture and programmatic deforming (according to the networks feature). Valid training data was generated by capturing live hand data while performing a series of movements and gestures at different orientations. During capture the hands were kept in clear view of an optical sensor.

To capture the live data, each frame of tracking data was processed, with the processing performed specific to the network the data was being captured for. In the case of the upper bone angles network, the angles between the bones were calculated. Algorithm 5.1 shows the process of calculating the angles between bones for the upper bone angles network.

Algorithm 5.1 - Pseudocode of the process for calculating the angles between bones for the upper bone angles network

```
Calculate Bone Angles for the Upper Bone Angles Deep Neural Network
Input: I : Input Hand

cross = Calculate cross product of the hands palmar and distal axes;
foreach Finger f in Hand do
    foreach Bone b in Finger do
        Skip the metacarpal bone;

        bone0dir = Current bone direction;
        bone1dir = Previous bone direction;
        bone0dir = ProjectOnPlane(bone0dir,cross);
        bone1dir = ProjectOnPlane(bone1dir, cross);

        angle = Calculate signed angle from bone0dir to bone1dir
         rotating around the cross axis;
        angle = Scale number from -180 to 180 range to 1 to -1;
        Store the angle in an array of bone angles for the hand;
    end
end
```

In the case of the bone length network, the length of each bone was calculated. The data calculated was mapped to a range of minus one to one and written to a feature specific log file. Figure 0.14 shows the code used to calculate and save the angle data for the upper bone angle network using the approach in Algorithm 5.1. Figure 5.4 contains an extract from a training data log produced for one of the deep neural networks, with the data normalized to the range of -1 to 1.

```
0.5,-0.0501574,-0.05202932,-0.05382819,-0.1545582,-0.1191251,0.1201349,
-0.1915136,-0.1361523,0.07578287,-0.2282825,-0.2709205,0.1348209,-0.14152,-0.07758912,
```

Figure 5.4 - Example data samples used for training one of the Deep Neural Networks (text wrapped for clarity)

In the case of bone length and spacing, valid training data was generated programmatically by randomly deforming an open hand within set thresholds. The valid data for bone length was generated by randomly adjusting the length of each bone to between 90% and 100% of the original length. The valid finger spacing data was generated by changing finger spacing randomly to between 90% and 100% of the original spacing. The valid length and spacing data used the 90-100% adjustment to provide small differences in the data that would be produced as the angle at which the fingers are viewed changes, during hand movements and interaction. Each row in the valid log files produced represents a single frame with the data comma separated. In the case of the bone length data, the length of each bone in each finger is stated, starting with the metacarpal bone of the thumb.

For each network, unique invalid data samples were generated by randomly programmatically deforming an open hand pose based on the network feature. The random values generated to deform the object were generated within pre-set un-natural thresholds based on the hand feature. In the case of bone length, each bone of each finger was randomly changed to between 0% and 60% of the original length. The invalid bone length range was determined by identifying the threshold at which the change in hand size results in a jarring user experience, even if the reduced size only lasts for a few frames. In the case of upper bone angles, the bones were bent backwards, rotating them within a set range (35-100 degrees). The invalid bend angles range was determined by identifying the natural angle limitations of the hand per the Leap tracking data, with a small tolerance added for user variance. Algorithm 5.2 shows a pseudocode implementation of the process to generate the invalid

data for the upper bone angles network. Each bone of each finger is randomly bent using the pseudocode shown in Algorithm 5.3 to rotate the bones. Figure 0.15 shows the code used to generate the invalid training data, resetting a stored hand to an open hand pose, with each bone of each finger randomly rotated to generate invalid data. As shown in Algorithm 5.2, once a sample had been created, the deformed hand was processed. During invalid data generation, the hands were processed using the same approach as the valid data, with network specific features calculated, mapped and stored in a separate network specific log file. The code implementation shown in Figure 0.14 for calculating the upper bone angles was used to create both the invalid and valid log files for the upper bone angle network.

Algorithm 5.2 - Pseudocode of the invalid data generation process for the upper bone angles network

```
Generate Invalid Data for Upper Bone Angles Deep Neural Network
for n ← 0 to N do
    validHandData = Create empty HandData object;
    openHand = Create a clone of the open hand pose;

    foreach Finger f in Hand do
        RotateBone(f,Random.Range(minAngle,maxAngle), 1,
         openHand.RadialAxis);
        RotateBone(f,Random.Range(minAngle,maxAngle), 2,
         openHand.RadialAxis);
        RotateBone(f,Random.Range(minAngle,maxAngle), 3,
         openHand.RadialAxis);
        RotateBone(f,Random.Range(minAngle,maxAngle), 4,
         openHand.RadialAxis);
    end
    GenerateSavedFrame(openHand);
      if n % 1000 == 0 then
        SaveFrames(savedFrames, filename);
        savedFrames.clear();
      end if
end
SaveFrames(savedFrames, filename);
savedFrames.clear();
```

Algorithm 5.3 - Pseudocode to rotate a bone by an angle around an axis

```
Rotate A Bone By An Angle Around An Axis
Input:  f : Current Finger, a : Rotation Angle, b : Bone Index, x :
 Rotation Axis
Output:  newFinger : Finger
newFinger = Create a new finger object and clone the current finger;
  if b > f.bones.length then
    b = b - 1;
    direction = Calculate current bones direction using the end position of it
      and the previous bone;
    P = RotatePointAroundPivot(Current bone end, current bone start,
      axis * angle);
    Set the end position of the current bone to P;
    Recalculate the bones direction using the start and new end position;
  else
    direction = Calculate current bones direction using the end position of it
      and the previous bone;
    P = RotatePointAroundPivot(Current bone end, current bone start,
      axis * angle);
    Set the end position of the current bone to P;
    Recalculate the bones direction using the start and new end position;

    for n ← b + 1 to NumBonesInFinger do
      direction = Calculate direction using end of current and previous bone;
      Set the start of the current bone using the end of the previous bone;
      P = Calculate new end position by applying the direction vector
        to current bones start position;
      Set the end of the current bone to the new end position P;
    end
  end if
```

Due to the degrees of freedom hand tracking enables, the number of potential invalid data samples is substantial. Thus, training a network to classify all of them is not feasible. As previously discussed, the networks used in the hand validation system are designed to learn the natural movement range of the human hand. Both the valid and invalid training data generated explore these limits. The valid using live capture and programmatic deforming outside the natural movement range for the invalid data. This enables the networks to learn the natural thresholds within which data is valid or invalid. Furthermore, this approach ensures the network is capable of validating hand data with high accuracy without training on all possibilities.

The machine learning approach of generative adversarial networks (GAN's) can be used to generate fake data. The model of GAN's is typically used in image generation such as generating human faces. A classifier is used in the GAN model, trained to classify the fake data generated by the generative network and real data, as either real or fake. For example, determining whether a photo of a human face is real or fake. As the networks train, the generative network becomes more effective at generating fake data. While, the classifier becomes more effective at classifying the fake data (Beckett, 2017). To generate the training data using a GAN would have taken considerable time to produce the high-quality data needed. In addition, the data generated would be classified by the classifier as real or fake as opposed to whether the hand features are within natural movement thresholds. Furthermore, the previously discussed invalid data generation process enables the movement ranges to be controlled. This ensures the networks are trained on the natural movement ranges by generating data at the thresholds.

Both the valid and invalid training data for each network were written to separate text files as previously discussed. The data classifications were also written to separate text files (one for each; 0's for invalid and 1's for valid). Thus, prior to use in the neural networks, the valid and invalid training data for a respective network were merged. The resulting text file contained all the invalid samples (one per line) followed by the valid samples (one per line). The same process was followed for the classifications/answers. The two text files containing the data samples and classifications respectively, were then identically shuffled to mix the invalid and valid data samples. The use of the same random shuffle ensured the classification labels were matched to the

appropriate data sample. This produced the training data and classification data for the corresponding network. The data was shuffled to ensure the networks were trained and tested on a mixture of invalid and valid data samples.

Each network loads the training and classification data from the corresponding text files produced with each line in the files representing a single data sample. Each line of the training data is processed, splitting the comma separated values, resulting in an array of arrays. Each record in the classification data was loaded into a one-dimensional array. The resulting arrays were then sliced, with the first 20% removed for testing and the remaining 80% used for training. Finally, the network is created before compiling and training the model. Figure 0.16 shows the python code used to create, train and export a deep neural network model.

Figure 5.5 and Figure 5.6 show graphs of the upper bone angle network model's categorical accuracy and loss respectively, over the three epochs trained. Analysis of Figure 5.5 shows the models accuracy to significantly increase between the end of the first and second epoch. This can also be seen in Figure 5.6 with the loss significantly decreasing over the same period. While, the variation in potential hand poses and thus input data is significant, the network is designed to classify hand data to one of two possible classifications. The networks are designed to learn the natural movement range of their respective feature. Therefore, it can be hypothesized that the significant improvement between epochs is due to the network learning the single threshold value either side of which the data is either valid or invalid. As the upper bone angle network is designed to the learn the natural bending angle

range, the network can identify a threshold value at which the angle is to large and is thus invalid. The figures show further improvement in the model performance after the third epoch but to less of an extent than the previous epoch.

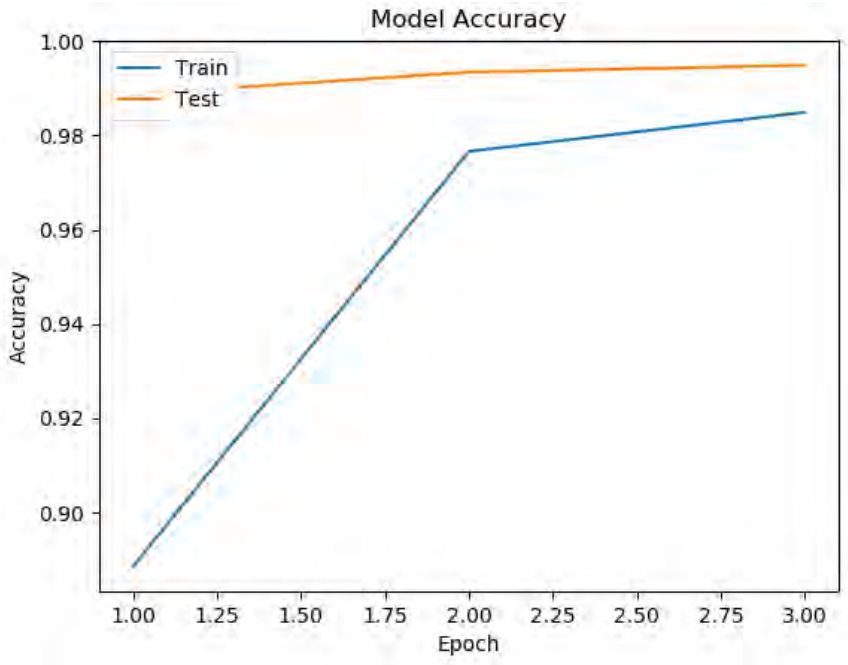


Figure 5.5 - Graph plotting Model accuracy against Epoch

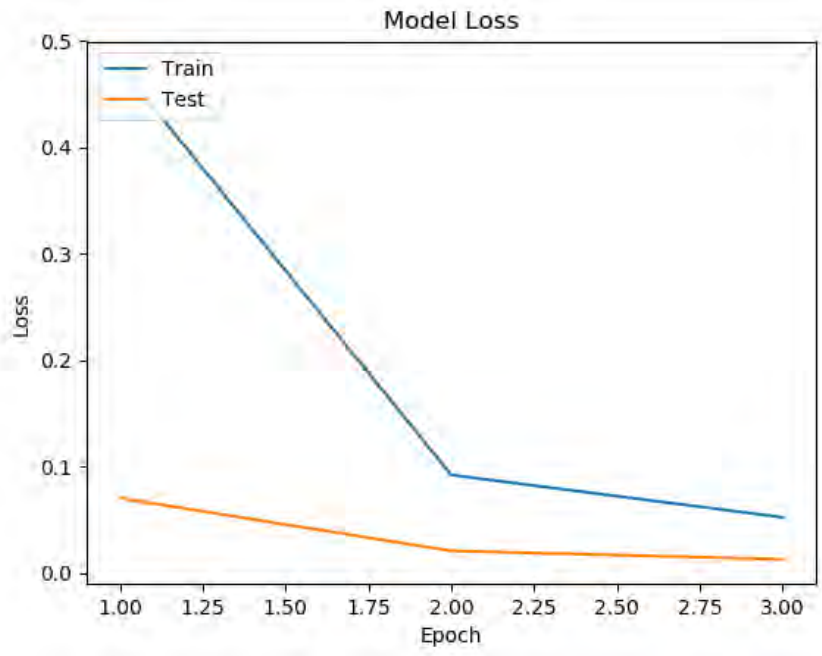


Figure 5.6 - Graph plotting categorical loss against epoch

Once fully trained, the models were saved and exported to '.pb' files for use within the MOT system using TensorFlowSharp, a C# based .NET binding to the TensorFlow library. The TensorFlowSharp library enabled, the exported models to be loaded and hand data evaluated against them. Figure 0.17 shows the code used to load a neural network model at runtime.

5.1.2 – Hand Validation Process

During the simulation, hand tracking data received is processed to calculate the necessary input data for the deep neural networks such as bone angles and length. The network models are loaded using TensorFlowSharp with the calculated data fed in as the inputs. The data is then validated by the networks and the confidences of the networks returned to Unity to perform the weighted average and classify the hand data. Figure 5.7 shows an overview of the validation process with the tracking data received by the validation system and pre-processed to calculate the input data for the networks. The input data is then sent to the corresponding network for analysis with the network's weights averaged using a weighted average to classify the hand data.

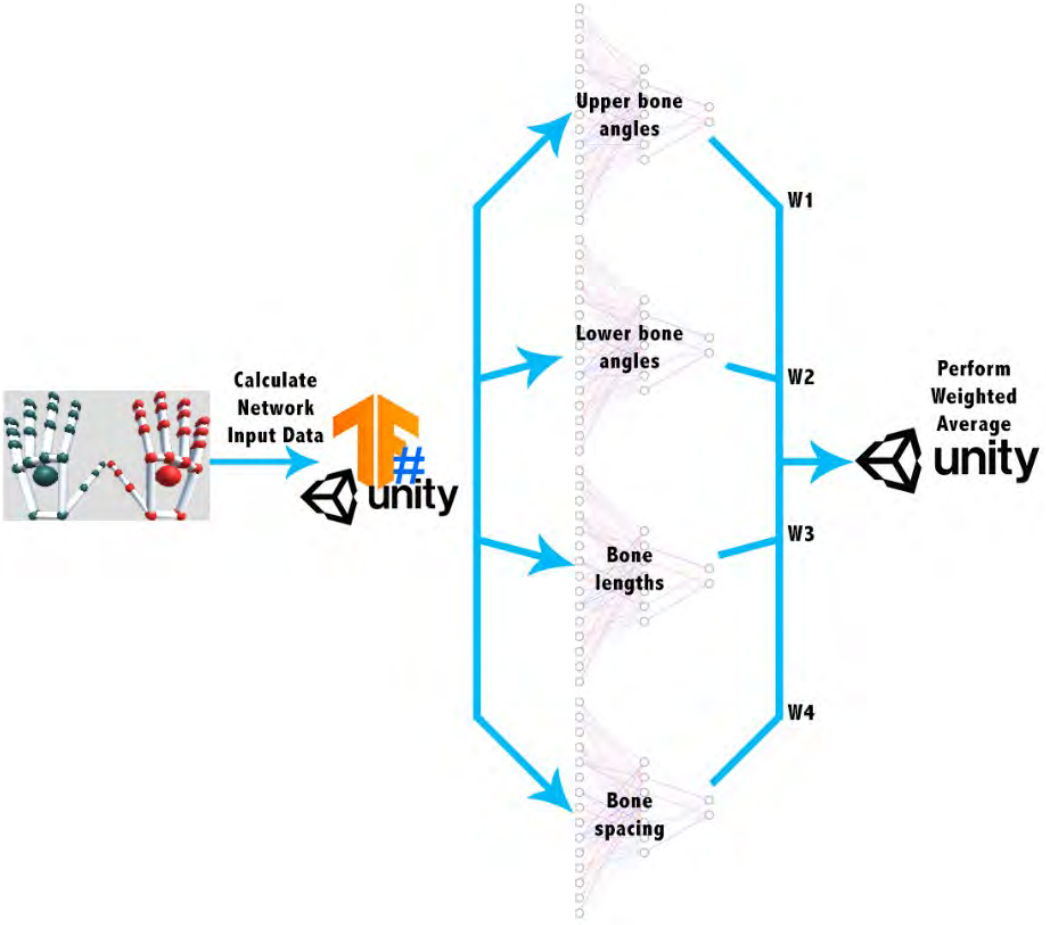


Figure 5.7 - Overview of the Validation Process

The validation system uses a static instance so it can easily be accessed anywhere within the MOT system. By default, the validation system uses a minimum confidence of 85% for valid classification, however this can be overridden. The validation system starts by calculating the input data for each network using the hand data to be validated. The input data is calculated using the same approach used for the generation of the training data samples, discussed in section 5.1.1. As previously discussed, in the case of the upper bone angles network, the angle between each bone of each finger is calculated. In the case of the bone length network, the length of each bone in

the hand is calculated. Figure 0.18 shows the code used to calculate the bones angles for validation using the approach shown in Algorithm 5.1.

Once the input data for each network has been calculated, the data is run through the corresponding network model for classification. Each network returns a two-dimensional float array containing the confidences for the valid and invalid classification. If a networks valid confidence is greater than both the invalid confidence and the minimum requirement of 85%, the valid confidence is returned. However, if either the invalid confidence is higher or the required confidence is not met, a confidence of zero is returned. Figure 0.19 shows the code used to run calculated hand data through a deep neural network model for validation.

Once the hand has been processed by each of the networks, the overall confidence is calculated using a weighted average of the valid classification confidences. The overall confidence is calculated using the network weighting detailed in Table 5.1 and the equation shown in Equation 5.1. Equation 5.1 shows the equation used to calculate the overall hand confidence, where ***N*** represents a networks confidence and ***W*** represents the networks weight.

Equation 5.1 - Equation used to calculate the validation system confidence in the validity of hand data

$$C = (N^1 * W^1) + (N^2 * W^2) + (N^3 * W^3) + (N^4 * W^4)$$

If the overall confidence meets the minimum confidence (85% by default) the hand is considered valid, otherwise the hand is classified as invalid. Figure 0.20 shows the code that validates the hand through the four networks and performs a weighted average of the resulting validity confidences.

5.2 – Occlusion Detection System

One of the biggest challenges with optical-based tracking is detecting and handling occlusion of user's hands to provide a consistent user experience. During the phase one testing, while reduced using multi-sensor aggregation, occlusion was still found to be a problem within the MOT system. This resulted in users having difficulty with two handed interactions; specifically, the pipette dropper. The main problem was the tracking instability and potential loss of tracking caused by occlusion. This negatively effects the user experience and introduces inconsistency. To address this issue, an occlusion detection system was developed and added to the extended MOT system.

The Occlusion detection system uses custom geometry and a custom raytracing-based approach to detect intersections. The system analyses hand data and detects hand-to-hand occlusion, such as one hand covering another. The occlusion detection system uses axis-aligned bounding boxes (AABB) and the Möller–Trumbore ray-triangle intersection algorithm (Möller & Trumbore, 1997; ScratchAPixel, 2019) for high performance computations and real-time analysis of hand data. The system generates custom cube-based geometry around each bone of the hand and the palm. This enables the system to test for direct intersections between hands such as if the user has their fingers interlaced. Figure 5.8 shows a user's hands interlocked, a gesture which significantly reduces the accuracy of tracking data by occluding parts of the fingers.

Figure 5.8 - Photo of Hands with fingers interlocked

The occlusion detection system starts by generating the custom rectangular-based geometry around each bone and the palm. The direction and length of each bone is used to calculate the corner positions of the bones bounding box. To calculate the corner positions, two vectors are calculated, the first out of the top of the bone (relative to palm direction) and the second out of the side. The upwards vector is calculated by rotating the bones direction 90-degrees around the radial axis (Figure 3.14). The side vector is calculated by rotating the upwards vector 90-degrees around the bone direction. Figure 5.9 shows the two bone vectors calculated to determine the corner positions of the bounding geometry: specifically, the upwards vector (green) and the side vector (yellow).

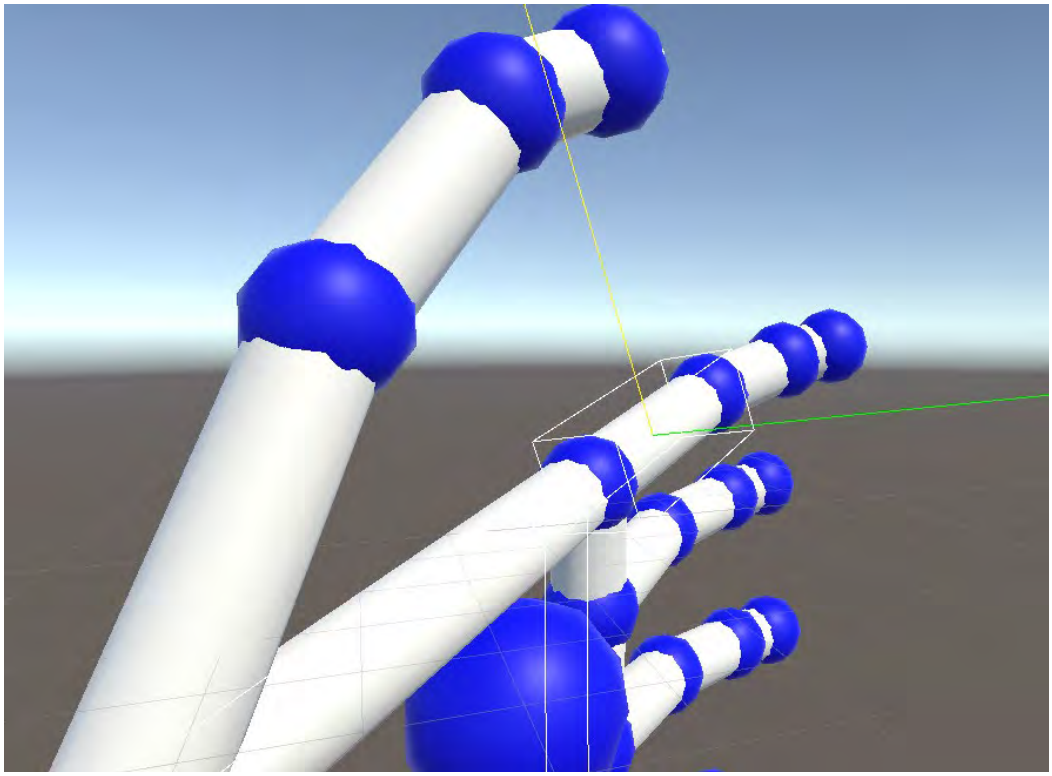


Figure 5.9 - Bone vectors calculated to calculate the custom bone geometry (yellow = side, green = up)

Once the vectors have been calculated the corner positions of the geometry are calculated using the start and end positions of the bone with the calculated vectors. The finger thickness is used to set the width and height. Algorithm 5.4 shows the process of calculating the four corner positions at one end of a bone. This process is applied to both the start and end position of each bone. Figure 0.21 shows the code used to calculate corner positions using the bones start position and end position with the calculated vectors shown in Figure 5.9. The start and end positions of the metacarpal bones are used to create the palm geometry. Figure 5.10 shows the resulting bone geometry with the palm geometry generated rendered in red and the centre positions marked with black markers.

Algorithm 5.4 - Pseudocode of the process to calculate the corner positions at one end of a bone

```
Calculate The Bound Corner Positions At One End Of A Bone
Input: S : Side Vector, U : Up Vector, P : Centre position at one end
 of the bone, R : Finger Radius

topLeft = P - (S * R) + (-U * R);
topRight = P + (S * R) + (-U * R);
bottomLeft = P - (S * R) + (U * R);
bottomRight = P + (S * R) + (U * R);
```

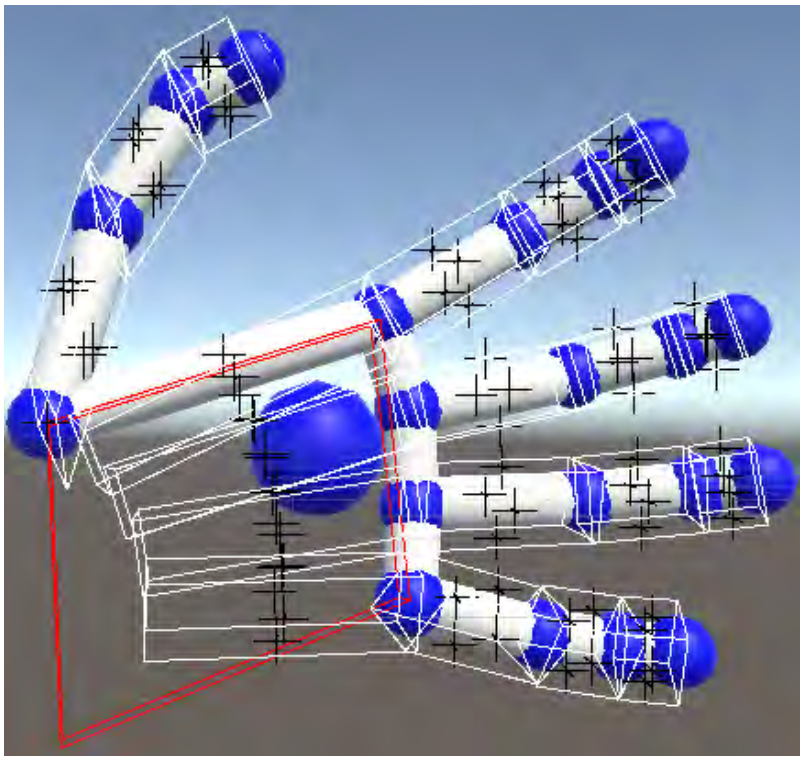


Figure 5.10 - Complete Hand Bound geometry rendered

Figure 5.11 shows the custom geometry generated around the bones and palm of a virtual hand representation, with the occluded bone geometry rendered in red.

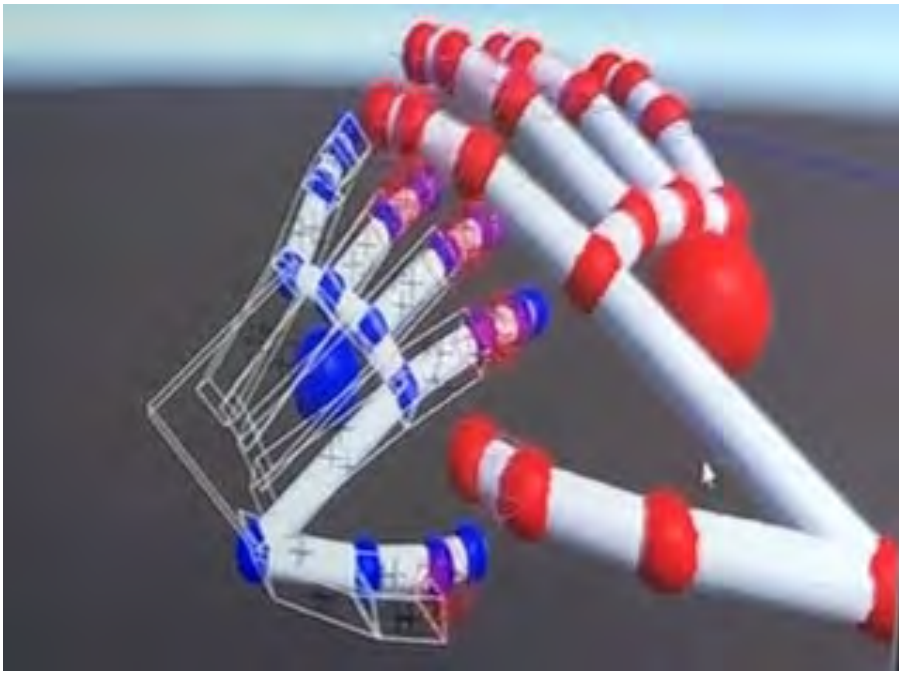


Figure 5.11 - Custom Cube-based Geometry generated around the virtual hand's bones and palm

To detect occlusion a test is performed using raytracing to determine if one hand is positioned in front of another, thereby either partially or fully occluding the other hand. A boxcast as shown in Figure 5.12 is projected from the centre of the HMD to the centre of each bone in the target hand. If the ray intersects with the opposing hand, the intersection distances for the two hands are calculated with a shorter distance for the opposing hand resulting in the target hand considered occluded.

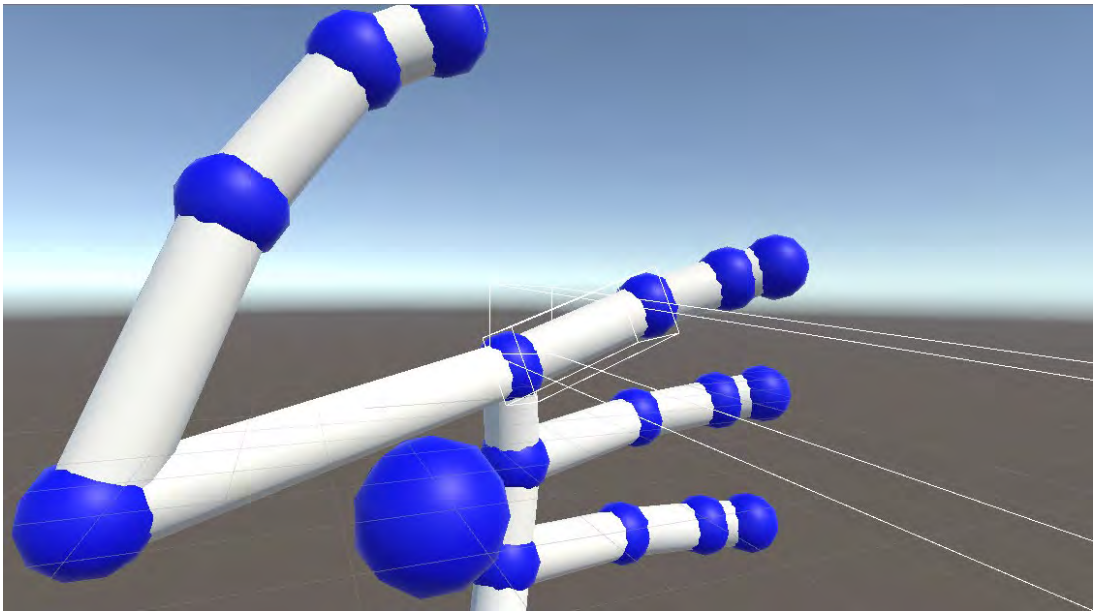


Figure 5.12 - Boxcast from the Occlusion Detection system rendered

To detect an intersection with the other hand, each of the opposing bones geometry is iterated, checking for a boxcast intersection with the hit distance returned. The bone geometry follows the Axis Aligned Bounding Box (AABB) approach, defining a set of parallel lines to the coordinate axis by its minimum and maximum points. A series of LineAABB intersection tests are performed using each of the four longest edges to find any intersection points and as a result, the minimum intersection distance. Figure 0.24 shows the code used to perform the AABB intersection using each of the corner points and the centre positions. Equation 5.2 shows the equation for expressing a ray.

Equation 5.2 - Equation showing how a ray can be expressed

$$P = O + D * t$$

In Equation 5.2, ***P*** is a position on the ray, ***O*** corresponds to the origin of the ray, ***D*** its direction and the parameter ***t*** can be any real value (negative, positive or zero). By changing the t-value, any point of the line defined by the

ray's position and direction can be found. The line equation for the x-axis component of the bounding volumes minimum bound position (**t0**) can be written as **y = B0x**, where **B0x** corresponds to the x-axis position of the minimum bounds position (lower bottom left) of the geometry**.** The axis intersection point can be calculated using **t0n = (B0n – On)/Dn**, where **n** is the value for the given axis. The same equation can be applied to the maximum bound position (***t1***).

To calculate the intersection distances, the raycast equation shown in Equation 5.2 is used, calculating the ***t0*** and ***t1*** values for each axis. Figure 0.23 shows the LineAABB intersection function handling each axis separately. The calculated min and max values are then compared to both each other and the current minimum and maximum values. If the t-value of a t0 intersection point on a given axis is greater than the current maximum t-value for a t1 axis, the ray does not intersect with the box. Figure 5.13 shows how evaluating t0 and t1 values can determine a ray does not intersect.

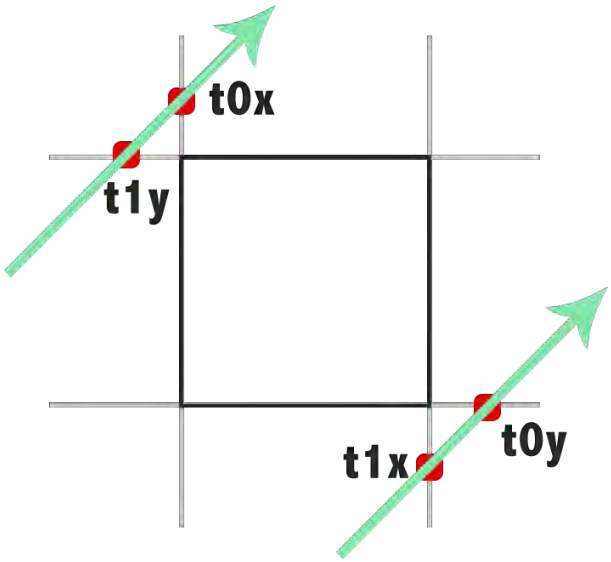


Figure 5.13 - Diagram showing how a t0 intersection point being greater than the t1 maximum indicates a ray does not intersect

However, if the ray intersects the box, the t-values for the three axes at both the minimum bound position (***t0***) and maximum bound position (***t1***) are compared to find the largest minimum (***tMin***) and smallest maximum respectively (***tMax***). The current min and max values are then updated and compared again to ensure the minimum does not exceed the maximum. If the minimum does not exceed the maximum an intersection has occurred on the given axis. Figure 0.22 shows the ClipLine function code used to calculate the intersection bounds. If an intersection occurs on all three axes, the intersection point is calculated using the equation shown in Equation 5.3 and the previously calculated t-values.

Equation 5.3 shows how the ray equation shown in Equation 5.2 can be rearranged to calculate the intersection point (***T***) on a line based on its origin (***O***), direction (***D***) and either it's minimum or maximum point (***P***). The equation can be used to calculate the positions where a line intersects the geometry planes parallel to the X, Y and Z axis.

Equation 5.3 - Equation for calculation the intersection point on a ray based on its origin, direction and minimum point of the bounding volume

$$T = \frac{(P - O)}{D}$$

Once the intersection point has been calculated, the distance is calculated using the magnitude of a vector between the ray's origin and the intersection point. Figure 0.23 shows the intersection distance calculation, firstly by calculating the direction through subtracting the minimum from the maximum position. The resulting vector is then applied to the minimum position with the minimum clip distance as the direction vector length to calculate the intersection point. Finally, the intersection distance is determined by

calculating the distance between the minimum bound position and the previously calculated position.

The intersection distance calculated is compared against the current minimum intersection distance and used to update it, should it be less than the current minimum. The previously discussed process in then repeated for the other bounding points of the current bone to calculate the overall minimum distance. Figure 0.24 shows the function used to calculate the minimum intersection distance for each point and the overall minimum as a result.

Once the minimum intersection distance with the other hand has been calculated. The intersection distance calculation is used to calculate the minimum intersection distance between the boxcast and the current bone being processed. As with the opposing hand a series of LineAABB intersection tests are performed to find the minimum intersection distance.

The two intersection distances are then compared with the bone classified as occluded if the intersection distance of the opposing hand is less than the intersection distance to the current bone. If the bone is occluded a counter is increased. Once all the bones have been processed the overall percentage coverage is calculated using the counter, with the hand considered occluded if a threshold value (33% by default) is exceeded. A threshold of 33% was selected based on developmental testing to determine the point at which the data becomes unstable. To determine the threshold, a series of occlusions were manually performed whilst displaying the percentage coverage. The occlusions were performed at different speeds to determine the maximum

coverage that could be supported whilst still ensuring a tolerance as a safety net, to ensure a consistent experience.

Due to factors such as noise, hand pose and natural body movement, there are edges of the tracking space where the hands can continuously switch between an occluded and non-occluded state. To provide a more consistent experience, time-based detection was added that requires 0.5 seconds of un-occluded data to return to the un-occluded state. The time detection threshold was determined by exploring a range of tolerances to identify the optimal time that ensured the hand is un-occluded and stable, whilst minimizing the time at which live tracking data is unused. The time-based detection ensures data stability and creates a smooth experience when performing gestures such as overlapping hands, whilst not introducing any perceivable ‘lag’ or delay upon returning to the un-occluded state. Hands marked as occluded are replaced with previously valid hands to prevent invalid or incorrectly positioned and rotated hands being presented to the user. Furthermore, during periods of occlusion, the position and rotation of tracked hands can vary significantly resulting in sudden ‘jumps’, therefore the hands are controlled using the Vive tracking system to ensure consistency and stability.

5.3 – Stability Detection System

Optical-based trackers such as the Leap Motion use camera technology and software analysis to track a user’s hands and as a result can suffer with stability issues due to factors such as noise. In addition, the tracking software used can often introduce instability upon recognising and tracking a hand, as the software refines the position and orientation the hand. The instability upon resuming tracking is particularly problematic during interactive VR simulations

due to the hand frequently moving in and out of the sensor tracking space. The phase one design of the MOT system used an inferred hand pose whilst tracking data was not available. However, once tracking was resumed the system would switch to the live data. This could result in the virtual hand experiencing a small jump or pop as the Leap API refined the hand position.

During developmental testing, the instability due to the sensor drivers refining the hands position was found to cause the virtual hand representation to experience visible ‘jumps’ or ‘pops’, emphasized during interpolation. As a user’s hand entered the detection space, the interpolation system detailed in section 5.4, would begin interpolating to smooth the transition to the live tracking data. However, the virtual hand would often ‘jump’ backwards momentarily due to the instability of the target data. Visual analysis of the tracking data showed the hand data experiences substantial changes in the first few seconds of tracking as the hands position and pose is refined. Figure 5.14 visualizes the substantial changes hand tracking data can experience during position refinement. In the figure, the blue hand represents a tracking frame with the purple hand representing the previous frame.

