VR HAND TRACKING FOR ROBOTICS APPLICATIONS

An Undergraduate Research Scholars Thesis

by

ZACHARY HELTON

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Prof. Stavros Kalafatis

May 2022

Major: Electrical Engineering

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Zachary Helton, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Research Faculty Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

VR Hand Tracking for Robotics Applications

Zachary Helton
Department of Electrical and Computer Engineering
Texas A&M University

Research Faculty Advisor: Stavros Kalafatis
Department of Electrical and Computer Engineering
Texas A&M University

In industrial and factory settings, there is no truly safe way for humans to interact with robots. Large industrial machines make too much noise for vocal commands to be used as a safe method of interaction. Electrical motors, machine tools, human operators all create noise that makes vocal commands less effective in a factory than in a quiet laboratory [1]. These issues put human floor workers in unsafe situations frequently. By developing a system that can effectively recognizing track hand gestures from a human and train a robot to respond accordingly, factory safety and human-robot interaction can be revolutionized.

The goal of the project is to develop and implement a program that will train a virtual robot to respond to hand gestures. Understanding concepts such as virtual and augmented reality, hand tracking, robotics, machine learning, and domain randomization will be central to the success of the project. Using Unity game design software, a virtual reality will be created in which a virtual robot will be trained to respond to human hand gestures. Thousands of environments will be simulated in VR so that a real robot will be prepared to respond to any possible command. The dataset developed in this research increased the average confidence

index of a machine learning gesture recognizer by 12.45%. The end application of this research is for factory automation and other fields where the safety of human-robot interaction is a priority.

ACKNOWLEDGEMENTS

Contributors

I would like to thank my faculty advisor, Prof. Stavros Kalafatis and Pranav Dhulipala, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

The materials used for VR Hand Tracking for Robotics Applications were provided by Pranav Dhulipala and the Department of Electrical and Computer Engineering at Texas A&M University. The analyses depicted in VR Hand Tracking for Robotics Applications was conducted in part by Pranav Dhulipala and these data are currently unpublished.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Undergraduate research was supported by the Department of Electrical and Computer Engineering at Texas A&M University.

No other funding was received for this project.

1. INTRODUCTION

The goal of the project is to develop and implement a program that will train a virtual robot to respond to hand gestures. Understanding concepts such as virtual and augmented reality, hand tracking, robotics, machine learning, and domain randomization will be central to the success of the project. Using Unity game design software, a virtual reality will be created in which a virtual robot will be trained to respond to human hand gestures. Thousands of environments will be simulated in VR so that a real robot will be prepared to respond to any possible command. The end application of this research is for factory automation and other fields where the safety of human-robot interaction is a priority. Figure 1.1 illustrates an overview of the design components in this project.

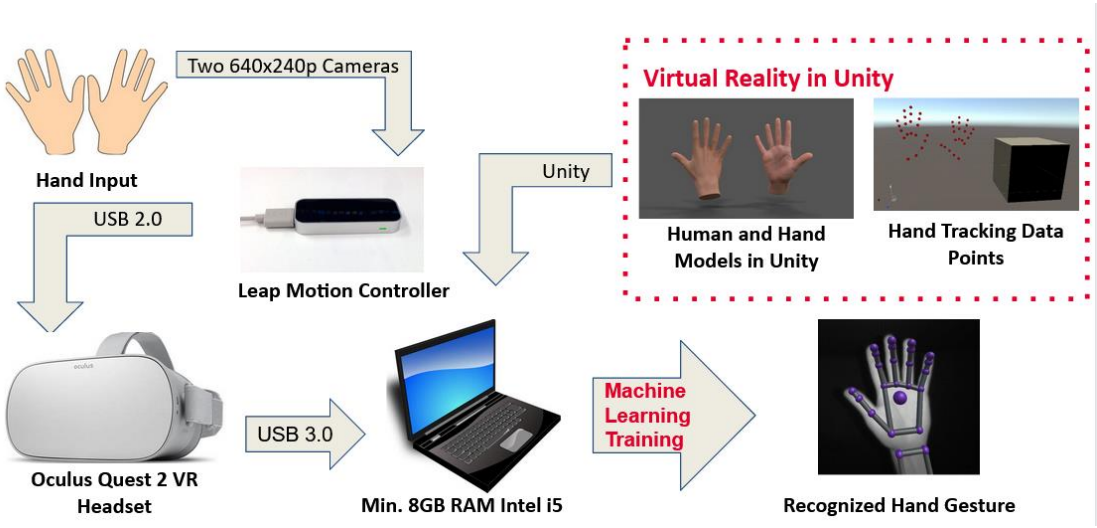

Figure 1.1: Subsystem Overview. The three main subsystems that were validated in this project were the Leap Motion Controller, the Virtual Reality in Unity, and the Machine Learning Gesture Recognition Model.

1.1 Background

The current system in place requires that responses be manually coded in or tested in the physical world. For instance, a human must show a robot a hand gesture and then hard code into the robot the response that they wish to receive. Previous research has been centered around hand tracking and training physical robots to respond to hand gestures. One publication describes the development of a system that allows a user to control a robotic hand using gesture recognition and hand tracking [2]. In this paper by Sharma, Sharma, and Yadav, a robotic hand is trained to mirror human hand gestures. Only more recently has virtual reality become a conversation. VR hand tracking and machine learning innovations present an opportunity for thousands of possible environments and scenarios to be simulated and uploaded to a physical robot in seconds.

1.1.1 Hardware Overview

A product manufactured by UltraLeap, the Leap Motion Controller [3], is the primary hardware component of this project. Two high accuracy infrared cameras and a simplistic USB 2.0 connection create an optical hand tracking module that provides high accuracy hand tracking in a small and compact sensor. It provides a full, virtual skeletal model of a hand that will be extremely useful in the recognition of different hand gestures in different environments.

1.1.2 Software Overview

Unity is a game and virtual reality design program that provides an abundance of tools that allow for maximum flexibility and simplicity in design [4]. While being user friendly, Unity also allows developers to create robust gaming environments for any application. In the case of this project, the virtual reality tools will be heavily utilized. Unity is also the program that has been used for the other sub-systems of the greater research project, so designing in Unity will provide the opportunity for a seamless assembly of sub-systems.

A deep learning neural network is the backbone of the software design in this project. The machine learning will train a virtual robot to recognize gestures by taking cases of all possible hand gestures into account. Once completed, a user should be able to input a hand gesture, and the machine learning will be able to classify that gesture.

1.1.3 Referenced Documents and Standards

For this project, it is essential to adhere to an abundance of standards. These standards can be seen in Table 1.1.

Table 1.1: Referenced Documents and Standards

Document Number	Revision/Release Date	Document Title
UH-003206-TC-6	Issue 6	Leap Motion Controller Datasheet
5.6-001N	Version 5.6 – 07/12/2017	Unity User Manual
UH-003207-TC-4	Issue 4 - 5/12/2019	VR Developer Mount Datasheet
DA-16298	Issue 2 – 10/08/2019	HTC Vive Pro Datasheet
USB-IF-20-106	Version 1.07 - 02/01/2019	USB 2.0 Electrical Compliance Test Specification
STD 01-12-002	Pub 8-1.3 09/21/1987	OSHA Guidelines for Robotics Safety
NIST-SP-800-121r2	Revision 2 – 05/01/2017	NIST Guide to Bluetooth Security
IJERT- 2278-0181	Vol.6 Issue 4 04/01/2017	Robotic Hand Control Using Gesture Recognition

1.2 Operating Concept

1.2.1 Scope

VR Hand Tracking for Robotics Applications is designed within the scope of a larger research project from Pranav Dhulipala. Pranav's research centers around human robot interaction. The VR hand tracking program that is being designed will be implemented into one

of Pranav's projects. The backbone of the hardware design of this project is the Leap Motion Controller (LMC). The LMC only has a range of 80cm, so the human must be near the sensor for the reading to be accurate. The virtual reality is being designed in Unity, so testing must be able to be completed with a simple USB cable connecting the sensor and the computer. The latter parts of the project will involve mounting the LMC to a VR headset. The HTC Vive Pro is the headset that will be used. This headset uses a USB 3.0 connection and is compatible with the LMC. [5]

1.2.2 Operational Description and Constraints

A virtual reality will be created so a virtual robot can be trained to respond to a variety of human commands. Unity is the software that will be used to design the virtual reality. Within this reality, thousands of different environments will be simulated to train the robot to correctly respond. In the case that the robot does not recognize the hand gesture, no response should be given. While this scenario is unlikely, safety still is a priority for this project. The user should feel confident that they are in full control of the robot with every hand gesture they make. The Leap Motion Controller will be used to track and upload real world hand commands into the virtual environment. Once the virtual training is completed, it will be uploaded into a physical robot that will be able to recognize and respond to human hand gestures.

1.2.3 System Description

The primary hardware system centers around the Leap Motion Controller. Technical specs of this controller include a 5V DC USB power supply, USB 2.0 data connection, two 640x240-pixel infrared cameras, and an ambient operating temperature of 32° to 113° F. (0° to 45° C). A more detailed description of the technical specs can be found on the data sheet for the

part [3]. The virtual reality will be developed within Unity. This reality will require a device with at least 16GB of RAM and 64-bit Windows 10 to run effectively.

1.2.4 Modes of Operation

1.2.4.1 Virtual Hand Tracking

Virtual hand tracking mode is the first operational mode of the system. This will be driven by taking a real-world input from a sensor. In this mode, the LMC should be able to track all motions and hand movements a user makes with a very high accuracy.

1.2.4.2 Gesture Recognition

The user will upload hand commands via the Leap Motion Controller, and then input how they would like the robot to respond to each command.

1.2.4.3 Training

Utilizing domain randomization, thousands of different environments will be simulated to train the robot. In this mode, the according robot responses can also be uploaded to a physical robot.

1.2.4.4 Testing

Testing mode will allow a user to approach a physical robot or VR headset, give a hand gesture, and have the robot respond. This final stage is a part of the larger scope of the project and will be implemented later in the project phase.

