University of Nevada, Reno

American Sign Language Gesture Recognition using Motion Tracking Gloves in VR

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Justice Steven Colby

Dr. Frederick C. Harris Jr., Advisor

May, 2022

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Justice Steven Colby

Entitled

American Sign Language Gesture Recognition using Motion Tracking Gloves in VR

be accepted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Dr. Frederick C. Harris, Jr., Advisor

Dr. Sergiu M. Dascalu,

Dr. Kristine E. Galek, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

May, 2022

Abstract

Gesture recognition has become an topic of great interest as it continues to advance the capabilities of human computer interaction. Research has shown that related technologies have the potential to facilitate highly accessible user interfaces, enabling users with various limitations to use different applications in a more intuitive way. This thesis presents a new contribution to this research by introducing a novel approach to performing gesture recognition on American sign language (ASL) hand gestures through virtual reality (VR) using motion tracking gloves. As a proof of concept, an application was developed using this approach which is capable of recognizing 34 ASL hand gestures performed by a user as they navigate a turorial-based environment. This application was evaluated through a user study to determine the effectiveness of the approach and any possible improvements that could be made. The hope is that the approach presented in this thesis could be expanded into a number of different applications aimed at propagating the use of ASL and improving the lives of those who use it regularly.

Dedication

I dedicate this thesis to my mom, Laura Diane Johnson, who helped me to persevere through many difficult times, and supported me through all of my endeavors.

Acknowledgments

I would like to thank my advisor Dr. Fredrick C. Harris, Jr. for his incredible support and guidance throughout my graduate school career. I would also like to thank committee members Dr. Sergiu M. Dascalu, and Dr. Kristine Galek for their time and support in helping me to complete this thesis. I would also like to thank Yifan Zhang and Daniel Enriquez for helping me to train the models used within the ASL gesture recognition application presented in this thesis. Also, thank you to Chase Carthen for helping me brainstorm project ideas and teaching me several useful concepts. I'd also like to thank all of the other friends I have made throughout my college career who have helped make the experience all the more enjoyable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There has been abundant research in the field of gesture recognition, with related applications spanning a wide range of technologies and disciplines. Gesture recognition refers to the process of detecting and interpreting human gestures by acquiring and analyzing related data [4]. Much of the research in this field has been dedicated towards hand gesture recognition, with sign language gestures being a particular area of interest. Sign Language is a method of communication used primarily within the Deaf and Hard of Hearing (DHH) community which uses hand gestures in the place of letters and words. Several applications have been developed with the goal of consistently and efficiently recognizing sign language gestures in order to ease the communication challenges commonly faced within the DHH community. While sign language gestures do follow specific rules and syntax as with any language, the gestures being performed can still vary greatly depending on the person performing them, as well as the shape and size of that person's hands. Because of this, machine learning techniques are often employed in these types of applications, since they are ideal for adapting to new data variations, and provide an efficient way of updating required gesture parameters as needed. Despite the successes of gesture recognition applications, they also face a number of challenges which can limit overall usability. One common issue with gesture recognition is occlusions, or obstructions within the area being detected, which can be caused by overlapping fingers, or forming fists. Many gesture recognition applications also lack the ability to recognize moving gestures as well as gestures which involve parts of the body that are outside of a typical field of view.

VR is another area of research which has seen substantial contributions in various disciplines. VR encompasses the creation and utilization of artificial simulations meant to replace one or more sensory stimuli provided by the real world [14]. A common technology used to experience VR, which is also the technology that will be discussed in this thesis, is a VR headset which provides visual and sometimes auditory simulations to a user. Since VR involves immersing the user into a virtual world with nearly endless possibilities, the field has massive potential to provide unique and intuitive user interfaces, especially when combined with different types of peripheral hardware. One approach to designing VR UIs is providing a simulated version of the user's hands which can replicate real-world movements within the virtual environment. There are a number of tools which achieve this in different ways, with each one having it's own benefits and trade-offs. Some tools include infrared sensor technologies such as the LEAP motion, as well as motion tracking gloves [23].