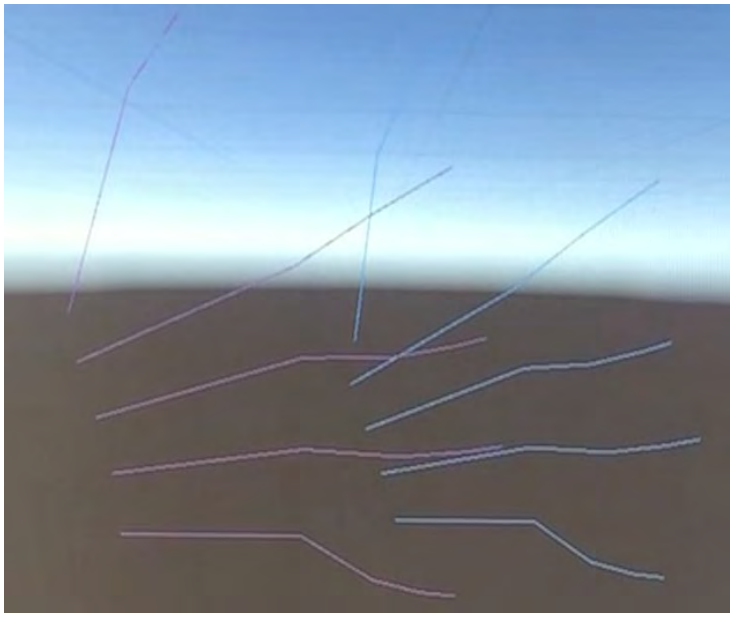


Figure 5.14 - Diagram illustrating the positional changes tracking data can experience

The Stability Detection system was designed to analyse incoming hand data and detect periods of tracking instability, toggling the interpolation system accordingly. In cases such as the user's hands entering the tracking space, the system will trigger the interpolation system to begin interpolating to the hand data controlled using the motion controllers. Once the incoming tracking data has stabilized, the interpolation target is switched, thus providing the user with a seamless transition to a live tracked virtual hand representation.

To determine data stability, the tracking data is compared against the previous tracking data to calculate the percentage difference. Each bone of each finger in the tracking data is processed, calculating the percentage difference between both the start and end positions in comparison to the previous data. Algorithm 5.5 shows the process of calculating the percentage difference between two hands.

Algorithm 5.5 - Pseudocode of the process to calculate the percentage difference between two hands

```
Calculate Percentage Difference Between Hands
Input:  P : Target Hand, I : Input Hand
total = 0;
possibleTotal = 0;
foreach Finger f in Hand do
    foreach Bone b in Finger do
        total += PercentageDifferenceBetweenVectors(I.b.nextJoint,
         P.b.nextJoint);
        total += PercentageDifferenceBetweenVectors(I.b.prevJoint,
         P.b.prevJoint);
        possibleTotal +=200;
    end
end
total += PercentageDifferenceBetweenVectors(I.palmPosition,
 P.palmPosition);
possibleTotal +=100;
result = RangeConvert(total,0,possibleTotal,100,0);
```

To calculate the percentage difference between two vectors, the axes are treated separately with the difference in the float values calculated using the equation in Equation 5.4. The three values are then totalled and mapped from a maximum of three hundred to a maximum of one hundred, with the result added to a total. To scale the total, Equation 3.2 detailed in section 3.4 is used. Equation 5.4 shows the equation used to calculate the percentage change between two numbers. Figure 0.25 shows the code used to calculate the percentage difference between two hands. Figure 0.26 shows the Vector3 and float percentage difference implementation.

Equation 5.4 - Equation used to calculate the percentage change between two numbers

$$\Delta = \left(\frac{A - B}{B}\right) * 100$$

Once all bones have been processed the total percentage difference is scaled using the equation in Equation 3.2 to out of one hundred. The resulting percentage difference is then compared against a set threshold (2%). A range of tolerances were explored by visually analysing the change in position as the virtual hand representation changes from the final interpolation position to the potentially stable live hand data. An open hand pose was used with the tolerance used to determine when the virtual hand representation could switch source. The tolerance was increased until the change was detectable. A 2% threshold was chosen as it provided the maximum value at which the change was not visually perceptible. If the 2% threshold is exceeded the tracking data is considered unstable. Thus, the current data is stored for comparison in the next update cycle and the stable data timer is reset to zero. However, if the threshold has not been exceeded, the stable data timer is compared against a set threshold (2 seconds). A range of time tolerances were explored to determine the minimum period of time after which the data can be considered to have stabilized if the changes remain within the set threshold. The two second threshold was chosen as it provided the minimum value at which the hand data can be consistently classified as stable.

If the data has been classified as stable continuously for more than two seconds, the data is considered stable and the interpolation system is re-targeted to transition the virtual hands to the live data. The entire process is then repeated for the other hand. Figure 0.27 shows the code used to calculate the percentage difference and classify the data based on the stability timer.

5.4 – Interpolation System

One of the problems with optical-based trackers is that they are subject to noise which can interfere with the tracking and cause instability. Furthermore, an inferred hand may not exactly match the stabilized live data as detected by the stability detection system in section 5.3. The phase one design of the MOT system used inferred hands to provide virtual hand representations when tracking data was not available. However, the system was unable to detect and handle small jumps/pops in the tracking data due to factors such as noise. In addition, the system was unable to synchronize the transition between the sensor sources. Furthermore, the system did not pre-emptively interpolate which could result in visible hand movements while its position was refined before the hand pose interpolation started. Therefore, a custom standalone interpolation system was designed to smooth inconsistencies in tracking data and transitions between data sources.

The stability detection system as described in section 5.3 enabled the MOT system to detect jumps/pops, substantial changes in hand position and general inconsistencies in tracking data. To smooth detected inconsistencies such as noise and glitches along with smoothing transitions between poses and data sources, an interpolation system was developed. The interpolation system ensures virtual hand representations move smoothly providing the user with a consistent experience by generating the transitional poses between two given hand poses. Figure 5.15 shows a start (blue) and end (red) pose with example transition poses (green and yellow) the interpolation system would generate to smooth the transition. During a simulation, the interpolation system produces significantly more than two poses to ensure a

smoother transition, with the number of poses produced dependent on factors such as movement speed and pose difference.

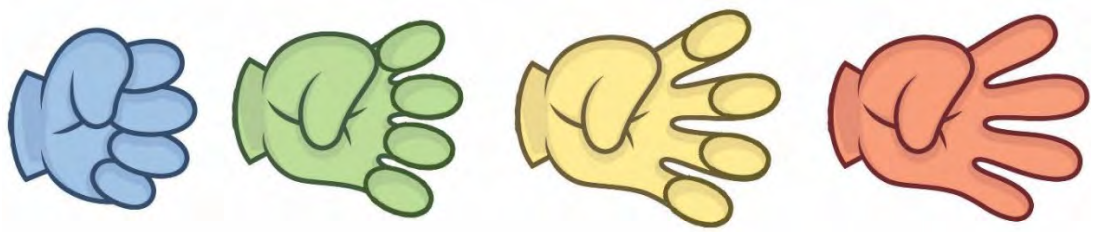


Figure 5.15 - Diagram showing the poses generated by the interpolation system

The interpolation system was designed to work alongside existing MOT sub-systems such as the mesh conformation system, providing a smooth animation/transition between the live and conformed hand pose. The interpolation system interpolates hand and finger positions separately, enabling fingers to be interpolated at higher speeds so live tracking can quickly be resumed, minimizing any sense of lag.

To prevent the user experiencing visible ‘pops’ in the virtual hands position or any visible change in hand movement speed, the interpolation system begins interpolating once the virtual hand is within seventy degrees of the user’s gaze direction. This ensures that the virtual hand representation does not get ahead of the live tracking position. In addition, it enables smooth synchronization between the virtual hand representation and the live tracking data. Furthermore, it ensures the user does not experience any change in movement speed or feeling of ‘lag’ by seeing the interpolation process start. As the system is already interpolating to the Vive tracking controlled hand position, the interpolation target can be changed to the live tracking data once

it has stabilized, providing a smooth transition to the live data. The tip of the middle finger is used to calculate the angle to the user's gaze direction as it is what the user will see first. If the angle exceeds seventy degrees, the interpolation stops as the user is unable to see the virtual hand. Figure 0.28 shows the code used to determine whether the Vive tracking controlled virtual hand is within the required range to automatically begin interpolating.

The interpolation speed used was dynamically adjusted based on the user's hand velocity as recorded by the Vive tracking system. To minimize any feeling of 'lag' whilst preventing the interpolation from completing too quickly, minimum (3) and maximum (12) interpolation speeds were used. The minimum interpolation speed dynamically changed, with a speed of zero used upon resuming tracking until the data had stabilized and the virtual hand had been synchronized with the live data. Limiting the maximum interpolation speed prevented the interpolation completing within a few frames during a fast gesture. This ensured the tracking data had time to stabilize, preventing a visible 'pop' in the virtual hands position. The dynamic adjustment of the minimum interpolation speed ensured a smooth experience upon resuming tracking whilst minimizing the effect of 'lag' on the user's experience. Figure 5.16 presents a visualization of the interpolation process, showing the live data (blue hand) and the interpolated hand (purple). In the figure, the user's hand has entered the tracking space and stopped moving.

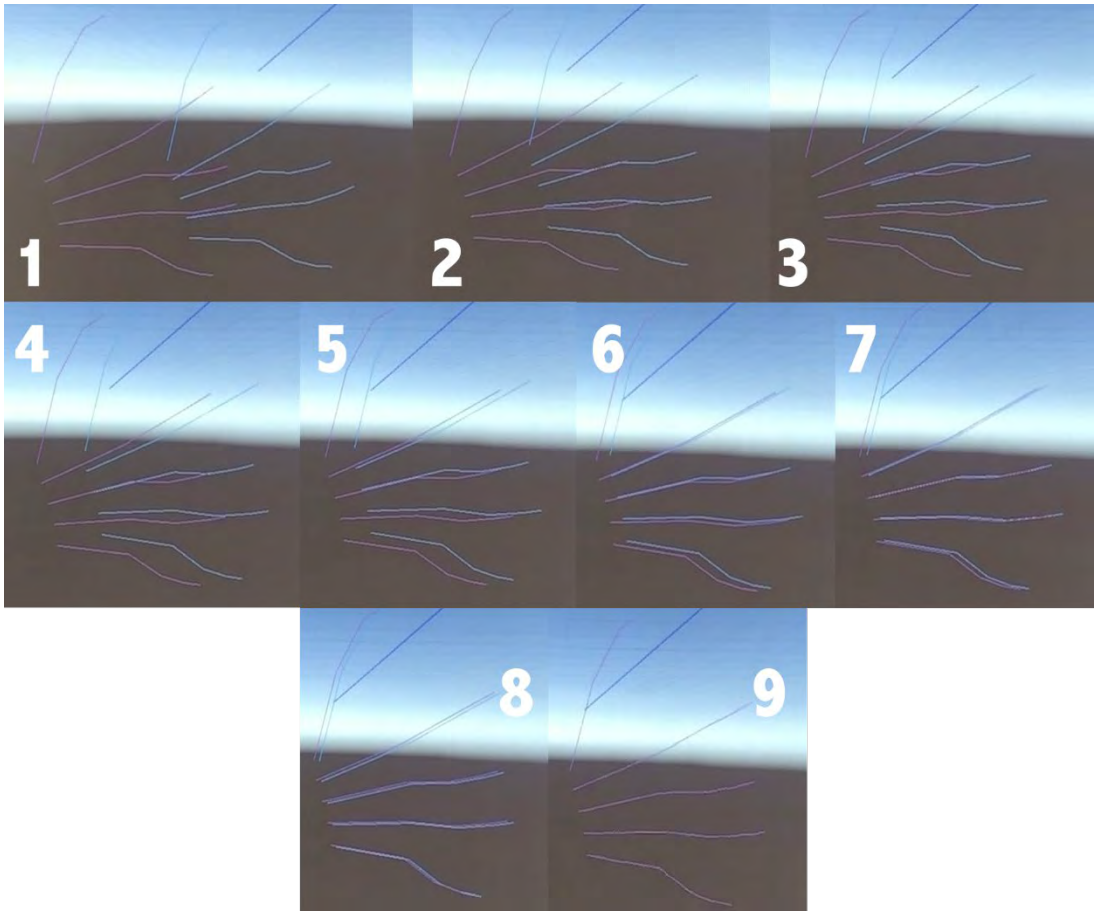


Figure 5.16 - Visualization of the interpolation process, the blue hand represents the live data with the purple the rendered interpolated hand

To prevent continuous interpolation, the system tests whether interpolation is required. To detect jumps/pops and trigger the interpolation, the system starts by comparing the hand position of the live tracking data against the current hand rendered. The distance between the palms of the two hands is compared against a set threshold (0.2 by default). A range of tolerances were explored to determine the maximum tolerance at which a change in position was not perceptible. To test tolerances an open hand was moved out of sensor tracking range and then moved to and held at the centre of the user's view. The virtual hand representation was visually analysed to detect any jumps as the virtual hand source switched from the interpolated to live tracking data. The tolerance

was increased until the change became perceptible to identify the maximum value. The 0.2 threshold was chosen as it provided the maximum value at which the change was not perceptible to the user, irrespective of movement speed.

If the threshold is exceeded a jump is detected. If a jump is detected, both the fingers and hand are checked to determine whether they require interpolation to smooth out any noise in the data. Figure 0.29 shows the code used to detect jumps, using tests to determine whether the fingers and/or hand position require interpolation.

The first test evaluates the fingers, processing each bone of each finger and checking it is within a required radius of the same bone on the target hand. The start and end positions of each bone are compared to ensure they are within a set radius of each other. The radius used is the radius of a virtual finger representation multiplied by a factor of 1.5 ensuring a high level of precision without over-interpolating. If the radius is exceeded, the percentage difference is returned on the scale of zero to one, with zero returned if the radius is not exceeded. The returned radius scores are totalled and the percentage of the maximum score (40 based on all bones exceeding the radius) is calculated with a lower percentage indicating the hands are closer together. The result is then reversed so 100% indicates a perfect alignment and compared against the set threshold of 99.8% to determine whether interpolation is required. The finger interpolation threshold was determined using the same approach for determining the jump detection threshold previously discussed. The interpolation process was triggered by exceeding the tracking range, with the virtual hands fingers visually analysed to detect

the transition between virtual hand representation data sources. The threshold was decreased from 100% until the change was perceptible to find the minimum threshold of 99.8%. If the set threshold of 99.8% is met the hand is classified as close enough that the fingers no longer require interpolation. Algorithm 5.6 shows the process for determining whether the fingers require interpolation. Figure 0.30 shows the code used to calculate the similarity between the finger poses of two virtual hands to determine whether interpolation is required.

Algorithm 5.6 - Pseudocode of the process for checking whether finger interpolation is required

```
Check Finger Interpolation Is Required
  Input:  P : Target Hand, I : Input Hand, R : Radius, T : Percentage
   Threshold
  temp = 0;
  foreach Finger f in Hand do
      foreach Bone b in Finger do
          temp += WithinRadius(b.nextJoint, target.b.nextJoint, radius);
          temp += WithinRadius(b.prevJoint, target.b.prevJoint, radius);
      end
  end
  S = Calculate temp percentage of maximum;
  A = 100 - S;
    if A >= T then
      Fingers do not require interpolation;
    else
      Fingers require interpolation;
    end if
```

The second test evaluates the current hand position against the position of the target hand using 1.5x finger thickness as the radius. The returned value is then converted to a percentage and subtracted from one hundred to invert it. The final value is then compared against a set threshold (99.8% by default) with values not reaching the threshold triggering the hand position to begin interpolating. Figure 0.31 shows the code used to determine whether two

vectors are within a given radius of each other, returning zero if within radius or the percentage increase if exceeded. The 99.8% threshold was determined during developmental testing to identify the point at which the transition between interpolated and live hand data could not be detected. To determine the threshold, a tracked hand was moved in and out of the tracking range to trigger the interpolation system. The virtual hand was visually analysed upon resuming tracking to determine whether the change was perceptible. The threshold was adjusted until the transition and any resulting ‘pops’ or ‘jumps’ were not perceptible to the user.

To interpolate fingers, each bone of each finger is processed, calculating the offset between the start position of the corresponding bone on the target hand and the target hands palm position. The calculated offset is then applied to the current hands palm position to calculate the new start position for the current bone. Linear interpolation is then performed from the current start position of the current bone to the calculated position using the current interpolation speed and Equation 5.5. Figure 0.32 shows the code implementation of the linear interpolation equation shown in Equation 5.5. Equation 5.5 shows the equation used for linear interpolation ***C*** with ***A*** representing the original vector, ***B*** as the target vector and ***T*** as the interpolation amount.

Equation 5.5 - Equation for Linear Interpolation of Vectors

$$C = (A + (B - A) * T)$$

The same process is then applied to the end position of the bone, before recalculating the bones direction and centre position. The adjustment of the finger positions based on the vector offsets to the palm position also rotates the current hand to align with the target hand. Algorithm 5.7 shows the process

for interpolating fingers to a target pose by interpolating the start and end position of each bone. Figure 0.33 shows the finger interpolation, with the new position resulting from linear interpolation to the calculated position.

Algorithm 5.7 - Pseudocode of process for interpolating fingers to a target pose

```
Interpolate Fingers To A Target Pose
  Input:  P : Target Hand, I : Input Hand, T : Interpolation Speed

  foreach Finger f in Hand do
      foreach Bone b in Finger do
          offset = P.f.b.nextJoint - P.palmPosition;
          newPosition = I.palmPosition + offset;
          b.nextJoint = LerpVector(b.nextJoint, newPosition, T);

          offset = P.f.b.prevJoint - P.palmPosition;
          newPosition = I.palmPosition + offset;
          b.prevJoint = LerpVector(b.prevJoint, newPosition, T);

          Recalculate bone direction;
          Recalculate bone centre position;
      end
      f.direction = LerpVector(f.direction, P.f.direction, T);
  end
```

To interpolate the hand position, the offset between the palm positions of the two hands is calculated. The offset is then applied to the start and end positions of each bone to calculate the new position. The bone is then interpolated to the new position using the same linear interpolation approach as used with the fingers. Algorithm 5.8 shows the process of interpolating a hands position by adjusting the positions of each bone.

Algorithm 5.8 - Pseudocode of the process for interpolating the position of a hand

```
Hand Position Interpolation
  Input:  P : Target Hand, I : Input Hand
  Offset = P.palmPosition - I.palmPosition;

  foreach Finger f in Hand do
    foreach Bone b in Finger do
      offsetPosition = b.nextJoint + offset;
      b.nextJoint = LerpVector(b.nextJoint, offsetPosition, t);

      offsetPosition = b.prevJoint + offset;
      b.prevJoint = LerpVector(b.prevJoint, offsetPosition, t);

      Recalculate bone direction;
      Recalculate bone centre position;
    end
  end
  Recalculate the palm normal;
```

Once both the start and end position have been updated, the bones direction and centre are recalculated. The palm position, wrist position, arm centre, arm direction, arm start and end vectors, are also interpolated using the same simple vector interpolation and offset. Once the position interpolation has completed, the resulting hand is analysed against the target hand. If the resulting hand is within the required threshold (99.8%) of the target hand, the interpolation is stopped. Figure 0.34 shows the code used to perform the hand interpolation, using the hand offset and the bones current positions.

As previously discussed, the hand and fingers are interpolated separately. If the fingers are not interpolating, the bone positions are updated by applying an offset to the hands palm position to calculate the new bone position. The finger updating process uses the same functionality as the finger interpolation with the new positions calculated and set as opposed to the result of interpolation. The offset calculated is between the position of the

corresponding bone on the target hand and the target hands palm position. The offset is applied to the current hands palm position to calculate the new start and end positions of the current bone. Once applied, the centre position and direction of the current bone are recalculated. Algorithm 5.9 shows the process of updating the finger positions when the hand is interpolating, with each bone adjusted using the relative offset. Figure 0.35 shows the code used to set the positions of the finger bones if they are not interpolating but the hands position is.

Algorithm 5.9 - Pseudocode of the process for updating the finger positions when the hand is interpolating

```
Update Finger Positions During Hand Position Interpolation
Input:  P : Target Hand, I : Input Hand

foreach Finger f in Hand do
    foreach Bone b in Finger do
        offset = P.f.b.nextJoint - P.palmPosition;
        newPosition = I.palmPosition + offset;
        b.nextJoint = newPosition;

        offset = P.f.b.prevJoint - P.palmPosition;
        newPosition = I.palmPosition + offset;
        b.prevJoint = newPosition;

        Recalculate bone direction;
        Recalculate bone centre position;
    end
end
```

Once the system has finished interpolating or if not interpolating, the resulting hand is stored as the starting point of the next interpolation. The final step in the interpolation system updates the ‘display/graphics hand’ object in the custom data structure using the result hand. The display/graphics hand is rendered if the system is interpolating and is used to detect jumps in the next update cycle.

The interpolation system was designed to work alongside the stability detection system as detailed in section 5.3. This ensures the live data has stabilised before interpolating to it, to synchronize the motion controlled and live tracking hands. Once tracking data is classified as stable as detailed in section 5.3, the interpolation system transitions the virtual hand representation from being Vive tracking system controlled to the live data. The Stability Checker is also used to detect noise within the live data and automatically trigger the interpolation system to smooth out any effects of the noise.

To support the implementation of the interpolation system, the MOT systems rendering pipeline was modified to support both MOT tracking data and purely graphical representations such as the results of the interpolation system. The custom data structure used throughout the MOT system was extended to support the two purely graphical hands previous discussed. The separation of graphics and tracking data enabled users to perform a fist gesture to interact with a virtual object, while the rendered virtual hand representation showed a smooth interpolation to a conformed hand holding the virtual object.

5.5 – Summary

This Chapter presented the phase two design of the extended Multiple Optical Tracking (MOT) system and the additional sub-systems of which it is comprised. This Chapter has shown the MOT systems novel approach to the validation of optical-based hand tracking data. This Chapter has also presented the design of complex systems designed to provide a more consistent user experience. The extended MOT system includes the design of a custom ray-based occlusion detection system. To smooth inconsistencies in optical tracking data and synchronize data sources, the design of a custom

interpolation system has been presented. Finally, this Chapter has shown a novel approach to the analysis of optical tracking to ensure data and position stability. The designs presented have been discussed in detail with justification for the selected methods used. In addition, details of developmental testing have been presented as part of the discussion behind design decisions made. The next Chapter presents the phase two experimental procedure and analysis of the results.

Chapter 6 – Results Phase Two

In this Chapter the experimental data from the second phase of experimentation is analysed. First, a thematic analysis of participant comments made during both experiences and the semi-structured interviews is presented.

Secondly, the hand data logs generated during the Chemical Engineering simulation and the open Likert-based questionnaires are analysed. The questionnaire answers were averaged and analysed to determine the effect of each condition on user experience.

Thirdly, the hand data logs generated during the Standardized Gesture Evaluation Test and the Likert-based questionnaire answers are analysed. The questionnaire answers were averaged to evaluate the performance of each condition on the gesture vocabulary and as a result, the effectiveness of the condition in resolving the key issues with optical trackers. Finally, a discussion of the results is presented.

The hand data logs from both parts were analysed to determine data validity and periods of invalid or missing data. The following tests were used; Chi-Square tests, Paired-Samples t-test, linear regression and several ANOVA tests to analyse the number of valid/null frames, total time without hands and the longest time without hands.

6.1 – Experimental Setup & Procedure

Fourteen adults (one female) participated in the study. All participants had perfect or corrected to, vision. Participants were pre-screened into one of four categories depending on their experience, namely; Gamer and VR Gamer (4), Gamer (8), VR Gamer (0) or None (2). Due to issues with data recording, two participants had to be removed from the study.

The experiments in this phase of testing were split into two parts. In the first part participants experienced a Chemical Engineering based VR simulation (Appendix 12) using two interaction conditions. The MOT system was compared against a single Leap approach to evaluate the effectiveness of the additional functionality implemented in the MOT system at creating a more consistent and natural user experience. Furthermore, the comparison enabled the evaluation of the additional functionality at producing more valid tracking data. Semi-structured interviews were conducted to understand participant experiences during the conditions.

For the second part of the experimentation a standardized gesture evaluation test was developed to evaluate optical based hand trackers in virtual reality. The gesture test was designed to evaluate key areas of an optical tracker's functionality; occlusion, field-of-view, accuracy and stability. In addition, the test evaluated usability factors including naturalism, ease of use and ease of learning. After performing each gesture, participants were asked a series of Likert-based questions to evaluate the conditions performance. Questions focused on whether there were any glitches or tracking issues experienced. Once all the gestures for a condition had been performed, participants were

asked a further series of Likert-based questions to ascertain their overall experiences in several areas, including naturalism and ease of use.

The entire experimental procedure was recorded (audio & video) for each participant to capture comments and reactions along with answers for transcription. A data logger ran in the background in both tests to capture large quantities of data such as the validity, confidence of incoming data and the status of the system at each point of the main loop. All the data logged was written to a text file for detailed analysis. Participants were informed that they could be as vocal as they wish on comments, criticisms or feelings experienced during the simulations.

6.1.1 – Chemical Engineering Simulation

In this section the experimental design and procedure for the evaluation of the MOT system and Leap Motion conditions are presented. The MOT system was compared to a single Leap Motion to determine if the MOT system results in a statistically significant increase in the number of valid tracking data frames produced, as a baseline comparison. In addition, the MOT system is compared against a single Leap Motion sensor to evaluate the effectiveness of the additional functionality implemented in resolving the four main issues with optical trackers.

The Chemical Engineering simulation is a seated VR experience that takes participants through the process of configuring and starting a mini-continuous distillation unit running an alcohol and water solution. During the simulation users interact with a series of knobs, levers and dials along with a chemical

beaker. The simulation was developed with expert input from academics within the Chemical Engineering department.

From an interaction perspective, the simulation presents the user with three different task types; translation/rotation, virtual user interface (VUI) and precise manipulation of virtual controls. The translation/rotation tasks require the user to put on safety goggles along with pouring a mixture into a large beaker. The simulation also features tasks that require precise manipulation of virtual controls such as adjusting the flow rates on the flow meters. Figure 6.1 shows a screenshot of the simulation and virtual equipment.

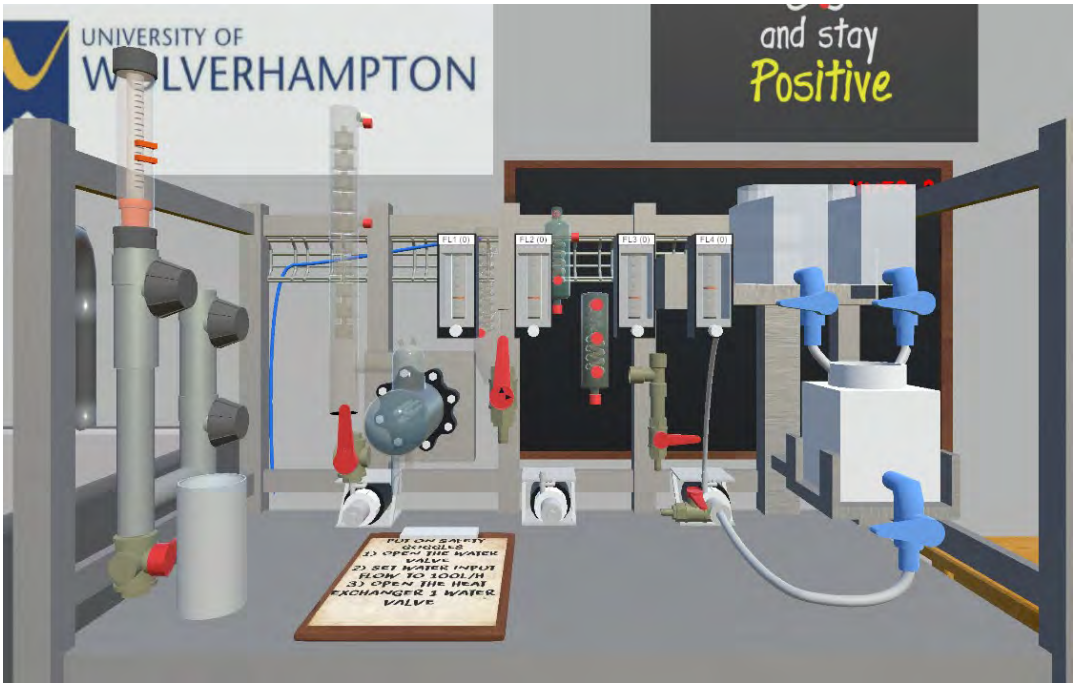


Figure 6.1 - Render of the Chemical Engineering Simulation

Finally, the simulation features virtual UI; specifically, a simple touch screen interface for the boiler. The user interacts with the virtual UI by tapping the buttons using their index finger. Figure 6.2 shows the virtual user interface of the control box.

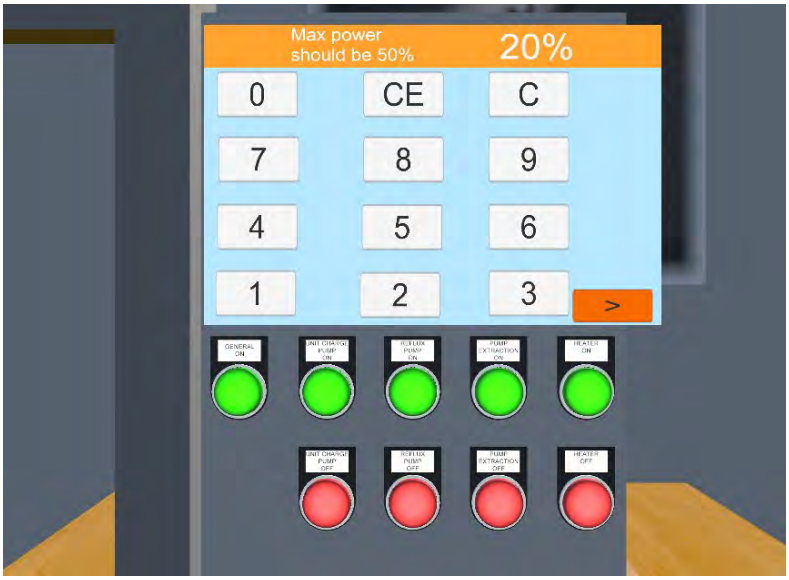


Figure 6.2 - Render of the Boiler VUI

As with the previous experiment the users are presented with a series of instructions via a virtual clipboard with clarification provided by the instructor if necessary. An audible tick sound effect was used to provide feedback along with the task displayed as green on the clipboard.

Participants completed the scenario twice, once for the MOT system followed by the standard Leap condition. The conditions were experienced in a fixed order with the MOT system providing an initial baseline for the comparison and removing the effect of order-based response bias in favour of the MOT system.

Before participants started the simulation, they were shown a screenshot of the virtual environment, with the experimental procedure and the object interaction mechanics explained. Once the experimental procedure had been explained, participants were asked to make themselves comfortable and begin the first simulation. After completing the first simulation, participants were asked a series of questions in a semi-structured interview. Once the first set

of questions had been completed, participants experienced the Leap condition before being asked the same series of questions to ascertain their experiences of the difference between the two conditions. Table 6.1 shows the interview questions asked during the semi-structure interviews.

The interview questions were designed to evaluate the usability and realism of the interaction conditions and the effect on user experience. The questionnaire contains some questions based on the System Usability Scale (SUS) of Brooke (1996) and the work of Witmer & Singer (1998). During the interviews, the questions were clarified to participants if they were unsure of the context. In the questionnaire, engagement is defined as the feeling of wanting to continue the use of the interaction condition. The question regarding ease was designed to assess how difficult the interaction approaches were to use, how easy was it to complete a desired action. The question regarding resemblance to real life was designed to compare the interaction approaches to that of the real-world. This enabled the effectiveness of the interaction approaches at transferring real-world skills and experience to be evaluated. The immersion question focused on the simulation, object behaviour and the replication of the real-world senses per the definition discussed in Chapter 2.4 (Meehan, et al., 2002; Slater, 2003; Kim, et al., 2017; Park, et al., 2019). The questionnaire did not use questions regarding presence as the “sense of being there” for two reasons. Firstly, it was anticipated that due to unfamiliarity with the simulation, clarification on instructions would have to be given. Secondly, as part of the detailed analysis of user experience, clarification and discussion of comments made by participants during the simulation would take place. Both cases would

significantly reduce the user's sense of presence as talking with the interviewer would remind them of the existence of the real world.

Table 6.1 - Interview Questions asked during the semi-structured interviews

Semi-Structured Interview Questions	**Likert Based**
What was your perception of the virtual hands during the experience? a) Did you experience any issues? Please explain.	No
How would you rate your level of control over the virtual hands? Please explain	Yes
On the following scale, how would you rate the accuracy of the virtual hands? Please explain	Yes
On the following scale, how would you rate the reliability of the virtual hands? Please explain	Yes
On the following scale, how would you rate the ability to extend your arm whilst keeping tracking? Please explain	Yes
How did you find the interaction with the controls and objects? a) How did you find the representation of the hand when interacting with objects?	No
On the following scale, how did you find the naturalism of the hands? a) Did you find the system uncomfortable/restrictive, if so how and why?	Yes
On the following scale, how easy did you find the controls to use?	Yes
On the following scale, how did you find the engagement of the controls?	Yes
On the following scale, how would you rate the simulations level of immersion?	Yes
On the following scale, how closely did you find the hand controls resemble real life?	Yes

6.1.2 – Standardized Gesture Evaluation

In this section the experimental design and procedure for the evaluation of the MOT system and Leap Motion conditions is presented. The conditions were compared using a standardized gesture evaluation test developed to provide consistent evaluation of optical tracking sensors. The test evaluates optical tracker performance and user experience, without external factors such as a virtual environment to deviate user focus from the interaction experience.

The gestures used were designed to stress the four key issues of optical based trackers; occlusion, field-of-view, accuracy and stability. The reaching, moving away and side gestures require the user to test both the horizontal and vertical field-of-view limitations and how the interaction mechanism handles the instability of the tracking data. The hand occlusion gesture was designed to test how the interaction mechanism handles occlusion, the resulting data instability and resuming the process of tracking the hand. Finally, the fist and beaker gestures were designed to test the overall accuracy of the interaction mechanism, with fist gesture focusing on rotation and the beaker gesture focusing on translation.

Participants completed the test twice, once for the MOT system followed by the standard Leap condition. Before participants started the test, they were shown the gestures they would need to perform, specifically; circular motions (Figure 6.3), hands over each other (Figure 6.4), facing hands moving away (Figure 6.5), hands to the sides (Figure 6.6), reaching gesture (Figure 6.7), fist rotation (Figure 6.8), and moving a beaker (to the left with the left hand and to the right with the right hand).