1.2.5 Users

VR Hand Tracking for Robotics Applications will allow for a wide range of users and use cases. While the research will be designed with a variety of end applications in mind, the envisioned users are those who work in factory automation. It will allow for factory floors to be safer for people to interact with. For example, a large industrial Amazon warehouse can be extremely loud. With large industrial robots moving around, safety is a huge issue. Vocal commands often do not work in this setting because of the level of noise. This is where hand tracking would benefit human factory

floor workers. A human would be able to give a hand gesture to a robot to have it to stop, reverse, go forward, or move in a different direction. This would in turn revolutionize safety for any users.

1.2.6 Support

Support will come in the form of a document that describes to the user how to accurately set up the system. It should instruct the user on how to upload basic hand gestures and their according responses. Additional support for users would come from Unity manuals and Leap Motion Controller data sheet. These are the two primary aspects of the system as a whole and many issues that could be encountered would be best troubleshooted through these two avenues.

1.3 Scenario(s)

1.3.1 Factory Automation

One use case for this project is in industrial factories and warehouses. One of the largest problems in this industry are that the robots used make a lot of noise. This can lead to several safety concerns. The hand tracking software and gesture recognition could be used to train robots to respond to hand gestures in a factory setting. This application is the primary focus for this project.

1.3.2 Hand Mirroring

Another possible application for this project would be utilizing the hand tracking and gesture recognition to have a robotic hand/arm mirror the movements of a human hand [6]. This could revolutionize human robot interaction and be applied in a wide variety of industries.

1.3.3 Autonomous Vehicles

The hand tracking system in VR could be used to train autonomous vehicles to respond to hand gestures and commands [7]. This would still allow for a “hands free” driving experience but would give the user more control over the car.

1.3.4 Sign Language Translation

In this scenario, a similar dataset and machine learning algorithm could be applied to the field of sign language recognition and translation. Previous projects have used a Leap Motion Controller to translate sign language gestures [8].

1.4 Analysis

1.4.1 Summary of Proposed Improvements

VR Hand Tracking for Robotics Applications will revolutionize human robot interaction, providing a safer, more efficient, and unique way of communication with robots. The hand tracking and gesture recognition would eliminate any concerns about loud noise in a factory environment. Additionally, utilizing domain randomization would greatly increase the speed of training times [9]. Thousands of environments could be simulated in seconds instead of what previously would have to be manually uploaded.

1.4.2 Disadvantages and Limitations

The system will be limited to the amount of hand gestures that a user can input and create. One disadvantage is that if a human were to do a slightly different hand gesture, then what was uploaded, the robot may not correctly respond. For example, if a human holds up three fingers up instead of four accidentally, the robot may respond differently. Another concern is the range of the sensor used. The Leap Motion Controller only has a range of 80cm, so the human must be near the sensor for the reading to be accurate

1.4.3 Alternatives

Alternatives for this solution could center around remote controls or voice commands, but both have their downfalls. Voice commands would not work in virtual environments and would be very difficult to train via virtual reality [1]. There is significantly more potential for human error

when a robot is remote controlled. Tracking high accuracy hand gestures would be more accurate. [10].

1.4.4 Impact

This project will revolutionize safety in a factory setting and in the field of human robot interaction. The hand gesture recognition and tracking will provide unique avenues and applications that will keep workers safe in a noisy factory setting [11]. Ethical considerations will center around what to do if an improper hand gesture is given, does the robot continue to execute the previous command or freeze? This could be an important potential safety and ethical issue, as it is extremely important to train the robot for human error cases. This project will allow for a human to safely give a gesture to a robot, and have it respond accordingly. The project will help avoid unnecessary injuries caused by human robot interaction.

Once the research is implemented, factories automation robots will be much safer to interact with [11]. Factory workers will feel safer, companies will have peace of mind knowing their employees are safe. It could be expected that improved safety would motivate workers to return to factory jobs, as one of the main drawbacks in the industry is safety concerns. The workforce returning to the factory would have a very positive economic impact on society.

2. METHODS

In the field of human-robot interaction, safety must be one of the first things considered. In a factory setting, loud noise is a prominent safety issue [12]. Robots cannot respond to verbal commands because of the loud machinery [1]. To combat this issue, a VR hand tracking and gesture recognition system shall be developed. The system will train a virtual robot to recognize and respond to hand gesture commands. The training should be able to be uploaded to a real-world robot. A Leap Motion Controller (LMC) shall be the central hardware part utilized in this design. The LMC shall be linked with a VR headset to provide detail hand tracking data. The system will all be developed in a virtual reality designed specifically for hand tracking and gesture recognition.

Concerning the scope of the project, the system will be able to recognize and classify a set of hand gestures into eleven different categories. There will be ten classes of gestures and one class of an unrecognized gesture. The gesture recognition system will be designed so that the classes can be easily changed. For example, an open hand gesture will be associated with the class “stop” and a thumbs up gesture will be associated with the class “go.” Fifteen gestures are being used to verify that the system is functional but there will be opportunities to test the system with more gestures. The gestures being used are similar to the gestures used in a dataset created by Rishahb Arya [13]. Using similar gestures to this research will allow for improvement metrics to be obtained when comparing old and new datasets.

2.1 System Requirements

2.1.1 System Definitions

VR Hand Tracking for Robotics Applications will be a hand gesture recognition system designed for improving safety in a factory setting. The LMC will connect to a virtual reality headset and upload hands to a virtual reality.

The input will come from a human through the LMC. This could be any variety of hand gesture like a “thumbs up” or holding up four fingers. Hand tracking will be done by the built-in software in the LMC, where a hand will appear and several different “joints” that all move together as a hand does. The Leap Motion Controller assigns tracking points to each of the user’s hand joints [14]. Within the virtual reality, the hand motions or signs a user makes should be able to be recognized as distinct gestures. Based on what gesture is given, a command will be sent to a virtual robot as the output.

2.1.2 Characteristics

2.1.2.1 Functional/Performance Requirements

The system shall utilize the LMC to take input readings and track hand movements of a user. The LMC will connect to a computer using a USB 3.0 connection [3]. Leap Motion Orion software will be installed on the device so that hand tracking data can be interpreted.

Unity will be the software that the virtual reality is designed in. A Makehuman model will be in Unity. The finger bones of the model will be rigged and mapped to the Leap Motion modules. A human hand input should be able to be mirrored by the Unity hand model.

A combination of the LMC and VR gesture recognition tools should be able to distinguish a pre-loaded list of hand gestures from a diverse data set. The data set should exhaust all possible environments so that a real-world robot will always be prepared to respond

accordingly. The data set will number in the thousands in length. This data set will contain Makehuman models performing a variety of different hand gestures. Physical characteristics like skin color, hand size, clothes, hair color, etc. will be randomized to allow for learning in a multitude of environments.

2.1.2.2 Physical Characteristics

The system should be less than 1 kg in mass, or 2.2 pounds. The VR headset accounts for well over half of this number (800g). The added weight of the system should be light enough for a person to be comfortable wearing the headset. The LMC is only 80g and compact (30mm x 80mm x 11.30 mm), so this will not present much of an issue.

The LMC shall be mounted onto the Vive Pro headset. Previous research has identifies that using the LMC with the HTC Vive is the most effective method for hand tracking [15]. Ideally, the sensor should be mounted in an area that will provide a highly accurate measurement from the LMC [3].

2.1.2.3 Electrical Characteristics

The main input is from the Leap Motion Controller. The LMC shall be able to recognize inputs that are hand signs or movements. The system should be able to accurately track hand gestures at a maximum range of 80 cm [3]. The LMC will be attached to the VR headset and will load the inputs into virtual reality.

The input voltage level for the VR Hand Tracking System shall be +5 VDC with a current +0.5 A. The Leap Motion Datasheet states these as the ideal operating values. No information is stated on the performance of the device at different voltage and current levels. For this reason, the LMC will only be used at these operating values. The LMC shall document and transfer external commands given to the system. External commands will come in the form of

hand gestures given to a virtual robot. Inputs will be read by two 640x240-pixel infrared cameras withing the LMC. These cameras are spaced 40 cm apart and operate in a spectral range of 850 +/- 25 nm.

VR Hand Tracking for Robotics Applications shall output commands in a virtual reality to a virtual robot. These commands will be coded into Unity. VR Hand Tracking for Robotics Applications shall output a video of what is happening inside the virtual reality. This video should be able to be projected on a headset or on a monitor for users to see. The VR Hand Tracking for Robotics Applications will connect to a VR headset via USB 2.0 connection. A 2/3 hybrid cable is package with and recommended for the LMC, but any certified cable will suffice. The project will follow all USB 2.0 connection standards [16].

2.1.2.4 Environmental Requirements

VR Hand Tracking for Robotics Applications will be designed for use in factory or lab setting. The project will not be required to withstand any extreme conditions. The ideal operating altitude for the LMC is between 0 to 10,000 feet or 0 to 3048 meters [3]. Per the LMC datasheet, the ambient operating temperature range is from 32°F to 113°F or 0°C to 45°C. Ideal safe storage temperature for the device is 14°F to 122°F or -10°C to 50°C. There are no waterproof capabilities designed into the LMC. It should not be exposed to any foreign substance or liquid as this could damage the sensor and effect the measurements taken. Per the LMC datasheet, the relative humidity of the system ranges from 5% to 85% (non-condensing).

2.1.2.5 Failure Propagation

VR Hand Tracking for Robotics Applications will have built in failure systems for gesture recognition. In the case that an incorrect or not recognized gesture is given, no command

will be sent to the robot, and it will not respond. When this occurs, a message stating "unrecognized gesture" will be displayed in the virtual reality headset.

The project will have a built-in test program for calibrating the LMC and the virtual reality gesture recognition. This test program will be run and will allow a user to upload simple hand gestures to ensure the both the hardware and software components are working properly. The program will act as the diagnostics for the system and will allow for a user to effectively troubleshoot issues that may occur. The BIT program shall have the ability to distinguish if a user's hands are being accurately tracked. The user will be instructed to display a set of simple hand gestures and movements to test if the system is accurately tracking hand motions. This shall be the first method of troubleshooting. This BIT program shall determine if the gesture recognition software in Unity is properly working. This will instruct a user to input simple and specific hand gestures. The program will test if the device can accurately recognize hand gestures.