This thesis presents a novel approach to performing ASL hand gesture recognition within a VR environment. This approach has been implemented into a usable application which successfully recognizes ASL gestures being performed by a user. The hardware used with the application includes the Vive Cosmos VR Headset along with the VRFree Sensoryx motion capture gloves. Using a feed-forward neural network to classify gestures, all letters of the ASL alphabet are successfully recognized along with 8 words/phrases. The gesture recognition technique employed by this application has the capability to recognize moving gestures as well as gestures which must be performed at specific parts of the body, which has been a limitation with many previous related applications. The VRFree gloves also overcome the occlusion issue of hand gesture recognition by providing distinct sensor data from each joint of each finger. In order to effectively leverage the VR portion of the application, an intuitive tutorial-based interface was created for the user which displays examples of gestures to be performed, and enables the user to perform gestures one at a time until each one is performed successfully.

The rest of this thesis is structured as follows: Chapter 2 provides background

knowledge integral to understanding the material of this work. This chapter starts by providing general information regarding sign language along with a review of technologies that have been used to promote sign language education and communication. It then provides information regarding VR, motion tracking hardware, and machine learning, with a focus on areas that are more relevant to the topics presented in this thesis. Chapter 3 discusses the framework that has been created for recognizing ASL gestures within a virtual environment. It also discusses the application that was created using this framework, and the implementation of them both. Chapter 4 outlines the details of a user study that was performed to evaluate the effectiveness of the ASL gesture recognition application and the intuitiveness of the user interface. The results of the user study are then analyzed in Chapter 5 based on a number of metrics used along with input from the participants. Finally, conclusions as well as future work are present in Chapter 6.

Chapter 2

Background and Related Work

2.1 Sign Language

2.1.1 Overview

Sign Language can be defined as a method of communication using hand gestures, facial expressions, and body posture, which is utilized primarily by the Deaf and Hard of Hearing (DHH) community. The language incorporates hand gestures representing "signs" which can be performed using one or both hands, with individual signs representing letters, words, or phrases. Along with the hand gestures themselves, facial expressions and body positioning can also be incorporated to present a more accurate interpretation of the information being communicated. While not everyone in the DHH community uses sign language as a primary method of communication, a great number prefer it due to it's capability to produce a descriptive image with the entire body, which can often pale in comparison with emphasized speech [31].

While sign language is often associated with those in the DHH community, several other groups throughout history have also used a similar type of communication in different situations [31]. For example, Native Americans have used signs to communicate with different tribes due to the high difficulty in mastering many spoken dialects. Also, hunting groups within Africa continue to use signs to communicate silently when sneaking up on prey. In fact, nearly everyone does perform some kind of gestural or nonverbal communication despite sometimes being so subtle that it goes unnoticed. For example, one might touch their chin to indicate that they are

thinking about something, or give a passing smile to indicate their comfortableness with someone [31].

Just as with spoken languages, different types of sign languages exist depending on the community and/or region in which they are being performed. The application presented in this thesis focuses on recognizing gestures from American Sign Language or ASL, which is used in North America. The gestures of the ASL alphabet are displayed in Figure 2.1. Along with the alphabet, there are also various words and phrases that can be performed in ASL using either a single hand gesture or multiple hand gestures combined.

The basic parameters which form a sign in ASL are presented in [31] and are outlined below :

- **Handshape** - the shape of a the hands when forming a sign. This can stay the same or change while performing the sign. Two-handed sign gestures can have the same handshape for both hands, or they can be different [31].

- **Orientation** - the position of the hand(s) in relation to the body. An example of this could be which direction the palm is facing when performing a sign (facing towards or away from the body, towards the ground, or upward) [31].

- **Location** - the place in space where a sign is formed. Some signs can be stationary, such as “MY” or “LOVE”, and some can move from one location in the signing space to another, such as “HELLO” or “NICE” [31].

- **Movement** - the direction in which the hand moves in relation to the body when performing a sign. Some signs may have a simple small movement, such as “NICE”, while some have a more complex larger movement, such as the gesture for “SIGN LANGUAGE” [31].

- **Nonmanual markers** - Additional body movements which are added to signs to create meaning. These can include facial expressions, head tilting, shoulder movement, and mouth movements [31].