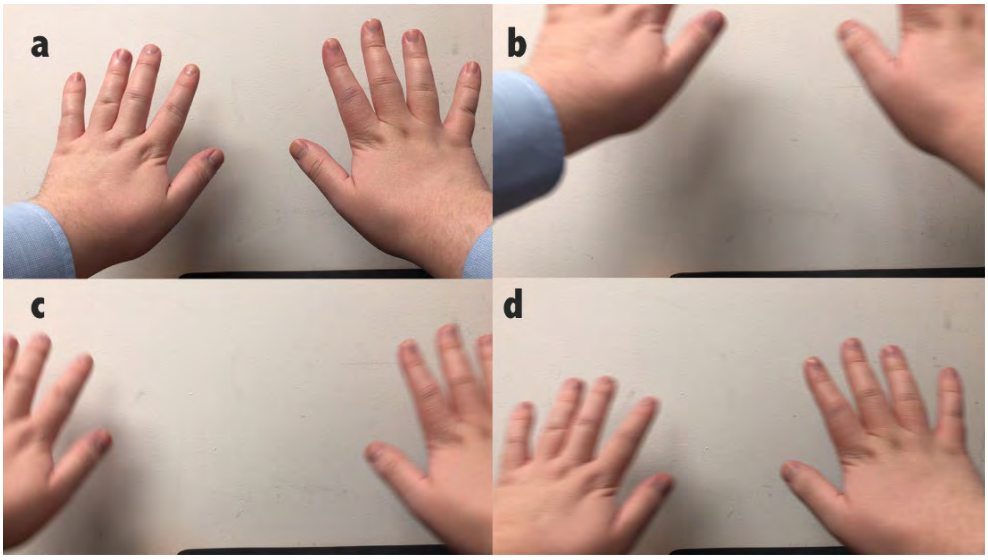


Figure 6.3 - Circle Gesture (Left Hand Anti-Clockwise, Right Hand Clockwise)

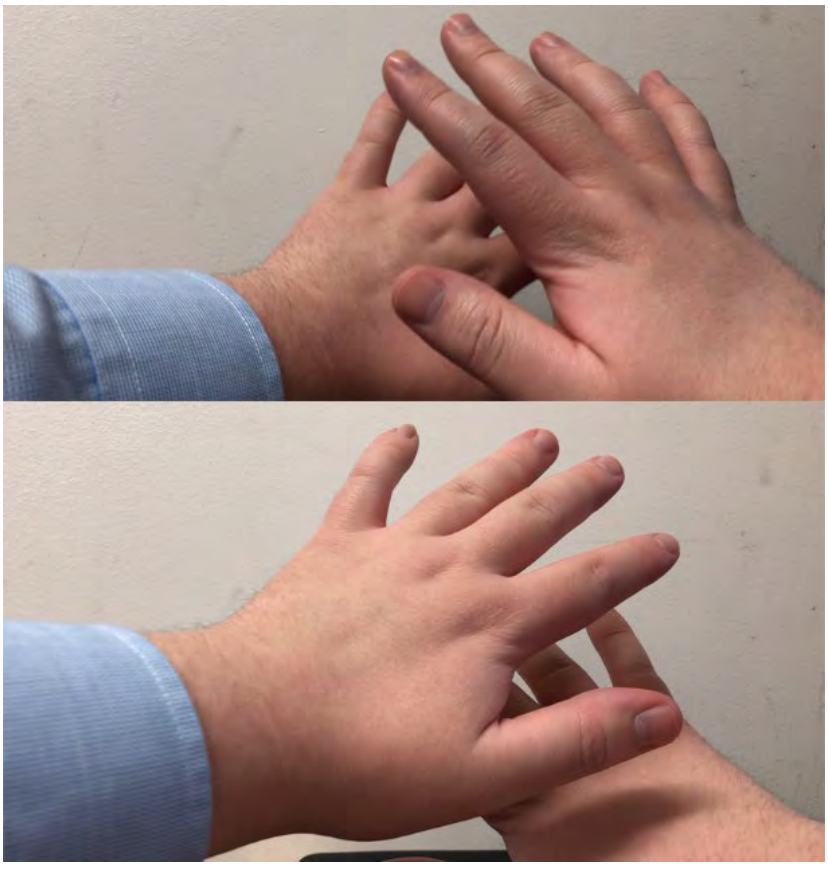


Figure 6.4 - Hand Occlusion Gestures (top = Right over left, bottom = left over right)

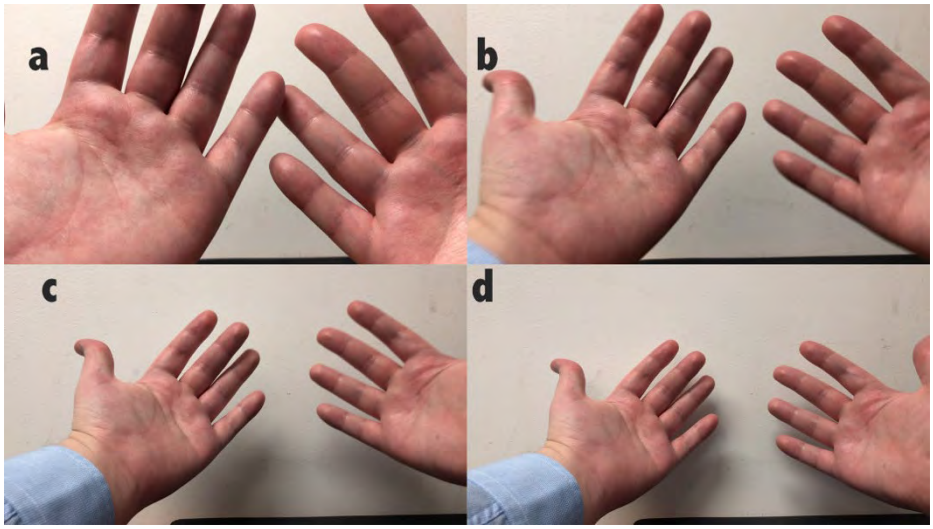


Figure 6.5 - Hands Moving Away Gesture

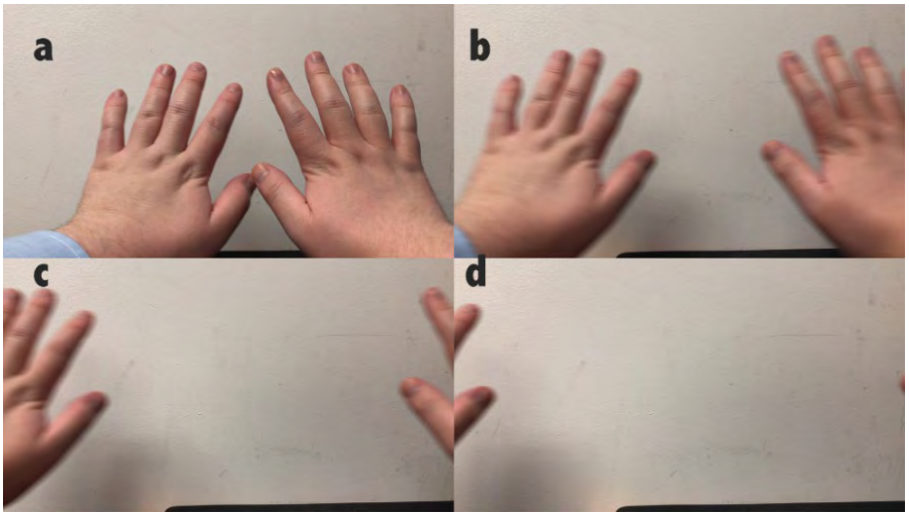


Figure 6.6 - Hands to the sides gesture

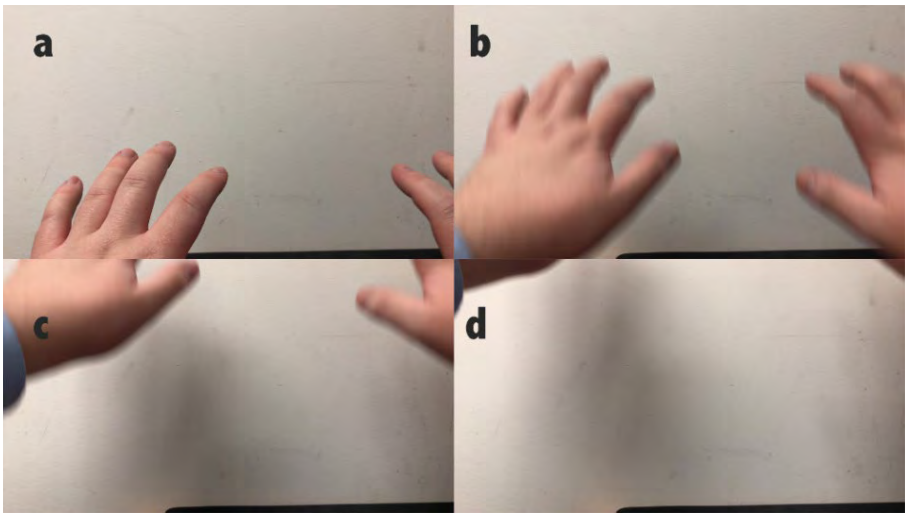


Figure 6.7 - Reaching Gesture

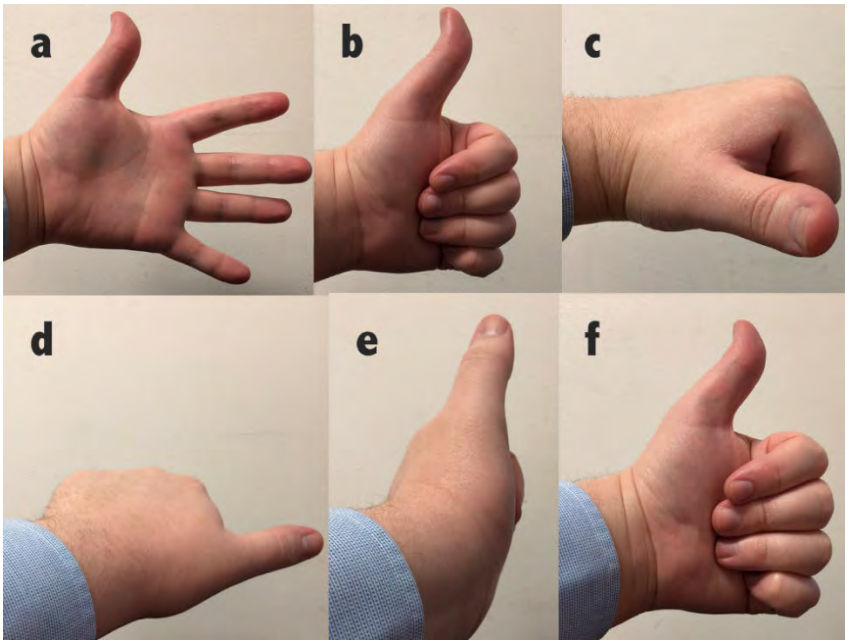


Figure 6.8 - Fist Gesture

Once all the gestures were completed, participants were asked a series of Likert-based questions to ascertain the participants overall experience of the condition while performing the gestures. The questions are broken down into several sections, specifically: overall accuracy, naturalness, satisfaction, ease of use and ease of learning. The questions were based on Brooke (1996) and Lund (2001) with inspiration from Kim et al (2015). The questions regarding naturalness and accuracy were inspired by questionnaires previously used in human-computer interaction research, the research presented in Chapter two and the results of the first phase of experimentation. The standardized gesture questionnaire can be found in Appendix 13.

6.2 – Simulation Experimentation

In this section the experimental data from the Chemical Engineering simulation is analysed. First, the analysis of the hand data logs generated during the simulation to determine data validity and periods of invalid or missing data, is presented.

Secondly, the Likert-based question answers are analysed to evaluate user experience of the interaction conditions and the virtual environment. Cronbach's alpha was used to determine the reliability of the Likert scale-based interview questions. In addition, a Spearman Correlation was performed to measure association between the questions (Statistics, 2018). The questionnaire answers were also analysed using paired sample t-tests to determine whether the mean difference between the two conditions was statistically significant. The semi-structured interviews were analysed by thematic analysis (section 10.1) with inter-rater reliability testing (Cohen's Kappa).

The initial hypothesis was that the use of an additional sensor and the MOT system would provide the user with a more consistent user experience. In addition, the MOT system would provide an increased field-of-view and accuracy with the occlusion and stability detection systems helping to provide more stable data. As a result, the MOT system would provide more valid frames than the front-mounted sensor and significantly reduce the time hands could not be tracked.

6.2.1 – Tracking Data Validity

In this section the results of the Quantitative analysis are detailed, analysing log files generated during participant experiences. The log files were analysed to determine the effect of the conditions on data validity and periods in which invalid or missing tracking data was observed. Furthermore, the validity of the tracking data produced by the MOT system is compared against that of a single Leap Motion sensor. In the figures presented “Aggregated Hands” refers to the hands produced by the MOT system and “External” refers to the 45-degree sensor.

6.2.1.1 – Chi Square Test

A Chi-Square test was conducted to analyse hand validity and determine the significance between the sensors and MOT system on hand validity produced. The test would enable the 45-degree sensor positioning to be evaluated to determine whether it had increased hand validity as previously hypothesized. Furthermore, the test would evaluate the effectiveness of the aggregation process at further improving hand validity. Table 6.2 shows the details of the Chi-Square test including the null and alternative hypotheses.

Table 6.2 - Details of the Chi-Square test conducted

Null Hypothesis	Assumes that there is no association between the two variables.
Alternative Hypothesis	Assumes that there is an association between the two variables.
Outcome	One Null Hypothesis Not Rejected, Two Alternative Accepted

Figure 6.9 shows the average percentage of total frames in which both hands were valid for each sensor and the MOT system. Figure 6.10 shows the average percentage of total frames in which the left and right hands were valid for each sensor and the MOT system. In Figure 6.9 and Figure 6.10, higher bars represent a greater percentage of the simulation tracking data being valid, thus demonstrating greater performance. Both Figure 6.9 and Figure 6.10 show the MOT system to produce more valid hand data than the two sensors.

The validation data was analysed using Chi-Square tests to determine the significance between the counts of valid and null hands observed by a Leap Motion sensor (front-facing or 45-degree angle) and the MOT system. The tests were run for both left and right hands and compared the two sensors against the MOT system and each other.

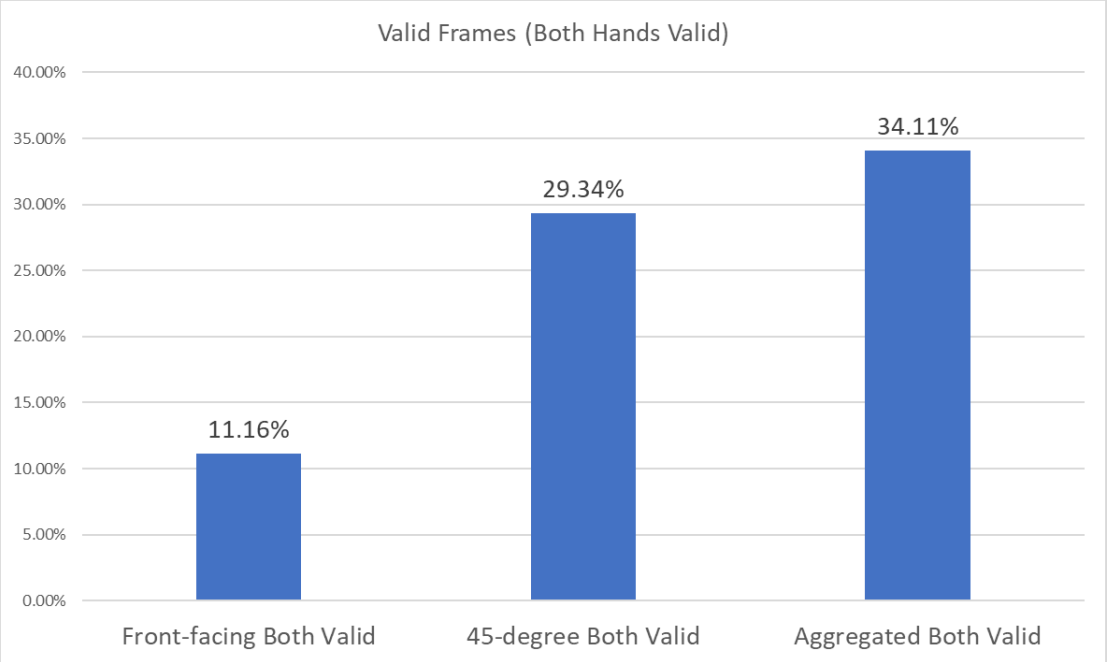


Figure 6.9 - Average of total frames in which both hands were valid

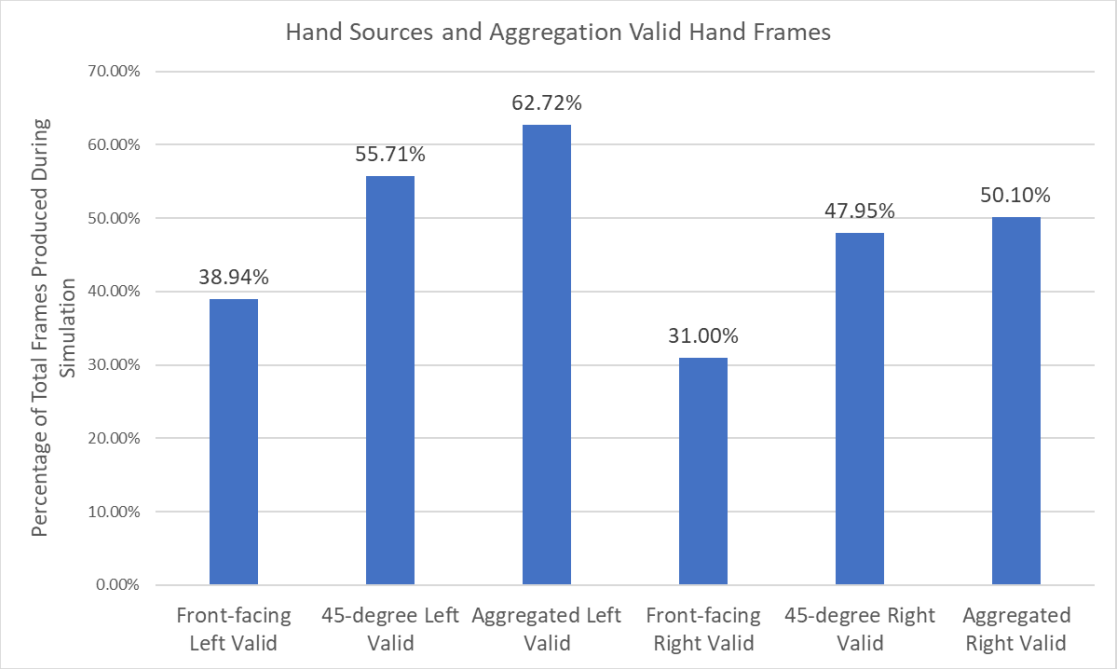


Figure 6.10 - Average Percentage of Total Frames Valid for each sensor and the MOT system for each hand

The p-value of the tests from the front-facing Leap Motion and the MOT system were all significant at $p < .01$. The p-value of the tests from the 45-degree angle Leap Motion sensor and the MOT system were not significant. The p-value of the tests from the front-facing Leap Motion and the 45-degree angle Leap Motion sensor were all significant at $p < .05$, with the test for both hands valid being significant at $p < .01$. The results show that the MOT system and 45-degree angle sensor were able to produce significantly more valid hands than the front-facing Leap Motion sensor.

Analysis of the chi-square statistics indicates greater data similarity between the 45-degree sensor and the MOT system than the front-facing sensor and MOT system. This would indicate that a 45-degree angle Leap Motion sensor as presented in this work is better positioned to recognize user hands in VR than a Leap Motion mounted on the front of a headset. The p-value results

indicate statistically insignificant results suggesting the two are not related. However, the small difference in hand validity between the 45-degree sensor and MOT system, results in a small difference between the observed and expected values. Thus, the results suggest the null hypothesis cannot be rejected. The effect of the 45-degree sensor on the MOT system is further evaluated by Linear Regression tests in section 6.2.1.6.

The results show the MOT system can produce significantly more valid hands than the front-facing Leap Motion sensor. Figure 6.10 indicates the MOT system offers an improvement in individual valid hand recognition over a bracket mounted Leap Motion. However, as shown in Figure 6.9 this improvement increases when the number of times that two valid hands were generated by a sensor or the MOT system is considered. The results show that the MOT system was able to generate two valid hands for a frame more often than a bracket mounted Leap Motion.

The results strongly correlate with the results of the pilot study conducted in the first phase of testing. The results show the 45-degree sensor to produce significantly more valid frames than the front-facing sensor. In addition, the results show the MOT system to produce significantly more valid frames than the front-facing sensor. Furthermore, the results support those of the pilot study in answering one of the academic questions, showing the 45-degree sensor to be the most effective sensor configuration. However, the results show no statistically significant difference between the 45-degree sensor and the MOT system, in contrast to the results of the pilot study in phase one.

6.2.1.2 – Paired Samples T-Test

A Paired Samples T-Test was conducted to determine whether the mean difference between the sensors and the MOT systems hand validity was statistically significantly different from zero. The valid and null data of each hand was compared across both sensors and the MOT system. In addition, the number of times two valid hands were generated was compared. Table 6.3 shows the details of the Paired Samples T-tests including the null and alternative hypotheses.

Table 6.3 - Details of the Paired Samples T-tests conducted

Null Hypothesis	Assumes that the mean difference between paired values is equal to zero.
Alternative Hypothesis	Assumes that the mean difference between paired values is not equal to zero.
Outcome	Null Hypothesis Rejected, Alternative Accepted

Table 6.4 presents the results of the Paired-Samples T-tests on hand validity, comparing the two sensors and the MOT system on each hand and the two hands together. The results show statistical significance in all the tests performed.

Table 6.4 - Results of Paired-Samples T-tests on hand validity

Pairs	Normality (Shapiro-Wilk)	First Mean	Second Mean	Paired Differences (95% CI)	t(11)	Significance (p)
Front Left -> MOT Left	0.257	38.94% +- 14.14%	62.72& +- 21.41%	-23.78% (-32.97% to -14.58%)	-5.690	<.0001
Front Right -> MOT Right	0.156	31% +- 21.68%	50.10% +- 25.61%	-19.1% (-26.79% to -11.41%)	-5.467	<.0001
Front Left -> External Left	0.836	38.94% +- 14.14%	55.71% +- 22.42%	-16.77% (-27.78% to -5.76%)	-3.351	0.006
Front Right -> External Right	0.307	31% +- 21.68%	47.95% +- 25.42%	-16.94 % (-25.48% to -8.41%)	-4.368	0.001
External Left -> MOT Left	Assessed by Skewness & Kurtosis	55.71% +- 22.42%	62.72% +- 21.41%	-7% (-11.68% to -2.32%)	-3.294	0.007
External Right -> MOT Right (Sign Test)	Not normally distributed	47.97% (Median)	48.49% (Median)		3.175 (z score)	0.001
Front Both -> MOT Both	0.058	11.16% +- 16.74%	34.11% +- 30.09%	-22.95% (-35.34% to -10.57%)	-4.079	0.002
Front Both -> External Both	0.236	11.16% +- 16.74%	29.34% +- 27.35%	-18.18% (-28.86% to -7.50%)	-3.746	0.003
External Both -> MOT Both	Assessed by Skewness & Kurtosis	29.34% +- 27.35%	34.11% +- 30.09%	-4.77% (-8.96% to -0.59%)	-2.512	0.029

The results of the Paired-Samples T-tests show statistically significant differences between all the pairs. This indicates statistically significant differences between the two sensors and both sensors compared against the MOT system. The results also show the effectiveness of the 45-degree sensor in producing more valid frames than the front-facing sensor. In addition, the results show the effectiveness of the MOT system's aggregation system at producing even more valid frames.

Analysis of the results shows that the mean difference between the 45-degree and MOT sensor is not zero. This shows that while the 45-degree sensor offers a more effective configuration over a front-facing sensor, the aggregation process used by the MOT system further improves hand validity. However, while the results show a significant difference in validity, the Chi-Square test indicates there is no association between the 45-degree sensor and the MOT system. This suggests that despite the significant improvement the MOT

system offers over the 45-degree sensor, the two are not related. Thus, the MOT validity cannot be predicted based on the 45-degree sensors validity. The effect of the 45-degree sensor on the MOT system is further analysed using a Linear Regression test in section 6.2.1.6.

6.2.1.3 – One-way Repeated Measures ANOVA on Valid/Null Hands

A one-way within subjects (or repeated measures) ANOVA was conducted to compare the effect of each sensor and the MOT system on hand validity. The test was used to determine whether there were any statistically significant differences in the average hand validity of the two sensors and the MOT system. Thus, determining the effectiveness of the sensor configuration and aggregation process at improving hand validity. Table 6.5 shows the details of the one-way within subject's ANOVA including the null and alternative hypotheses.

Table 6.5 - Details of the One-Way Within Subjects ANOVA test conducted

Null Hypothesis	The null hypothesis states that the related population means are equal.
Alternative Hypothesis	The alternative hypothesis states that the related population means are not equal.
Outcome	Null Hypothesis Rejected, Alternative Accepted in all but two tests

Table 6.6 presents the results of the one-way repeated measures ANOVA on hand validity for each hand and both hands together, between the two sensors and the MOT system. The results show statistically significant increases in valid hands and decreases in null hands in all tests. However, the 'both valid' and 'both null' tests, between 45-degree sensor and the MOT system show no statistically significant difference was found. The results concur with the previous tests showing the 45-degree sensor to provide more valid hands than

the front-mounted sensor, with the MOT system providing significantly more valid frames than both.

Table 6.6 - One-way repeated measures ANOVA results for valid and null hands

Hand Data	Sphericity	Epsilon Correction	Within-Subjects Effects	Post hoc Analysis (Bonferroni Correction) (95% CI)		
				Front-Facing - 45-degree	Front-Facing - MOT System	45-degree - MOT System
Left Hand Valid	0.012 (Violated)	0.631	F(1.262,13.882) =19.044, p<.0005	-16.77 (-30.88 to -2.66), p <.0005	-23.772 (-35.553 to -11.991), p<.0005	-7.002 (-12.996, -1.007), p<.05
Left Hand Null	0.019 (Violated)	0.646	F (1.291,14.202) =20.705, p<.0005	17.915 (4.150 to 31.680), p<.05	24.133 (12.493 to 35.774), p<.0005	6.218 (0.148 to 12.289), p<.05
Right Hand Valid	<.0005 (Violated)	0.519	F(1.038,11.419) =23.658, p<.0005	-16.945 (-27.884 to -6.005), p <.01	-19.100 (-28.952 to -9.248), p<.01	-2.155 (-4.147, -0.164), p<.05
Right Hand Null	<.0005 (Violated)	0.520	F(1.039,11.434) =23.976, p<.0005	16.943 (6.093 to 27.794), p<.01	19.074 (9.292 to 28.857), p<.01	2.131 (0.137 to 4.125), p<.05
Both Hands Valid	<.01 (Violated)	0.581	F (1.161,12.771) =14.959, p<.01	-18.181 (-31.869 to -4.493), p <.01	-22.954 (-38.825 to -7.084), p<.01	-4.773 (-10.133 to 0.586), p>.05
Both Hands Null	<.01 (Violated)	0.559	F (1.118,12.296) =22.147, p<.0005	16.669 (4.687 to 28.651), p<.01	20.189 (10.635 to 29.744), p<.0005	3.520 (-0.447 to 7.487), p>.05

The results of the post hoc analysis show that hand validity significantly increased from the front-facing sensor to both the 45-degree sensor and MOT system. In addition, the results show hand validity significantly increase from the 45-degree sensor to the MOT system. The results of the tests for both hands show no statistically significant improvement between the 45-degree sensor and MOT system. Based on the results of the single hand conditions and previous tests, this is likely due to the 45-degree sensor having the greatest impact on the MOT system validity. In some cases, the tracking data from the front-facing sensor will have provided any missing hands, resulting in a slight increase compared to the 45-degree sensor. However, as the MOT system will have been mostly dependent upon the 45-degree sensor, if it did not have two hands available, typically the MOT system would not either. Thereby, resulting in the insignificant difference between the two.

6.2.1.4 – One-way Repeated Measures ANOVA on Total Time with No Hands

A one-way within subjects (or repeated measures) ANOVA was conducted to compare the effect of each sensor and the MOT system on the total time in which no hands were available. The test was used to determine whether there were any statistically significant differences in the average percentage of simulation duration in which no hand data was available. Thus, determining the effectiveness of the sensor configuration and aggregation process at improving field-of-view, stability and hand validity. Table 6.7 shows the details of the one-way within subject's ANOVA including the null and alternative hypotheses.

Table 6.7 - Details of the One-Way Within Subjects ANOVA test conducted

Null Hypothesis	The null hypothesis states that the related population means are equal.
Alternative Hypothesis	The alternative hypothesis states that the related population means are not equal.
Outcome	Null Hypothesis Rejected, Alternative Accepted in all but one test

Figure 6.11 shows the average percentage of simulation length in which the sensors and MOT system were unable to provide tracking data for the specific hand. Figure 6.12 shows the average percentage of simulation length in which the sensors and MOT system were unable to provide tracking data for both hands. In both Figure 6.11 and Figure 6.12, smaller bars show tracking data was available for more of the simulation, indicating better performance. Figure 6.11 and Figure 6.12 show the MOT was able to provide tracking data for more of the simulation, having the lowest total time for each hand and the two hands together. The results also show the front-facing sensor was unable to provide

tracking data for a substantial part of the simulation with the right hand missing 66.22% of the simulation on average. Table 6.8 shows the results of the one-way repeated measures ANOVA on total time without hands.

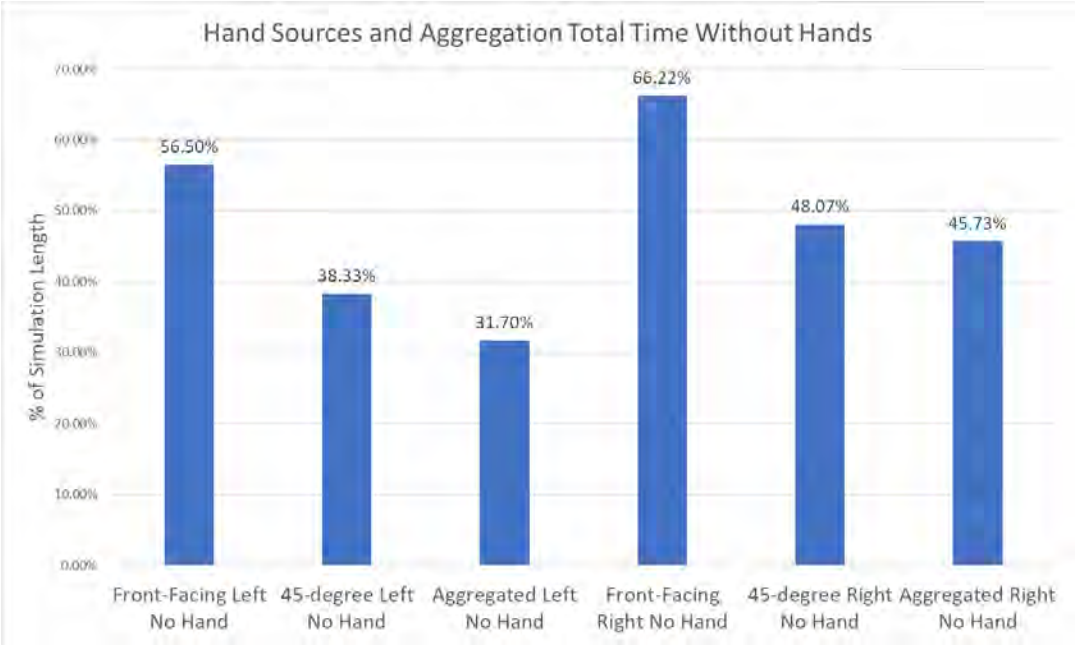


Figure 6.11 - Average total Time without hands (% of Simulation Length) for each source and the MOT system

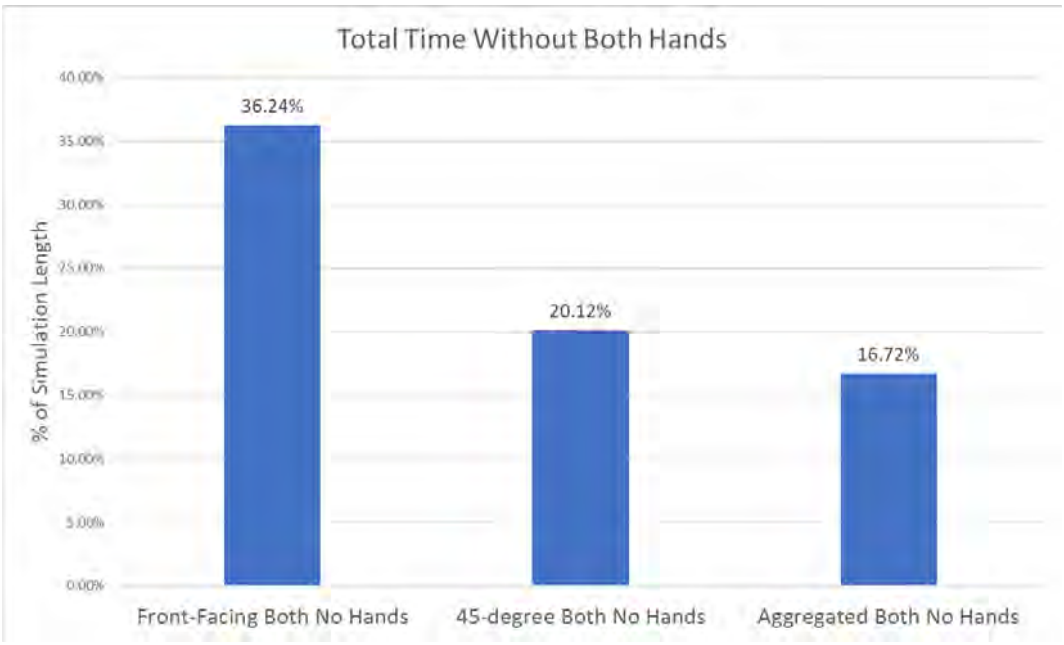


Figure 6.12 - Average total Time without both hands (% of Simulation Length) for each source and the MOT system

Table 6.8 - One-way repeated measures ANOVA results for total time with no hands

Hand Data	Sphericity	Epsilon	Within-Subjects Effects	Post hoc Analysis (Bonferroni Correction) (95% CI)		
				Front-Facing - 45-degree	Front-Facing - MOT System	45-degree - MOT System
No Left Hand	0.021 (Violated)	0.650	F(1.300,14.297) =20.398, p<.01	18.170 (3.792 to 32.549), p<.05	24.803 (13.173 to 36.433), p<.01	6.633 (0.022 to 13.244), p<.05
No Right Hand	<.01 (Violated)	0.521	F(1.043,11.471) =23.325, p<.01	18.155 (6.336 to 29.974), p<.01	20.492 (9.879 to 31.106), p<.01	2.337 (0.080 to 4.594), p<.05
No Hands	<.01 (Violated)	0.565	F(1.130,12.432) =22.003, p<.01	16.125 (4.547 to 27.703), p<.01	19.524 (10.221 to 28.828), p<.01	3.400 (-0.516 to 7.315), p>.05

The results of the ANOVA show a statistically significant decrease in the all the tests except for the total time with no hands (both hands missing) where there was no statistically significant difference between the 45-degree sensor and the MOT system. The results of the post hoc analysis between the front-facing and 45-degree sensor further demonstrate the effectiveness of the 45-degree sensor configuration. The results show a 32% and 27% decrease in time without hands for the left and right hand between the front-facing and 45-degree sensor, respectively.

The results concur with the one-way repeated measures ANOVA on the number of valid hands produced by the sensors and the MOT system. The increased number of valid frames produced by the 45-degree sensor and MOT system as shown in section 6.2.1.3 will have reduced the time in which no hand data was available. The results align with the results of the previous tests, showing the total time in which hand data was not available decreases as hand validity increases. Thus, reducing the periods of no hands and as a result the time in which an inferred hand must be used. These results further support the effectiveness of the MOT system design as an effective approach

to using multi-sensor data aggregation to improve optical tracker-based hand interaction approaches.

6.2.1.5 – Longest Time without Hands

Two Friedman tests were conducted to analyse the effect of the front-facing sensor, 45-degree sensor and MOT system at reducing the time in which no hand data is available. The test was used to determine whether there were any statistically significant differences in the average of the longest period in which no hand data was available. Thus, determining the effectiveness of the sensor configuration and aggregation process at improving field-of-view, stability and hand validity. Table 6.9 shows the details of the one-way within subject's ANOVA including the null and alternative hypotheses.

Table 6.9 - Details of the Friedman tests conducted

Null Hypothesis	The null hypothesis states that the distribution of scores in each group are the same
Alternative Hypothesis	The alternative hypothesis states at least two of the group's distributions differ.
Outcome	Some Null Hypothesis Not Rejected, Some Alternative Accepted

Figure 6.13 shows the average of the longest time as a percentage of the simulation length that each sensor and the MOT system had no data for a hand. Figure 6.14 shows the average of the longest time as a percentage of the simulation length that each sensor and the MOT had no hand data for both hands. In Figure 6.13 and Figure 6.14, smaller bars show a shorter period before tracking data was resumed (even for just a single frame), indicating better performance as the source was able to keep track of the users hands more consistently. The figures show the right hand experienced longer periods

of missing tracking data compared to the left hand, with a small improvement from the 45-degree sensor to the MOT system.

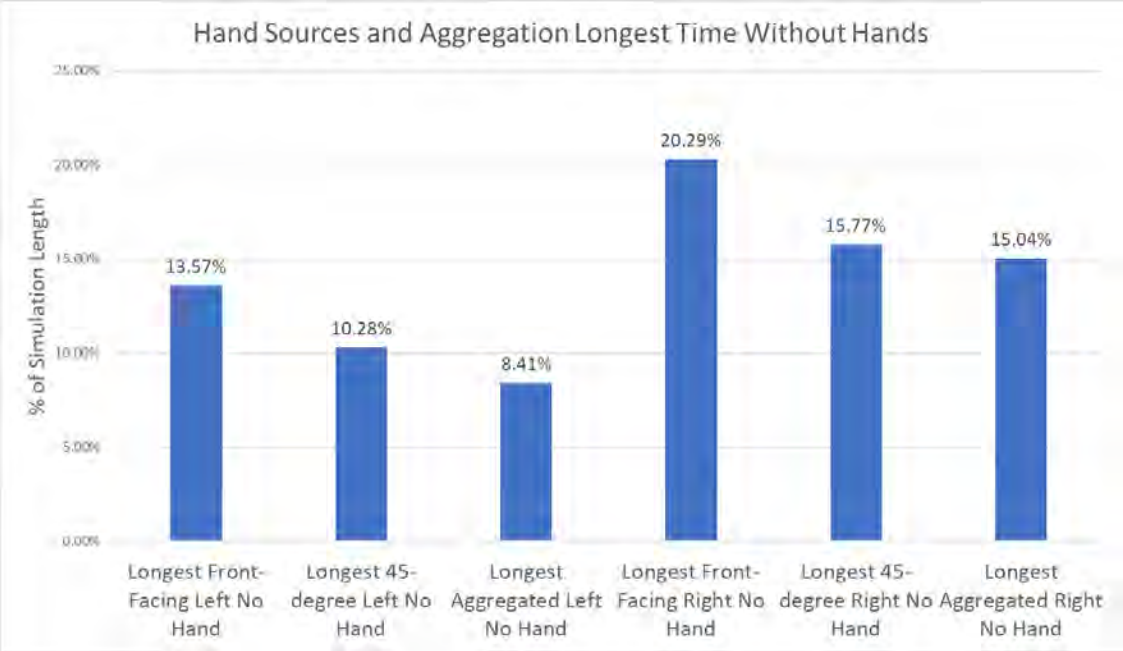


Figure 6.13 - Average longest time without hands (% of Simulation Length) for each source and the MOT system

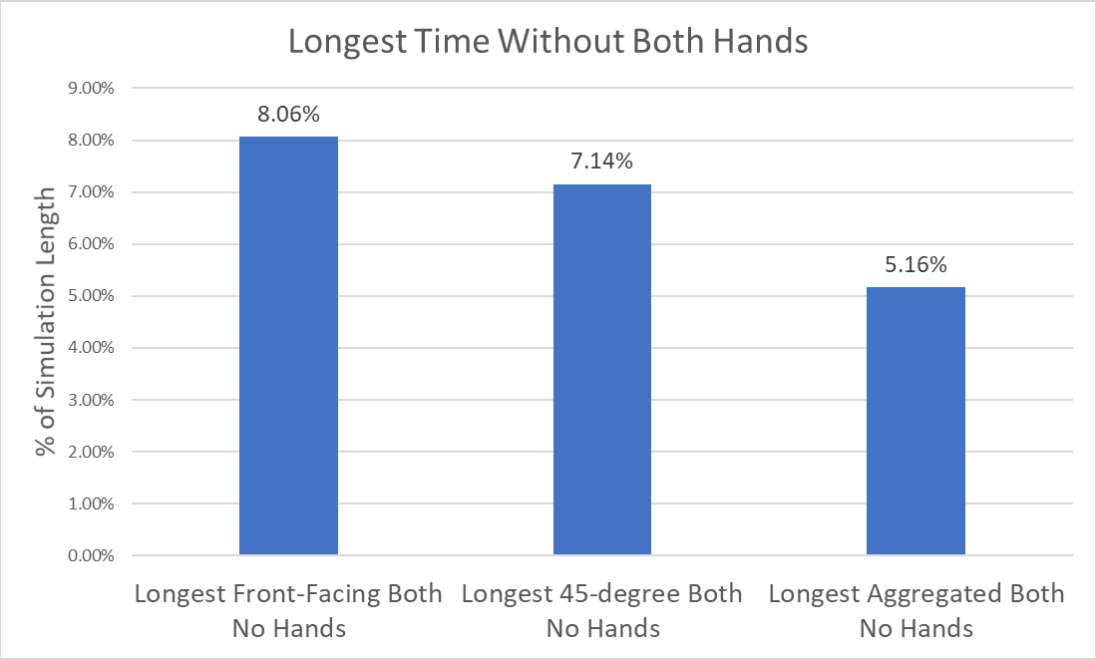


Figure 6.14 - Average longest time without both hands (% of Simulation length) for each source and the MOT system

The results show a statistically significant decrease ($p<.05$) for both the left and right hands, between the front-facing and 45-degree sensor. The left hand also shows a statistically significant decrease between the front-facing sensor and the MOT system ($p<.05$). However, there was no statistically significant difference between the 45-degree sensor and the MOT system for the left hand. The right hand saw no statistically significant decrease between both the front-facing to 45-degree sensor and 45-degree sensor to MOT system. A one-way repeated measures ANOVA was conducted which showed no significant decrease at all for the longest time with both hands missing.