The BIT shall remember each gesture recognized or not recognized by the system. From this, the user will be able to see each command given and whether the system recognized it or not. The system shall be isolate errors and commands from the BIT. If an input is given that is not a hand gesture or is incorrect, the system will not respond. This will help to prioritize safety and will also account for human error.

2.1.3 Support Requirements

Per the Leap Motion Controller datasheet, a computer running Windows 7+ or Mac OS X 10.7+ is required. The device must have an Intel Core i3/i5/i7 or AMD Phenom II processor. 2 GB of RAM and a USB 2.0 port are required. Unity requires a system running Windows 10 and

a x86 or x64 CPU. The HTC Vive Pro system requirements document recommends a device have an Intel Core i5, 8GB of RAM, a 1.4 DisplayPort, and at least 1 USB 3.0 port. Combining these together specifications together, a Windows 10 machine with 8GB of RAM, an Intel Core i5 processor, at least 1 USB 3.0, and at least 1 USB 2.0 compatible port is required for VR Hand Tracking for Robotics Applications.

A support document and user manual will be provided for the end users of this project. With this support document will also be the instructions for the BIT diagnostic testing mentioned in section 3.2.5 of this report. These instructions will assist the end user in the setup of the LMC, HTC Vive Pro, and Unity VR. The goal of this support is to provide a summary of all common problems a user might encounter and their solutions.

2.2 System Interface

2.2.1 Physical Interface

The physical design components of this system are the VR Headset and the Leap Motion Controller. The LMC will be connected to the headset via USB 2.0 connection. Presently, the assumption is that the LMC will be mounted to the front of the HTC Vive Pro using a 3D printed Leap Motion Developer mount [17].

2.2.1.1 Weight

A defined goal for this project is to keep the system weight around 1 kg. Since physical the components of the system will eventually combined be on a wearable headset, it is ideal for the system to be lightweight and easily maneuverable. The LMC has a mass of 80 grams [3]. As stated in the HTC Vive Pro Datasheet [5], the VR headset being used for this project has a mass of 800 grams. Per the VR Developer Mount Datasheet, one of the potential mounts for the leap motion controller has a mass of 20 grams.

2.2.1.2 Dimensions

According to the LMC Datasheet [3], the Leap Motion controller measures 30mm x 80mm x 11.30 mm. No physical dimensions of the Vive Pro are given on the datasheet. In order to obtain these, a physical measurement must be done on the headset itself. However, the datasheet does recommend that the user of the headset be in an open space 3.5m x 3.5 m in dimensions. This much space is given to avoid physical contact with other objects and safety hazards. The VR Developer Mount Datasheet states that the dimensions of the mount are 13mm x 80mm x 30mm.

2.2.2 Mounting Locations

Part of the research of this project will be experimenting with different mounting locations and mounting mechanisms. At the present time, it appears the most likely solutions will be to mount the LMC to the VR headset. The mount used will either be a purchased part or a custom 3D printed part. One thing to account for is the camera lenses at the front of the HTC Vive Pro. These lenses can be covered, but adhesive used in mounting could possibly damage them.

The VR Developer Mount as seen in Figure 2.1 is an orderable part designed specifically for the LMC. It is intended to be compatible with any VR headset and attaches via an adhesive strip. One issue with this method of mounting is the fact that an adhesive strip would have to be attached over the camera lenses of the HTC Vive Pro. This is not ideal and is why other mounting options are being considered.

Figure 2.1: VR Developer Mount from Leap Motion

A custom 3D printed option allows for maximum flexibility for mounting. A solution can be specifically designed for the HTC Vive Pro Headset. An example from a public LMC forum [18] is shown in Figure 2.2. The design in the figure would be attached to the headset via tie straps to allow for easy removal and security. An example of this can be seen in Figure 2.3.

Figure 2.2: 3D printed mount for HTC Vive Pro headset

Figure 2.3: Example from Leap forums [19] of sensor mounted using 3D printed mount

2.2.3 Thermal Interface

2.2.3.1 VR Headset Cooling

Among other researchers, there have been reviews that the HTC Vive Pro gets warm to the touch after 2 plus hours of continuous usage [20]. The area around the face gets very sweaty and warm. Possible solutions to this problem include installing a fan to blow air onto the headset or installing heat gaskets that help to absorb heat from the headset and from the skin. They provide separation from the headset with a layer of cool fabric. HTC Vive public forum discussions leave the impression that this issue is very circumstantial, some users have significant issues with cooling, others have none. If necessary, a heat gasket like the one shown in Figure 2.4 will be installed into the headset to improve cooling.

Figure 2.4: Facial heat gasket designed for HTC Vive Pro. This layer of fabric attached to the back of the headset and creates a cooler gaming experience by separating the users face from the headset.

2.2.3.2 Leap Motion Controller Cooling

The leap motion controller has a relatively low ambient operating temperature range of 0°C to 45° C. Because of this, cooling for the LMC will not likely be a concern and a thermal interface will likely not be needed.

2.2.4 Electrical Interface

The LMC shall be powered from the HTC Vive Pro Headset. The HTC Vive Pro will connect to a computer via USB 3.0 or 2.0 and will be powered from that connection.

2.2.4.1 Primary Input Power

The primary input power of the system will come from the computer that the HTC Vive is connected to via USB. The input power range for the Vive Pro is 2.5W to 3.5W. The LMC will be powered plugged into the Vive headset and requires a minimum input power of 2.5 V. The Vive Pro can also be powered wirelessly with up to 6 hours of battery life, however for this project the headset will mostly remain plugged in when in use. When combined, an approximate power of 6W is expected.

2.2.4.2 Signal and Video Interfaces

From the Vive Pro Datasheet, it is known that the HTC Vive Pro has a refresh rate of 90 Hz. According to the Leap Motion Controller Datasheet, the LMC typically operates at 120 Hz. The HTC Vive Pro displays an image with a 2880 x 1600p resolution. This will follow all standards in accordance with the IEEE VR specifications outlined in IEEE P2048. The LMC has two 640 x 240 cameras. Because the resolution of the LMC is significantly lower than that of the HTC Vive Pro, there may be some image quality issues within the virtual reality. To resolve this, the LMC must be mounted in a position where the highest accuracy and highest resolution reading can be acquired.

2.2.5 Communications / Device Interface Protocols

2.2.5.1 Bluetooth Communications (BLE)

The HTC Vive Pro does have Bluetooth capabilities. These capabilities will most likely not be used in this project. When BLE Communication is used, image quality is often sacrificed.

2.2.5.2 Host Device

The host device on the HTC Vive Pro is a USB 3.0 connection. Similarly, the LMC uses a USB 2.0 connection. The LMC will be plugged into the HTC Vive Pro, and the headset will be plugged into a computer. All USB 3.0 and 2.0 standards referenced from www.usb.org will be kept.

2.2.5.3 Video Interface

The HTC Vive Pro connects via a DisplayPort 1.2 or newer. This connection runs from the computer to a link box. The headset is then connected to the link box by a headset cable connector. The referenced connections shall meet all standards from DP-1.2, VESA DisplayPort Standard Version 1.2.

2.3 Subsystems Overview

The two design components of the research are the Leap Motion Controller setup and a unity virtual reality. A brief overview of both subsystems will be highlighted below.

2.3.1 Leap Motion Controller

2.3.1.1 Subsystem Introduction

The Leap Motion Controller (LMC) is a high accuracy hand tracking sensor that maps the hand movements and gestures of the user. The LMC is accompanied the by the Leap Motion Orion Software. This software allows for the user to see hand tracking data in a window called the Leap Motion Visualizer (LMV). Software to aid integration into unity was also installed, called the Leap Unity Modules. Seen in Figure 2.5, the LMC was tested to validate the accuracy, range, and light sensitivity of the sensor. This test aided in verifying that the LMC was the best hand tracking sensor for the project.

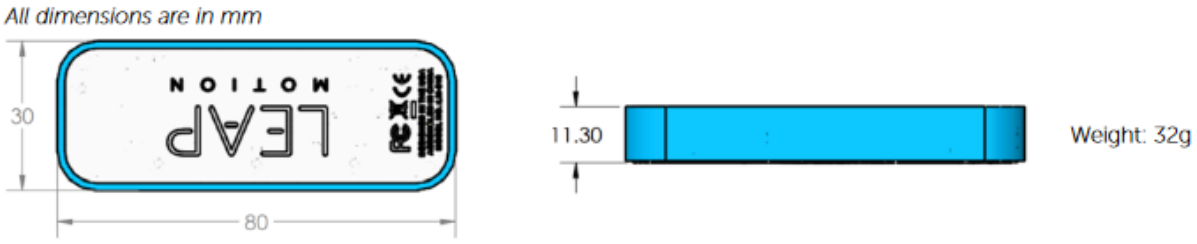

Figure 2.5: Dimensions of Leap Motion Controller. This figure comes directly from the referenced Leap Motion Controller Datasheet and shows a sketch of the sensor. The bottom view (left) and side view (right) can be seen.

2.3.1.2 Subsystem Details

According to the LMC Datasheet, the sensor utilizes two 640x240-pixel infrared cameras to track hand movements. The datasheet also states the cameras have a maximum range of 80cm,

but the ideal range is up to 60cm. This range was tested to verify the stated specifications. The dimensions of the LMC are 30mm x 80mm x 11.3mm and the sensor weighs 32g (LMC Datasheet).

The Leap Motion Orion Version 4.1.0 software allows for hand tracking readings to be viewed through the LMV. This environment is the is where all testing for the LMC was completed. There are two different modes of operation for the LMV: desktop mode and head mounted mode. For testing, desktop mode was used. The LMV was chosen for testing because it is simple, easy to use, and allowed for the most accurate and efficient testing. An image of the hand readings seen through the Leap Motion Visualizer can be seen in Figure 2.6. Additionally, the Leap Unity Modules were installed and allowed for the user to implement LMC tracking data into a Unity Virtual Reality.