The results indicate greater inconsistency between the left and right hands with the right hand having experienced a longer period of unavailability on average. Furthermore, based on the statistically insignificant difference between the sensors and MOT for the right hand (except for the front-facing and 45-degree) and the position of the objects relative to the user. It can be hypothesized that the control box task was a significant contributor. The proximity of the control box would likely have resulted in the user's hand being too close to the surface of the sensors, affecting their view of the user's hand. Overall, analysis shows the results to follow the trend as seen in previous results with the 45-degree sensor outperforming the front-facing sensor and the MOT have better performance than both sensors.

6.2.1.6 – Linear Regression on MOT Hand Validity

A Linear regression test was performed to assess the linear relationship between the sensors and MOT hand validity. The test was used to determine the statistical significance of any relationship and how much variation in the MOT systems hand validity is explained by a sensor's validity. Table 6.10 shows the details of the conducted linear regression test including the null and alternative hypotheses.

Table 6.10 - Details of the Linear Regression test conducted

Null Hypothesis	Assumes that there is no relationship between the two variables. The null hypothesis states that the coefficient of the slope is equal to zero.
Alternative Hypothesis	Assumes that there is a relationship between the two variables. If there is a significant linear relationship between the independent variable X and the dependent variable Y, the coefficient of the slope will not equal zero.
Outcome	Null Hypothesis Rejected, Alternative Accepted

Table 6.11 shows the results from the Linear Regression test, indicating the regression equation for each hand of each sensor along with the effect sizes. Analysis of the effect sizes (Adjusted R^2) shows the 45-degree sensor had a large effect with the front-facing having a medium effect on the MOT system validity (Cohen, 2013). The results show the valid hand data from a sensor statistically significantly predicted MOT hand validity at $p < .0005$ except for the front-facing sensors left hand which was $p < .01$.

Table 6.11 - Results from the Linear Regression Test

Task	Regression Equation	ANOVA	R^2	Adjusted R^2	Increase in MOT Validity
45-degree Left Valid -> MOT Left Valid	12.464 + 0.902 * external left validity	F(1,10)=82.796, p<.0005	94.5	89.2	0.902% (95% CI, 0.681% to 1.123%)
Front-facing Left Valid -> MOT Left Valid	18.995 + 1.123 * front left validity	F(1,10)=12.201, p<.01	74.1	55	1.123% (95% CI, 0.407% to 1.839%)
Front-facing Right Valid -> MOT Right Valid	17.787 + 1.042 * front right validity	F(1,10)=35.059, p<.0005	88.2	77.8	1.042% (95% CI, 0.650% to 1.435%)
45-degree Right Valid -> MOT Right Valid	2.003 + 1.003 * external right validity	F(1,10)=1087.524, p<.0005	99.5	99.1	1.003% (95% CI, 0.935% to 1.071%)

These results concur with the results of the previous tests, showing the effectiveness of the 45-degree sensor at producing significantly more valid hand data than the front-facing sensor. Analysis of the results for the one-way ANOVA on hand data validity showed statistically significant differences at $p<.05$ between the 45-degree sensor and the MOT for all tests except those requiring both hands. In contrast the front-facing sensors showed statistically significant differences to $p <.01$. The large effect of the 45-degree sensor on the MOT system suggests greater similarity with regards to data validation and thus reduces statistically difference, increasing the p value. The results concur with the analysis of the Chi-Square test results which indicated the 45-degree sensor and MOT system were not related. These results show the 45-degree sensor to have a large effect on the MOT system, thus explaining the small difference between the expected and actual values in the Chi-Square resulting in the null hypothesis not being rejected.

Furthermore, these results concur with those of the previous tests such as the repeated measures ANOVA in 6.2.1.4, showing the significant effect of the 45-degree sensor on the validity of the MOT system.

6.2.2 – Questionnaire Analysis

The semi-structured interviews were divided into two sections with the same questionnaire completed after each condition. The interviews comprised of open questions and questionnaire styled questions using a Likert scale 1-5 (1. Strongly disagree, 5. Strongly agree). Cronbach's alpha was used to determine the reliability of the questions (Statistics, 2018), with the questionnaire reaching excellent reliability (Taylor, 2013), $\alpha = 0.907$.

Figure 6.15 shows the mean Likert scores for questions 2 to 11 of the questionnaire. The results show the MOT system was the most natural (Q7) and provided the greatest sense of engagement (Q9), scoring 2.08% and 7.3% higher, respectively. However, in the other questions the Leap condition either matched or outperformed the MOT condition by between 1.92% (Q8) and 11.11% (Q10).

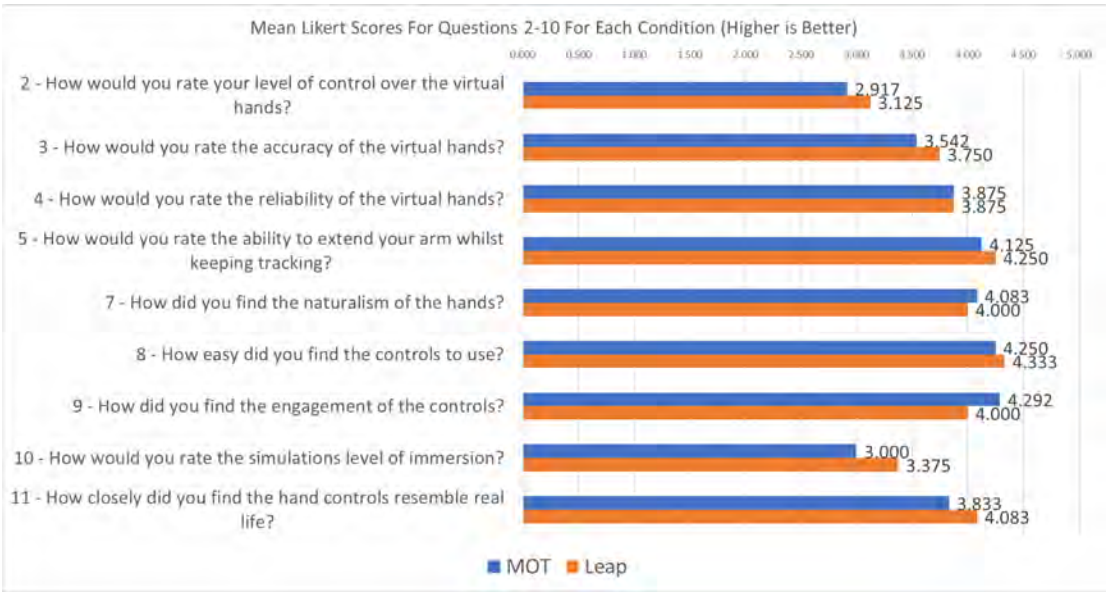


Figure 6.15 - Mean Likert scores for the interview questions

A Spearman's rank-order correlation was run to assess the relationship between the questions. There was a statistically significant, strong positive correlation between all the questions except for Q10 regarding immersion which showed no correlation with any of the other questions. Analysis of the results shows most of the questions have a strong correlation with significance of p<.01 and below. Appendix 14 shows the full results of the spearman correlation test.

Table 6.12 shows the results of a Paired-Samples T-test on the results of the questionnaire to determine whether the mean difference between the two conditions was statistically significant. The results of the paired samples t-test show no statistically significant difference for any of the interview questions.

Question	Normality (Shapiro-Wilk)	Leap Mean	MOT Mean	Paired Differences (Leap - MOT)	t (11)	Significance (p)
Control	0.378	3.125 +- 0.8013	2.917 +- 0.5149	-0.2083 (95% CI, -0.7053 to 0.2886)	-0.923	0.376
Accuracy	0.49	3.75 +- 0.866	3.542 +- 0.7217	-0.2083 (95% CI, -0.8918 to 0.4752)	-0.671	0.516
Reliability	0.051	3.875 +- 0.9654	3.875 +- 0.7424	0.0 (95% CI, -0.6058 to 0.6058)	0.0	1.0
Extension	Assessed by Skewness & Kurtosis	4.25 +- 0.8919	4.125 +- 0.9077	-0.125 (95% CI, -0.4317 to 0.1817)	-0.897	0.389
Naturalism	0.122	4 +- 0.7385	4.083 +- 0.6337	0.0833 (95% CI, -0.4724 to 0.6391)	0.330	0.748
Difficulty	0.122	4.33 +- 0.985	4.25 +- 0.965	-0.083 (95% CI, -0.871 to 0.705)	-0.233	0.820
Engagement	0.077	4 +- 1.1282	4.292 +- 0.8649	0.2917 (95% CI, -0.6316 to 1.2150)	0.695	0.501
Immersion	0.296	3.375 +- 0.9324	3 +- 0.8528	-0.3750 (95% CI, -0.9356 to 0.1856)	-1.472	0.169
Real life	0.0	4.08 +- 0.669	3.83 +- 0.835	-0.250 (95% CI, -0.645 to 0.145)	-1.393	0.191

Table 6.12 - Results of Paired Samples T-test for the Questionnaire

The mean Likert scores shown in Figure 6.15 show the Leap Motion either matched or outperformed the MOT system in most areas. However, the results of the paired samples t-test show that the differences are not significant enough to draw a conclusion on the effect of the conditions on user experience. These results contradict the quantitative results previously discussed in section 6.2.1. The quantitative results previously discussed show the MOT system to produce significantly more valid hands than the front-facing sensor. This was particularly evident in the tests regarding the longest and total period in which hand data was not available. The results of these tests showed the front-facing sensor to be missing hand data for more than half the simulations duration on average. While the results of the Likert scores show the two conditions to have identical average scores. It can be hypothesized that the intricacy of the simulation diverted participant focus away from the level of interaction and differences between the two interaction conditions. Furthermore, the simulation requires users to interact with various pieces of apparatus to perform tasks with specific settings/values required. The focus required is likely to have diverted the user's attention away from the behaviour of the virtual hands. This would have enabled periods of missing or invalid hands with the single Leap as indicated by the quantitative results, to have gone un-noticed. In addition, many of the object interactions required the user to be looking directly at the object they are interacting with. The small interaction space will have made it unlikely for the user to exceed the field-of-view limitations of the interaction conditions. Resulting in the field-of-view limitations being mostly untested and users not experiencing the interpolation synchronization in the MOT system.

6.3 – Standardized Gesture Evaluation

In this section the experimental results from the Standardized Gesture Evaluation Test are analysed. First, the analysis of the hand data logs generated during the simulation to determine data validity and periods of invalid or missing data is presented.

Secondly, the Likert-based test answers are analysed to evaluate the two optical tracking conditions through the validation of both the four key areas and user experience. A Spearman Correlation was performed to measure association between the questions (Statistics, 2018). The questionnaire answers were also analysed using paired sample t-tests to determine whether the mean difference between the two conditions was statistically significant. Participant comments made were analysed by thematic analysis (section 6.4) with inter-rater reliability testing (Cohen's Kappa).

6.3.1 – Tracking Data Validity

In this section the results of the Quantitative analysis are detailed, analysing log files generated during participant experiences to determine the effect of the conditions on data validity and periods in which invalid or missing tracking data was observed. Furthermore, the validity of the tracking data produced by the MOT system is compared against that of a single Leap Motion sensor. In the figures used "Aggregated Hands" refers to the hands produced by the MOT system. In addition, "External" refers to the 45-degree sensor.

6.3.1.1 – Chi Square Test

A Chi-Square test was conducted to analyse hand validity and determine the significance between the sensors and MOT system on hand validity produced. The test was used to determine the significance between the counts of valid and null hands observed by a Leap Motion sensor (front-facing or 45-degree angle) and the MOT system. The tests were run for both left and right hands and compared the two sensors against the MOT system and each other. The test would enable the 45-degree sensor positioning to be evaluated to determine whether it had increased hand validity as previously hypothesized. Furthermore, the test would evaluate the effectiveness of the aggregation process at further improving hand validity. Table 6.13 shows the details of the Chi-Square test including the null and alternative hypotheses.

Table 6.13 - Details of the Chi-Square test conducted

Null Hypothesis	Assumes that there is no association between the two variables.
Alternative Hypothesis	Assumes that there is an association between the two variables.
Outcome	Two Null Hypothesis Not Rejected, One Alternative Accepted

Figure 6.16 shows the average percentage of total simulation frames in which both hands were valid for each sensor and the MOT system. Figure 6.17 shows the average percentage of total simulation frames in which the left and right hands were valid for each sensor and the MOT system. Furthermore, in both figures higher bars indicate a greater number of valid frames were produced and thus better performance. The results concur with the results of the Simulation experimentation detailed in section 6.2.1.1, showing the MOT system to produce the most valid frames and the front-facing the least.

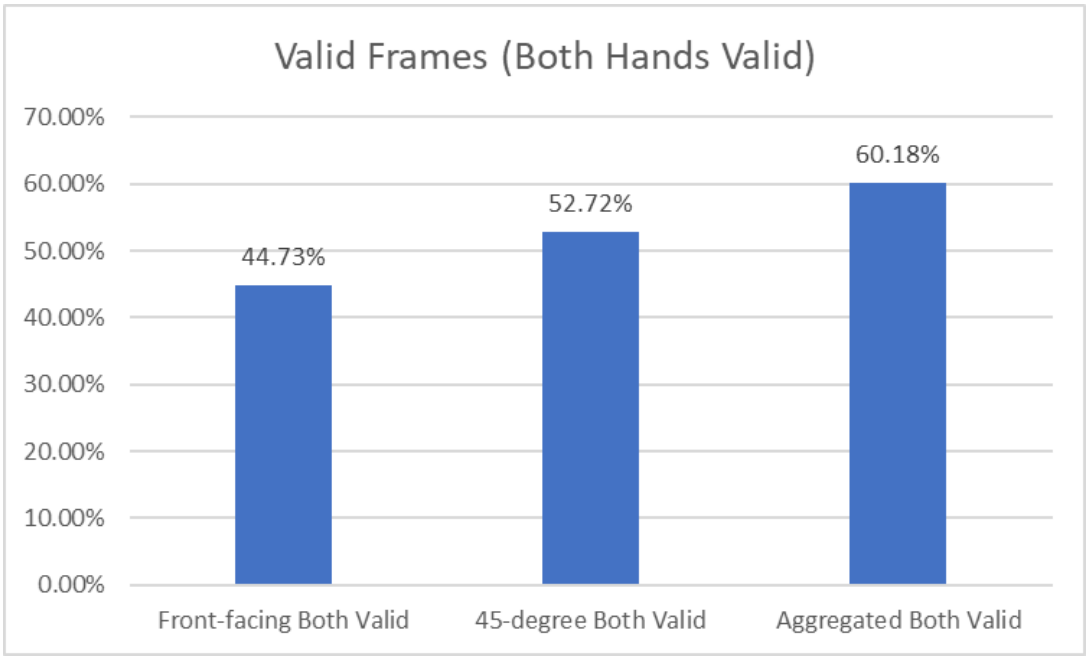


Figure 6.16 - Average of total frames in which both hands were valid during gesture experimentation

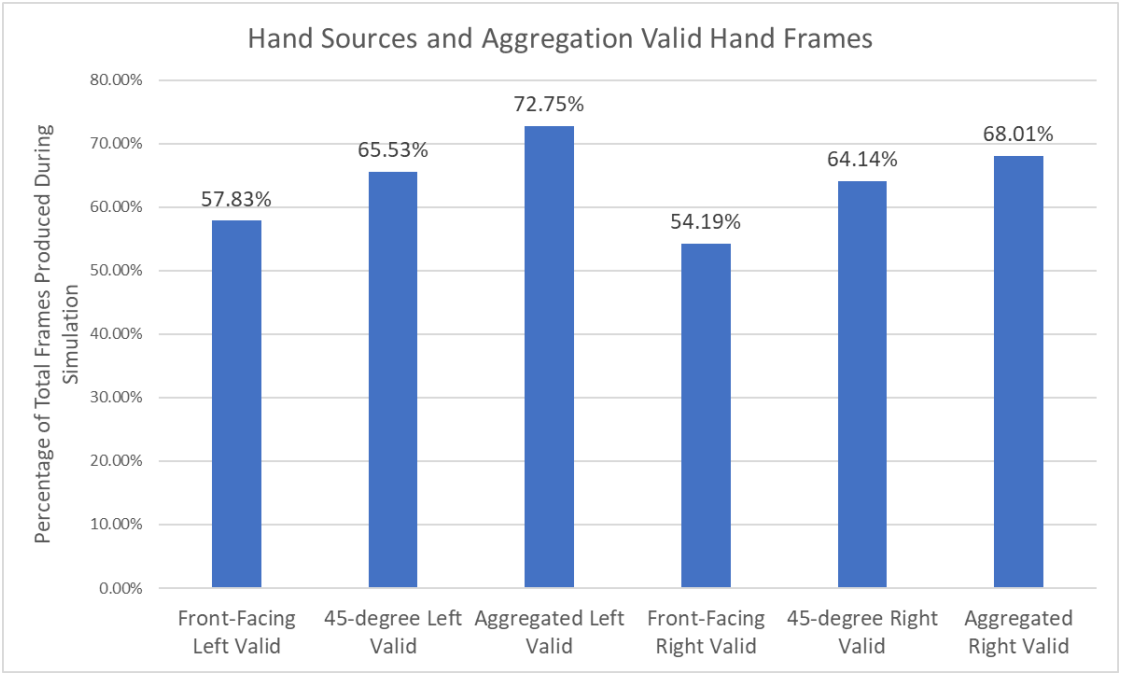


Figure 6.17 - Average Percentage of Total Frames Valid for each sensor and the MOT system for each hand

The p-value of the tests from the front-facing Leap Motion and the MOT system were all significant at $p<.05$. The tests between the 45-degree angle Leap Motion sensor and the MOT system were not significant. The tests between the front-facing and 45-degree angle Leap Motion sensor were not statistically significant. The results show that the MOT system was able to produce significantly more valid hands than the front-facing Leap Motion sensor.

Analysis of the chi-square statistics indicate greater data similarity between the 45-degree sensor and the MOT system than the front-facing sensor and the MOT system. This supports the results of the previous Chi-Square tests, indicating the 45-degree sensor as presented in this work is better positioned to recognize user hands in VR than a front-mounted configuration. As with previous Chi-Square test, the p-value results indicate statistically insignificant results between the 45-degree sensor and the MOT system. However, the small difference in validity between the two as seen in Figure 6.17, results in a small difference between the observed and expected values. Thus, the results suggest the null hypothesis cannot be rejected. The effect of the sensors on the MOT system is further evaluated by Linear Regression tests in section 6.3.1.6.

The results concur with the previous tests and the results of the pilot study detailed in section 4.3, showing the MOT system can produce significantly more frames than either sensor. In addition, the results further support those presented in sections 4.3 and 6.2.1 showing the MOT system offers an improvement in individual hand recognition over a bracket mounted sensor. This improvement is further increased when two valid hands being generated is considered.

6.3.1.2 – Paired Samples T-Test

A Paired Samples T-Test was conducted to determine whether the mean difference between the sensors and the MOT systems hand validity was statistically significantly different from zero. The valid and null data of each hand was compared across both sensors and the MOT system. In addition, the number of times two valid hands were generated was compared. In one case a Wilcoxon Signed Rank test was used due to the data not being normally distributed. A Sign test was used in three cases where the data was neither normally distributed nor symmetrically shaped. Table 6.14 shows the details of the Paired Samples T-tests including the null and alternative hypotheses. Table 6.15 shows the results of the paired-samples t-tests on hand validity for each hand and the two hands together, for each sensor and MOT system.

Table 6.14 - Details of the Paired Samples T-tests conducted

Null Hypothesis	Assumes that the mean difference between paired values is equal to zero.
Alternative Hypothesis	Assumes that the mean difference between paired values is not equal to zero.
Outcome	Alternative Accepted in all but two tests where the Null Hypothesis could not be rejected

Table 6.15 - Results of Hand Validity Paired-Sample T-test

Pairs	Normality (Shapiro-Wilk)	First Mean	Second Mean	Paired Differences (95% CI)	t(11)	Significance (p)
Front Left -> MOT Left	Assessed by Skewness & Kurtosis	57.83% +- 21.36%	72.75% +- 19.52%	-14.92% (-22% to -7.84%)	-4.638	0.001
Front Right -> MOT Right (Sign Test)	Not normally distributed	57.04% (Median)	77.33% (Median)		3.175 (z score)	0.001
Front Left -> External Left	0.277	57.83% +- 21.36%	65.53% +- 23.10%	-7.70% (-17.98% to 2.59%)	-1.648	0.128
Front Right -> External Right (Sign Test)	Not normally distributed	57.04% (Median)	71.96% (Median)		2.598 (z score)	0.009
External Left -> MOT Left	Assessed by Skewness & Kurtosis	65.53% +- 23.1%	72.75% +- 19.52%	-7.22% (-11.92% to -2.52%)	-3.381	0.006
External Right -> MOT Right	0.137	64.14% +- 27.45%	68.01% +- 26.83%	-3.87% (-5.86% to -1.88%)	-4.283	0.001
Front Both -> MOT Both (Sign Test)	Not normally distributed	43.15% (Median)	64.15% (Median)		3.175 (z score)	0.001
Front Both -> External Both (Wilcoxon Signed Rank Test)	Not normally distributed	43.15% (Median)	55.45% (Median)	2.2971% Median Increase	1.177 (z score)	0.239
External Both -> MOT Both	0.139	52.72% +- 29.77%	60.18% +- 28.70%	-7.46% (-11.13% to -3.8%)	-4.483	0.001

The results show statistically significant differences in all tests except for the front-facing to 45-degree comparison for the left hand and the front-facing to 45-degree comparison for both hands being valid. These results support the results of the first phase of testing and the Chemical Engineering simulation test, indicating the 45-degree sensor produces significantly more valid frames than the front-facing sensor. Analysis of the results shows that the mean difference between the 45-degree and MOT sensor is not zero. This shows that while the 45-degree sensor offers a more effective configuration over a front-facing sensor, the aggregation process used by the MOT system further improves hand validity.

The results show no statistically significant difference for the left hand between the front-facing and 45-degree sensor. The simulation-based experiment detailed in section 6.2, showed statistically significant differences between all pairs. The difference between the two scenarios suggests that participants

typically kept their left hand in view of the two sensors for longer than the right hand. This could be due to user handedness, resulting in a hand preference when performing gestures, with in the opposing hand remaining stationary. The discrepancy could also be the result of user preference for grasping the virtual beaker with the left hand due to conformation issues, as reported by a few participants. The conformation issues reported by a few participants is discussed further within the thematic analysis in section 6.4.

6.3.1.3 – One-way Repeated Measures ANOVA on Valid/Null Hands

A one-way within subjects (or repeated measures) ANOVA was conducted to compare the effect of each sensor and the MOT system on hand validity. The test was used to determine whether there were any statistically significant differences in the average hand validity of the two sensors and the MOT system. Thus, determining the effectiveness of the sensor configuration and aggregation process at improving hand validity. Table 6.16 shows the details of the one-way within subject's ANOVA including the null and alternative hypotheses.

Table 6.16 - Details of the One-Way Within Subjects ANOVA test conducted

Null Hypothesis	The null hypothesis states that the related population means are equal.
Alternative Hypothesis	The alternative hypothesis states that the related population means are not equal.
Outcome	Alternative Accepted in all but one test where the Null Hypothesis could not be rejected

Table 6.17 shows the results of a one-way repeated measures ANOVA test. The results showed statistical significance in all but one test between the 45-degree sensor and the MOT system, with the exception being the both null hands test. The results also showed statistically significant results in all but

two tests between the front-facing sensor and the MOT system, with the exceptions being the 'Right Hand Valid' and 'Right Hand Null' tests. There were no statistically significant results between the front-facing and 45-degree sensors.

Table 6.17 - One-way repeated measures ANOVA for valid and null hands

Hand Data	Sphericity	Epsilon Correction	Within-Subjects Effects	Post hoc Analysis (Bonferroni Correction) (95% CI)		
				Front-Facing to 45-degree	Front-Facing to MOT System	45-degree to MOT System
Left Hand Valid	0.003 (Violated)	0.593	F(1.186,13.044)= 9.090, p=.008	7.700 (5.478 to 20.877), p=0.383	14.922 (5.848 to 23.995), p<.01	7.222 (1.198 to 13.246), p<.05
Left Hand Null	0.003 (Violated)	0.591	F(1.182,13.0)= 9.123, p=.008	-7.723 (-20.951 to 5.505), p=0.384	-14.985 (-24.052 to -5.919), p<.01	-7.263 (-13.307 to -1.218), p<.05
Right Hand Valid	<.0005 (Violated)	0.521	F(1.043,11.469)= 5.830, p=.032	9.959 (-4.762 to 24.680), p=0.249	13.827 (-0.075 to 27.730), p>.05	3.868 (1.321 to 6.416), p<.01
Right Hand Null	<.0005 (Violated)	0.522	F(1.043,11.473)= 5.830, p=.032	-9.927 (-24.659 to 4.804), p=0.252	-13.847 (-27.754 to 0.060), p>.05	-3.920 (-6.480 to -1.359), p<.01
Both Hands Valid	<.0005 (Violated)	0.561	F(1.121,12.331)= 6.105, p=.026	7.983 (-7.878 to 23.844), p=0.551	15.448 (1.561 to 29.334), p<.05	7.465 (2.769 to 12.160), p<.01
Both Hands Null	0.009 (Violated)	0.622	F(1.243,13.675)= 10.836, p=.004	-9.679 (-20.211 to 0.852), p=0.075	-13.278 (-22.115 to -4.440), p<.01	-3.598 (-7.916 to 0.720), p>.05

Analysis of the results in comparison to those of the simulation experiment detailed in section 6.2.1, shows statistically significant differences between the two sensors and MOT system. However, the results presented in Table 6.17 show no statistically significant differences between the front-facing and 45-degree sensors. This is likely due to the difference in user movement requirements between the chemical engineering simulation and the gesture evaluation test. The simulation experience required a much greater range of movement from the user to interact with the various objects within the virtual environment whilst monitoring the virtual equipment. In contrast, the gesture evaluation enabled users to keep their hands within a relatively consistent space typically within the field of view of both sensors. Some of the gestures such as the reaching gesture, resulted in the front-facing field-of-view being

exceeded. The results of which can be seen Figure 6.17 with the increased validity for the 45-degree sensor compared to the front-facing sensor. However, during the experience the user's hands will have generally remained within the field-of-view of both sensors. This will have resulted in the relatively small difference in validity compared to those seen in Figure 6.10 for the simulation. Thus, the difference was found to be statistically insignificant.

The results of the front-facing and MOT system comparison concur with the previously discussed results showing the effectiveness of the aggregation approach presented at improving optical hand tracking. Furthermore, the results help to further answer the academic question, showing how multi-sensor aggregation improves factors such as the tracking field-of-view and the validity of the data produced. The insignificant results between the 45-degree sensor and MOT system for both hands being null, concurs with the previously discussion showing the significant effect of the 45-degree sensor on MOT system validity. As the 45-degree sensor has the greatest impact on the MOT system, in events where the sensor has no hand tracking data available, the MOT system will be dependent on the front-facing sensor. However, as previously discussed, the results have shown, in such cases the front-facing sensor will typically have no hand data. In some cases, hand data will be available which results in the small statistically insignificant increase shown in Figure 6.16.

6.3.1.4 – One-way Repeated Measures ANOVA on Total Time with No Hands

A one-way within subjects (or repeated measures) ANOVA was conducted to compare the effect of each sensor and the MOT system on the total time in which no hand data was produced. The test was used to determine whether there were any statistically significant differences in the total time in which no hand data was produced by the sensors and MOT system. Thus, determining the effectiveness of the sensor configuration and aggregation process at improving the field-of-view and tracking stability. Table 6.18 shows the details of the one-way within subject's ANOVA including the null and alternative hypotheses. Figure 6.18 shows the average total time without each hand for each sensor and the MOT system. Figure 6.19 shows the total time without both hands for each sensor and the MOT system.

Table 6.18 - Details of the One-Way Within Subjects ANOVA test conducted

Null Hypothesis	The null hypothesis states that the related population means are equal.
Alternative Hypothesis	The alternative hypothesis states that the related population means are not equal.
Outcome	Some Null Hypothesis Not Rejected, Some Alternative Hypothesis Accepted

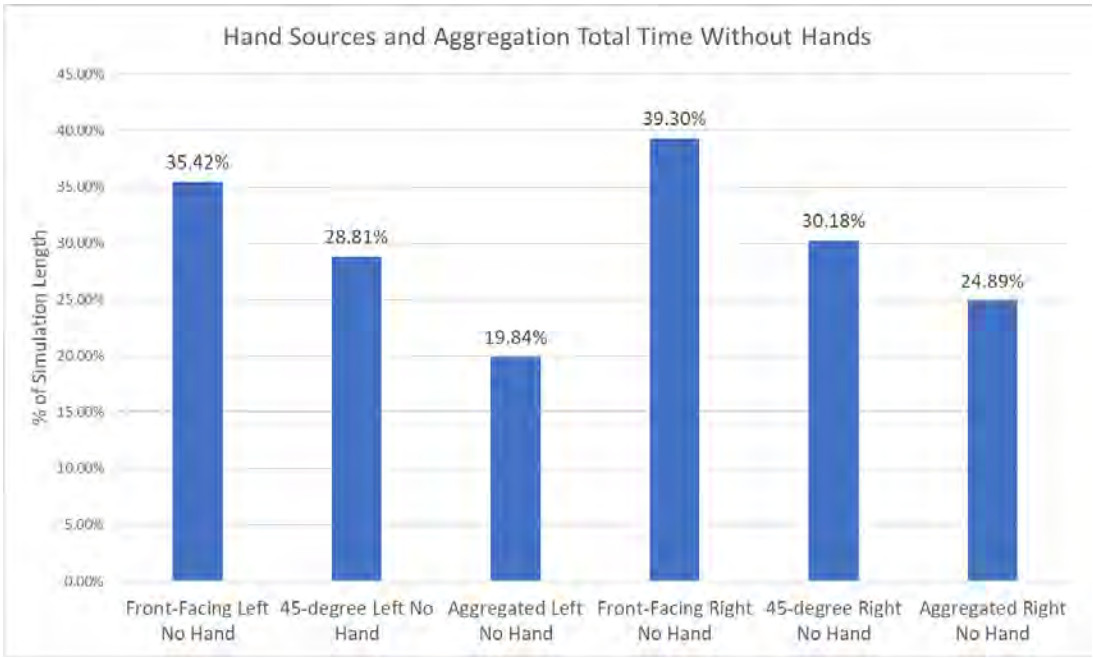


Figure 6.18 – Average total Time without hands (% of Simulation Length) for each source and the MOT system

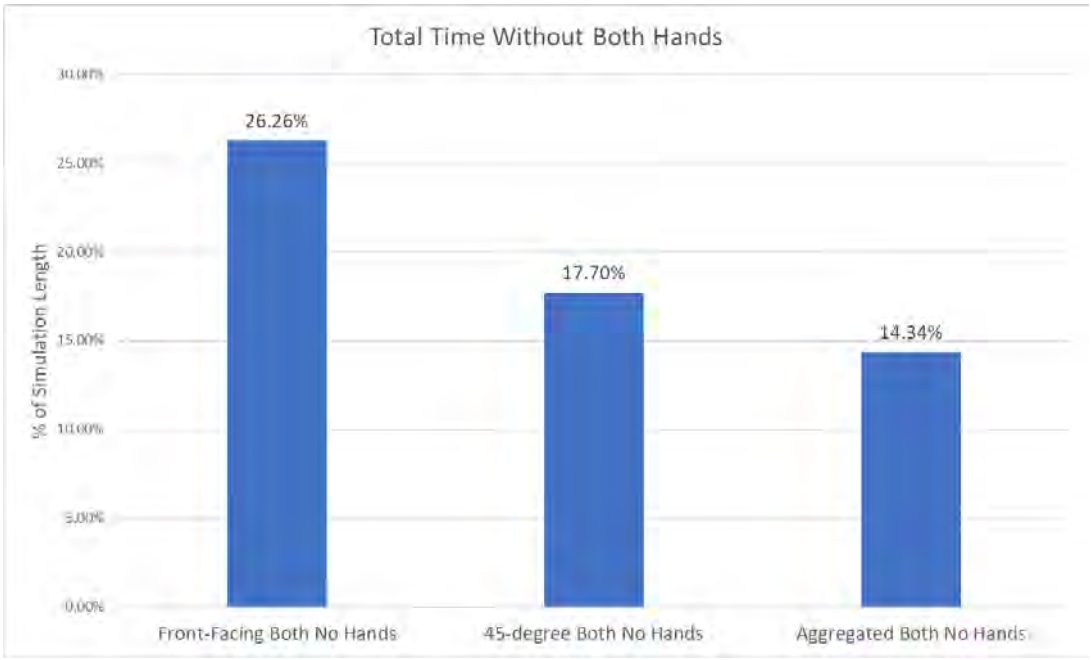


Figure 6.19 – Average total time without both hands (% of Simulation Length) for each source and the MOT system

The results in Table 6.19 show no statistically significant difference in tests between the sensors. In addition, no statistically significant difference was found in the “No Hands” test between the 45-degree sensor and the MOT system. However, statistically significant results were found in all three tests between the front-facing sensor and the MOT system, with significant results also found for the “No Left Hand” and “No Right Hand” tests between the 45-degree sensor and the MOT system.

Table 6.19 - One-way repeated measures ANOVA results for total time with no hands

Hand Data	Sphericity	Epsilon Correction	Within-Subjects Effects	Post hoc Analysis (Bonferroni Correction) (95% CI)		
				Front-Facing to 45-degree	Front-Facing to MOT System	45-degree to MOT System
No Left Hand	0.012 (Violated)	0.630	F(1.260,13.857)= 10.241, p<.01	-6.614 (-19.558 to 6.329), p=0.532	-15.581 (-23.571 to -7.591), p<.01	-8.967 (-16.284 to -1.649), p<.05
No Right Hand	0.001 (Violated)	0.563	F(1.126,12.386)= 6.550, p<.05	-9.111 (-23.082 to 4.859), p=0.279	-14.407 (-27.666 to -1.148), p<.05	-5.296 (-9.283 to -1.310), p<.01
No Hands	0.019 (Violated)	0.646	F(1.293,14.221)= 10.271, p<.01	-8.562 (-18.148 to 1.024), p=0.086	-11.919 (-20.019 to -3.819), p<.01	-3.357 (-7.598 to 0.883), p>.05

The results indicate the sensors had similar time periods in which they were unable to track the hands with no statistical difference between the two but both statistically different to the MOT system. The results of the “No Hands” test supports the results of the previous tests showing the 45-degree sensor provides significantly more valid frames. This is further supported by the results showing no statistically significant difference between the 45-degree sensor and the MOT system, but a strong statistical difference between the front-facing and MOT system. This indicates that times where the MOT system had no hands were caused by the 45-degree sensor being unable to see the hands with the front-facing sensor having already lost track.

6.3.1.5 – Longest Time with No Hands

Two Friedman tests were conducted to analyse the effect of the front-facing sensor, 45-degree sensor and MOT system at reducing the time in which no hand data is available. The test was used to determine whether there were any statistically significant differences in the average of the longest period in which no hand data was available. Thus, determining the effectiveness of the sensor configuration and aggregation process at improving field-of-view, stability and hand validity. Table 6.20 shows the details of the one-way within subject's ANOVA including the null and alternative hypotheses. Figure 6.20 shows the average of the longest time without a given hand for each sensor and the MOT system.

Table 6.20 - Details of the Friedman tests conducted

Null Hypothesis	The null hypothesis states that the distribution of scores in each group are the same
Alternative Hypothesis	The alternative hypothesis states at least two of the group's distributions differ.
Outcome	Some Null Hypothesis Not Rejected, Some Alternative Accepted

The results show the MOT system to have the smallest maximum period in which tracking data for a hand was not available. Furthermore, the results show the effectiveness of the 45-degree sensor in comparison to a front-facing sensor with better performance for both the left and right hand. The results show a statistically significant decrease ($p<.01$) for both hands, between the front-facing and MOT system. However, there was no statistically significant difference between the two sensors or the 45-degree sensor and the MOT system, for either hand. A one-way repeated measures ANOVA was

conducted which showed no significant decrease for the longest time with neither hand.

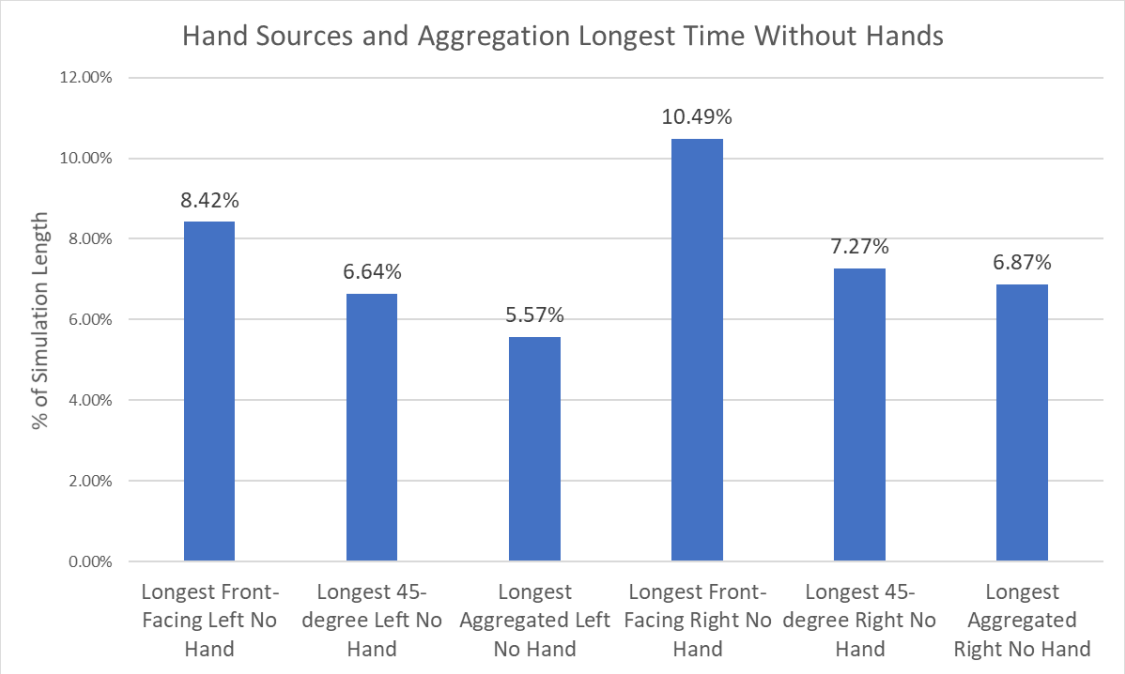


Figure 6.20 - Average longest time without hands (% of Simulation Length) for each source and the MOT system

The results show a statistically significant difference between the front-facing sensor and MOT system. However, no statistically significant difference was found between the two sensors or the 45-degree sensor and the MOT system. The results contrast with those of the simulation which showed statistically significant decreases between the sensors and between the 45-degree sensor and MOT system in the case of the right hand. Analysis of the performance of the MOT system shown in Figure 6.20 further demonstrates the effectiveness of the MOT systems aggregation approach with the time statistically significantly reduced compared to the front-facing sensor.

Analysis of the results in comparison with those of the one-way repeated measures ANOVA discussed in section 6.3.1.4 shows the results concur on the effect of the 45-degree sensor on the MOT system. As previously discussed in section 6.3.1.3, the insignificant difference between the two sensors is likely due to the user's hands generally remaining within the tracking space of both sensors during gestures. Some of the gestures performed required the field-of-view of sensors to be exceeded, which can be seen in the difference between the two as shown in Figure 6.20. However, the user's hands were typically within the optimal detection space resulting in the statistically insignificant results found.

6.3.1.6 – Linear Regression on MOT Hand Validity

A Linear regression test was performed to assess the linear relationship between the sensors and MOT hand validity. The test was used to determine the statistical significance of any relationship and how much variation in the MOT systems hand validity is explained by a sensor's validity. Table 6.21 shows the details of the conducted linear regression test including the null and alternative hypotheses. Table 6.22 shows the results of the linear regression test performed to assess the linear relationship between the sensors and MOT hand validity.

Table 6.21 - Details of the Linear Regression test conducted

Null Hypothesis	Assumes that there is no relationship between the two variables. The null hypothesis states that the coefficient of the slope is equal to zero.
Alternative Hypothesis	Assumes that there is a relationship between the two variables. If there is a significant linear relationship between the independent variable X and the dependent variable Y, the coefficient of the slope will not equal zero.
Outcome	Null Hypothesis Rejected, Alternative Accepted

The results indicate the 45-degree sensor had a large effect size with the front-facing at a medium effect size according to (Cohen, 2013). The results from previous tests indicate stronger evidence against the null hypothesis of the relationship between the front-facing sensor and the MOT system than the 45-degree sensor and MOT system. However, the results concur with those of the previous linear regression tests (4.3.4 and 6.2.1.6) demonstrating the 45-degree sensor was much more effective in producing valid hand data for the MOT system.

Table 6.22 - Results of a Linear Regression Test using Validity data from the Gestures Test

Task	Regression Equation	ANOVA	R^2	Adjusted R^2	Increase in MOT Validity
Front-facing Left Valid -> MOT Left Valid	27.550 + 0.782 * local left validity	F(1,10)=27.194, p<.0005	73.1	70.4	0.782% (95% CI, 0.448% to 1.116%)
45-degree Left Valid -> MOT Left Valid	19.941 + 0.806 * external left validity	F(1,10)=99.970, p<.0005	90.9	90	0.806% (95% CI, 0.626% to 0.985%)
Front-facing Right Valid -> MOT Right Valid	24.552 + 0.802 * local right validity	F(1,10)=17.269, p<.01	63.3	59.7	0.802% (95% CI, 0.372% to 1.232%)
45-degree Right Valid -> MOT Right Valid	5.730 + 0.971 * external right validity	F(1,10)=775.913, p<.0005	98.7	98.6	0.971% (95% CI, 0.893% to 1.049%)

The results concur with the analysis of the Chi-Square test results which indicated the 45-degree sensor and MOT system were not related. These results concur with that of previous tests, showing the 45-degree sensor to have a large effect on the MOT system. Thus, explaining the small difference between the expected and actual values in the Chi-Square resulting in the null hypothesis not being rejected. Furthermore, these results concur with those of the previous tests such as the repeated measures ANOVA in 6.3.1.4, showing

the significant effect of the 45-degree sensor on the validity of the MOT system.

Analysis of the results in comparison to the Linear regression test conducted on the simulation in section 6.2.1.6 shows the front-facing sensor to have a different impact. The left hand shows a 28% increase with the right hand experiencing a 23% decrease with regards to the adjusted R^2. The improvement seen in the left hand is likely due to the user's hands remaining within the detection space of both sensors during several of the gestures. While the decrease in the right hand is due to a combination of factors. The first being the absence of the virtual control box users interacted with during the simulation. The task was typically performed using the right hand which kept the hand within the optimal tracking range of the front-facing sensor for a prolonged period. The second factor being the position in which users rested their hands between gestures resulting in variance in hand visibility and by extension validity, based on the hands visibility to each sensor.

6.3.2 – Standardized Questionnaire Analysis

The gesture evaluation questionnaire was divided into several sections with the same questions asked after each gesture. A series of overall sections were asked once the gestures had been completed. The questions used a Likert scale 1-7 (1. Strongly disagree, 7. Strongly agree). Cronbach's alpha was used to determine the reliability of the questions (Statistics, 2018), with the questionnaire reaching excellent reliability (Taylor, 2013), $\alpha = 0.918$.

Figure 6.21 shows the mean Likert scores for the different gestures completed and the sections regarding the overall experience such as; Naturalness and Satisfaction. The results show the MOT system to outperform the Leap condition in most categories. Gestures such as the occlusion gesture demonstrated the effectiveness of the MOT systems occlusion detection system, with a 93.83% increase in total score compared to the Leap condition. The results also show the effectiveness of the stability detection, interpolation and inferred hand pose systems at providing a consistent experience and increased field-of-view. These systems were particularly effective during gestures such as the “Sides Gesture” and “Reaching Gesture” where the user must move their hands outside of the sensors field-of-view and back into view.

In addition, the results concur with the results of the thematic analysis where users indicated they felt the Leap condition had a smaller field-of-view and they could see the hands disappear as a result. Analysis of the scores given for the hands disappearing question on the “Sides Gesture” shows the Leap condition with an average score of 4.42 compared to an average of 6.63 for the MOT condition, indicating a significant improvement. These results demonstrate the improvement the additional MOT system functionality implemented provides, enabling users to exceed the tracking space with a smooth and consistent experience.

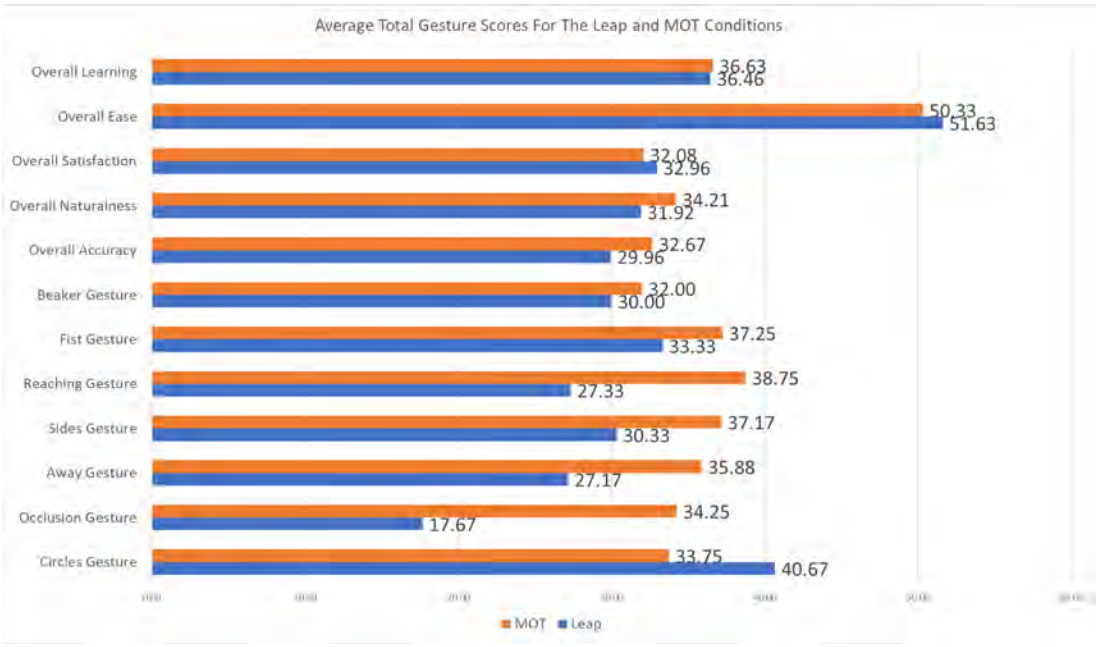


Figure 6.21 - Average Total Gesture Scores for the Leap and MOT conditions

The average scores for Learning, Ease and Satisfaction show very little difference between the two conditions. In the case of the Learning and Ease this is likely due to the two conditions using the same interaction approach, using a user's hands for interaction. The similarity in the satisfaction results is likely due to a combination of the simplistic test environment and using the same interaction approach. The naturalness question showed a small improvement in favour of the MOT condition (34.21% compared to 31.92%). Based on the results of the thematic analysis this is likely due to visual appearance of the hand which had been described as "terminator-ish" limiting the natural feeling of the interaction despite the much more positive feedback the MOT system received.

The questions relating to the beaker translation task showed a small improvement in favour of the MOT condition (32% compared to 30%). Analysis of the thematic analysis showed participants liked the mesh conformation systems effect on the appearance of the virtual hand. However, a few participants did report issues with the right hand's conformation. Furthermore, the simplicity of the task and the identical object interaction mechanics will have resulted in the only difference between the two conditions being the mesh conformation system.

The circles gesture shows the Leap condition to outperform the MOT condition (40.67% compared to 33.75%). Based on the feedback received and the thematic analysis, this is likely due to the sensitivity of the interpolation system introducing slight elements of the lag, most commonly in the right hand. As further discussed in section 6.4.5, this occurred due to participants hands repeatedly exceeding the sensors field-of-view causing the interpolation system to be triggered each time to smooth out any noisy data.

A Spearman's rank-order correlation was run to assess the relationship between the questions. There was a statistically significant strong correlation between almost all the questions except for the circle gesture which showed no correlation with any of the other questions. The 'Naturalness' questions showed significant correlation with "Satisfaction", "Ease" and "Learning" indicating more natural interaction provides users with the familiarity of real-world interactions thus leading to a more intuitive and satisfying experience. Analysis of the results shows most of the questions strongly correlate with many having significance of $p<.01$. The complete results of the spearman correlation test can be found in Appendix 15.

A paired samples t-test was run to determine whether the mean difference between the two conditions was statistically significant. The results of the test show a statistically significant difference between the two conditions for the circles, occlusion, away and reaching gesture questions. The other sections showed no statistically significant difference. The results show field-of-view and occlusion to be the greatest factors in user experience, with gestures that tested these aspects of the two conditions demonstrating statistically significant differences. Table 6.23 shows the results of a Paired-Samples T-test conducted using the results of the gesture questionnaire.

Table 6.23 - Results of Paired Samples T-test for Gesture Questionnaire

Gestures (Leap->MOT)	Normality (Shapiro-Wilk)	Leap Mean	MOT Mean	Paired Differences (95% CI)	t (11)	p
Circles	0.924	40.67 +- 3.420	33.75 +- 7.124	6.917 (1.435 to 12.399)	2.777	0.018
Occlusion	0.185	17.67 +- 6.583	34.25 +- 7.275	-16.583 (-24.064 to -9.102)	-4.879	<.0005
Away	0.233	27.167 +- 9.134	35.875 +- 7.01	-8.7083 (-15.1173 to -2.993)	-2.991	0.012
Sides	0.155	30.33 +- 11.5	37.17 +- 4.174	-6.833 (-14.394 to 0.727)	-1.989	0.072
Reaching	0.445	27.33 +-12.025	38.75 +- 4.751	-11.417 (-18.432 to -4.402)	-3.582	0.004
Fist	Skewness and Kurtosis	33.33 +- 9.3258	37.25 +- 5.4083	-3.9167 (-10.1166 to 2.2833)	-1.390	0.192
Beaker	0.281	30 +- 8.235	32 +- 6.368	-2 (-7.126 to 3.126)	-0.859	0.409
Overall	0.967	29.958 +- 9.502	32.667 +- 6.7161	-2.7083 (-10.3645 to 4.9479)	-0.779	0.453
Naturalness	0.548	31.917 +- 9.13	34.208 +- 9.4951	-2.2917 (-9.3206 to 4.7373)	-0.718	0.488
Satisfaction	0.612	32.958 +- 6.777	32.083 +- 7.7131	0.8750 (-3.5563 to 5.3063)	0.435	0.672
Ease	0.362	51.625 +- 9.852	50.333 +- 11.0296	1.2917 (-4.7306 to 7.3139)	0.472	0.646
Learning	Skewness and Kurtosis	36.458 +- 5.255	36.625 +- 5.6614	-0.1667 (-2.1728 to 1.8395)	-0.183	0.858

6.4 – Thematic Analysis

Interview results and observations were transcribed by the main investigator and an analysis carried out in line with the six phases of Thematic Analysis as proposed by Braun and Clarke (2006). Four main themes and seventeen subthemes were identified. Figure 6.22 shows an overview of the themes, structure and relative code distribution within each theme. The diagram showing the breakdown of percentage subtheme coverage within a theme was generated in NVivo 11, software used as part of the presented thematic analysis. To validate the themes an inter-rater reliability test was conducted using Cohen's Kappa with a scoring of 0.79, indicating substantial agreement (Stephanie, 2014). As per the guideline suggested by Lombard et al (2002), a sample size of three (25%) was randomly selected for the inter-rater reliability testing.

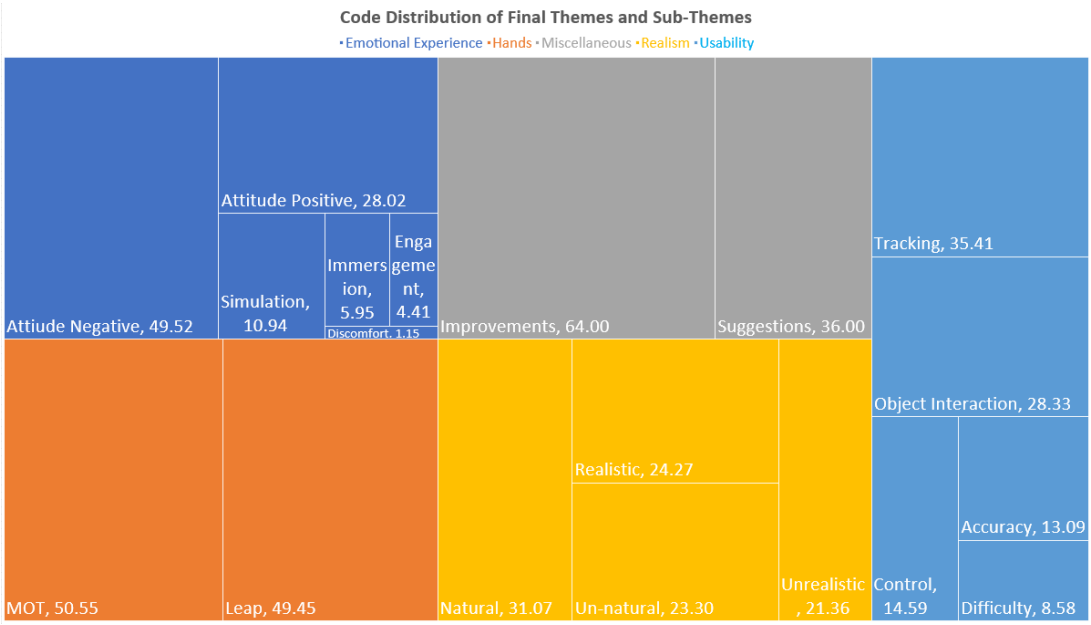


Figure 6.22 - Code distribution of the final themes and subthemes (Top Left – Emotional Experience, Bottom Left – Hands, Top Middle – Miscellaneous, Bottom Middle – Realism, Right Side – Usability)

Emotional experience focuses on the participants psychological feelings towards the experience. Miscellaneous explores participants comments on improvements that could be made or future experiences. Realism summarizes participants thoughts on the degree to which the simulation and interaction techniques replicate real world interactions. Usability summarizes participants thoughts on the ease and manner of interaction. The coding distribution of the themes indicates that participants main topic of discussion included feelings of the naturalism, realism and the hands tracking. In the following subsections, the main thematic themes that emerged are analysed.

6.4.1 – Emotional Experience

Analysis of the interviews shows that participants generally considered the MOT system to provide the more consistent experience of the two interaction conditions. A few participants made minor comments regarding the weight of the HMD and the Vive controllers positioned on their arms, “quite heavy [VR controllers on arm], well you know reasonably heavy”. The weight of the HMD could be due to the additional weight of the leap sensors and bracket. However, none of the participants explicitly stated this or indicated that the bracket had any effect on the VR experience. The Leap condition was found to provide a smaller field-of-view with the hands quickly dropping off as they move away from the optimal tracking area. In contrast the MOT system was found to have a much wider field of view, maintaining its accuracy over a larger area, “so like here, I’ve lost them but I’m expecting to see them about here”, “because of the fact that you get to here and It’s all gone”.

During experimentation, several participants pushed the systems to see if they could break them, performing gestures such as interlocking fingers, “I wanted to see what I could do with them so I was like crossing my fingers and seeing how they responded and stuff like that”. The MOT system was able to detect the hand occlusion and present the user with inferred hands to ensure the virtual hands did not disappear and allow the user to perform the gesture/pose.

The results show the effectiveness of the occlusion detection system at detecting hand occlusion while performing complex gestures/poses and triggering the inferred hand pose system before the tracking data becomes unstable. Furthermore, the results show the effectiveness of the stability detection system with inferred hands used until the tracking data had stabilized.

The combination of the occlusion detection system, stability detection system and the inferred hand pose system, enabled users to perform complex poses resulting in data instability with a consistent user experience. Overall, participants found both conditions engaging due to the ability to interact with their hands, “it was a nice change of pace to be honest to actually feel like I’m just using my own hands rather than holding a controller”.

6.4.2 – Hands

Participants reported the Leap condition as providing a sense of slightly higher accuracy when the hands are within the optimal tracking area, “I found it was better, I found it like the interaction just to be a little better, a little bit crisper, maybe a little bit more precise”. However, as with the field-of-view, the accuracy quickly dropped off as the hands moved away from the optimal

tracking area resulting in the hands disappearing. In contrast, the MOT system had slightly lower accuracy in comparison within the optimal tracking area but maintained its accuracy over a larger area, “at one point I was adjusting the strap on my wrist here and I was moving my fingers and it [MOT] registered that I was moving my finger so, I think that was very good cause I could see that”. These results show the effectiveness of both the stability detection and interpolation systems at supporting the aggregation process to provide a more consistent and stable experience.

The MOT systems inferred hand pose system was heavily praised with participants commenting on the fact that they could always see their hands, “I had taken for example, for granted that the other hands [MOT] yeah, the other hands [MOT] were always just there”, “…in the first version [MOT] upon entering the periphery and being seen I can see the previous hand but with this version [Leap] I see nothing but then my real hands pose”. In contrast the Leap condition was reported by some as being rather intermittent “they kept disappearing on me, as soon as I looked up even an inch, one or two hands would disappear, so my perception was that their presence was very intermittent”. These results show the positive effect the inferred hand pose system had on creating a consistent user experience. Furthermore, the results show the improvement the use of the extended functionality provides to the inferred hand pose system. The extended functionality enabled instability or invalid hands to be detected with the hands smoothly transitioned to an inferred pose for a consistent experience. This provided the user with hands that could always be seen.

For both conditions, some participants suggested that a more realistic hand appearance along with the addition of an arm model would help the hands feel more connected and immersive. As opposed to seeing “two hands floating around” which were described as feeling “terminator-ish”, “you haven’t exactly got a skin on them to make them look like real hands” or resembling that of a Gorillapod tripod. Research such as that of Lugrin et al (2015) has shown user sensitivity to virtual arm appearances with participants often complaining if the arm length did not match. However, Tran et al (2017) showed no significant difference in user perception of “hand only”, “hand and forearm” or “hand and entire arm”. Furthermore, the results showed “hand only” to significantly outperform an entire arm representation and often the forearm representation. Lin & Jörg (2016) concur with Lugrin et al (2015) on user sensitivity to appearance with their results indicating sensitivity as the appearance becomes more realistic.

Participant feedback concurs with the results of research of Lin & Jörg (2016), Lugrin et al (2015) and Tran et al (2017). The feedback shows most participants found the hands to be an engaging experience with the appearance of the virtual hand representation having little impact. Some participants suggested a more realistic hand representation would provide a more immersive experience. However, research suggests that virtual hand appearance is subject to the Uncanny Valley effect with hands that are too realistic having a negative impact. The Uncanny Valley hypothesis suggests that more human characteristics equal more acceptance up to a certain point after which there occurs a sudden dip in response due to subtle imperfections of appearance and/or movement.

6.4.3 – Realism

The natural movement used by both conditions to control the respective virtual hand representations was praised for feeling natural and intuitive with many of the real-world movements replicated, “my instinct was the exact movement to do or what expected to do, what you expected me to do was exactly what was natural to me”. However, in both conditions participants stated that the virtual hand representation had led them to believe that they would be able to interact with finer control than that which was available in the simulation, “as I say I struggled with the fine control in terms of getting the accuracy compared to the real world in terms of how much movement do I have to put in”. The extended MOT system functionality provided users with a smoother, more consistent, and stable experience. However, neither condition provided users with the fine hand control the virtual hand representation had led them to assume. The fine control the MOT system provides is an area for further work and improvement.

Participants reported issues with hands disappearing in the Leap condition, as they approached the edges of their field-of-view and when the two virtual hand representations were in proximity or interacting with each other, “it did lose of track on the very edges my hands disappeared”, ”the fact that the hand underneath was disappearing it was not tracked perfectly”, “if I did bring my two hands together it was a problem” . In contrast the MOT system enabled users to intersect fingers and occlude hands due to the occlusion detection system working with the inferred pose system to ensure a virtual hand was displayed. Participant feedback indicates the occlusion detection system and stability detection system provide an effective approach to handling their respective issues. As previously discussed in section 6.4.1, the occlusion

detection system enabled participants to perform complex and natural, two handed interactions such as interlocking fingers. The stability detection system combined with the increased field-of-view through aggregation enabled users to explore the full tracking range of the sensors with consistent hand tracking.

Participants felt the object interaction mechanisms were a bit un-realistic partly due to the simplicity of the interaction. However, the mesh conformation system received positive feedback with users enjoying the natural looking grasp pose “ah there we go, see that’s what I wanted to see [hand conforming around beaker], that’s better”. During the Leap condition, some participants experienced issues with the beaker pouring task, as they were unable to see their virtual hand and therefore were uncertain on whether the fist gesture was being performed, “the fingers were disappearing into the beaker I couldn’t tell whether the hand was around the beaker whether the fingers were in the right position to hold it”. A few participants experienced issues with the MOT systems right-hand deforming when attempting to conform around the beaker, “the right had severe issues whereas the left sort of snapped on and then I was able to tilt it anyway that I wanted to whereas the right was just sort of like rubber banding everywhere”. The left hand was reported as conforming perfectly, with the right hand sometimes taking a few attempts for it to conform correctly.

The object controls implemented into the virtual environment used a simplistic design as detailed in Appendix 11. Some participants reported the object controls as being too simplistic compared to what they were expecting and stated a preference for the object interactions to use more realistic and complex controls. In addition, some participants reported frustration with the

virtual object controls due to the automatic handling once near the required setting, “some of the interactions it seemed that they were almost scripted like if it were to turn a dial as soon as I approached it”.

6.4.4 – Usability

One of the big factors affecting usability was tracking stability, with the single Leaps disappearing hands found to be rather frustrating, “they would often disappear, and I would have to wiggle my fingers for them to come back”. In the case of the MOT system, the inferred hand pose system would provide the user with a virtual hand representation. However, sometimes the representation provided would not perfectly align the orientation of the users hand, “well the one thing that surprised me is that they did seem to flicker like when I was trying to press the buttons on the right, I was like this [pointing with palm down] and the virtual hand was like this [pointing with palm up, slightly off his hands rotation] and I would have to shake my hand a bit and it would sort of flicker back into place”.

The results show that the inferred hand pose system is effective in providing the user with a virtual hand representation. However, the tracking information from the Vive tracking system and the inferred hand pose are not suitable for all scenarios. Participants who had VR experience identified the functionality of the inferred hand system stating that they noticed it appeared to be presenting either the previously valid hand or an approximation. In the case of most participants, the presence of a non-interactable virtual hand representation was found to cause a minor sense of confusion. However, all participants reported the disappearance of virtual hands in the Leap condition, with most finding it frustrating, particularly when trying to interact with a virtual

object. The results show that having a non-interactable virtual hand is effective in providing a consistent user experience. However, its effects are limited to purely visual representations and basic object interaction, using the inferred hand pose and mesh conformation systems. Furthermore, the results show the limitations of the motion controller tracking data with it unable to detect real-time changes in hand orientation. This can result in a discrepancy between the orientation of the inferred and live hand, resulting in the user having to compensate.

Participant feedback has also shown that the weight and mounting of the controllers can result in them moving on the user's arm. This can result in a discrepancy in virtual arm length until the user's hands are within the required proximity of the optimal tracking position and the arm length is re-calculated. Some participants reported the movement of the trackers to be annoying as they had to re-adjust them. In addition, some participants reported finding the trackers to be "a bit heavy" on their arms, particularly towards the end of the experiment.

The occlusion detection system was shown to be effective in providing an inferred virtual hand representation during periods of occlusion. However, as a virtual hand was rendered, participants presumed it was fully interactable and thus participants reported feedback such as "they didn't disappear but the bottom one isn't functioning" and "top hand is fine but once again the bottom hand, glitchy, it doesn't disappear but it's not moving really".

6.4.5 – Thematic Discussion

The results of the thematic analysis show participants found the MOT system to provide a more consistent experience. Participant feedback indicates the MOT system addresses the key issues of field-of-view, occlusion and stability with accuracy partially addressed. Participants reported the Single Leap condition as having a smaller field-of-view often resulting in the virtual hands disappearing when participants expected to see them. The smaller field-of-view quickly became apparent during general movement with feedback such as “so like here, I’ve lost them but I’m expecting to see them about here” while the MOT system was reported as “always being there”. The single Leap condition were reported as feeling more precise when the hands were within the optimal tracking area. However, the smaller field-of-view resulted in the accuracy quickly decreasing as the hands moved outside the optimal tracking area.

In contrast the MOT system was found to maintain accuracy over a wider field-of-view but have slight latency in comparison due to the interpolation system. The results show the interpolation system was effective in smoothing the transition/synchronization to the live tracking data from a motion-controlled hand. However, the real-time detection of “jumps” in tracking position and the automatic angle interpolation was found to be “sensitive” with some users experiencing periods of slight latency. The main occurrence was during the circles gesture as part of the Standardized Gesture Evaluation Test where the hand would frequently partially exceed the sensors field-of-view, triggering interpolation and an inferred hand.

The inferred hands system was found to be successful in ensuring the user consistently had two virtual hands. However, an issue highlighted by several participants was the perceived control over the inferred virtual hand, due to believing it was their hand being tracked. In contrast, while the disappearance of the virtual hands during the single Leap condition was found to be frustrating, if the hand was visible it was generally controllable. The results show that an inferred hand provides a more consistent user experience, displaying a visual representation and handling basic object interactions. However, the absence of presumed control can have minor negative effects on a user's experience during complex interactions.

One of the general issues identified with both interaction conditions was the visual appearance and functionality of the virtual hand representations. Participants reported the virtual hands as resembling "a terminator" with their robotic appearance and the lack of an arm resulting in two floating hands to feel a bit un-natural. Furthermore, the ability to see virtual hands within the simulation led some participants to report that the fine control they had anticipated was not present.

However, it should also be noted that this could be down to the rather simplistic nature of the object control implemented within the simulation. The results indicate the simplistic design of the object interactions (detailed in Appendix 11) did not provide sufficiently complex interactions to fully evaluate the dexterity of the interaction conditions. Furthermore, the un-realistic nature of the object interactions limited the user's sense of immersion despite the naturalness of the interaction, due to the perceived level of fine control the appearance of the virtual objects provided.

6.5 – Complexity Analysis

The MOT system is designed to run at more than 90FPS as previously discussed in Chapter 3. The Leap Motion sensors used can provide tracking data in excess of 200FPS (Han & Gold, 2014). The constant stream of data provided to the MOT system and greater FPS of the sensors ensures the most recent tracking data is used. Furthermore, tracking frame data is always available for the MOT system to process. This results in minimal latency within the system, such that it is not perceivable to users during simulation experiences. The low latency and thus imperceptible effect have been shown during both developmental testing and in both phases of experimentation with participants not reporting any issues regarding tracking latency.

The MOT systems components process all hands of the current tracking data, applying the same processes to each hand. The results of the complexity analysis in Table 6.24 show the linear fashion of the MOT systems processing, with the processing required increasing with each additional hand. As most of the MOT systems components processing is linear, based on the number of hands available, the MOT system should scale well. As previously discussed in section 3.4, the data aggregation system processes ***B*** frames for each of the ***A*** sources, with three frames used per the results of developmental testing. Table 6.24 shows the data aggregation system to have a complexity of **O(AB)**, meaning the running time approaches being linearly proportional to **AB**, as the system can support more than two sources with more than three frames for each. As previously discussed, three frames were determined to be the optimal number of frames, therefore the complexity of the data aggregation system can be simply presented as linear or **O(N)**.

As detailed in Appendix 10, the mesh conformation system was modified during the development of the extended functionality to improve performance. The mesh conformation system was modified to calculate the surface position for the metacarpal bone of each finger ***F*** with the remaining bone positions determined mathematically. The jagged array is then used to process each triangle ***T*** that uses the identified vertex to find the position on the surface of the object, closest to the given position. Table 6.24 shows the complexity analysis for the mesh confirmation system for the creation of the data structures with processing and just the processing if the data structures were previously created. The first step in the mesh conformation system is the creation of a jagged array using the mesh triangle data. The process of creating the jagged array involves iterating over each triangle **(a)** to determine how many times each vertex is used. Each vertex is then iterated **(b)**, creating the empty jagged array structure using the vertex usage counts to determine array size. The triangles are then iterated over again **(a)** and the jagged array is populated, resulting in a complexity of **O(a + b + a)**. Once the jagged array has been created, the Kd-Tree is created using the vertex data, with the creation of the Kd-Tree taking **O(log(n))**. Finally, each finger **(f)** is iterated with the Kd-Tree accessed to find the closest vertex **log(n)**, with the resulting triangle array accessed from the jagged array **O(1)** and iterated to triangulate the closest surface position **(t)**. This results in the complexity of **O(f * log(n) * t)**.

However, as also shown in Table 6.24, if the jagged array and Kd-Tree have previous been created, the complexity of the mesh conformation system is reduced. The previously discussed process is used with each finger **(f)**

iterated, getting the closest vertex **log(n)**, accessing the jagged array to get the triangles for that vertex **O(1)** and then processing the triangles **(t)** that use it to calculate the position.

Table 6.24 - Complexity Analysis (Time) for the MOT system components

MOT System Component	**Big O Notation**
Data Processor	O(n)
Data Aggregation System	O(A * B)
Deep NN Validation System	O(n)
Post Aggregation System	O(n)
Occlusion Detection System	O(n)
Vive Arm Tracking System	O(n)
Stability Detection System	O(n)
Inferred Hand Pose System	O(n)
Mesh Conformation System	O(a + b + a) + O(log(n)) + O(f * log(n) * t) **or** O(1) + O(f * log(n) * t)
Interpolation System	O(n)

As previously discussed, the mesh conformation system uses a Kd-Tree and Jagged array. Table 6.25 presents a complexity analysis of the data structures used in comparison to alternative approaches. Analysis of the performance of a Kd-Tree in comparison to alternative approaches such as a B-Tree shows similar performance. While alternative approaches such as brute force by processing every vertex in the mesh array grows linearly, resulting in increasing processing times.

The jagged array used requires three iterations the first to determine the structure of the two-dimensional array by processing each triangle and counting how many times each vertex is used. The second iteration is used to create the two-dimensional array to the required sizes. The third iteration processes each triangle and stores the triangle number in the jagged array under the appropriate vertices. An alternative approach using a dictionary of

lists would require each triangle to be processed. For each triangle the dictionary would have to be examined to check whether the corresponding vertex key existed. Finally, the triangle number would have to be appended to the end of the corresponding vertex list. The complexity of the dictionary approach would require more processing than that of the jagged array used.

Table 6.25 - Complexity Analysis (Time) for Data Structures used within the Mesh Conformation System and Alternatives

Data Structure	Access Type	Big O Notation
Array	Write	O(n)
Array	Read	O(n)
Kd-Tree	Write	O(log(n))
Kd-Tree	Read	O(log(n))
B-Tree	Write	O(log(n))
B-Tree	Read	O(log(n))
Binary Search Tree	Write	O(n)
Binary Search Tree	Read	O(n)
Jagged Array Used	Write	O(a + b + a)
Jagged Array Used	Read	O(1)
Dictionary of Lists	Write	O(t * v * a)
Dictionary of Lists	Read	O(1)

6.6 – Discussion

Analysis of user feedback collected during the Chemical Engineering simulation showed the MOT system to provide the most natural and engaging experience. The results of the simulation questionnaire showed the single Leap condition to outscore the MOT system in all other questions. However, the results of the quantitative analysis, specifically the tracking data logs, disagrees with that of the questionnaire. Analysis of the hand tracking data in sections 6.2.1 and 6.3.1 shows the MOT system to produce significantly more valid hand data than the front-facing sensor. As a result, the total period in which tracking data was not available was significantly reduced, with any

missing hands handled via the inferred hand pose system. The poor validity and resulting long periods of time without hands achieved by the front-facing sensor will have resulted in hands disappearing during the single sensor configuration simulation experience. However, this is not reflected in the questionnaire feedback provided. The tracking data was logged for each frame of the simulation to provide a reliable analysis of the user experience. While, the questionnaire is based on the user's perceived experience and thus the tracking data provides a more accurate representation of the experience. Based on the questionnaire feedback and the tracking performance of the two conditions, it can be concluded that the intricacy and complexity of the simulation task kept users focused on the simulation. This resulted in many of the issues within the single sensor configuration and solutions presented in the MOT system being unperceived. Thus, resulting in the poor perception of the two interaction conditions. Based on user feedback and the 2.08% difference in mean Likert score for the naturalism question, it can be concluded that the un-realistic appearance of the hands was a limiting factor in both conditions.

The results of the gesture questionnaire analysis support the hypothesis that the intricacy and complexity of the simulation diverted user focus. The results showed the MOT system to outperform the Leap condition in most categories. Furthermore, the results showed the effectiveness of the additional MOT system functionality. The average scores for the hand occlusion gesture showed the effectiveness of the occlusion detection system at handling self-occlusion to provide a stable and consistent experience. The MOT system saw a 93.8% score increase compared to that of the single front-facing sensor

condition. The results of the qualitative analysis concur with participants reporting hand stability in the MOT condition with the bottom hand remaining in position and visible. However, in the single Leap condition, the bottom hand is reported as frequently disappearing with both hands disappearing on some occasions.

The results presented concur with those of the research previously discussed in Chapter 2, demonstrating hand to hand interaction to be a significant factor in occlusion. In addition, the results concur with research such as that of Jin et al (2016), Marin et al (2014), Clark and Moodley (2016) and Zou, et al (2019), showing the use of multiple sensors to reduce the effects of occlusion. The presented solution contributes to the answer of one of the research questions showing that multi-sensor data aggregation can improve optical tracking by reducing the effect of occlusion. Furthermore, the results of the analysis comparing the performance of the two sensors, concurs with previously discussed research such as Shao (2016). The configuration of the 45-degree sensor reduces the effects of self-occlusion by proving more of a top-down view. This enables the user's fingers to be tracked while the front-facing sensor is occluded by the back of the hand. The tracking validity analysis of the two sensors helps answer one of the research questions, showing the 45-degree sensor to be a much more effective configuration.