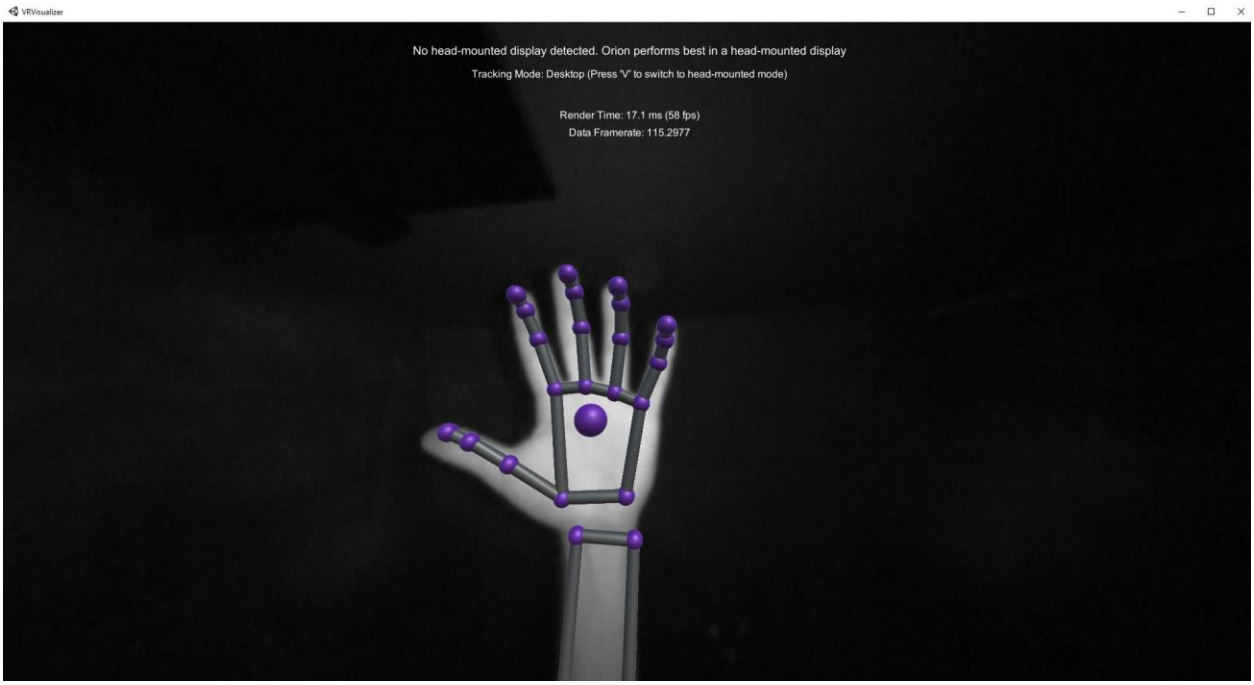

Figure 2.6: Leap Motion Visualizer. The environment used to test the LMC and track hand movements can be seen. Desktop mode is being used.

2.3.2 Unity Game Design Software

2.3.2.1 Subsystem Introduction

Unity is a game development software that is used to develop the virtual reality the hand tracking will be done in. The Unity subsystem contains the Leap Motion Unity modules mentioned in the previous section as well as a Makehuman model with a full bone structure. Makehuman models have been used in Unity for previous applications. Motion capturing of these models has been applied to other industries. Sports science professionals have found Makehuman skeletons to be extremely comparable to real humans when animated [21]. The final product of this subsystem is the implementation the Leap Motion Tracking mapped to the Makehuman model in Unity. Testing was completed with the LMC modules and Makehuman models to verify the case that allows for them to function the best. Once the tracking data was mapped to the hand and finger bones of the human model, further testing was done to determine the best location of the virtual LMC.

2.3.2.2 Subsystem Details

The LMC Unity modules contain three plugins designed to be used in Unity. These plugins were downloaded from the Leap Motion website and include the Core, Hands, and Interaction Engine. The Core module provides the user with the necessities for implementing the LMC in Unity such as a virtual representation of the LMC. This virtual LMC is confined to the same restrictions in VR as it is in the real world. This means that any tracking done in VR must be done at a range between 20cm and 60cm. [22] The Hands module provides the user with a diverse amount of pre-rigged human hand models and examples to map Leap tracking data to. An example of one of these hand models can be seen in Figure 2.7. The Interaction Engine was

not used in this project. These modules were used to help validate that Leap tracking data can correctly be mapped in Unity.

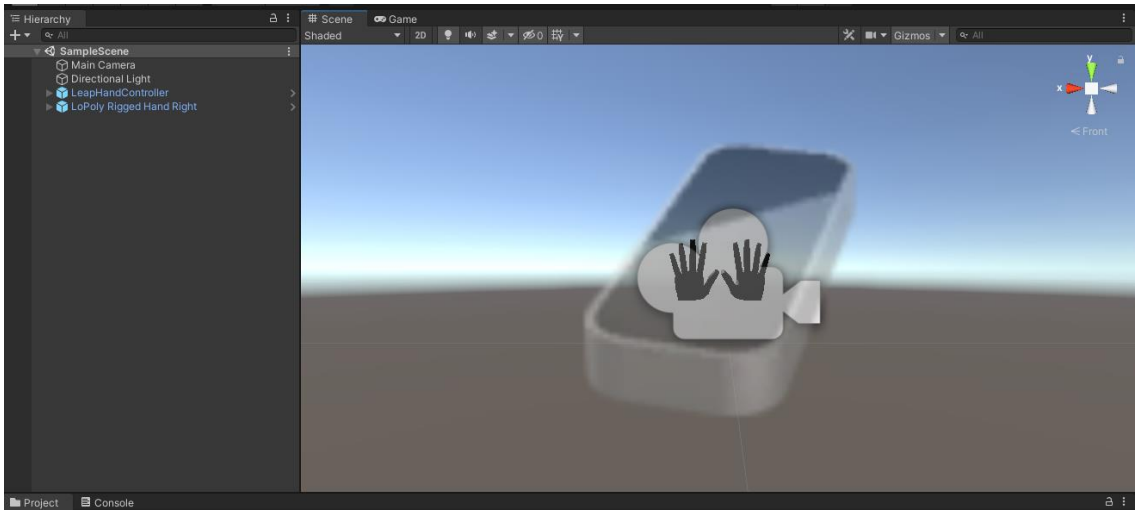

Figure 2.7: Hands in Unity. Example of a pre-loaded Leap Hands module and LMC loaded into Unity VR.

Makehuman is a software used to design human models for game engines. One can customize the height, weight, gender, appearance, etc. of a human model. Additionally, a skeleton can be added to the model. Figure 2.8 illustrates the skeletal structure of the model. This proved to be very useful for hand and finger tracking in Unity, as tracking points can be rigged to the bones of a model. The ".massproduce" plugin in Makehuman was also used in this project. This randomly generates a specified number of models with completely randomized physical traits. This will allow for tracking to be implemented in a variety of different environments and will be utilized to build a dataset as the project moves to the implementation stage. Validation was required to determine that the randomization was sufficient for a diverse dataset.

The rigged model hands in Unity map the LMC tracking to the joints of the Makehuman model. This is done using the animation of the human model in Unity combined with the virtual

LMC module in Unity. The “rigged hands” script must be used to identify the bones and fingers that are being tracked by the LMC. Validation must be completed to determine the best location for the LMC in Unity. Shown in Figure 2.9 is the first step of rigging the hand bones from the Makehuman model in Unity. Once hand models have been rigged and mapped to the LMC, a user can input a hand gesture and have the virtual hand mirror the movements of a physical hand input.

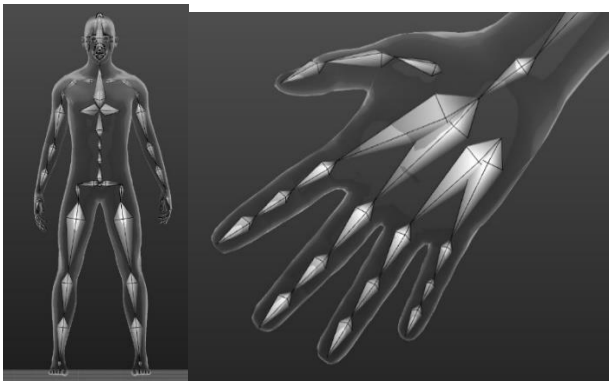

Figure 2.8: Makehuman Skeleton. Bones are assigned to a full human model (right) and used for tracking in Unity. Primary care abouts are finger and hand bones (left).

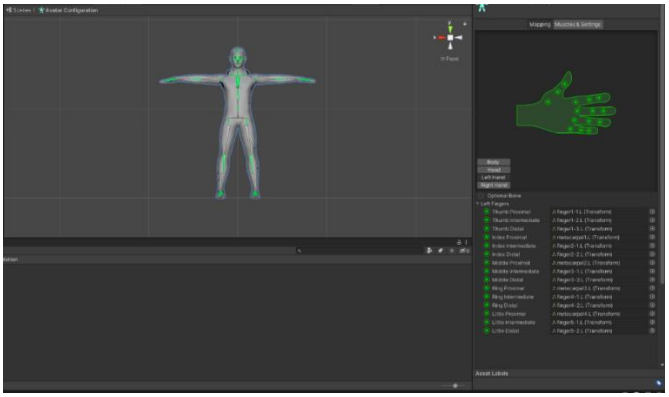

Figure 2.9: Rigged Fingers and Hands. The bones from the Makehuman hand model are mapped to Unity.

3. RESULTS

3.1 Results

3.1.1 Leap Motion Controller

The Leap Motion Controller was tested in a wide range of settings. Different lightings, skin tones, distances, and gloves on hands were all tested to see the effect they have on hand tracking. The two that had the greatest effect on hand tracking accuracy were distance from the sensor and the type of glove on one's hand. The highlights of results from distance testing can be seen in Figure 3.1. Complete LMC range testing results can be seen in Tables 3.1 – 3.3. It was determined from this testing that the ideal range of the sensor is 20-60cm. The Leap Motion Controller will track hand joints in this range with an accuracy of 0.2mm [23].