Gestures that evaluated field-of-view limitation demonstrated the increased tracking range that the MOT system provides. Participants reported the Leap condition to have a smaller field-of-view with hands often disappearing still within view, while the MOT system was able to perform the gestures. The results presented concur with the previously discussed research such as that

of Shao (2016), Marin et al (2014), Pinto et al (2015) and Jin et al (2016) on the limitations of optical tracking field-of-view. Furthermore, the results show the effect of the optimal tracking area in optical trackers (Colgan, 2015) and the degradation of tracking data outside it, with hands disappearing while still within view. In contrast the MOT system was found to have a much larger field-of-view firstly because of the aggregation process as suggested by research previously discussed in Chapter 2. Secondly, because of the stability detection system which was able to detect instability and trigger an inferred hand pose to ensure a consistent and stable experience. These results indicate the effectiveness of the interpolation and stability detection systems in creating a smooth and consistent experience. Furthermore, the results show the MOT system presents an effective approach to detecting and handling instability in optical tracking data.

Participant feedback indicates users did not experience invalid or disfigured hands, other than the few issues with the right-hand during mesh conformation but a more realistic looking hand could improve the realism. The absence of invalid or disfigured hands demonstrates the effectiveness of the deep neural network-based validation system in comparison to the previous approach. Furthermore, the combination of the validation system and occlusion detection system was found to be effective in providing support for complex gestures in which hands can disappear or become deformed. Participants were able to perform gestures such as interlocking fingers without fingers being mis-recognized resulting in an invalid deformed hand being presented.

A paired-samples t-test indicated statistically significant difference between the two conditions in gestures addressing the key issues with optical trackers; specifically, circle, hand-over, away and reaching. The results show more extreme gestures that evaluate the key issues are more effective for evaluating interaction conditions, with no significant differences found in gestures effecting more subtle factors. The similar results for both conditions on the questions regarding satisfaction, ease and learning, suggests users evaluated the conditions in terms of hand-based interaction as opposed to the specific differences between the two conditions.

The results of the tracking data validation tests concur with the pilot study conducted in the first phase of testing (section 4.3) showing the 45-degree sensor to provide more valid frames than a front-facing sensor. Furthermore, the results show the effectiveness of the MOT system in providing more valid tracking data using multiple sensors. Analysis from the results of the statistical analysis showed the MOT system to produce statistically significantly more valid frames than a front-facing sensor. In addition, the results showed the effectiveness of the 45-degree sensor in producing significantly more valid frames than a front-facing sensor. Linear regression showed the 45-degree sensor had a large effect on the MOT system data validity, compared to the medium effect of the front-facing sensor.

The results of the tracking validity analysis show the MOT systems data aggregation approach significantly improves the validity of tracking data. The combing of sensor data through an aggregator enables even better tracking of users' hands and more valid hands being presented to the user. These findings expand on that of Jin et al (2016) who demonstrated, in a non-VR

context, the tracking improvements of using an additional tripod mounted leap sensor in controlling a robotic arm. These results also concur with previously discussed research such as that of Jin et al (2016), Marin et al (2014), Clark and Moodley (2016) and Zou, et al (2019), showing the use of multiple sensors to improve hand tracking. Furthermore, the results answer one of the research questions, showing multi-sensor aggregation to improve the validity of optical tracking data. The tracking validity analysis of the two sensors helps answer one of the research questions, showing the 45-degree sensor to be a much more effective configuration. The next Chapter presents the conclusion of the work presented, a discussion of any limitations and of future work in the area.

Chapter 7 – Conclusion and Future Work

7.1 – Conclusion

This thesis has presented a hand-based interaction mechanism using optical tracking sensors and multi-sensor data aggregation to overcome the four main issues (occlusion, field-of-view, stability and accuracy) with optical tracking solutions. To address these issues, a hand-based interaction mechanism known as the MOT system, was developed. As part of the MOT system development, a custom 3D bracket was designed for additional sensor mounting. In addition, a real-time natural hand pose system that generates realistic hand interaction poses for a more realistic and immersive experience. Furthermore, the MOT system used a real-time occlusion detection, mesh conformation and deep neural network-based hand validation systems to provide a more natural and consistent user experience. The developed MOT system has been compared against traditional interaction approaches; specifically, single sensor setups, haptic gloves and motion controllers.

The original aims and objectives of the thesis were as follows:

- To identify and evaluate the relevant scientific literature, methods, tools and technologies to develop a critical understanding of the literature.
- To design and implement a real-time tracking optical-based hand interaction approach using multi-sensor aggregation.
- To evaluate the effect of different sensor positions and orientations on optical tracking data in factors including validity and stability.

- To evaluate the developed interaction approach against traditional approaches such as gloves and motion controllers at creating a sense of immersion and naturalism.
- To evaluate the developed optical-based interaction approach against a single sensor configuration.

The first aim was addressed by conducting thorough research into virtual reality, immersion & presence and hand-based input mechanisms. Researching both contact-based devices such as motion controllers and vision-based approaches using technologies such as optical-based trackers. Chapter two, the Literature Review, presents an analysis and discussion of background research, to place this research in an academic context and establish previous approaches. Research analysed has shown significant user preference for direct natural hand-based interaction providing users with higher degrees-of-freedom and a more engaging experience. Machine learning approaches, particularly neural networks have proven effective in gesture classification with finger angles one of the most effective hand features. However, such approaches cannot perform real-time recognition. The use of multiple optical tracking sensors can improve tracking accuracy and field-of-view, but the effect is limited in statistically positioned sensors due to hand and body occlusion during interaction.

The second aim was achieved through the development of a Multiple Optical Tracking (MOT) System. The MOT system uses multi-sensor data aggregation along with; a custom designed 3D printed bracket, motion controller tracking, inferred hands pose system, mesh conformation, deep neural-network base

hand validation, stability detection system, interpolation system and an occlusion detection system, to address the current limitations of optical tracking. A detailed discussion of the designs for the sub-systems of which the MOT system is comprised can be found in Chapters three and five.

As part of the development of the MOT system, different additional sensor positions and angles were evaluated using factors including hand validity and stability. During both phases of experimentation, the performance of a traditional front-facing sensor configuration was evaluated against a 45-degree angled sensor. Experimental results showed the custom mounted 45-degree angle sensor to produce significantly more valid hand data than the front-facing sensor. The phase two simulation experiment saw a 93% increase in hand validity between the two sensors when the validity of both hands was considered. The left and right hands saw a 60% and 35% increase, respectively. Furthermore, results of Linear regression tests showed the 45-degree sensor to a large effect on the MOT systems hand validity with the front-facing having a medium effect. The completion of the third aim through the evaluation of different positions and orientations along with the results of the comparisons answers the second research question. The results show a 45-degree sensor to be the most effective sensor configuration for optical-based hand interaction in virtual reality.

The fourth aim was achieved through the first phase of experimentation in which the MOT system was evaluated against motion controller and haptic gloves in a chemistry themed simulation. The experiment enabled user preference and the effect of each condition on a sense of immersion and engagement to be determined. Furthermore, the experimentation enabled the

strengths and weaknesses of both the MOT system and traditional approaches to be identified. The identified limitations were then used to improve the effectiveness of the MOT system during phase two. The results showed the MOT system to be the preferred interaction condition due to the more direct natural interaction. The results also showed the effectiveness of the aggregation approach with the MOT system producing significantly more valid hand data than both sensors. The results help to answer to the first research question showing multi-sensor aggregation can improve both the tracking field of view and the number of valid tracking frames produced. However, the results also showed that despite the increased field-of-view and number of valid tracking frames, the MOT system still had issues with occlusion and instability.

The second phase of experimentation evaluated the MOT system with additional functionality implemented based on research and the results of the first phase of experimentation. In the experiment the MOT system was compared against a front-mounted sensor in both an educational VR simulation and gesture-based evaluation. The results further demonstrated the 45-degree sensor offers the most effective sensor configuration with a 163% increase between the two sensors when considering the validity of both hands. The left and right hands saw a 43% and 55% increase, respectively.

The evaluation of the MOT system in comparison to a traditional front-facing sensor configuration combined with the results of the first phase of experimentation provide an answer to the first research question. The results show how multi-sensor aggregation can improve optical-based hand tracking by increasing the field-of-view and the number of valid hand tracking frames

produced. The aggregation process has been shown to enable the MOT system to maintain high accuracy over its enhanced field-of-view. However, the aggregation systems averaging process results in slightly decreased accuracy within the optimal tracking area in comparison a single front-facing sensor.

Furthermore, the completion of the fifth objective has shown that while multi-sensor aggregation can improve hand validity, it alone is not sufficient to overcome the issues with occlusion and instability. However, the results of the evaluation of the extended MOT system show that multi-sensor aggregation combined with complex software can address the issues of occlusion and instability to provide a more consistent user experience. The effectiveness of the additional functionality is particularly evident in the gesture evaluation test with gestures such as that testing occlusion seeing a 93.83% increase in score. The research presented in this work has shown an effective solution for addressing the key issues with optical trackers through a multi-sensor and software solution.

The results of the qualitative analysis showed the motion controller tracking and inferred hands pose system were found to be effective in providing a more stable and consistent user experience. The stability and occlusion detection systems were found to be effective in providing a consistent experience, detecting and handling invalid tracking data. The systems ensured stable hand representations even during challenging conditions such as hand occlusion. The mesh conformation system received positive feedback for its natural hand poses with only a few participants experiencing issues. Finally, the multi-network based deep neural network validation system proved to be effective

in validating hand data in real-time ensuring users had a smooth and consistent experience. The use of multiple networks focusing on the natural movement range of a specific hand feature resulted in high accuracy without the limitation of a gesture vocabulary.

7.2 – Research Limitations

The results of the research presented in this thesis demonstrate the effectiveness of the MOT system design and answer the two research questions presented in Chapter 1. However, some limitations should be noted. Firstly, the second phase of experimentation compares the extended MOT system to a traditional front-facing configuration. This experimentation was performed to evaluate the effectiveness of the MOT systems design at improving optical-based hand tracking solutions through addressing the four key issues. However, the extended functionality was not evaluated against the controller and glove-based conditions used in the first phase of experimentation. An evaluation against the controller and glove conditions would have enabled the effectiveness of the additional functionality in improving the MOT system to be further evaluated.

Finally, the two phases of experimentation provide statistically significant results to demonstrate the effectiveness of the MOT system at improving optical-based hand interaction solutions. Furthermore, detailed quantitative and qualitative analysis has been performed. However, the sample size of participants used in both phases of experimentation can be considered small, thus limiting the strength of the conclusions that can be drawn.

7.3 – Future Work

The research presented in this thesis demonstrates the potential for a Multiple Optical Tracking (MOT) System, that uses multi-sensor data aggregation and additional functionality to provide a natural and consistent hand interaction experience. However, there are several areas that would benefit from further work. Participant feedback indicated the presence of the inferred virtual hands led them to presume they were controllable. The research presented in this work has shown inferred hands provide a more consistent experience. However, there is also a need for more interactive approaches that can provide a greater level of interaction during periods of invalid/missing data. Future work will look to add more intelligent approaches for inferred hands, to improve the user experience and interaction consistency. Future work will also look to significantly reduce the footprint of the motion controllers using technologies such as microcontrollers and infrared sensors. The depth sensor data provided by the Leap Motion sensors could be useful for occlusion detection, therefore future work will look at using the depth data in the occlusion detection system. Finally, future work will look to improve the sense of realism through exploring physics-based object interactions and controls for a more realistic and natural user experience.

References

Achibet, M. et al., 2015. *Elastic-Arm: Human-scale passive haptic feedback for augmenting interaction and perception in virtual environments.* s.l., s.n., pp. 63-68.

Adams, A., Lunt, P. & Cairns, P., 2008. A qualitative approach to HCI research. In: *Research Methods for Human-Computer Interaction.* s.l.:Cambridge University Press, pp. 138-157.

Alavi, S., Arsenault, D. & Whitehead, A., 2016. Quaternion-based gesture recognition using wireless wearable motion capture sensors. *Sensors,* Volume 16, p. 605.

Alexandrovsky, D. et al., 2019. *Demonstrating VRBox: A Virtual Reality Augmented Sandbox.* s.l., s.n., p. 1–4.

Alfaro, L. et al., 2019. Virtual Reality Full Immersion Techniques for Enhancing Workers Performance, 20 years Later: A Review and a Reformulation. *Virtual Reality,* Volume 10.

Almeida, L. et al., 2019. *Towards natural interaction in immersive reality with a cyber-glove.* s.l., s.n., pp. 2653-2658.

Anthes, C., Garc\́ia-Hernández, R. J., Wiedemann, M. & Kranzlmüller, D., 2016. *State of the art of virtual reality technology.* s.l., s.n., pp. 1-19.

Argelaguet, F., Hoyet, L., Trico, M. & Lécuyer, A., 2016. *The role of interaction in virtual embodiment: Effects of the virtual hand representation.* s.l., s.n., pp. 3-10.

ART, 2020. *Flystick 3 Controller.* [Online]
Available at: https://ar-tracking.com/products/interaction/flystick-3/

Assila, A., de Oliveira, K. M. & Ezzedine, H., 2014. *Towards qualitative and quantitative data integration approach for enhancing HCI quality evaluation.* s.l., s.n., p. 469–480.

Bachmann, D., Weichert, F. & Rinkenauer, G., 2018. Review of three-dimensional human-computer interaction with focus on the leap motion controller. *Sensors,* Volume 18, p. 2194.

Beckett, J., 2017. *What is a Generative Adversarial Network? Inventor Explains _ The Official NVIDIA Blog.* [Online]
Available at: https://blogs.nvidia.com/blog/2017/05/17/generative-adversarial-networks/

Bentley, J. L., 1980. Multidimensional divide-and-conquer. *Communications of the ACM,* Volume 23, pp. 214-229.

Borja, E. F., Lara, D. A., Quevedo, W. X. & Andaluz, V. H., 2018. *Haptic stimulation glove for fine motor rehabilitation in virtual reality environments.* s.l., s.n., p. 211–229.

Bowman, D. A. & McMahan, R. P., 2007. Virtual reality: how much immersion is enough?. *Computer,* Volume 40.

Bowman, D. A., McMahan, R. P. & Ragan, E. D., 2012. Questioning naturalism in 3D user interfaces. *Communications of the ACM,* Volume 55, pp. 78-88.

Braun, V. & Clarke, V., 2006. Using thematic analysis in psychology. *Qualitative research in psychology,* Volume 3, pp. 77-101.

Breslauer, N., Galić, I., Kukec, M. & Samardžić, I., 2019. Leap Motion Sensor for Natural User Interface. *Tehni{\v{c}}ki vjesnik,* Volume 26, pp. 560-565.

Brooke, J., 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry,* Volume 189, pp. 4-7.

Buchanan, T., 2018. *A Hands on Look at the State of Input in VR - VRScout.* s.l.:s.n.

Caserman, P., Garcia-Agundez, A. & Goebel, S., 2019. A Survey of Full-Body Motion Reconstruction in Immersive Virtual Reality Applications. *IEEE transactions on visualization and computer graphics.*

Chapoulie, E., 2014. *Gestures and direct manipulation for immersive virtual reality,* s.l.: s.n.

Chapoulie, E. et al., 2014. Evaluation of direct manipulation using finger tracking for complex tasks in an immersive cube. *Virtual Reality,* Volume 18, pp. 203-217.

Chapoulie, E. et al., 2015. *Finger-based manipulation in immersive spaces and the real world.* s.l., s.n., pp. 109-116.

Chaudhary, A., Raheja, J. L., Das, K. & Raheja, S., 2013. Intelligent approaches to interact with machines using hand gesture recognition in natural way: a survey. *arXiv preprint arXiv:1303.2292.*

Choi, C., 2018. Computational Learning for Hand Pose Estimation.

Choi, I. et al., 2017. *Grabity: A Wearable Haptic Interface for Simulating Weight and Grasping in Virtual Reality.* s.l., s.n., pp. 119-130.

Choi, I. et al., 2016. *Wolverine: A wearable haptic interface for grasping in virtual reality.* s.l., s.n., pp. 986-993.

Choi, I. et al., 2018. *CLAW: A Multifunctional Handheld Haptic Controller for Grasping, Touching, and Triggering in Virtual Reality.* s.l., s.n., p. 654.

Choi, J.-H., Ko, N.-Y. & Ko, D.-Y., 2001. *Morphological gesture recognition algorithm.* s.l., s.n., pp. 291-296.

Clark, A. & Moodley, D., 2016. *A System for a Hand Gesture-Manipulated Virtual Reality Environment.* s.l., s.n., p. 10.

Clifford, R. M. S., Tuanquin, N. M. B. & Lindeman, R. W., 2017. *Jedi ForceExtension: Telekinesis as a Virtual Reality interaction metaphor.* s.l., s.n., pp. 239-240.

Cohen, J., 2013. *Statistical power analysis for the behavioral sciences.* s.l.:Routledge.

Colagrossi, M., 2019. *What would it take to create a fully immersive virtual reality?.* s.l.:s.n.

Colgan, A., 2015. *4 Design Problems for VR Tracking (And How to Solve Them).* [Online]
Available at: http://blog.leapmotion.com/4-design-problems-vr-tracking-solve/

Collins, K. & Borowski, K., 2018. *Experimental Game Interactions in a Cave Automatic Virtual Environment.* s.l., s.n., pp. 1-9.

Davis, S., Nesbitt, K. & Nalivaiko, E., 2015. *Comparing the onset of cybersickness using the Oculus Rift and two virtual roller coasters.* s.l., s.n., p. 30.

Derpanis, K. G., 2004. A review of vision-based hand gestures. *Department of Computer Science York University.*

Diliberti, N. et al., 2019. *Real-time gesture recognition using 3d sensory data and a light convolutional neural network.* s.l., s.n., p. 401–410.

Dittrich, J., 2017. *Leap Motion in VR with Vive using Unity.* [Online] Available at: https://www.youtube.com/watch?v=T6nDDoPhua0

Edwards, B. I., Bielawski, K. S., Prada, R. & Cheok, A. D., 2019. Haptic virtual reality and immersive learning for enhanced organic chemistry instruction. *Virtual Reality,* Volume 23, p. 363–373.

El Saddik, A., Orozco, M., Eid, M. & Cha, J., 2011. Haptics: general principles. In: *Haptics technologies.* s.l.:Springer, pp. 1-20.

ElKoura, G. & Singh, K., 2003. *Handrix: animating the human hand.* s.l., s.n., pp. 110-119.

Fahmi, F., Nainggolan, F., Andayani, U. & Siregar, B., 2018. Development of excavator training simulator using leap motion controller. *JPhCS,* Volume 978, p. 012034.

Fahmi, F. et al., 2020. Comparison study of user experience between virtual reality controllers, leap motion controllers, and senso glove for anatomy learning systems in a virtual reality environment. *system,* Volume 2, p. 3.

Feng, Z. et al., 2018. Immersive virtual reality serious games for evacuation training and research: A systematic literature review. *Computers \& Education,* Volume 127, pp. 252-266.

Figueiredo, L., Rodrigues, E., Teixeira, J. & Techrieb, V., 2018. A comparative evaluation of direct hand and wand interactions on consumer devices. *Computers \& Graphics,* Volume 77, pp. 108-121.

Freire, M. et al., 2016. Game learning analytics: learning analytics for serious games. In: *Learning, design, and technology.* s.l.:Springer Nature Switzerland AG, pp. 1-29.

Friedman, J. H., Bentley, J. L. & Finkel, R. A., 1976. An algorithm for finding best matches in logarithmic time. *ACM Trans. Math. Software,* Volume 3, pp. 209-226.

Fröhlich, T. et al., 2018. *VRBox: A Virtual Reality Augmented Sandbox for Immersive Playfulness, Creativity and Exploration.* s.l., s.n., p. 153–162.

Furht, B., ed., 2008. Immersive Virtual Reality. In: *Encyclopedia of Multimedia.* Boston(MA): Springer US, pp. 345-346.

Gabele, M., Schröer, S., Hußlein, S. & Hansen, C., 2019. An AR Sandbox as a Collaborative Multiplayer Rehabilitation Tool for Children with ADHD. *Mensch und Computer 2019-Workshopband.*

Gallotti, P., Raposo, A. & Soares, L., 2011. *v-Glove: A 3D virtual touch interface.* s.l., s.n., pp. 242-251.

Gao, Y., González, V. A. & Yiu, T. W., 2017. Serious games vs. traditional tools in construction safety training: a review. *LC3 2017,* Volume 1, pp. 4-7.

Ghinea, M. et al., 2018. *Perception of Absolute Distances Within Different Visualization Systems: HMD and CAVE.* s.l., s.n., pp. 148-161.

GoPro, 2020. *Protective Housing HERO8 Black.* s.l.:s.n.

GoTouchVR, 2020. *VRTouch.* [Online]
Available at: https://www.gotouchvr.com/copy-of-technology-devices-1

Guzsvinecz, T., Szucs, V. & Sik-Lanyi, C., 2019. Suitability of the Kinect sensor and Leap Motion controller—a literature review. *Sensors,* Volume 19, p. 1072.

Halarnkar, P. et al., 2012. A review on virtual reality. *IJCSI International Journal of Computer Science,* p. 6.

Han, J. & Gold, N. E., 2014. *Lessons learned in exploring the Leap Motion™ sensor for gesture-based instrument design.* s.l., Proceedings of the International Conference on New Interfaces for Musical Expression.

Hannema, D., 2001. Interaction in virtual reality. *Interaction in Virtual Reality.*

Han, S. & Kim, J., 2017. A study on immersion of hand interaction for mobile platform virtual reality contents. *Symmetry,* Volume 9, p. 22.

Han, Y., 2010. A low-cost visual motion data glove as an input device to interpret human hand gestures. *IEEE Transactions on Consumer Electronics,* Volume 56.

HaptX, 2020. *Technology | HaptX.* s.l.:s.n.

Heaney, D., 2019. *How VR Positional Tracking Systems Work - UploadVR.* s.l.:s.n.

Hinchet, R., Vechev, V., Shea, H. & Hilliges, O., 2018. *Dextres: Wearable haptic feedback for grasping in vr via a thin form-factor electrostatic brake.* s.l., s.n., pp. 901-912.

Höll, M., Oberweger, M., Arth, C. & Lepetit, V., 2018. *Efficient Physics-Based Implementation for Realistic Hand-Object Interaction in Virtual Reality.* s.l., s.n.

Horowitz, K., 2014. *Skeletal Tracking 101: Getting Started with the Bone API.* [Online]
Available at: http://blog.leapmotion.com/skeletal-tracking-101-getting-started-with-the-bone-api-and-rigged-hands/

ICVR-Interactive, 2018. *HaptX VR Gloves: In-depth Developer's Review.* [Online]
Available at: https://medium.com/@info_93973/haptx-vr-haptic-gloves-hands-on-review-62b27353d32b

Irfan, Q., Jensen, C., Ni, Z. & Hietpas, S., 2018. *Building an exoskeleton glove on virtual reality platform.* s.l., s.n., pp. 645-650.

Ishiyama, H. & Kurabayashi, S., 2016. *Monochrome glove: A robust real-time hand gesture recognition method by using a fabric glove with design of structured markers.* s.l., s.n., pp. 187-188.

Jadhav, S. et al., 2017. Soft robotic glove for kinesthetic haptic feedback in virtual reality environments. *Electronic Imaging,* Volume 2017, pp. 19-24.

Jarvis, M., 2016. *Developing for VR's hardware constraints is like going back to the N64.* [Online]

Available at: http://www.develop-online.net/interview/developing-for-vr-s-hardware-constraints-is-like-going-back-to-the-n64/0219596

Jin, H. et al., 2016. Multi-LeapMotion sensor based demonstration for robotic refine tabletop object manipulation task. *CAAI Transactions on Intelligence Technology,* Volume 1, pp. 104-113.

Keras, 2020. *Keras documentation: Dense layer.* [Online]
Available at: https://keras.io/api/layers/core_layers/dense/

Kim, H., Lee, G. & Billinghurst, M., 2015. *A Non-linear Mapping Technique for Bare-hand Interaction in Large Virtual Environments.* s.l., s.n., pp. 53-61.

Kim, J. & Maher, M. L., 2019. *Metaphors, Signifiers, Affordances, and Modalities for Designing Mobile and Embodied Interactive Systems.* s.l., s.n., pp. 542-545.

Kim, J.-S. & Park, J.-M., 2015. *Physics-based hand interaction with virtual objects.* s.l., s.n., pp. 3814-3819.

Kim, M., Jeon, C. & Kim, J., 2017. A study on immersion and presence of a portable hand haptic system for immersive virtual reality. *Sensors,* Volume 17, p. 1141.

Kim, M., Lee, J., Kim, C. & Kim, J., 2018. Tpvr: User interaction of third person virtual reality for new presence and experience. *Symmetry,* Volume 10, p. 109.

Kiselev, V., Khlamov, M. & Chuvilin, K., 2019. *Hand Gesture Recognition with Multiple Leap Motion Devices.* s.l., s.n., p. 23.

Ko, J. et al., 2019. An Epiduroscopy Simulator Based on a Serious Game for Spatial Cognitive Training (EpiduroSIM): User-Centered Design Approach. *JMIR serious games,* Volume 7, p. e12678.

Krokos, E., Plaisant, C. & Varshney, A., 2019. Virtual memory palaces: immersion aids recall. *Virtual Reality,* Volume 23, pp. 1-15.

Kumar, P., Gauba, H., Roy, P. P. & Dogra, D. P., 2017. A multimodal framework for sensor based sign language recognition. *Neurocomputing,* Volume 259, pp. 21-38.

Kyriakou, P. & Hermon, S., 2019. Can I touch this? using natural interaction in a museum augmented reality system. *Digital Applications in Archaeology and Cultural Heritage,* Volume 12, p. e00088.

Lahanas, V. et al., 2017. Virtual reality-based assessment of basic laparoscopic skills using the Leap Motion controller. *Surgical endoscopy,* Volume 31, p. 5012–5023.

Lang, B., 2013. *SMI Introduces 3D Glasses With Eye Tracking.* [Online] Available at: https://www.roadtovr.com/smi-3d-eye-tracking-glasses/

LaViola Jr, J. J. et al., 2017. *3D user interfaces: theory and practice.* s.l.:Addison-Wesley Professional.

Lee, E. A.-L., Wong, K. W. & Fung, C. C., 2010. How does desktop virtual reality enhance learning outcomes? A structural equation modeling approach. *Computers & Education,* Volume 55, p. 1424–1442.

Lee, H., Jung, T. H., tom Dieck, M. C. & Chung, N., 2019. Experiencing immersive virtual reality in museums. *Information & Management,* p. 103229.

Lee, N., 2016. *The challenges of creating games for virtual reality.* [Online] Available at: https://www.engadget.com/2016/03/16/the-challenges-of-creating-games-for-virtual-reality/

Li, L. & Dai, S., 2014. *Bayesian neural network approach to hand gesture recognition system.* s.l., s.n., pp. 2019-2023.

Lin, L. & Jörg, S., 2016. *Need a hand? how appearance affects the virtual hand illusion.* s.l., s.n., pp. 69-76.

Lin, L. et al., 2019. *The effect of hand size and interaction modality on the virtual hand illusion.* s.l., s.n., p. 510–518.

Lombard, M., Snyder-Duch, J. & Bracken, C. C., 2002. Content analysis in mass communication: Assessment and reporting of intercoder reliability. *Human communication research,* Volume 28, pp. 587-604.

Lougiakis, C., Katifori, A., Roussou, M. & Ioannidis, I.-P., 2020. *Effects of Virtual Hand Representation on Interaction and Embodiment in HMD-based Virtual Environments Using Controllers.* s.l., s.n., p. 510–518.

Lugrin, J.-L., Latt, J. & Latoschik, M. E., 2015. *Anthropomorphism and illusion of virtual body ownership.* s.l., s.n., pp. 1-8.

Lund, A. M., 2001. USE questionnaire: Usefulness, satisfaction, and ease of use. *Usability interface,* Volume 8, pp. 3-6.

Luzhnica, G., Simon, J., Lex, E. & Pammer, V., 2016. *A sliding window approach to natural hand gesture recognition using a custom data glove.* s.l., s.n., p. 81–90.

Madathil, K. C. et al., 2017. An empirical study investigating the effectiveness of integrating virtual reality-based case studies into an online asynchronous learning environment. *Computers in Education Journal,* Volume 8, p. 1–10.

Manjrekar, S. et al., 2014. *CAVE: An Emerging Immersive Technology--A Review.* s.l., s.n., pp. 131-136.

Marin, G., Dominio, F. & Zanuttigh, P., 2014. *Hand gesture recognition with leap motion and kinect devices.* s.l., s.n., pp. 1565-1569.

Marroquin, B., 2017. *HTC Vive Controller.* [Online]
Available at: http://www.336gamereviews.com/htc-vive-review/htc-vive-controller/

McCartney, R., Yuan, J. & Bischof, H.-P., 2015. *Gesture recognition with the leap motion controller.* s.l., s.n., p. 3.

Meehan, M., Insko, B., Whitton, M. & Jr, B. a. F. P., 2002. Physiological measures of presence in stressful virtual environments. *Acm transactions on graphics (tog),* Volume 21, pp. 645-652.

Microsoft, 2014. *Kinect for Xbox One.* [Online]
Available at: http://www.xbox.com/en-GB/xbox-one/accessories/kinect-for-xbox-one#fbid=jrnPQXaFq-x

Millar, G. C. et al., 2018. *Tangible landscape: A hands-on method for teaching terrain analysis.* s.l., s.n., p. 1–12.

Moehring, M. & Froehlich, B., 2011. Natural interaction metaphors for functional validations of virtual car models. *IEEE transactions on visualization and computer graphics,* Volume 17, pp. 1195-1208.

Möller, T. & Trumbore, B., 1997. Fast, minimum storage ray-triangle intersection. *Journal of graphics tools,* Volume 2, pp. 21-28.

Moore, A. G. et al., 2020. *A Formative Evaluation Methodology for VR Training Simulations.* s.l., s.n., p. 125–132.

Motion, L., 2013. *Leap Motion.* [Online]
Available at: https://www.leapmotion.com/product/desktop

Muender, T. et al., 2019. *Does It Feel Real? Using Tangibles with Different Fidelities to Build and Explore Scenes in Virtual Reality.* s.l., s.n., p. 1–12.

Muhanna, M. A., 2015. Virtual reality and the CAVE: Taxonomy, interaction challenges and research directions. *Journal of King Saud University-Computer and Information Sciences,* Volume 27, pp. 344-361.

Murthy, G. R. S. & Jadon, R. S., 2009. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management,* Volume 2, pp. 405-410.

Nabioyuni, M. & Bowman, D. A., 2015. *An evaluation of the effects of hyper-natural components of interaction fidelity on locomotion performance in virtual reality.* s.l., s.n., pp. 167-174.

Nan, X. et al., 2014. vDesign: a CAVE-based virtual design environment using hand interactions. *Journal on Multimodal User Interfaces,* Volume 8, pp. 367-379.

Nasim, K. & Kim, Y. J., 2018. Physics-based assistive grasping for robust object manipulation in virtual reality. *Computer Animation and Virtual Worlds,* Volume 29, p. e1820.

Navarro, D. & Sundstedt, V., 2019. *Evaluating player performance and experience in virtual reality game interactions using the htc vive controller and leap motion sensor.* s.l., s.n., pp. 103-110.

Nicholson, C., 2020. *A Beginner's Guide to Neural Networks and Deep Learning.* [Online]
Available at: https://pathmind.com/wiki/neural-network#:
:text=Earlier%20versions%20of%20neural%20networks,qualifies%20as%20
%E2%80%9Cdeep%E2%80%9D%20learning.

Nielsen, J., 1993. Iterative user-interface design. *Computer,* Volume 26, p. 32–41.

Nunes de Vasconcelos, G. et al., 2019. Do we still need CAVEs?.

Oculus, 2019. *Oculus Rift S.* [Online]
Available at: https://www.oculus.com/rift-s/

Oprea, S. et al., 2019. A visually realistic grasping system for object manipulation and interaction in virtual reality environments. *Computers & Graphics,* Volume 83, p. 77–86.

Pallister, K., 2017. *How far are we from interactivity in VR?.* s.l.:s.n.

Park, H. et al., 2017. *Visual Representation of Gesture Interaction Feedback in Virtual Reality Games.* s.l., s.n., pp. 20-23.

Park, W., Heo, H., Park, S. & Kim, J., 2019. A Study on the Presence of Immersive User Interface in Collaborative Virtual Environments Application. *Symmetry,* Volume 11, p. 476.

Patel, K. et al., 2006. *The effects of fully immersive virtual reality on the learning of physical tasks.* s.l., s.n., p. 87–94.

Perret, J. & Vander Poorten, E., 2018. *Touching virtual reality: a review of haptic gloves.* s.l., s.n., pp. 1-5.

Pinter, C. et al., 2020. SlicerVR for Medical Intervention Training and Planning in Immersive Virtual Reality. *IEEE Transactions on Medical Robotics and Bionics,* Volume 2, p. 108–117.

Pinto, J., Dias, P., Eliseu, S. & Sousa Santos, B., 2015. *Interactive configurable virtual environment with Kinect navigation and interaction.* s.l., s.n.

Placidi, G. et al., 2017. *A Virtual Glove System for the Hand Rehabilitation based on Two Orthogonal LEAP Motion Controllers..* s.l., s.n., pp. 184-192.

Postolache, O., Lourenço, F., Pereira, J. D. & Girão, P., 2017. *Serious game for physical rehabilitation: Measuring the effectiveness of virtual and real training environments.* s.l., s.n., p. 1–6.

Poupyrev, I., Billinghurst, M., Weghorst, S. & Ichikawa, T., 1996. *The go-go interaction technique: non-linear mapping for direct manipulation in VR.* s.l., s.n., pp. 79-80.

Preece, J., Sharp, H. & Rogers, Y., 2015. Data Analysis, Interpretation and Presentation. In: *Interaction design: beyond human-computer interaction.* s.l.:John Wiley & Sons, pp. 275-319.

Pulijala, Y. et al., 2018. Effectiveness of immersive virtual reality in surgical training—a randomized control trial. *Journal of Oral and Maxillofacial Surgery,* Volume 76, p. 1065–1072.

Qingchao, X. & Jiangang, C., 2017. *The application of leap motion in astronaut virtual training.* s.l., s.n., p. 012015.

Quesenbery, W., 2001. *What does usability mean: Looking beyondease of use'.* s.l., s.n., p. 432–436.

Ranjan, C., 2019. *Rules-of-thumb for building a Neural Network.* [Online] Available at: https://towardsdatascience.com/17-rules-of-thumb-for-building-a-neural-network-93356f9930af

Rautaray, S. S. & Agrawal, A., 2015. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review,* Volume 43, pp. 1-54.

Ren, Z., Yuan, J., Meng, J. & Zhang, Z., 2013. Robust part-based hand gesture recognition using kinect sensor. *IEEE transactions on multimedia,* Volume 15, pp. 1110-1120.

Schenker-Tech, 2020. *Prime One: The New Era of Hand Tracking.* [Online] Available at: https://www.schenker-tech.de/en/manus-vr

Schmidt, S. et al., 2018. *Impact of virtual environments on motivation and engagement during exergames.* s.l., s.n., pp. 1-6.

Schwind, V., Knierim, P., Chuang, L. & Henze, N., 2017. *" Whereś Pinky?" The Effects of a Reduced Number of Fingers in Virtual Reality.* s.l., s.n., pp. 507-515.

Schwind, V. et al., 2017. *" These are not my hands!" Effect of Gender on the Perception of Avatar Hands in Virtual Reality.* s.l., s.n., p. 1577–1582.

ScratchAPixel, 2019. *A Minimal Ray-Tracer: Rendering Simple Shapes (Sphere, Cube, Disk, Plane, etc).* s.l.:s.n.

Shao, L., 2016. Hand movement and gesture recognition using Leap Motion Controller.

Shehade, M. & Stylianou-Lambert, T., 2020. Virtual Reality in Museums: Exploring the Experiences of Museum Professionals. *Applied Sciences,* Volume 10, p. 4031.

Silva, L., Dantas, R., Pantoja, A. & Pereira, A., 2013. *Development of a low cost dataglove based on arduino for virtual reality applications.* s.l., s.n., pp. 55-59.

Slater, M., 2003. A note on presence terminology. *Presence connect,* Volume 3, pp. 1-5.

Slater, M., 2009. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical Transactions of the Royal Society B: Biological Sciences,* Volume 364, pp. 3549-3557.

Stabbert, T., Fröhlich, T., Alexandrovsky, D. & Malaka, R., 2017. Extending Augmented Sandboxes with Virtual Reality Interaction. *Mensch und Computer 2017-Workshopband.*

Statistics, L., 2018. *Cronbachś Alpha in SPSS Statistics - procedure, output and interpretation of the output using a relevant example | Laerd Statistics..* s.l.:s.n.

Statistics, L., 2018. *Pearsonś product-moment correlation using SPSS Statistics.* s.l.:s.n.

Statistics, L., 2018. *Spearmanś correlation using SPSS Statistics.* s.l.:s.n.

Stephanie, 2014. *Cohen's Kappa Statistic.* s.l.:s.n.

Stone, R. J., 2000. *Haptic feedback: A brief history from telepresence to virtual reality.* s.l., s.n., p. 1–16.

Strickland, J., 2007. *How Virtual Reality Gear Works.* s.l.:s.n.

Studio, T., 2018. *UX Design Glossary: How to Use Affordances in User Interfaces.* s.l.:s.n.

Sutherland, I. E., 1968. *A head-mounted three dimensional display.* s.l., s.n., pp. 757-764.

Taylor, J., 2013. *Confusing Stats Terms Explained: Internal Consistency - Stats Make Me Cry Consulting.* s.l.:s.n.

Teleb, H. & Chang, G., 2012. *Data glove integration with 3d virtual environments.* s.l., s.n., pp. 107-112.

Tian, H., Wang, C., Manocha, D. & Zhang, X., 2018. Realtime Hand-Object Interaction using Learned Grasp Space for Virtual Environments. *IEEE transactions on visualization and computer graphics.*

Tran, T. Q., Shin, H., Stuerzlinger, W. & Han, J., 2017. *Effects of virtual arm representations on interaction in virtual environments.* s.l., s.n., p. 40.

Trigueiros, P., Ribeiro, F. & Reis, L. P., 2012. *A comparison of machine learning algorithms applied to hand gesture recognition.* s.l., s.n., pp. 1-6.

TutorialsPoint, 2020. *Keras - Dense Layer.* [Online]
Available at: https://www.tutorialspoint.com/keras/keras_dense_layer.htm#:~:text=Dense%20layer%20is%20the%20regular,input%20and%20return%20the%20output.

Valentini, P. P. & Pezzuti, E., 2017. Accuracy in fingertip tracking using Leap Motion Controller for interactive virtual applications. *International Journal on Interactive Design and Manufacturing (IJIDeM),* Volume 11, pp. 641-650.

Valve, 2019. *Valve Index.* [Online]
Available at: https://www.valvesoftware.com/en/index/controllers

Visbox, 2020. *CAVE Automatic Virtual Environment.* [Online]
Available at: http://www.visbox.com/products/cave/

Vive, 2020. *HTC Vive Pro.* [Online]
Available at: https://www.vive.com/uk/product/vive-pro/

Wang, R. Y. & Popović, J., 2009. *Real-time hand-tracking with a color glove.* s.l., s.n., p. 63.

Weichert, F., Bachmann, D., Rudak, B. & Fisseler, D., 2013. Analysis of the accuracy and robustness of the leap motion controller. *Sensors,* Volume 13, pp. 6380-6393.

Witmer, B. G. & Singer, M. J., 1998. Measuring presence in virtual environments: A presence questionnaire. *Presence,* Volume 7, p. 225–240.

Wozniak, P. et al., 2016. *Possible applications of the LEAP motion controller for more interactive simulated experiments in augmented or virtual reality.* s.l., s.n., p. 99460P.

Yang, J. et al., 2018. Interaction with three-dimensional gesture and character input in virtual reality: recognizing gestures in different directions and improving user input. *IEEE Consumer Electronics Magazine,* Volume 7, pp. 64-72.

Yan, X. & Aimaiti, N., 2011. *Gesture-based interaction and implication for the future,* s.l.: s.n.

Yusnita, L. et al., 2017. *Implementation of real-time static hand gesture recognition using artificial neural network.* s.l., s.n., p. 1–6.

Zeutzheim, B., 2019. Natural Menu Interactions in VR with Leap Motion.

Zhang, J. et al., 2018. *Development of laparoscopic cholecystectomy simulator based on unity game engine.* s.l., s.n., pp. 1-9.

Zhang, L., Bowman, D. A. & Jones, C. N., 2019. *Exploring Effects of Interactivity on Learning with Interactive Storytelling in Immersive Virtual Reality.* s.l., s.n., p. 1–8.

Zhang, Z., McInerney, T., Zhang, N. & Guan, L., 2014. *A cave based 3D immersive interactive city with gesture interface.* s.l., s.n.

Zhao, W., Zhang, J., Min, J. & Chai, J., 2013. Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics (TOG),* Volume 32, p. 207.

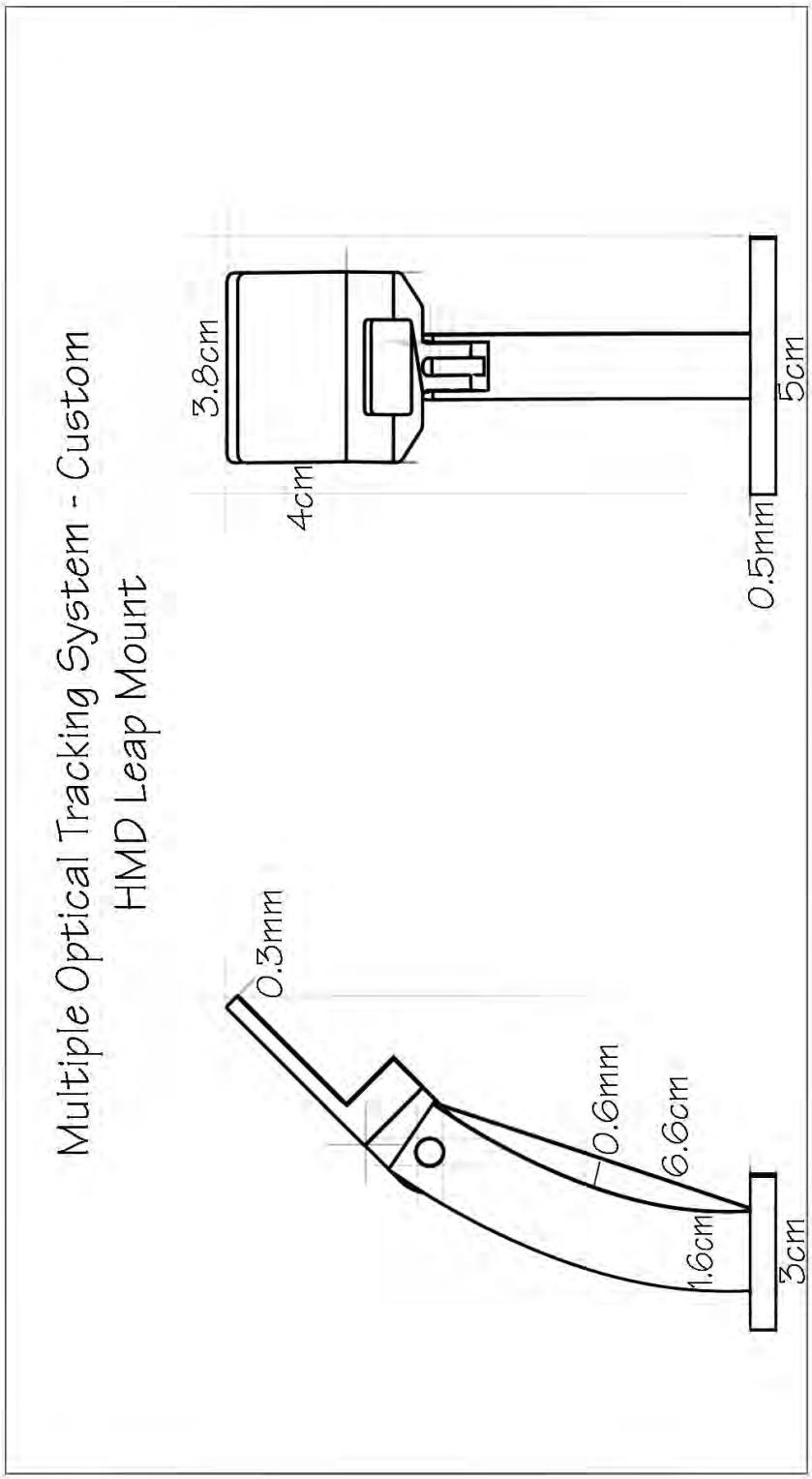
Zhou, T. et al., 2016. A comparative study for telerobotic surgery using free hand gestures. *Journal of Human-Robot Interaction,* Volume 5, pp. 1-28.

Zilak, M., Car, Z. & Jezic, G., 2018. *Educational virtual environment based on oculus rift and leap motion devices.* s.l., s.n.

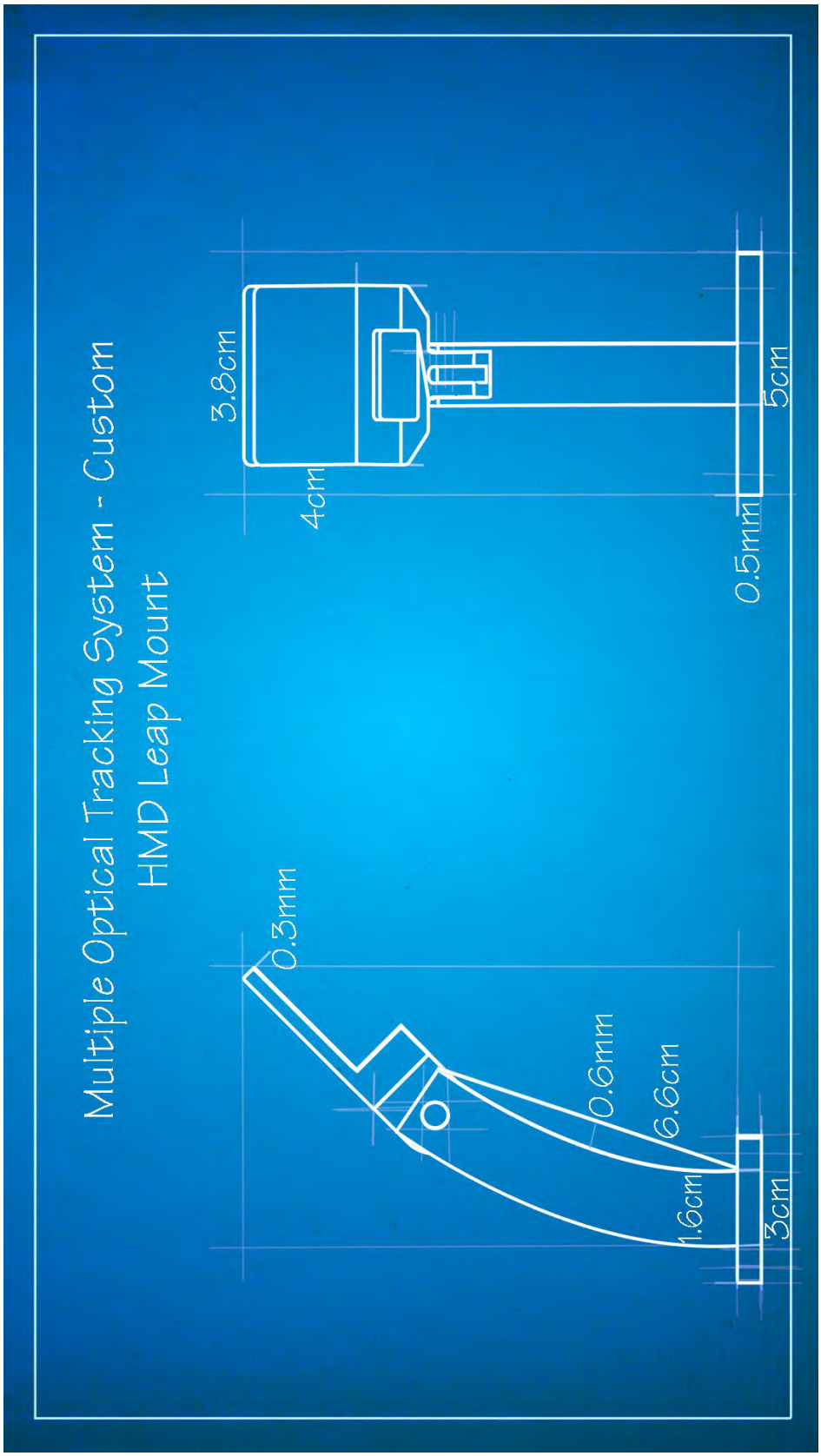
Zou, Y., Liu, H. & Zhang, J., 2019. *Real-time grasp type recognition using leap motion controller.* s.l., s.n., pp. 472-480.

Appendix

Appendix 1



Appendix 2


Multiple Optical Tracking System - Custom
HMD Leap Mount
3.8cm
4cm
0.5mm
5cm
0.3mm
0.6mm
6.6cm
1.6cm
3cm

Appendix 3 – Phase One Implementation Code

3.1 – Data Aggregation System

```
float CalculateSensorWeight(Leap.Vector position)
{
    float HighestX = 0.3f, LowestX = -0.3f;
    float HighestY = 0.4f, LowestY = 0.01f;
    float HighestZ = 0.3f, LowestZ = -0.3f;

    float clampedX = (1f - (-1f)) / (HighestX - LowestX) * (position.x - HighestX) + 1f;
    float clampedY = (1f - (-1f)) / (HighestY - LowestY) * (position.y - HighestY) + 1f;
    float clampedZ = (1f - (-1f)) / (HighestZ - LowestZ) * (position.z - HighestZ) + 1f;

    float totalClamp = clampedX + clampedY + clampedZ;

    float weight = (1f - 0f) / (3f - (-3f)) * (totalClamp - 3f) + 1f;
    weight = ((weight - 0f) / (1f - 0f)) * (1f - (-1f)) + (-1f);
    weight = 1f - Mathf.Abs(weight);

    //Round the weight to 2 decimal places for asthetics and ease
    weight = Round(weight, 2);
    return weight;
}
```

Figure 0.1 - Code used to calculate the weight of a sensor based on hand position

```
if (i == 0)
{
    currentFrame.hands[h].Fingers[f].bones[b].Length
        = data.hands[h].Fingers[f].bones[b].Length;
    currentFrame.hands[h].Fingers[f].bones[b].Width
        = data.hands[h].Fingers[f].bones[b].Width;
    currentFrame.hands[h].Fingers[f].bones[b].Type
        = data.hands[h].Fingers[f].bones[b].Type;

    currentFrame.hands[h].Fingers[f].bones[b].Direction
        = (data.hands[h].Fingers[f].bones[b].Direction * weightMultipliers[i]);
    currentFrame.hands[h].Fingers[f].bones[b].NextJoint
        = (data.hands[h].Fingers[f].bones[b].NextJoint * weightMultipliers[i]);
    currentFrame.hands[h].Fingers[f].bones[b].Center
        = (data.hands[h].Fingers[f].bones[b].Center * weightMultipliers[i]);
    currentFrame.hands[h].Fingers[f].bones[b].PrevJoint
        = (data.hands[h].Fingers[f].bones[b].PrevJoint * weightMultipliers[i]);
    currentFrame.hands[h].Fingers[f].bones[b].Rotation = Quaternion.Euler(
        data.hands[h].Fingers[f].bones[b].Rotation.ToQuaternion().eulerAngles
        * weightMultipliers[i]).ToLeapQuaternion();
}else
{
    currentFrame.hands[h].Fingers[f].bones[b].Direction +=
        (data.hands[h].Fingers[f].bones[b].Direction * weightMultipliers[i]);

    currentFrame.hands[h].Fingers[f].bones[b].NextJoint +=
        (data.hands[h].Fingers[f].bones[b].NextJoint * weightMultipliers[i]);

    currentFrame.hands[h].Fingers[f].bones[b].Center +=
        (data.hands[h].Fingers[f].bones[b].Center * weightMultipliers[i]);

    currentFrame.hands[h].Fingers[f].bones[b].PrevJoint +=
        (data.hands[h].Fingers[f].bones[b].PrevJoint * weightMultipliers[i]);

    currentFrame.hands[h].Fingers[f].bones[b].Rotation = Quaternion.Euler(
        currentFrame.hands[h].Fingers[f].bones[b].Rotation.ToQuaternion().eulerAngles+
        (data.hands[h].Fingers[f].bones[b].Rotation.ToQuaternion().eulerAngles *
         weightMultipliers[i])).ToLeapQuaternion();
}
```

Figure 0.2 - Screenshot of the Code for averaging hands using the frame history for a given source

```
void CalculateEachSensorWeightBasedOnTotal(ref Dictionary<string,float>weights)
{
    if (weights.Keys.Count == 0) return;

    float totalWeight = 0f;
    float actualTotalWeight = 0f;
    foreach(string key in weights.Keys)
    {
        totalWeight += weights[key];
    }
    actualTotalWeight = totalWeight;

    for(int i = 0; i < weights.Keys.Count; i++)
    {
        if (weights[weights.Keys.ElementAt(i)] == 0) continue;

        float newValue = (1f - 0f) / (totalWeight - 0f) *
                         (weights[weights.Keys.ElementAt(i)] - totalWeight) + 1f;
        weights[weights.Keys.ElementAt(i)] = newValue;
    }
}
```

Figure 0.3 - Code for calculating the sensors overall weights

3.2 – Hand Validation System

```
public static bool EquiDistantFingers(Leap.Finger[] fingers)
{
    float AverageSpacing = 0f;
    float thresholdPercentage = 10f;
    float threshold = 0f;
    List<float> spacing = new List<float>();

    for(int i = 1; i < fingers.Length - 1; i++)
    {
        Leap.Bone b1 = fingers[i].GetBone(
            Leap.Bone.BoneType.TYPE_METACARPAL);
        Leap.Bone b2 = fingers[i + 1].GetBone(
            Leap.Bone.BoneType.TYPE_METACARPAL);

        float space = Vector3.Distance(
            b1.NextJoint.ToVector3(),b2.NextJoint.ToVector3());
        AverageSpacing += space;
        spacing.Add(space);
    }
    AverageSpacing = AverageSpacing / spacing.Count;
    threshold = (AverageSpacing / 100f) * thresholdPercentage;

    for(int i = 0; i < spacing.Count-1; i++)
    {
        if (WithinTolerance(spacing[i],
                AverageSpacing, threshold)==false)
        {
            return false;
        }
    }
    return true;
}
```

Figure 0.4 - Finger Spacing Validation Testing Code

```
public static bool VectorIsWithinBoneLength(Vector3 center, Vector3 direction,
    float length, Vector3 vectorToCheck)
{
    return (vectorToCheck.z >= (center - (direction * (length / 2f))).z) &&
           (vectorToCheck.z <= (center + (direction * (length / 2f))).z);
}

public static bool VectorIsWithinRadiusOfAnother(Vector3 center,
    float radius, Vector3 vectorToCheck)
{
    var sqrDistance = Mathf.Pow(center.x - vectorToCheck.x,2) +
                      Mathf.Pow(center.y - vectorToCheck.y,2) +
                      Mathf.Pow(center.z - vectorToCheck.z,2);
    return (sqrDistance < Mathf.Pow(radius,2));
}
```

Figure 0.5 - Code to determine if two bones are within range and within bone length of each other

```
public static bool FingersIntersect(Leap.Finger[] fingers)
{
    foreach(Leap.Finger f in fingers)
    {
        foreach(Leap.Bone.BoneType b in System.Enum.GetValues(
            typeof(Leap.Bone.BoneType)))
        {
            if (b == Leap.Bone.BoneType.TYPE_DISTAL ||
                b == Leap.Bone.BoneType.TYPE_METACARPAL ||
                b== Leap.Bone.BoneType.TYPE_INVALID) continue;
            Leap.Bone outerFinger = f.GetBone(b);
            foreach (Leap.Finger finger in fingers)
            {
                if (finger.Equals(f)) continue;
                Leap.Bone innerFinger = finger.GetBone(b);
                bool withinRadius = VectorIsWithinRadiusOfAnother(
                    outerFinger.Center.ToVector3(),
                    outerFinger.Width / 2f, innerFinger.Center.ToVector3());

                bool withinLength = VectorIsWithinBoneLength(
                    outerFinger.Center.ToVector3(),
                    outerFinger.Direction.ToVector3(), outerFinger.Length,
                    innerFinger.Center.ToVector3());

                if(withinRadius && withinLength)
                {
                    return true;
                }
            }
        }
    }
    return false;
}
```

Figure 0.6 - Fingers Intersecting Validation Code

3.3 – Mesh Conformation System

```
objSpacePt = objectCollider.InverseTransformPoint(
    h.Fingers[f].bones[b-1].NextJoint.ToVector3()
    +(currentHand.Fingers[f].bones[b].Direction.ToVector3()
    * currentHand.Fingers[f].bones[b].Length));

meshPt = NearestPointOnMesh(objSpacePt, verts, kd, mesh.triangles, vt,
    currentHand.PalmNormal.ToVector3(),objectCollider.parent.transform);

worldPt = objectCollider.TransformPoint(meshPt);

float targetDistance = (currentHand.Fingers[f].bones[b].NextJoint -
                        currentHand.Fingers[f].bones[b-1].NextJoint).Magnitude;

float actual = (worldPt.ToVector() - h.Fingers[f].bones[b - 1].NextJoint).Magnitude;

Vector3 direction = worldPt - h.Fingers[f].bones[b - 1].NextJoint.ToVector3();
direction = direction.normalized;
float lengthDifference = targetDistance - actual;

direction = currentHand.Fingers[f].bones[b].Direction.ToVector3().normalized;
worldPt += (direction * lengthDifference);

h.Fingers[f].bones[b].NextJoint = worldPt.ToVector();
h.Fingers[f].bones[b].PrevJoint = h.Fingers[f].bones[b - 1].NextJoint;

h.Fingers[f].bones[b].RecalculateCenter();
h.Fingers[f].bones[b].RecalculateDirection();
h.Fingers[f].bones[b].RecalculateLength();
```

Figure 0.7 - Code to calculate the conformed bone position and modifiy the virtual hand representation

Appendix 4 – Vive Interaction System

The HTC Vive controllers support several buttons, a trigger and a D-Pad to support a wide variety of simulation genres and enable developers to create innovative interaction techniques. Figure 0.8 shows an image of the HTC Vive controllers. The Vive based interaction mechanism uses the rear trigger to perform ‘grab’ and ‘release’ actions, with the trigger held down to continue holding the virtual object. One of the key aspects to the Vive’s interaction design was keeping its reliance on buttons to a minimum as it would reduce the natural feel of the controllers, potentially giving the hand-based approaches bias. In addition, buttons would not be available in the hand-based approaches, therefore their use would result in too significant a difference between the different approaches.

Figure 0.8 - HTC Vive Controller (Front & Rear View)

The entirety of the d-pad is used to toggle the dropper mode and the user interacts with the centrifuge UI by tapping the virtual buttons with the controller as if it were the user’s hand. The ‘Child of Controller’ approach as detailed in Appendix 9 was used for object interactions with the Unity plugin VRTK (Virtual

Reality Toolkit) used to aid in its implementation. In order to detect when the user has picked up and released an object, the VRTK events “InteractableObjectGrabbed” and “InteractableObjectReleased” were used respectively.

In some cases, additional functionality is needed when interacting with a virtual object such as the pipette in the case of the ‘Chemistry Fun’ VR simulation, where the d-pad is needed to toggle the between pipette modes. During an “Object Picked Up” event, an event listener is added to the “Touchpad Pressed” event with it then removed during the “Object Released” event. This ensures that object specific functionality cannot be used once the object has been released.

Appendix 5 – Manus Interaction System

The Manus Gloves interaction system is a hand-based interaction system for VR that uses bend-sensor based gloves to integrate user's hands into a virtual environment. The system comprises two Manus VR gloves; one for each hand. Figure 0.9 show the Manus VR gloves.

The Manus Gloves interaction system was designed to support non-physics-based interaction as with the other two conditions. The design of the Manus Interaction system and the non-physics-based interaction mechanism was influenced by elements of the MOT systems design; specifically, it's interaction mechanics. The original mechanism for determining the hands state remains, however the subsequent functionality was changed. The Manus Interaction System used the same object interaction mechanism as the MOT system detailed in Appendix 11. During the update cycle of the Manus Interaction system object interactions are handled and the virtual hand representations are updated, by analysing the bend-sensor data. The interaction states (open or closed/fist) of the virtual hand representations are determined by comparing the bend sensor data against set thresholds.

Figure 0.9 - Photo of the Manus VR glove, with the hand in an open hand pose.

Appendix 6 – Chemistry Fun Simulation Design

This section details the design of the ‘Chemistry Fun’ virtual reality simulation, the interactable objects and the three different task types; translation/rotation, two handed and virtual user interface (VUI), that are used within the simulation. Each of the interactable objects types are detailed, including their individual designs and any object-specific interactions that had to be designed, whilst ensuring the same level of control was available in all three conditions.

Beakers

In the simulation there are five beakers; four small chemical beakers and a larger “mixing” beaker. The four chemical beakers are dynamically assigned a chemical and fluid colour at run-time, with a label texture generated and applied to the beaker. The interaction for the chemical beakers follows the simple ‘grab’ and ‘release’ approach. To pour the chemicals, the beaker must be tipped and within proximity of the mixing beaker. The fluid levels on a beaker is adjusted while pouring by adjusting the z-axis scale to represent the increase of decrease in fluid. Once the beaker has been emptied, it snaps back to its original position and becomes un-interactable.

Test Tubes

In the simulation there a two test tubes; one for use with the centrifuge and the other for the Bunsen burner. In order to add or remove fluid, the pipette must be used. The interaction with the test tubes is minimal, predominantly consisting of translation tasks. They can be picked up and moved around the virtual environment to position them in either the Bunsen burner or centrifuge and can be filled or drained using the pipette.

Pipette

The pipette is designed to have a minimalist appearance with the end of the handle changing colour to represent its current state; suction or release. The pipette is used to transfer chemicals between beakers and test tubes for use in both the centrifuge and Bunsen burner. To prevent the user from making mistakes the pipette will not release a chemical into a container containing a different chemical and will not change state within interactive proximity of a container. In the hand-based techniques, the pipettes mode is switched by tapping on a cube on the end of the handle with the palm of the hand and in the case of the controller by clicking anywhere on the d-pad. Figure 0.10 shows the virtual representation of the pipette dropper.

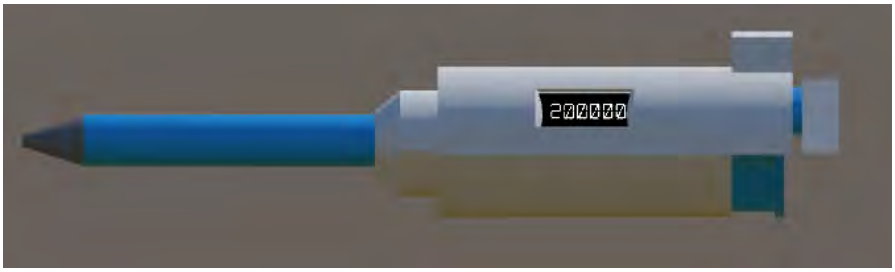


Figure 0.10 – Dropper used in the simulation

Centrifuge

The centrifuge is designed to automatically load test tubes containing chemicals within proximity and present the user with a virtual UI to configure the machine. The user can adjust both the speed and duration of the spin cycle along with the necessary start button. The centrifuge UI uses a trigger-based interaction mechanic enabling the user to interact by tapping the virtual button. If the user has entered incorrect settings an error message is displayed on the

virtual UI informing them. Figure 0.11 shows the centrifuge virtual user interface.

Figure 0.11 – Screenshot of the simulation showing the Centrifuge Configuration UI

Bunsen Burner

The Bunsen burner (seen in Figure 0.11 above) uses a trigger-based proximity mechanic with the heating process beginning once a test tube is placed near the flame. To provide the user with visual feedback on the process, a pop UI is displayed with a flame silhouette that slowly fills and turns green once the heating process has completed.

Clipboard

The clipboard is positioned on the back wall and presents the user with a series of instructions on how to complete the experiment. The clipboard is automatically updated upon task completion with completed tasks displayed in green.

Appendix 7 - Hand Calibration System

During the first phase of testing one aspect of the MOT system that users commented on was the extent to which they must close their hand for the system to detect their hand as closed and grab a virtual object. Some participants reported issues with their virtual hands picking up objects with their hand slightly cupped due to being detected as a fist gesture.

In the phase one design detailed in Chapter 3, the MOT system used the Leap API to detect when a finger is extended, with a first gesture recognized when none of them are extended. To provide users with a more tailored and accurate experience, a new Hand calibration system was designed to replace the existing functionality, enabling the user to calibrate the system to their hands movement range. At the start of the simulation the user performs both a fist and open hand pose which are stored for calibration.

The system is designed so that each frame all the bones in the MOT tracking data are iterated with the angle for bone calculated along with the angles of the same bone in the stored open and closed poses. To calculate the bone angle, the direction of the current and next bone are projected on planes (Figure 3.26) using the negative of the radial axis (blue axis in Figure 3.14) as the plane normal. The signed angle between the two projected vectors is then calculated. The calculated angle for the bone is then mapped from the range of the angles for the open and closed poses to the range of zero to one. Once all the bones have been processed, the total weight of all the bones in mapped to the scale of zero to one and compared against a set threshold (0.6 by default). The hand is classified as closed (fist) if the threshold is exceeded.

Appendix 8 - Tracking Data Debugging Tool

During the final stages of development, debugging the MOT system became increasingly difficult due to the Unity Gizmos tool only being able to display a single frame (the frame I pause the editor on) with the hand renderings of the different stages of the update cycle overlaid.

To aid development, a tracking data debugging tool was developed that can store the hand data at different stages of the MOT update cycle for each frame as separate render layers that can be toggled providing much greater control and depth of information. All the data is stored in a timeline that can be scrolled through to visualize the data at any point since runtime. The hand data is rendered using the Unity Gizmos tool, enabling the simulation to be paused within the Unity editor, preventing further data logging while still allowing the tool to be used.

A custom inspector window was designed to provide an easy to use interface for the tool (Figure 0.13), such as the timeline slider at the top of the interface, with next and previous buttons available for frame by frame analysis. Both the left and right hand can be analysed but to limit performance requirements, only one hand can be rendered at once, with the current hand selected via a drop-down menu. Figure 0.12 shows the tool in use with three layers enabled for rendering and the timeline been searched through frame by frame.

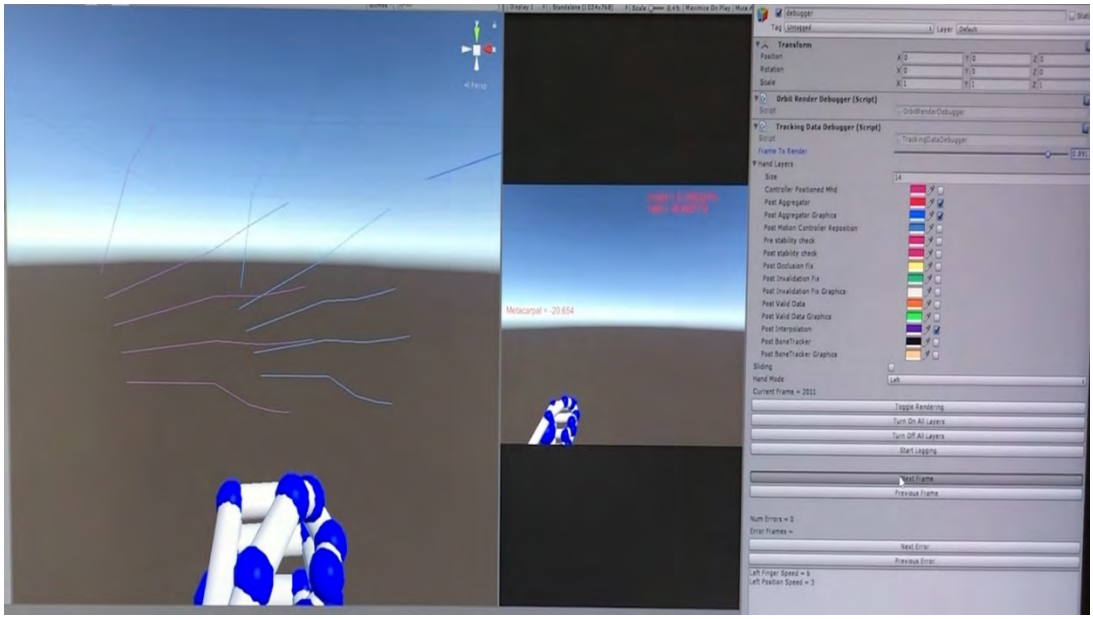


Figure 0.12 - Photo of the Tracking Data Debugging Tool in use

As the default Unity console can become quite cluttered with debug messages which cannot be linked to a given frame, the inspector features a console output at the bottom. The console displays any messages logged with the select frames hand data to provide additional debugging information. Finally, two buttons can be used to jump back-and-forth between errors logged in the tools console output, making error finding easier. The combination of the error jumping and frame-by-frame searching makes it easier to track down errors by jumping to the next error and then analysing the previous and post error frames to determine the conditions that led to the error.

Figure 0.13 - Tracking Data Debugging Tool

Appendix 9 – Phase Two Implementation Code

```
public void GenerateSavedFrame(Hand h)
{
    SavedFrame sf = new SavedFrame();
    foreach (Finger f in h.Fingers)
    {
        for (int b = 1; b < f.bones.Length; b++)
        {
            Vector3 bone0Dir = f.bones[b].Direction.ToVector3();
            Vector3 bone1Dir = f.bones[b-1].Direction.ToVector3();
            bone0Dir = Vector3.ProjectOnPlane(bone0Dir,
                Vector3.Cross(h.PalmarAxis(), h.DistalAxis()));
            bone1Dir = Vector3.ProjectOnPlane(bone1Dir,
                Vector3.Cross(h.PalmarAxis(), h.DistalAxis()));
            float angle = Vector3.SignedAngle(bone0Dir, bone1Dir,
                -Vector3.Cross(h.PalmarAxis(), h.DistalAxis()));
            //normalize to -1 to 1 for NN
            angle = (1f - (-1f)) / (180f - (-180f))
                    * (angle - 180f) + 1f;
            sf.bonesRotations.Add(angle);
        }
    }
    savedFrames.Add(sf);
}
```

Figure 0.14 - Code to calculate angles between the bones and normalize them for Deep Neural Network training data

```
for(int n = 0; n < numRecordsToGenerate; n++)
{
    invalidhData = new HandData();
    invalidhData = invalidhData.CopyFrom(capturedhData);
    Hand ha = invalidhData.GetHand((handToUse == Chirality.Left));

    //Reset hand back to default open state
    Hand newHand = new Hand();
    newHand = newHand.CopyFrom(ha);
    for (int f = 0; f < newHand.Fingers.Count; f++)
    {
        newHand.Fingers[f] = RotateBone(newHand.Fingers[f],
            boneAngles[f].bone1MinAngle,1, newHand.RadialAxis());
        newHand.Fingers[f] = RotateBone(newHand.Fingers[f],
            boneAngles[f].bone2MinAngle,2, newHand.RadialAxis());
        newHand.Fingers[f] = RotateBone(newHand.Fingers[f],
            boneAngles[f].bone3MinAngle,3, newHand.RadialAxis());
        newHand.Fingers[f] = RotateBone(newHand.Fingers[f],
            boneAngles[f].bone4MinAngle,4, newHand.RadialAxis());
    }
    ha = ha.CopyFrom(newHand);

    if (ha != null)
    {
        for (int f = 0; f < ha.Fingers.Count; f++)
        {
            ha.Fingers[f] = RotateBone(ha.Fingers[f],
                Random.Range(minAngleRange,maxAngleRange),1, ha.RadialAxis());
            ha.Fingers[f] = RotateBone(ha.Fingers[f],
                Random.Range(minAngleRange, maxAngleRange),2, ha.RadialAxis());
            ha.Fingers[f] = RotateBone(ha.Fingers[f],
                Random.Range(minAngleRange, maxAngleRange),3, ha.RadialAxis());
            ha.Fingers[f] = RotateBone(ha.Fingers[f],
                Random.Range(minAngleRange, maxAngleRange),4, ha.RadialAxis());
        }
        GenerateSavedFrame(ha);
    }
    if (n % 1000 == 0)
    {
        //we have done another 1000 records, so lets save again
        SaveFrames(savedFrames,filename);
        savedFrames.Clear();
    }
}
SaveFrames(savedFrames,filename);
savedFrames.Clear();
Debug.Log("Invalid frames generated successfully");
```

Figure 0.15 - The code to generate invalid hand data for training the upper bone angles Deep Neural Network

```
with open("outputData.txt") as textFile:
    lines = [line for line in textFile]

x_train =[]
temp = []
for string in lines:
    num_array = string.split(',')
    for num in num_array:
        if num.strip() != "" and num.strip() !="\n":
            temp.append(float(num))
    x_train.append(temp)
    temp = []

x_train = np.array(x_train, dtype=float)

with open("outputAnswers.txt") as textFile:
    linesA = [line for line in textFile]

y_train=[]
for string in linesA:
    y_train.append(int(string))

y_train = np.array(y_train, dtype = int)
x_test = x_train[:round((len(lines)/100)*20)]
x_train = x_train[round((len(lines)/100)*20):]
y_test = y_train[:round((len(linesA)/100)*20)]
y_train = y_train[round((len(linesA)/100)*20):]
y_train = to_categorical(y_train,2)
y_test = to_categorical(y_test,2)

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(name="input_node"))
model.add(tf.keras.layers.Dense(15,activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(7,activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(2,activation=tf.nn.softmax,
    name="output_node"))

model.compile(optimizer='adam', loss='categorical_hinge',
    metrics=['categorical_accuracy'])
model.fit(x_train,y_train, validation_split=0.30, epochs=3)

model.save('leapModel.model')
print(model.summary())

frozen_graph = freeze_session(K.get_session(),
    output_names=[out.op.name for out in model.outputs])
tf.train.write_graph(frozen_graph, "./", "leap_model.pb",
 as_text=False)
```

Figure 0.16 - Screenshot of the Neural Network Code using TensorFlow and Keras for Bone Angle Classification

```
void Awake()
{
    instance = this;
    graph = new TFGraph();
    if (File.Exists(Application.dataPath +
       "/MultiLeap_V2/ML & Validator/MultiLeap_Training/leap_model.pb") == false)
    {
        Debug.LogError("Path to file is incorrect, unable to load model");
        fileFound = false;
    }
    graph.Import(File.ReadAllBytes(Application.dataPath +
         "/MultiLeap_V2/ML & Validator/MultiLeap_Training/leap_model.pb"));
    session = new TFSession(graph);
}
```

Figure 0.17 - Code to load a saved Deep Neural Network model in the simulation at runtime

```
float[] CalculateAngles(Hand h)
{
    List<float> d = new List<float>();
    Vector3 cross = Vector3.Cross(h.PalmarAxis(), h.DistalAxis());

    for(int f=0;f<h.Fingers.Count;f++)
    {
        for (int b = 1; b < h.Fingers[f].bones.Length; b++)
        {
            Vector3 bone0Dir = h.Fingers[f].bones[b].Direction.ToVector3();
            Vector3 bone1Dir = h.Fingers[f].bones[b-1].Direction.ToVector3();

            bone0Dir = Vector3.ProjectOnPlane(bone0Dir, cross);
            bone1Dir = Vector3.ProjectOnPlane(bone1Dir, cross);

            float angle = Vector3.SignedAngle(bone0Dir, bone1Dir, -cross);
            angle = (1f - (-1f)) / (180f - (-180f)) * (angle - 180f) + 1f;
            d.Add(angle);
        }
    }
    return d.ToArray();
}
```

Figure 0.18 - Code to calculate angle input data for the neural network

```
private float MakePrediction(float[] data, float minConfidence = 0.85f)
{
    var runner = session.GetRunner();
    float[][] inputs = new float[][] { data };

    runner.AddInput(graph["sequential_input"][0], new TFTensor(inputs));
    runner.Fetch(graph["output_node/Softmax"][0]);

    var output = runner.Run();

    TFTensor result = output[0];
    float[,] outputs = (float[,])result.GetValue();

    float invalid = outputs[0, 0];
    float valid = outputs[0, 1];

    if (valid > invalid && (valid > minConfidence))
    {
        return valid;
    }
    return 0f;
}
```