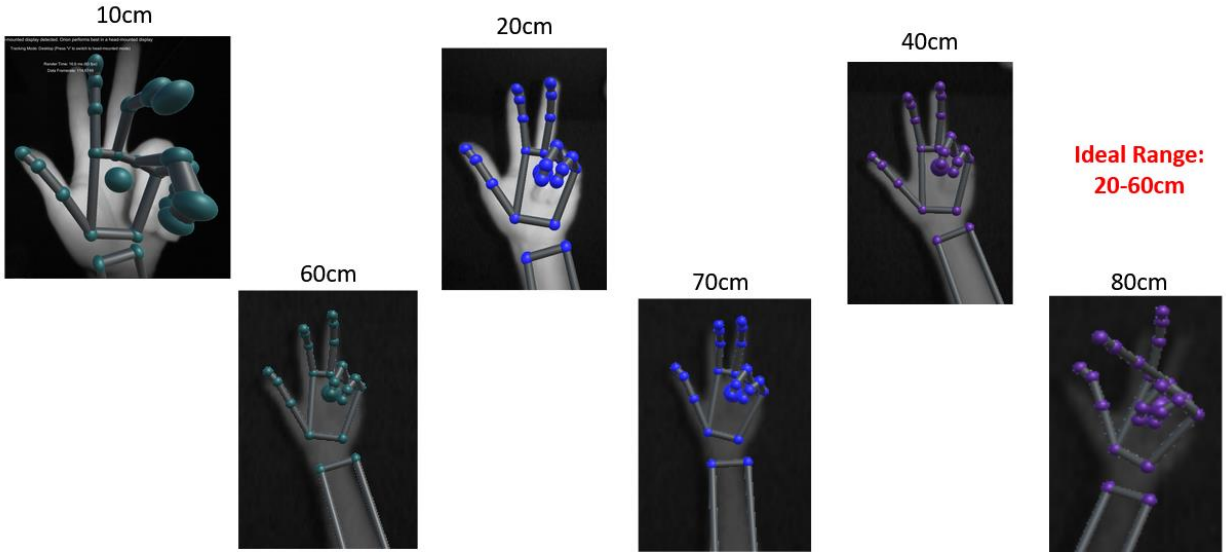

Figure 3.1: Leap Motion Range Testing. The same gesture was tested at six different distances from the sensor. The sensor provided no accuracy at 80cm and decreased tracking accuracy after 70cm.

Table 3.1: Leap Motion Range Testing When Changing Light Intensity

Leap Motion Tracking with Changes in Light Intesntity			
Range (cm)	**Bright Light**	**Medium Light**	**Dim Light**
5	NO	NO	NO
10	YES	NO	NO
15	YES	YES	NO
20	YES	YES	YES
25	YES	YES	YES
30	YES	YES	YES
35	YES	YES	YES
40	YES	YES	YES
45	YES	YES	YES
50	YES	YES	YES
55	YES	YES	YES
60	YES	YES	YES
65	YES	YES	NO
70	YES	NO	NO
75	NO	NO	NO
80	NO	NO	NO

‘

Figure 3.2 highlights Leap Motion tracking with different kinds of gloves. The findings from this illustrate that the Leap Motion Controller will not track any kind of glove with a two-tone palm. Single color gloves like the middle glove shown in Figure 3.2 can be accurately tracked. As seen in Table 3.2, wearing a solid glove decreases the maximum range of the LMC by 23.07%. When in bright light, while two tone gloves are not recognized at all. Table 3.3 shows that when gloves must be worn, it is best to be in an environment with bright light. It appears no other research has been done testing the effect wearing gloves has on Leap Motion Controller accuracy. From this data, the best hand tracking data will be received from the LMC in a brightly lit environment when a hand has no glove on. The range in this environment is 10 to 70cm.

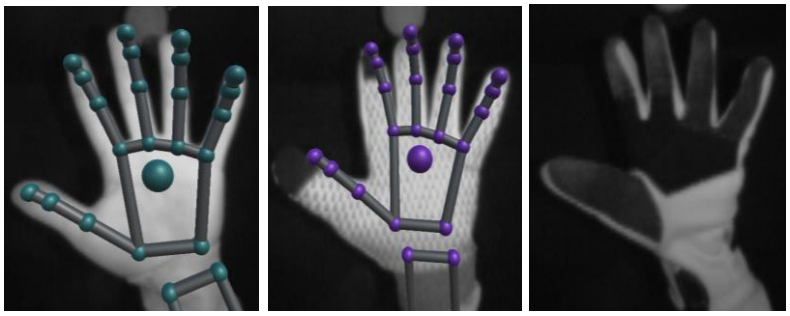

Figure 3.2: Leap Motion Tracking with Gloves. Hands with no gloves (left), a single-color glove (middle), and a two-tone glove (right), were tracked in the Leap Motion Visualizer at a range of 20cm.

Table 3.2: Leap Motion Range Testing When Changing Gloves

Leap Motion Tracking with Changes in Gloves			
Range (cm)	**No Gloves**	**Single Color**	**Two-tone**
5	NO	NO	NO
10	YES	NO	NO
15	YES	YES	NO
20	YES	YES	NO
25	YES	YES	NO
30	YES	YES	NO
35	YES	YES	NO
40	YES	YES	NO
45	YES	YES	NO
50	YES	YES	NO
55	YES	NO	NO
60	YES	NO	NO
65	YES	NO	NO
70	NO	NO	NO
75	NO	NO	NO
80	NO	NO	NO

Table 3.3: Leap Motion Range Testing When Changing Light Intensity with Gloves

Leap Motion Tracking with Single Color Gloves			
Range (cm)	**Bright Light**	**Medium Light**	**Dim Light**
5	NO	NO	NO
10	NO	NO	NO
15	YES	YES	YES
20	YES	YES	YES
25	YES	YES	YES
30	YES	YES	YES
35	YES	YES	YES
40	YES	YES	NO
45	YES	NO	NO
50	YES	NO	NO
55	NO	NO	NO
60	NO	NO	NO
65	NO	NO	NO
70	NO	NO	NO
75	NO	NO	NO
80	NO	NO	NO

3.1.2 Makehuman Hand Models

The Makehuman ".massproduce" function was used to generate 1,000 unique human models. These models have different hand sizes, skin tones, and clothing. Figure 3.3 highlights two of the virtual human models in the Makehuman environment. These models were used to create the dataset of hand gestures. One of the most important factors in any gesture recognition dataset is the variation [24]. Domain randomization for computer vision gesture recognition is a concept that can easily be accounted for using Unity [9].

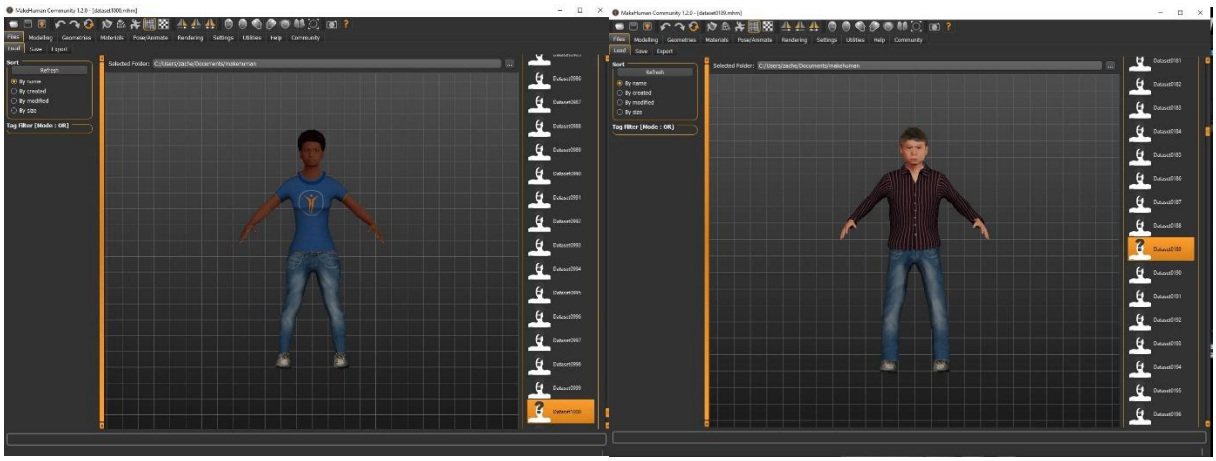

Figure 3.3: Makehuman Models. Models were given a skeletal structure and imported into Unity to be given a set hand gesture.

3.1.3 Machine Learning Dataset

Using the hand models, a dataset of hand gestures was created to be used in the machine learning training. The first iteration of the dataset consisted of 1,000 total hands, containing 100 hands for each of the 10 still gestures. This dataset was used for proof of concept and to verify that the gesture recognition program and LSTM machine learning was properly working. Once verified, the dataset size was expanded to 150,000. Previous research shows that increasing dataset size increases the confidence index of a machine learning gesture recognition algorithm [25]. 10 still hand gestures and 5 moving gestures each had 10,000 images from the human models. To gauge the improvement of this method of dataset creation enables, a more generic dataset was used to compare results. This dataset was developed by R. Arya and uses black and white images that only change finger shape and size [13]. This data contains 24,000 hands compared to the 150,000 used in the virtual hand dataset. A comparison of the two datasets can be seen in Figure 3.4.

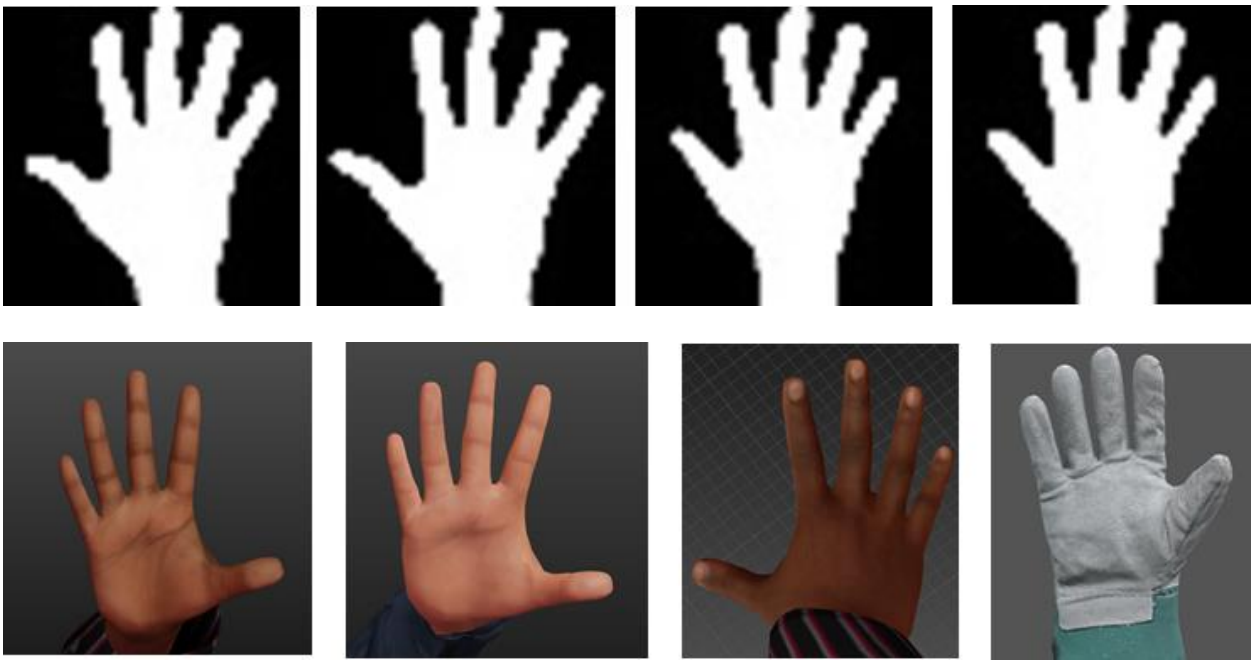

Figure 3.4: Hand Datasets. A generic dataset of pixelated hands (top) is compared to a newly developed virtual hand dataset (bottom).

To obtain a dataset that is more effective for training, domain randomization was used. Domain randomization involves changing different variables in a given dataset to diversify the type of knowledge that the neural network will store [9]. In this dataset, the variables that were randomized were, image background color, skin tone, hand size, glove color, hand orientation, and gesture type.

3.1.3.1 Image Background

To account for all possible background possibilities, hands were placed in environments with randomized color. A spectrum of the colors used can be seen in Figure 3.5. In addition to the randomized color, 500 Google search images of "factory automation setting" to familiarize the ML with common things that will be seen in a factory [11].

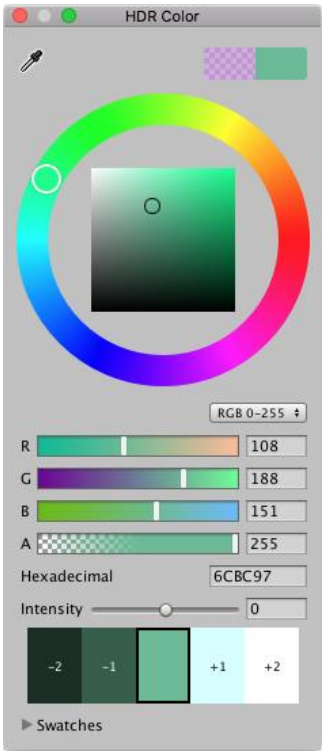

Figure 3.5: Unity Color Wheel. Within Unity, 9,500 random colors were assigned to be the background of the VR.

3.1.3.2 Skin Tone

The Makehuman ".massproduce" function randomly created a new skin tone for each model that it generates. 1,000 unique human models were generated, which created 1,000 unique skin tones. For each of the fifteen gesture classes, 10,000 images were created. Hands were assigned skin tone colors in groups of ten for uniformity purposes.

3.1.3.3 Hand Size

Once again, mass producing models in Makehuman allowed for complete randomization of hand size. 1,000 unique models were made, meaning there are 1,000 unique hand sizes utilized in this dataset. Hand sizes for models can be manually adjusted with Makehuman sliders

if one desires. The parameters for hand size can be seen in Figure 3.6 and included fingers distance, fingers diameter, fingers length, scale hand, and hand position.

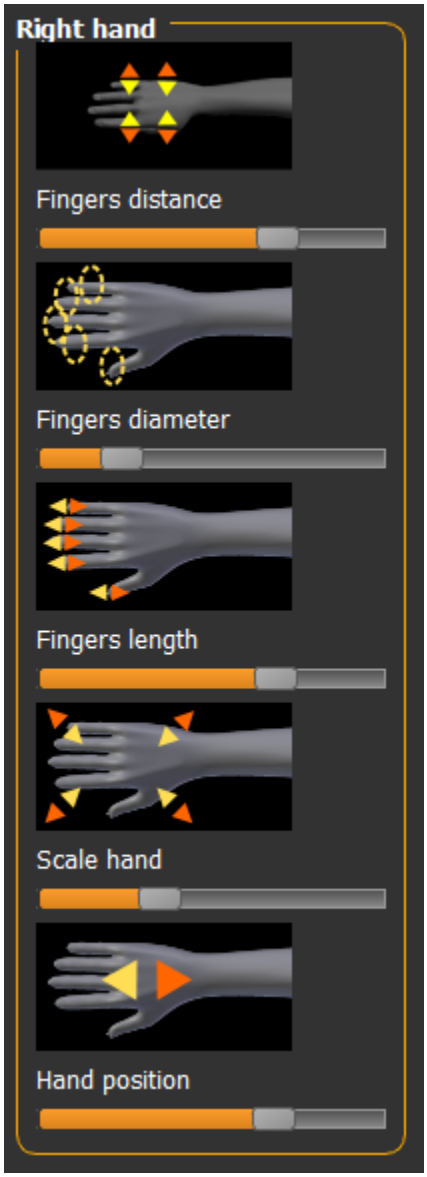

Figure 3.6: Hand Size Variation. Each model created placed each of the five sliders in a unique position which created 1,000 unique hand sizes.

3.1.3.4 Glove Color

One innovation that this research presented over previous studies is the inclusion of gloves in a hand model dataset. No previous studies were found to include hands with gloves in a virtual dataset. This was done because factory automation workers often are required to wear gloves for safety purposes. [11]. Glove color was randomized using the same Unity color spectrum in Figure 3.5. Makehuman has a wide variety of gloves available for use. 1,000 of the 10,000 models for each gesture were given gloves.

3.1.3.5 Gesture Type

For the ten still gestures, Unity sliders were used to create a preset for each human model that was uploaded. A visual depiction of the Unity sliders can be found in Figure 3.7. These sliders provide a simple and repeatable way to create unique gestures. For the five moving gestures, it was necessary to first create model animations in Blender and then upload those animations to Unity.

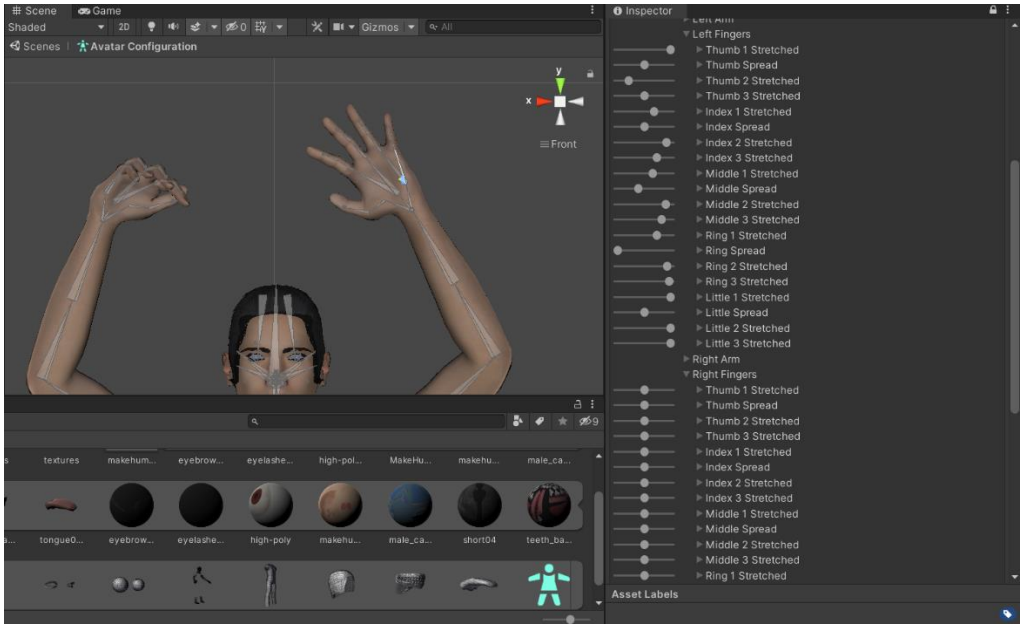

Figure: 3.7 Unity Sliders: The skeletal structure of a model is used to adjust position

3.1.4 Gesture Recognition Model

A gesture recognition program for Leap Motion was used to determine how using a virtual dataset affects the confidence index of gesture recognition. The program used is based on research done at the University of Auckland [26]. The output of the program is a graph that shows the recognized gesture, confidence index of the recognized gesture, movement of the hands, and angularity of the hands. The biggest care about in this graph is the confidence index line. An image of a sample graph output can be seen in Figure 3.8.