Figure 0.19 - Code to evaluate hand data against a Deep Neural Network

```
public bool ValidateHand(Hand h, float minConfidence = 0.85f)
{
    if (h == null) return false;
    float[] data = CalculateAngles(h);

    if (enabled == false) return true;

    if (data.Length > 0)
    {
        float validPercentage = MakePrediction(data, minConfidence);

        float lowerValidity = Tensorflow_LowerValidator.instance.MakePrediction(data,
            minConfidence);

        float spacingValidity = TensorFlow_SpacingValidator.instance.ValidateHand(h);

        float bLength = Tensorflow_BoneLengthValidator.instance.ValidateHand(h,
            minConfidence);

        float totalValidity = (validPercentage * 0.30f) + (bLength * 0.30f) +
                              (spacingValidity * 0.30f) + (lowerValidity * 0.10f);

        if(totalValidity >= minConfidence)
        {
            return true;
        }
    }
    return false;
}
```

Figure 0.20 - Code to determine hand validity using a weighted average of deep neural network confidences

```
public BoneBounds(Hand h, Bone b)
{
    b.RecalculateDirection();
    direction = b.Direction.ToVector3();
    forwardBone = Quaternion.AngleAxis(90f, h.RadialAxis()) * direction;
    rightBone = Quaternion.AngleAxis(90f, b.Direction.ToVector3()) * forwardBone;
    b.RecalculateCenter();
    center = b.Center.ToVector3();

    Vector3 topleft = b.PrevJoint.ToVector3() - (rightBone.normalized *
        (SPHERE_RADIUS*scale)) + (-forwardBone.normalized * (SPHERE_RADIUS*scale));
    Vector3 topright = b.PrevJoint.ToVector3() + (rightBone.normalized *
        (SPHERE_RADIUS*scale)) + (-forwardBone.normalized * (SPHERE_RADIUS*scale));
    Vector3 bottomleft = b.PrevJoint.ToVector3() - (rightBone.normalized *
        (SPHERE_RADIUS*scale)) + (forwardBone.normalized * (SPHERE_RADIUS*scale));
    Vector3 bottomright = b.PrevJoint.ToVector3() + (rightBone.normalized *
        (SPHERE_RADIUS*scale)) + (forwardBone.normalized * (SPHERE_RADIUS*scale));

    ConfigureBoneBounds(b,topleft,topright,bottomleft,bottomright,
       b.Direction.ToVector3(),b.Length);
}
private void ConfigureBoneBounds(Bone b, Vector3 topL, Vector3 topR, Vector3 bottomL,
    Vector3 bottomR, Vector3 dir, float depth)
{
    this.height = depth;
    this.direction = dir;
    upperTopLeft = topL;
    upperTopRight = topR;
    upperBottomLeft = bottomL;
    upperBottomRight = bottomR;

    //Calculate lower positions
    lowerTopLeft = b.NextJoint.ToVector3() - (rightBone.normalized *
       (SPHERE_RADIUS*scale))+ (-forwardBone.normalized * (SPHERE_RADIUS*scale));
    lowerTopRight = b.NextJoint.ToVector3() + (rightBone.normalized *
       (SPHERE_RADIUS*scale))+ (-forwardBone.normalized * (SPHERE_RADIUS*scale));
    lowerBottomLeft = b.NextJoint.ToVector3() - (rightBone.normalized *
       (SPHERE_RADIUS*scale))+ (forwardBone.normalized * (SPHERE_RADIUS*scale));
    lowerBottomRight = b.NextJoint.ToVector3() + (rightBone.normalized *
       (SPHERE_RADIUS*scale))+ (forwardBone.normalized * (SPHERE_RADIUS*scale));

    center = (lowerBottomLeft + upperTopRight) * 0.5f;
}
```

Figure 0.21 - Code to create bone geometry

```
static bool ClipLine(int d, BoneBounds hb, Vector3 v0, Vector3 v1,
    ref float f_low, ref float f_high)
{
    float f_dim_low, f_dim_high;

    f_dim_low = (hb.GetMinAxis(d) - GetVector3Axis(v0, d)) /
                (GetVector3Axis(v1, d) - GetVector3Axis(v0, d));
    f_dim_high = (hb.GetMaxAxis(d) - GetVector3Axis(v0, d)) /
                 (GetVector3Axis(v1, d) - GetVector3Axis(v0, d));

    if (f_dim_high < f_dim_low)
        swap(ref f_dim_high, ref f_dim_low);

    if (f_dim_high < f_low)
        return false;

    if (f_dim_low > f_high)
        return false;

    f_low = Mathf.Max(f_dim_low, f_low);
    f_high = Mathf.Min(f_dim_high, f_high);

    if (f_low > f_high)
        return false;

    return true;
}
```

Figure 0.22 - Code to check whether a ray intersects with the planes of the geometry

```
static bool LineAABBIntersection(BoneBounds hb, Vector3 v0, Vector3 v1,
    out Vector3 vecIntersection, out float flFraction)
{
    float f_low = 0f;
    float f_high = 1f;

    vecIntersection = v0;
    flFraction = 1000f;

    if (!ClipLine(0, hb, v0, v1, ref f_low, ref f_high))
    {
        return false;
    }

    if (!ClipLine(1, hb, v0, v1, ref f_low, ref f_high))
    {
        return false;
    }

    if (!ClipLine(2, hb, v0, v1, ref f_low, ref f_high))
    {
        return false;
    }
    Vector3 b = v1 - v0;
    vecIntersection = v0 + (b * f_low);
    flFraction = Vector3.Distance(v0, vecIntersection);
    return true;
}
```

Figure 0.23 - Code to perform LineAABB intersection

```
static bool BoneBoundsBoxRaycastIntersection(BoneBounds bb, BoxCast bx1)
{
    float minDistance = 1000f;
    float distance = 1000f;
    Vector3 intersection;

    //Top left
    LineAABBIntersection(bb, bx1.UpperTopLeft, bx1.LowerTopLeft,
        out intersection, out distance);
    if (distance < minDistance){minDistance = distance;}

    //Top right
    LineAABBIntersection(bb, bx1.UpperTopRight, bx1.LowerTopRight,
        out intersection, out distance);
    if (distance < minDistance){minDistance = distance;}

    //Bottom left
    LineAABBIntersection(bb, bx1.UpperBottomLeft, bx1.LowerBottomLeft,
        out intersection, out distance);
    if (distance < minDistance){minDistance = distance;}

    //Bottom right
    LineAABBIntersection(bb, bx1.UpperBottomRight, bx1.LowerBottomRight,
        out intersection, out distance);
    if (distance < minDistance){minDistance = distance;}

    //Middle
    LineAABBIntersection(bb, bx1.UpperMiddle, bx1.LowerMiddle,
        out intersection, out distance);
    if (distance < minDistance){minDistance = distance;}

    return (minDistance<1000f);
}
```

Figure 0.24 - Code to determine if there is an intersection between custom bone bounds geometry and a custom boxcast

```
public static float CalculatePercentageDifferenceBetweenHands(Hand previous, Hand current)
{
    float total = 0f;
    float possibleTotal = 0f;

    for (int f = 0; f < current.Fingers.Count; f++)
    {
        for (int b = 0; b < current.Fingers[f].bones.Length; b++)
        {
            total += Extensions_Math.PercentageDifferenceBetweenVectors(
                previous.Fingers[f].bones[b].NextJoint.ToVector3(),
                current.Fingers[f].bones[b].NextJoint.ToVector3());

            total += Extensions_Math.PercentageDifferenceBetweenVectors(
                previous.Fingers[f].bones[b].PrevJoint.ToVector3(),
                current.Fingers[f].bones[b].PrevJoint.ToVector3());

            possibleTotal += 200f;
        }
    }

    total += Extensions_Math.PercentageDifferenceBetweenVectors(
        previous.PalmPosition.ToVector3(),
        current.PalmPosition.ToVector3());
    possibleTotal += 100f;

    float result = Extensions_Math.RangeConvert(total, 0f, possibleTotal, 100f, 0f);
    return result;
}
```

Figure 0.25 - Code to calculate the percentage difference between two hands

```
public static float PercentageDifference(float val, float target)
{
    float increase = val - target;
    float diff = (increase / target) * 100f;
    return diff;
}

public static float PercentageDifferenceBetweenVectors(Vector3 a,
    Vector3 b)
{
    float x = Mathf.Abs(PercentageDifference(a.x, b.x));
    float y = Mathf.Abs(PercentageDifference(a.y, b.y));
    float z = Mathf.Abs(PercentageDifference(a.z, b.z));
    return RangeConvert((x + y + z), 0f, 300f, 100f, 0f);
}
```

Figure 0.26 - Code to calculate the percentage difference between two Vectors

```
if (returningToLive_left && boneTracker.GetMergedData.LeftHand !=null && mhd !=null)
{
    if (returnToLiveFrame_left == null)
    {
        returnToLiveFrame_left = new Hand().CopyFrom(
            HandInterpolator.TransformHandToViveTrackerPosition(
            Chirality.Left,lastvalid.LeftHand));
        leftFingersInterpolating = true;
        leftPositionInterpolating = true;
        leftStabilityStartTime = Time.time;
    }
    else
    {
        float diff =
            HandInterpolator.CalculatePercentageDifferenceBetweenHands(
                returnToLiveFrame_left,boneTracker.GetMergedData.LeftHand);

        if (diff >= threshold)
        {
            returnToLiveFrame_left = returnToLiveFrame_left.CopyFrom(
                boneTracker.GetMergedData.LeftHand);
            leftDataStabilized = false;
            leftStabilityStartTime = Time.time;
        }
        else
        {
            if ((Time.time - leftStabilityStartTime) > stabilityTimeRequired)
            {
                leftDataStabilized = true;
                returningToLive_left = false;
                leftFingersInterpolating = true;
                leftPositionInterpolating = true;
                returnToLiveFrame_left = null;
                leftMinInterpPosSpeed = 3f;
            }
        }
    }
}
```

Figure 0.27 - Code to check the stability of the left-hand tracking data

```
if (leftTargetHand != null && validationData.LeftHand == null)
{
    //Check the angle to see if we should start interpolating
    Vector3 tipPosition = (leftTargetHand.Fingers[2].TipPosition.ToVector3());

    Vector3 dir = tipPosition - transform.position;
    dir = Vector3.ProjectOnPlane(dir,transform.forward);
    float ang = Vector3.SignedAngle(transform.up, dir, transform.forward);

    if (ang >= -70f && ang <= 70f)
    {
        leftPositionInterpolating = true;
        leftFingersInterpolating = true;
    }
    else if(ang <-70f || ang > 70f)
    {
        leftPositionInterpolating = false;
        leftFingersInterpolating = false;
    }
}
```

Figure 0.28 - Code to automatically begin interpolation as the Vive tracking system controlled virtual hand representation re-enters the users view

```
if (leapHand != null && displayHandLeft != null)
{
    if(HandInterpolator.DetectJump(displayHandLeft,leapHand,0.2f)
       && leftWithinOptimalRange)
    {
        if (leftFingersInterpolating == false)
        {
            leftFingersInterpolating =
                HandInterpolator.InterpolationRequired(displayHandLeft,
                    previousLeftHand,leapHand,radius*1.5f,99f);
        }
        if (leftPositionInterpolating == false)
        {
            leftPositionInterpolating =
                HandInterpolator.CheckPositionInterpolationRequired(
                    displayHandLeft,previousLeftHand,leapHand,radius*1.5f,99f);
        }
    }
}
```

Figure 0.29 - Code to check for jumps and trigger interpolation if detected

```
public static bool InterpolationRequired(Hand current, Hand previous, Hand target,
    float radius, float targetVal = 99.8f)
{
    if (current != null && previous != null && target != null)
    {
        float temp = 0f;

        for (int f = 0; f < current.Fingers.Count; f++)
        {
            for (int b = 0; b < current.Fingers[f].bones.Length; b++)
            {
                temp += WithinRadius(current.Fingers[f].bones[b].NextJoint.ToVector3(),
                    target.Fingers[f].bones[b].NextJoint.ToVector3(),radius);

                temp += WithinRadius(current.Fingers[f].bones[b].PrevJoint.ToVector3(),
                    target.Fingers[f].bones[b].PrevJoint.ToVector3(),radius);
            }
        }
        float val = Extensions_Math.Round((100f - ((temp / 40f) * 100f)), 2);
        if (val >= targetVal)
        {
            return false;
        }
        else
        {
            return true;
        }
    }
    return false;
}
```

Figure 0.30 - Code to determine if finger interpolation between two hands is required

```
static float WithinRadius(Vector3 previous, Vector3 current, float radius)
{
    float distance = (previous - current).magnitude;
    if (Extensions_Math.Round(distance,3) <= radius)
    {
        return 0f;
    }
    else
    {
        float increase = distance - radius;
        float percentageIncrease = (increase / radius) * 100f;
        float val = (percentageIncrease / 100f);
        return Mathf.Clamp01(val);
    }
    return 0f;
}
```

Figure 0.31 - Code to determine if two positions are within a set radius of each other and return the percentage increase on the scale of 0-1

```
public static Leap.Vector LerpVector(Leap.Vector a, Leap.Vector b,
    float t)
{
    return (a + (b - a) * t);
}
```

Figure 0.32 - Code to perform interpolation of a vector

```
for (int f = 0; f < h.Fingers.Count; f++)
{
    for (int i = 0; i < h.Fingers[f].bones.Length; i++)
    {
        offset = pose.Fingers[f].bones[i].NextJoint - pose.PalmPosition;
        Vector newPosition = input.PalmPosition + offset;
        h.Fingers[f].bones[i].NextJoint = LeapExtensions.LerpVector(
            h.Fingers[f].bones[i].NextJoint, newPosition, t);

        offset = pose.Fingers[f].bones[i].PrevJoint - pose.PalmPosition;
        newPosition = input.PalmPosition + offset;
        h.Fingers[f].bones[i].PrevJoint = LeapExtensions.LerpVector(
            h.Fingers[f].bones[i].PrevJoint, newPosition, t);

        h.Fingers[f].bones[i].Direction = (h.Fingers[f].bones[i].NextJoint
                                           - h.Fingers[f].bones[i].PrevJoint);
        h.Fingers[f].bones[i].Center = (h.Fingers[f].bones[i].PrevJoint
                                        + h.Fingers[f].bones[i].NextJoint) / 2f;
    }
    h.Fingers[f].Direction = LeapExtensions.LerpVector(
        h.Fingers[f].Direction, pose.Fingers[f].Direction, t);
}
```

Figure 0.33 - Code to interpolate the fingers of a hand object to a target pose

```
Vector offsetPosition;
Vector offset = pose.PalmPosition - input.PalmPosition;
for (int f = 0; f < h.Fingers.Count; f++)
{
    for (int i = 0; i < h.Fingers[f].bones.Length; i++)
    {
        offsetPosition = input.Fingers[f].bones[i].NextJoint + offset;
        h.Fingers[f].bones[i].NextJoint = LeapExtensions.LerpVector(
            h.Fingers[f].bones[i].NextJoint, offsetPosition, t);

        offsetPosition = input.Fingers[f].bones[i].PrevJoint + offset;
        h.Fingers[f].bones[i].PrevJoint = LeapExtensions.LerpVector(
            h.Fingers[f].bones[i].PrevJoint, offsetPosition, t);

        h.Fingers[f].bones[i].Direction = (h.Fingers[f].bones[i].NextJoint -
                                           h.Fingers[f].bones[i].PrevJoint);

        h.Fingers[f].bones[i].Center = (h.Fingers[f].bones[i].PrevJoint +
                                        h.Fingers[f].bones[i].NextJoint) / 2f;
    }
}
```

Figure 0.34 - Code for hand position interpolation

```
for (int f = 0; f < h.Fingers.Count; f++)
{
    for (int i = 0; i < h.Fingers[f].bones.Length; i++)
    {
        offset = pose.Fingers[f].bones[i].NextJoint - pose.PalmPosition;
        Vector newPosition = input.PalmPosition + offset;
        h.Fingers[f].bones[i].NextJoint = newPosition;

        offset = pose.Fingers[f].bones[i].PrevJoint - pose.PalmPosition;
        newPosition = input.PalmPosition + offset;
        h.Fingers[f].bones[i].PrevJoint = newPosition;

        h.Fingers[f].bones[i].Direction = (h.Fingers[f].bones[i].NextJoint
                                           - h.Fingers[f].bones[i].PrevJoint);
        h.Fingers[f].bones[i].Center = (h.Fingers[f].bones[i].PrevJoint
                                        + h.Fingers[f].bones[i].NextJoint) / 2f;
    }

    h.Fingers[f].Direction = pose.Fingers[f].Direction;
}
```

Figure 0.35 - Code used to set finger positions if they are not interpolating

Appendix 10 – MOT System Phase Two General Improvements

During the second phase of development, a few modifications were made to the MOT system to improve performance and streamline the system. In addition, a few small bugs were fixed. One of the first improvements made was the implementation of a pose caching system to improve the performance of the system when using inferred hand poses such as during periods of invalid data. In the original implementation the hand pose would be loaded from a file each time the hand was requested, which could potentially be multiple times during an update cycle. The pose caching system loads all stored poses at runtime and serves copies of the poses upon request, providing a small but significant performance improvement.

To support the implementation of the functionality detailed above by providing access to the required data, a new external client application was developed using the Unity3D engine. The new client application was developed using the code from the original Win32 C# client application, with the Unity Leap API providing more data and functionality than the Leap C# API.

Based on the feedback received during phase one of experimentation, the mesh conformation system was extended to also support the conformation of the real-time tracking data. The conformation system conforms virtual finger representations upon intersection with the surface of a virtual objects mesh, with live tracking of the finger resumed when the live data indicates the intersection has ceased. This enables users to adjust their virtual finger positioning without having to release the object.

The sensor weighting system used in the data aggregator was extended to calculate a sensor weight for each hand as opposed to an overall weight for the sensor, enabling each hand to prioritize the sensor with the highest weighting for the given hand. In addition, with the change to a Unity3D based external client application, some of the computations such as calculating frame weighting were moved to the client application with the MultiLeapTransferPacket library extended to support the transfer of the additional data.

Finally, the MOT system was refactored into more modular components for easier working and tailoring of the user experience with features easily toggled.

Appendix 11 – MOT Object Interaction Mechanic

The MOT system uses the “Child of Controller” mechanic for object interaction, with a virtual object becoming a ‘child’ of the virtual hand representation while held, maintain the relative position and rotation observed when picking up the object. The current state (open or closed/fist) of the virtual hands is determined by analysing the extension state of each finger, with the hand state overwritten while the Vive tracking system is used. The ‘closed’ state requires at least three fingers to be positioned into a fist pose or the hand must be holding or within proximity of a virtual object in the case of Vive tracking control.

In order to detect when a virtual hand is within proximity of a virtual object, the Unity3D physics trigger-based system is used that fires events when the collider of a virtual object intersects with that of the virtual hand representation. The virtual hands have sphere-based collider child objects that use Unity’s trigger system to detect these intersections with virtual objects and fire the events.

The Unity trigger system uses three events for the start of the intersection, during and once the intersection has ended; specifically, ‘OnTriggerEnter’, ‘OnTriggerStay’ and ‘OnTriggerExit’. During the ‘OnTriggerEnter’ event the hands state (open or closed/fist) is analysed to determine whether the object should be picked up. To pick up the virtual object it is parented to the collider object with which it has intersected.

A script ‘TriggerBinder’, attached to the hands sphere collider is used to track virtual objects within proximity as detected by the Unity Trigger system, storing the objects in a list. During the MOT systems update cycle, the trigger binders

used for both the left and right hands automatically release any currently held objects if the hand is open (Figure 0.36) and remove them from the intersection list, along with any objects that are no longer within proximity. If the items were not removed from the intersection list, the user could perform a fist gesture and pickup an object not within its proximity.

```
if(transform.childCount > 0)
{
    if (LeapPI_HandData.Instance.HandOpened(hType) &&
        MultiLeap_DataAnalysisV2._instance.GetViveHand(hType)==false)
    {
        LeapPhysicsManager.Instance.ReleaseAllObjects_Trigger((hType == HandType.LEFT));
    }
}
```

Figure 0.36 - MOT Trigger Binder Update Method

The main interaction functionality is handled by the ‘LeapGrabber’ script which handles the ‘Child of Controller’ mechanic for a given hand, both grabbing and releasing virtual objects in accordance with the hands state. During the update cycle, if the hands state is closed, it is not holding an object and the interactable object is not held by the other hand, the ‘Grab_Trigger’ method is used to pick up the first object in the corresponding hands ‘TriggerBinder’ interaction list (Figure 0.37).

```
if (LeapPI_HandData.Instance.HandClosed(HandType))
{
    foreach (var rb in TriggerBinder.CollidingObjects)
    {
        if (!_grabbedItems_Trigger.Contains(rb) && rb != null &&
            _grabbedItems_Trigger.Count < MaxAmountToGrab)
        {
            if(TriggerBinder.transform.childCount > 0)
            {
                continue;
            }
            Grab_Trigger(rb,true);
            _grabbedWithFist = true;
        }
    }
}
```

Figure 0.37 – Leap Grabber Update Grab Objects

The ‘Grab_Trigger’ method is used to perform the ‘Child of Controller’ interaction mechanic, first storing the objects current parental hierarchy so it can be restored when the object is released. The object is then made a child object of the hands trigger binder and the Unity3D physics system is instructed to ignore any further collision or intersection detection between the object and the hands trigger binder. The physics system ignores further trigger event calls between the virtual hand and the object, as the hand will remain within proximity of the object, with the hand in a closed state, resulting in continuous physics event calls negatively impacting performance.

In the hand open state, the “ReleaseItems” function is used to release any objects held by the hand, with each objects parent hierarchy restored and the “WaitUntil” coroutine used to ensure the hand has moved out of proximity before the physics system recognises trigger events calls. The “WaitUntil” coroutine is used to prevent the user immediately picking up the object again

in the next update cycle due to minimal time difference between frames, resulting in the hand still positioned within proximity of the virtual object, while in the closed state (Figure 0.38). The “WaitUntil” yield instruction will prevent the coroutine from continuing to execute until the supplied delegate evaluates to true, which in this case means the distance between the virtual object and the hands sphere collider exceeds a set threshold.

```
private IEnumerator ReleaseObject_Trigger(Rigidbody rigidbody)
{
    if (rigidbody != null)
    {
        _grabbedItems_Trigger.Remove(rigidbody);
        rigidbody.transform.parent = _grabbedItems_Trigger_Parents[rigidbody];
        _grabbedItems_Trigger_Parents.Remove(rigidbody);

        var interactable = rigidbody.GetComponent<Interactable>();
        // Wait till the hand is out of the object
        if (interactable != null)
            yield return new WaitUntil(() => Vector3.Distance(rigidbody.position,
                                TriggerBinder.gameObject.transform.position) > 0.4f);

        IgnoreCollision(rigidbody, false);

        Debug.Log("ReleaseObject_Trigger:: releasing " + rigidbody.gameObject.name);

        PickupListener pl = rigidbody.gameObject.GetComponent<PickupListener>();
        if (pl != null)
        {
            pl.Released();
        }
        if (MultiLeap_DataAnalysisV2._instance != null)
        {
            MultiLeap_DataAnalysisV2._instance.HandReleaseObject(
                (this.HandType == HandType.LEFT));
        }
    }
}
```

Figure 0.38 - Code to release an object in MOT object interaction mechanic

Appendix 12 – Chemical Engineering Simulation Design

In this section the detailed design of a Chemical Engineering based VR simulation is presented including the design of the interactable objects featured in the simulation. Additionally, the design of the interaction techniques/mechanisms for the virtual objects will also be presented. Finally, the design of the simulations scientifically accurate background systems will be detailed such as the management of fluid mixtures ratios and realistic temperature control using principles such as specific heat capacity along with Newtons Laws of Cooling.

Fluid Storage

The simulation features a mini-continuous distillation unit (MCDU) which is comprised of a series of tanks, cooling chambers, flow meters, pump, etc, with fluid being transferred both in and out of all of them. As each of the components of the MCDU would share common functionality, an object-orientated programming approach was taken with the development of a 'FluidStorage' class, supporting the transference of fluids both to and from multiple sources, whilst ensuring the component does not exceed a maximum fluid capacity.

As the simulation was designed to replicate the real-world equipment as accurately as possible, the simulation would require scientific principles to be applied such as calculating the resulting temperature from two mixing fluids using the calculated specific heat capacities of the mixtures and equations such as Newton's Law of Cooling.

Objects such as the pumps and flow meters were considered to have a negligible capacity in the context of the simulation, due to their high flow rate and the incoming liquid flowing in and straight out via a tube. To simulate the negative pressure acting on the flow of liquids throughout the system, the fluid storage class was designed to support a “middle-man” behaviour with fluid directly transferred from the input to output and reverse. For example, the pump that connects the pre-heater and large mixing tank, transfers any overflow in the pre-heater to the large mixing tank simulating the opposing pressure to the pump.

Beakers

In the simulation there is a beaker containing a pre-mixed solution of water and alcohol that the user must pour into the large mixture tank so that it can be pumped into the unit. Figure 0.39 shows the beaker the user must pour using a simple ‘grab’ and ‘release’ approach using the ‘Child of Controller’ mechanic as detailed in Appendix 11. To pour the solution, the beaker must be within proximity of the mixture tank and angled appropriately to pour, with a water particle effect displayed for positive feedback. Once the beaker has been emptied the pouring effect stops and the user can place the beaker back onto a table.

Figure 0.39 - Screenshot of the Beaker in the Chemical Engineering Simulation

Mini-Continuous Distillation Unit Control Box

To control the various features of the mini-continuous distillation unit and monitor variables such as boiler and heat exchanger temperatures, the user interacts with the control box. To adjust the power setting of the boiler and thereby its maximum temperature, the user interacts with a virtual UI. Figure 0.40 shows the virtual user interface used to control the boiler power.

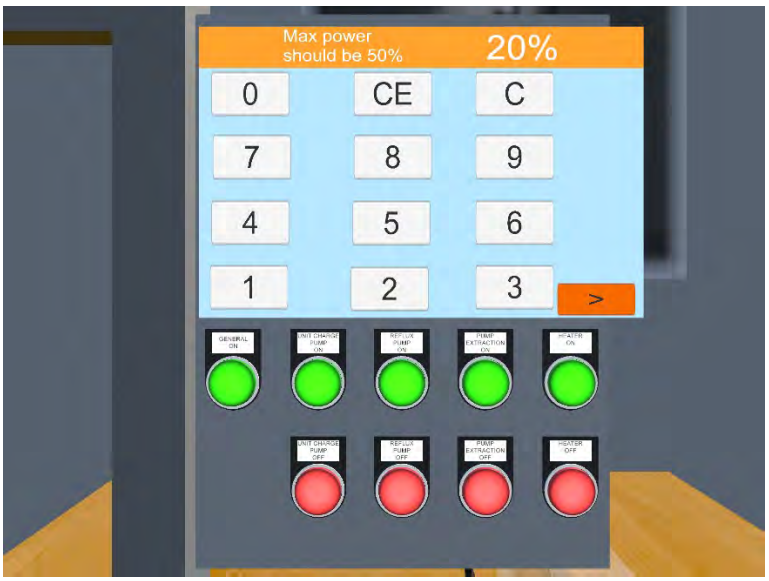


Figure 0.40 - Render of the Mini-Continuous Distillation Unit Control Box

The control box uses a same trigger-based approach therefore to interact with one of the virtual buttons, the user must tap the virtual button. If the user has entered incorrect settings an error message is displayed on the virtual UI informing them.

Levers

The simulation features a series of levers as shown in Figure 6.1 which are used to control the flow of liquids around the unit. The levers inherit from the FluidStorage class and use the “middle-man” approach detailed in Appendix 12. The levers can either control the flow or transfer the fluid to a selected output connection depending on the selected mode. In ‘Tap’ mode the lever acts as a toggle, transferring the fluid to the output (and any overflow back to the input). In ‘Selection’ mode, the lever is constantly transferring incoming fluids to the selected output connection (and any overflow back to the input). In ‘Selection’ mode, the selected output is determined using the levers rotation with each output connection selected at a set rotation +/- a 10-degree tolerance. The levers use the Unity3D hinge joint system to limit their rotation to that of the real-world equipment.

To interact with the levers the user must perform a fist gesture within proximity of the lever, with the lever then moving with the user’s virtual hand representation by continuously pointing towards it. To stop interacting with the virtual lever the user must open their hand.

Flow Control Valves

The simulation features a series of valves which are used to set the flow rate of liquids around the unit using the “middle-man” approach with amount

transferred based on the set flow rate and overflow sent back to the input. The flow rate is determined by converting the current angle from its movement range to the flow range. To aid users, the flow meters feature an additional ball float (orange) to visualize the rate the user should set the flow rate to. Figure 0.41 shows the four flow meters used within the simulation.

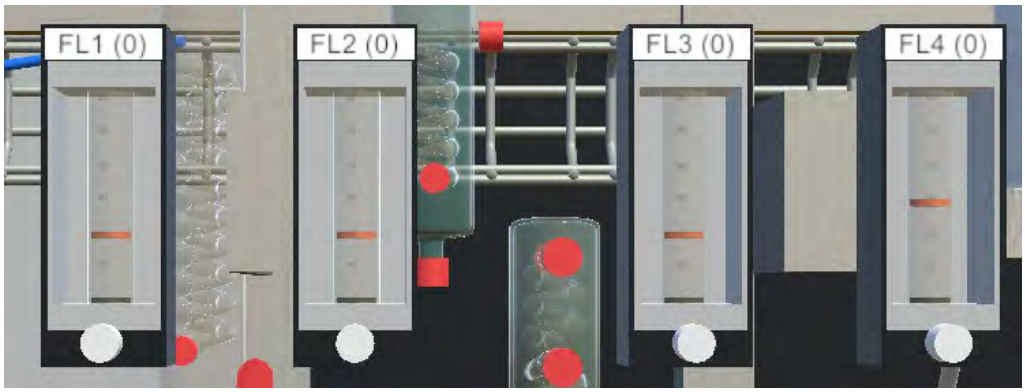


Figure 0.41 - Render of the Four Flow Meters within the Chemical Engineering Simulation

Appendix 13 – Standardized Gesture Evaluation Questionnaire

Gestures:

1) Circular Motions (left anti-clockwise and right clockwise, with palms facing away)

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									
The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

2) Hands over each other and hold for five seconds (Left over right, then right over left)

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									
The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

3) Facing hands moving away (hands start near headset with palms facing and user slowly extends arms)

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									

The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

4) Palms facing towards with fingers upwards, hands move to the sides out of view, then move back into view to the original position

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									
The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

5) Reaching gesture (move hand outwards and upwards, i.e. diagonally, as if reaching a shelf above)

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									
The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

6) Fist rotate around to different orientations
    a. Palm facing
    b. Thumb facing
    c. Fingers facing away
    d. Thumb facing upwards
    e. Palm facing

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									
The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

7) Moving beaker from left to right

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									
The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

Overall Accuracy

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands tracked perfectly									
I did not observe any glitchy behaviour									

The hands were stable and reflected my own hands perfectly									
I felt in full control of the hands									
I did not perceive any lag or delay in the tracking									
The system never lost track of my hands									

Naturalness:

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
The hands looked realistic while holding objects									
The hands behaved realistically									
Object interactions were realistic									
I did not have to limit my hand poses to perform actions									
I had full range tracking movement									
My movements were fully tracked									
My hands did not appear deformed									

Satisfaction:

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
I am satisfied with it									
It is fun to use									
It works the way I want it to work									
It is pleasant to use									
I think that I would like to use this system frequently									
I thought there was too much inconsistency in this system									

Ease of Use:

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
It is easy to use									
It is simple to use									
It is user friendly									
It is flexible									
Using it is effortless									
I don't notice any inconsistencies as I use it									
I found the system unnecessarily complex									
I found the system very cumbersome to use									
I feel very confident using the system									

Ease of Learning:

	Strongly Disagree	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Strongly Agree**
I learned to use it quickly									
It is easy to learn to use it									
I quickly became skilful with it									
I can easily remember how to use it									
I needed to learn a lot of things before I could get going with this system									
I would imagine that most people would learn to use this system very quickly									

Appendix 14 – Simulation Questionnaire Spearman Correlation

		Questionnaire_Control	Questionnaire_Accuracy	Questionnaire_Reliability	Questionnaire_Arm Extension	Questionnaire_Naturalism	Questionnaire_Difficulty	Questionnaire_Engagement	Questionnaire_Immersion	Questionnaire_Real life
Questionnaire_Control	Correlation Coefficient	1.000	.796**	.763**	.604**	.571**	.648**	.735**	.290	.622**
	Sig. (2-tailed)	.	.000	.000	.002	.004	.001	.000	.169	.001
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Accuracy	Correlation Coefficient	.796**	1.000	.721**	.555**	.675**	.683**	.635**	.157	.494*
	Sig. (2-tailed)	.000	.	.000	.005	.000	.000	.001	.464	.014
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Reliability	Correlation Coefficient	.763**	.721**	1.000	.637**	.554**	.642**	.758**	.036	.716**
	Sig. (2-tailed)	.000	.000	.	.001	.005	.001	.000	.867	.000
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Arm Extension	Correlation Coefficient	.604**	.555**	.637**	1.000	.523**	.638**	.591**	.237	.698**
	Sig. (2-tailed)	.002	.005	.001	.	.009	.001	.002	.264	.000
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Naturalism	Correlation Coefficient	.571**	.675**	.554**	.523**	1.000	.546**	.534**	.224	.545**
	Sig. (2-tailed)	.004	.000	.005	.009	.	.006	.007	.294	.006
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Difficulty	Correlation Coefficient	.648**	.683**	.642**	.638**	.546**	1.000	.532**	.073	.483*
	Sig. (2-tailed)	.001	.000	.001	.001	.006	.	.008	.736	.017
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Engagement	Correlation Coefficient	.735**	.635**	.758**	.591**	.534**	.532**	1.000	.117	.639**
	Sig. (2-tailed)	.000	.001	.000	.002	.007	.008	.	.587	.001
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Immersion	Correlation Coefficient	.290	.157	.036	.237	.224	.073	.117	1.000	.402
	Sig. (2-tailed)	.169	.464	.867	.264	.294	.736	.587	.	.051
	N	24	24	24	24	24	24	24	24	24
Questionnaire_Real life	Correlation Coefficient	.622**	.494*	.716**	.698**	.545**	.483*	.639**	.402	1.000
	Sig. (2-tailed)	.001	.014	.000	.000	.006	.017	.001	.051	.
	N	24	24	24	24	24	24	24	24	24

Appendix 15 – Gesture Questionnaire Spearman Correlation Results

			Circle Gesture	Occlusion Gesture	Away Gesture	Sides Gesture	Reaching	Fist	Beaker	Overall	Naturalness	Satisfaction	Ease	Learning
Spearman's rho	Circle Gesture	Correlation Coefficient	1.000	-.281	-.057	.104	-.083	.231	.250	.251	.207	.260	.237	.138
		Sig. (2-tailed)	.	.184	.793	.630	.699	.278	.240	.236	.332	.220	.265	.521
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Occlusion Gesture	Correlation Coefficient	-.281	1.000	.562**	.331	.587**	.408*	.327	.296	.330	.161	.162	.160
		Sig. (2-tailed)	.184	.	.004	.115	.003	.048	.119	.160	.115	.452	.449	.456
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Away Gesture	Correlation Coefficient	-.057	.562**	1.000	.392	.499*	.506*	.540**	.394	.326	.351	.394	.436*
		Sig. (2-tailed)	.793	.004	.	.058	.013	.012	.006	.057	.120	.093	.057	.033
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Sides Gesture	Correlation Coefficient	.104	.331	.392	1.000	.679**	.766**	.520**	.519**	.499*	.513*	.579**	.497*
		Sig. (2-tailed)	.630	.115	.058	.	.000	.000	.009	.009	.013	.010	.003	.014
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Reaching	Correlation Coefficient	-.083	.587**	.499*	.679**	1.000	.646**	.548**	.542**	.352	.415*	.445*	.456*
		Sig. (2-tailed)	.699	.003	.013	.000	.	.001	.006	.006	.092	.044	.029	.025
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Fist	Correlation Coefficient	.231	.408*	.506*	.766**	.646**	1.000	.782**	.696**	.616**	.687**	.790**	.729**
		Sig. (2-tailed)	.278	.048	.012	.000	.001	.	.000	.000	.001	.000	.000	.000
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Beaker	Correlation Coefficient	.250	.327	.540**	.520**	.548**	.782**	1.000	.751**	.736**	.815**	.801**	.786**
		Sig. (2-tailed)	.240	.119	.006	.009	.006	.000	.	.000	.000	.000	.000	.000
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Overall	Correlation Coefficient	.251	.296	.394	.519**	.542**	.696**	.751**	1.000	.852**	.806**	.774**	.652**
		Sig. (2-tailed)	.236	.160	.057	.009	.006	.000	.000	.	.000	.000	.000	.001
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Naturalness	Correlation Coefficient	.207	.330	.326	.499*	.352	.616**	.736**	.852**	1.000	.876**	.809**	.683**
		Sig. (2-tailed)	.332	.115	.120	.013	.092	.001	.000	.000	.	.000	.000	.000
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Satisfaction	Correlation Coefficient	.260	.161	.351	.513*	.415*	.687**	.815**	.806**	.876**	1.000	.932**	.889**
		Sig. (2-tailed)	.220	.452	.093	.010	.044	.000	.000	.000	.000	.	.000	.000
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Ease	Correlation Coefficient	.237	.162	.394	.579**	.445*	.790**	.801**	.774**	.809**	.932**	1.000	.946**
		Sig. (2-tailed)	.265	.449	.057	.003	.029	.000	.000	.000	.000	.000	.	.000
		N	24	24	24	24	24	24	24	24	24	24	24	24
	Learning	Correlation Coefficient	.138	.160	.436*	.497*	.456*	.729**	.786**	.652**	.683**	.889**	.946**	1.000
		Sig. (2-tailed)	.521	.456	.033	.014	.025	.000	.000	.001	.000	.000	.000	.
		N	24	24	24	24	24	24	24	24	24	24	24	24