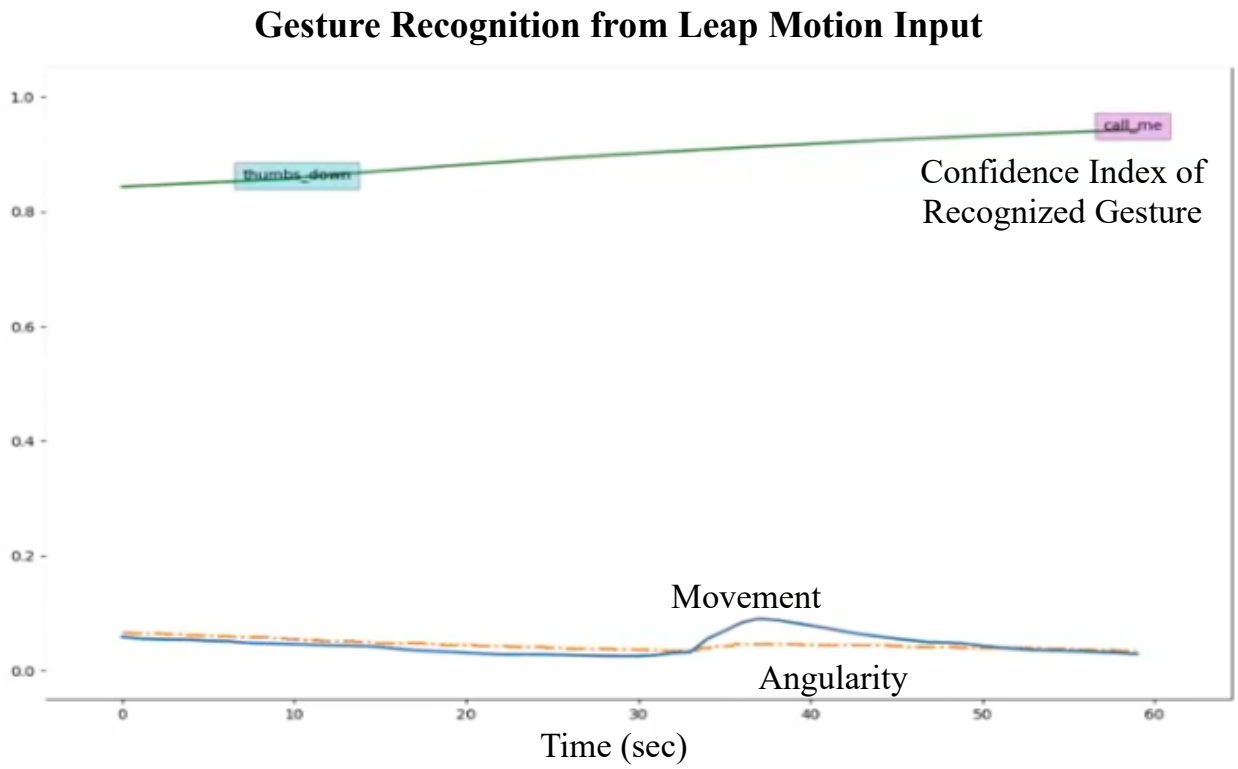

Figure 3.8: Leap Motion Gesture Recognition. The recognized gesture, confidence index (green), movement (blue), and angularity (yellow) are all graphed.

This gesture recognition graph was used twice for each of the ten still gestures. In the first iteration, a generic dataset was used [13]. The second test was done with the new virtual

dataset. For the five moving gestures, only the new dataset was used as previous researchers did not incorporate moving gestures into their dataset. The fifteen gestures can be seen in Figure 3.9. Graphs comparing the confidence index of these recognized gestures will be shown below.

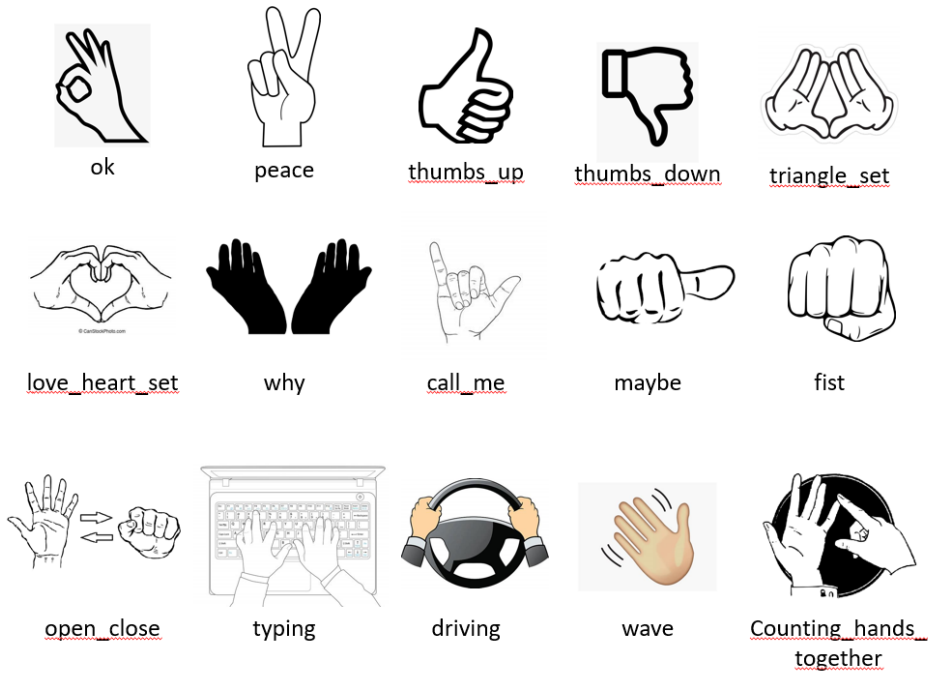

Figure 3.9: Fifteen Gesture Classes. Ten still gestures (top and middle) and five moving gestures (bottom).

3.1.5 Gesture Recognition Results Comparison

To accurately get improvement metrics from using virtual reality dataset, results were first collected using a standard gesture recognition dataset made publicly available [13]. Confidence index data was recorded at intervals of 5, 10, 15 and 20 seconds to accurately illustrate how the confidence index increased over time. It is clear from the data that the confidence index of a gesture increased at an average rate of 0.56% per second with the old dataset and 0.39% per second with the new dataset. The "overall CI" is an average of the four

confidence index measurements and is the measurement used to calculate overall improvement from the system. The data showed the overall confidence index increased by an average of 12.45% for each still gesture when the virtual dataset developed is used. The data showed that while the confidence index increases at a slower rate using the new dataset, the overall increase in confidence is significant. The full results for each individual gesture can been seen in Table 3.4, Table 3.5, Table 3.6, Table 3.7, Table 3.8, Table 3.9, Table 3.10, Table 3.11, Table 3.12, Table 3.13, Table 3.14, Table 3.15, Table 3.16, Table 3.17, and Table 3.18. For moving gestures, no public dataset was available to test beforehand, so no improvement metrics could be calculated. The five moving gestures had an average overall CI of 90.01%.

One interesting note from the data is that the overall CI is slightly lower for gestures that are similar (thumbs_up, thumbs_down, maybe). This can be seen in Tables 3.6, 3.7, and 3.12.

Table 3.4: Confidence Index of "ok" Gesture

Confidence Index (%) of "ok"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	77.03	80.67	82.98	86.05	81.6825	0.60133333
New	85.96	88.19	90.95	93.87	89.7425	0.52733333
				% Improvement	9.867475	

Table 3.5: Confidence Index of "peace" Gesture

Confidence Index (%) of "peace"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	74.85	79.46	83.12	85.24	80.6675	0.69266667
New	87.14	90.36	92.97	95.28	91.4375	0.54266667
				% Improvement	13.3511	

Table 3.6: Confidence Index of "thumbs_up" Gesture

Confidence Index (%) of "thumbs_up"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	61.87	66.34	68.74	72.65	67.4	0.71866667
New	78.85	82.96	83.65	84.39	82.4625	0.36933333
				% Improvement	22.34792	

Table 3.7: Confidence Index of "thumbs_down" Gesture

Confidence Index (%) of "thumbs_down"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	67.89	70.43	72.76	75.34	71.605	0.49666667
New	82.45	85.67	88.33	91.05	86.875	0.57333333
				% Improvement	21.32533	

Table 3.8: Confidence Index of "triangle_set" Gesture

Confidence Index (%) of "triangle_set"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	81.54	84.63	88.01	90.64	86.205	0.60666667
New	85.29	87.17	89.64	92.08	88.545	0.45266667
				% Improvement	2.71446	

Table 3.9: Confidence Index of "love_heart_set" Gesture

Confidence Index (%) of "love_heart_set"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	80.47	82.95	84.74	87.03	83.7975	0.43733333
New	85.94	86.67	87.96	89.64	87.5525	0.24666667
				% Improvement	4.481041	

Table 3.10: Confidence Index of "why" Gesture

Confidence Index (%) of "why"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	75.62	78.96	81.9	84.77	80.3125	0.61
New	88.12	91.56	93.34	95.78	92.2	0.51066667
				% Improvement	14.80156	

Table 3.11: Confidence Index of "call_me" Gesture

Confidence Index (%) of "call_me"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	78.21	81.14	84.35	85.92	82.405	0.514
New	85.67	88.75	89.43	90.12	88.4925	0.29666667
				% Improvement	7.387294	

Table 3.12: Confidence Index of "maybe" Gesture

Confidence Index (%) of "maybe"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	63.45	67.28	70.06	72.99	68.445	0.636
New	79.87	82.56	83.75	85.02	82.8	0.34333333
				% Improvement	20.97304	

Table 3.13: Confidence Index of "fist" Gesture

Confidence Index (%) of "fist"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
Old	82.74	85.92	86.13	87.2	85.4975	0.29733333
New	88.34	90.02	93.15	95.32	91.7075	0.46533333
				% Improvement	7.26337	

Table 3.14: Confidence Index of "open_close" Gesture

Confidence Index (%) of "open_close"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
New	85.12	87.95	91.69	94.27	89.7575	0.61

Table 3.15: Confidence Index of "typing" Gesture

Confidence Index (%) of "typing"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
New	85.67	89.23	91.01	94.83	90.185	0.61066667

Table 3.16: Confidence Index of "driving" Gesture

Confidence Index (%) of "driving"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
New	88.92	91.56	93.19	95.05	92.18	0.40866667

Table 3.17: Confidence Index of "wave" Gesture

Confidence Index (%) of "wave"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
New	83.94	86.32	89.06	91.99	87.8275	0.53666667

Table 3.18: Confidence Index of "counting_hands_together" Gesture

Confidence Index (%) of "counting_hands_together"						
Dataset / Time (sec)	5	10	15	20	Overall CI	Slope
New	87.45	89.52	91.28	92.11	90.09	0.31066667

4. CONCLUSION

4.1 Conclusion

For this research, conclusions can be drawn about each of the three main subsystems. It was concluded that the Leap Motion Controller is a highly accurate hand tracking sensor that works best in a controlled environment. When variables begin to change, the range of the sensor decrease by up to 23.07%. Makehuman, Unity, and Blender are excellent to use to develop large scale datasets in a virtual reality. Early results from the machine learning and gesture recognition testing are showing an average increase of 12.45% in the confidence index of each of the ten still gesture classes. The five moving gestures had no previous reference datasets but are consistently being recognized with an average confided index of 90.01%. It has been noticed that gestures that are similar (for example thumbs up and thumbs down) have a lower confidence index. The use of virtual hand models to train a robot in virtual reality is showing early promise, but more testing is needed.

Due to the fact this research is new in the field of human robot interaction, it is probable that a similar approach to improve gesture recognition could be taken in other fields. One very intriguing area to explore is sign language recognition and translation. Theoretically, a similar dataset of sign language letters and words could be used to improve interpretation and translation of sign language in the virtual and physical world. Research involving sign language gesture recognition has already been done in a study from *Applied Sciences*. [8] In this study, the Leap Motion Controller was used to track and recognize sign language gestures. A similar virtual dataset could be applied to this project, and one would expect to see similar improvements in both the recognition and the confidence index of the machine learning.

4.2 Recommendations

At the present time, the recommendations for those implementing or continuing research in this field are as follows:

1. Utilize virtual datasets in future projects to more efficiently create robust datasets that allow for more randomization and variation.
2. Create datasets that are in the millions in length. Current results from this project show that as dataset size increases, the confidence index of the gesture recognizer also increases. A dataset this size should consistently provide classifications with a confidence index of above 95% [27].
3. Explore other possible options for hand tracking hardware to be used in a factory automation setting. The Leap Motion Controller provides adequate readings in a lab or more controlled environment. Other hand tracking hardware may provide more accurate readings in uncontrolled environments [28].
4. Apply virtual machine learning hand datasets to sign language recognition and translation with the Leap Motion Controller [8]. Utilizing Unity and Blender would allow for previous research projects that used only the sign language alphabet to expand into moving gestures [29]. Other sign language gesture recognition programs can recognize moving gestures with an average confidence index of 72.79% [30]. If a virtual hand dataset were to be used for a project like this, it would be expected that a similar 12.45% confidence index increase could be seen.

REFERENCES

[1] H. Silaghi, U. Rohde, V. Spoiala, A. Silaghi, E. Gergely, and Z. Nagy, "Voice command of an industrial robot in a noisy environment," 2014 International Symposium on Fundamentals of Electrical Engineering (ISFEE), Nov. 2014.

[2] S. Sharma, S. Sharma, and P. Yadav, "Design and implementation of robotic hand control using gesture recognition," International Journal of Engineering Research and, vol. V6, no. 04, Apr. 2017.

[3] "Leap Motion Controller Datasheet - Ultraleap," Leap Motion, 2021. [Online]. Available: https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf. [Accessed: 2022].

[4] U. Technologies, "Unity user manual 2020.3 (LTS)," Unity, 2020. [Online]. Available: https://docs.unity3d.com/Manual/index.html. [Accessed: 2022].

[5] "Vive Pro Specs & User Guide," VIVE Pro Specs & User Guide - Developer Resources, 2018. [Online]. Available: https://developer.vive.com/resources/hardware- guides/vive-pro-specs-user-guide/. [Accessed: 2022].

[6] F. Adolfsson, "Linköpings universitet SE- Linköping + A Model-Based Approach to Hands Overlay for Augmented Reality." Accessed: Apr. 02, 2022. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1547156/FULLTEXT01.pdf

[7] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33–55, Mar. 2016, doi: 10.1109/tiv.2016.2578706.

[8] A. Vaitkevičius, M. Taroza, T. Blažauskas, R. Damaševičius, R. Maskeliūnas, and M. Woźniak, "Recognition of american sign language gestures in a virtual reality using Leap Motion," Applied Sciences, vol. 9, no. 3, p. 445, 2019.

[9] S. Z. Valtchev and J. Wu, "Domain randomization for neural network classification," Journal of Big Data, vol. 8, no. 1, Jul. 2021, doi: 10.1186/s40537-021-00455-5.

[10] T. B. Sheridan, “Human–Robot Interaction,” Human Factors: The Journal of the Human Factors and Ergonomics Society, vol. 58, no. 4, pp. 525–532, Apr. 2016, doi: 10.1177/0018720816644364.

[11] T. Heimonen, J. Hakulinen, M. Turunen, J. P. P. Jokinen, T. Keskinen, and R. Raisamo, “Designing Gesture-Based Control for Factory Automation,” Human-Computer Interaction – INTERACT 2013, pp. 202–209, 2013, doi: 10.1007/978-3-642-40480-1_13.

[12] M. Vasic and A. Billard, “Safety issues in human-robot interactions,” 2013 IEEE International Conference on Robotics and Automation, May 2013, doi: 10.1109/icra.2013.6630576.

[13] R. Arya, “Hand gesture recognition dataset,” Kaggle, 04-Sep-2021. [Online]. Available: https://www.kaggle.com/datasets/aryarishabh/hand-gesture-recognition-dataset. [Accessed: 2022].

[14] L. Shao, G. Wetzstein, and R. Konrad, “Hand movement and gesture recognition using Leap Motion Controller Stanford EE 267, Virtual Reality,Course Report, Instructors.” [Online]. Available: https://stanford.edu/class/ee267/Spring2016/report_lin.pdf

[15] G. Caggianese, L. Gallo, and P. Neroni, “The Vive Controllers vs. Leap Motion for Interactions in Virtual Environments: A Comparative Evaluation,” Intelligent Interactive Multimedia Systems and Services, pp. 24–33, Jun. 2018, doi: 10.1007/978-3-319-92231-7_3.

[16] “USB 2.0 Specification | USB-IF,” Usb.org, Dec. 21, 2018. https://www.usb.org/document-library/usb-20-specification

[17] Thingiverse.com, “Leap Motion VR Developer Mount by LeapMotion,” @thingiverse, 2022. https://www.thingiverse.com/thing:445866 [Accessed: 2022].

[18] “3D printing file for VR Developer Mount,” Ultraleap (Leap Motion) Community Forums. https://forums.leapmotion.com/t/3d-printing-file-for-vr-developer-mount/1657 [Accessed: 2022].

[19] “VR Mount for HTC Vive Pro : now available!,” Ultraleap (Leap Motion) Community Forums. https://forums.leapmotion.com/t/vr-mount-for-htc-vive-pro-now- available/7243 [Accessed: 2022].

[20] M. Huzaifa et al., “ILLIXR: An Open Testbed to Enable Extended Reality Systems Research,” IEEE Micro, pp. 1–1, 2022, doi: 10.1109/mm.2022.3161018.

[21] A. Shingade and A. Ghotkar, “Animation of 3D Human Model Using Markerless Motion Capture Applied To Sports,” International Journal of Computer Graphics & Animation, vol. 4, no. 1, pp. 27–39, Jan. 2014, doi: 10.5121/ijcga.2014.4103.

[22] T. Novacek, C. Marty, and M. Jirina, “Project MultiLeap: Fusing Data from Multiple Leap Motion Sensors,” 2021 IEEE 7th International Conference on Virtual Reality (ICVR), May 2021, doi: 10.1109/icvr51878.2021.9483819.

[23] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, “Analysis of the Accuracy and Robustness of the Leap Motion Controller,” Sensors, vol. 13, no. 5, pp. 6380–6393, May 2013, doi: 10.3390/s130506380.

[24] C. M. de Melo, B. Rothrock, P. Gurram, O. Ulutan, and B. S. Manjunath, “Vision-Based Gesture Recognition in Human-Robot Teams Using Synthetic Data,” 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2020, doi: 10.1109/iros45743.2020.9340728.

[25] A. L’Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, “Machine Learning With Big Data: Challenges and Approaches,” IEEE Access, vol. 5, pp. 7776–7797, 2017, doi: 10.1109/access.2017.2696365.

[26] I. U. Rehman et al., “Fingertip Gestures Recognition Using Leap Motion and Camera for Interaction with Virtual Environment,” Electronics, vol. 9, no. 12, p. 1986, Nov. 2020, doi: 10.3390/electronics9121986.

[27] A. Bailly et al., “Effects of Dataset Size and Interactions on the Prediction Performance of Logistic Regression and Deep Learning Models,” Computer Methods and Programs in Biomedicine, p. 106504, Oct. 2021, doi: 10.1016/j.cmpb.2021.106504.

[28] K. Tanjung, F. Nainggolan, B. Siregar, S. Panjaitan, and F. Fahmi, “The Use of Virtual Reality Controllers and Comparison Between Vive, Leap Motion and Sensor Gloves Applied in The Anatomy Learning System,” Journal of Physics: Conference Series, vol. 1542, no. 1, p. 012026, May 2020, doi: 10.1088/1742-6596/1542/1/012026.

[29] D. Naglot and M. Kulkarni, "Real time sign language recognition using the leap motion controller," *2016 International Conference on Inventive Computation Technologies (ICICT)*, 2016, pp. 1-5, doi: 10.1109/INVENTIVE.2016.7830097.

[30] T.-W. Chong and B.-G. Lee, “American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach,” Sensors, vol. 18, no. 10, p. 3554, Oct. 2018, doi: 10.3390/s18103554.