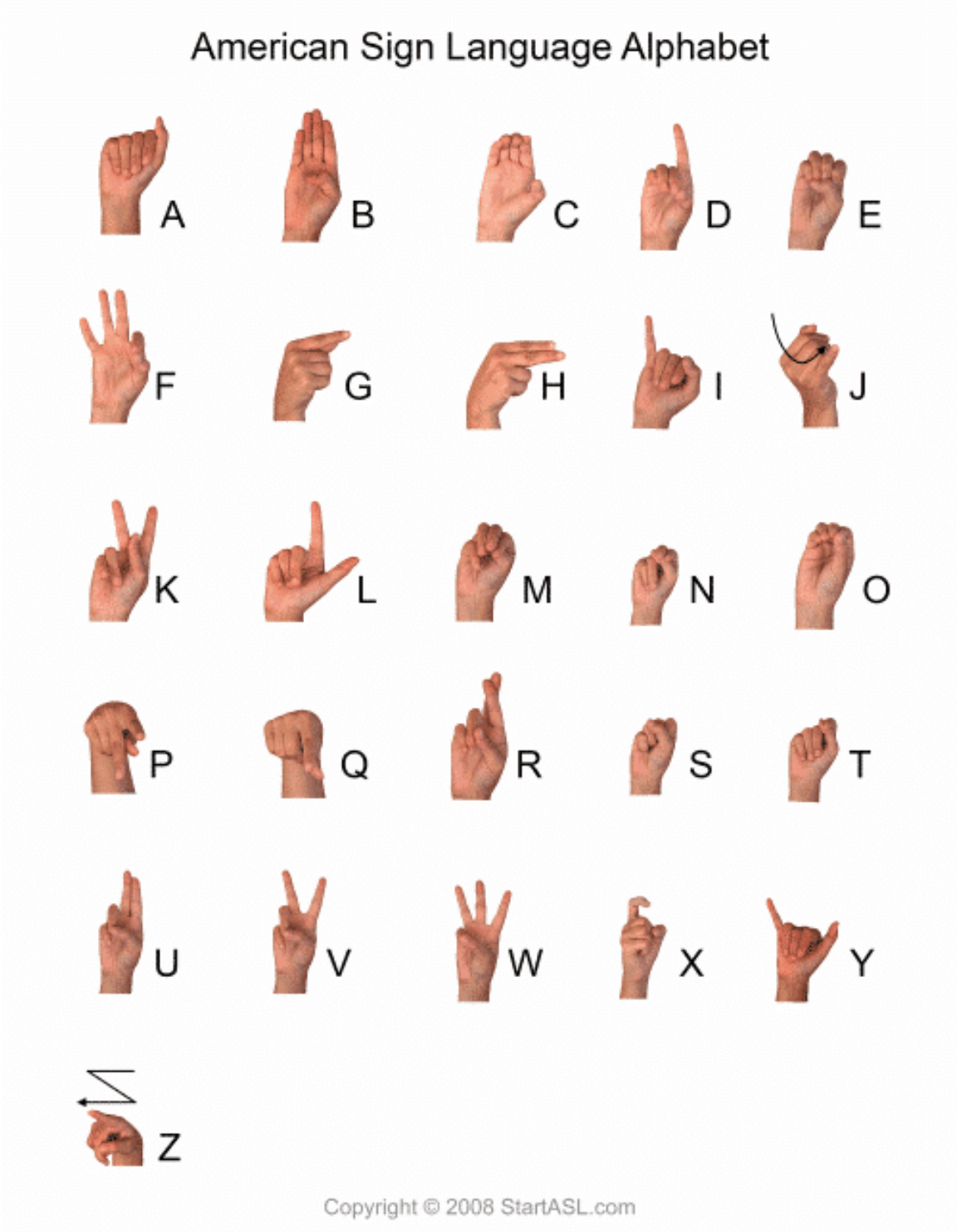


Figure 2.1: The American Sign Language Alphabet[30].

These five parameters must be considered when performing any sign in ASL because if one of them changes, the entire meaning of the sign can also change.

2.1.2 Technology and Sign Language

Members of the DHH community continue to face a wide range of difficulties throughout day-to-day life due to the high significance auditory abilities have in various situations [24]. This is a problem that researchers have striven to solve in a number of different and creative ways. As research has advanced, a number of software and hardware tools have been created to help make communication and education easier within the DHH community.

Historically some common methods of education for sign language users included face-to-face learning methods, in which an experienced signer demonstrates ASL hand gestures for a student to replicate, along with literature which displays appropriate gestures matched with their corresponding meaning [19]. However, As technology has advanced, there has been excitement among researchers in the area of sign-language technologies, prompting those in the field of Human Computer Interaction (HCI) to investigate its future potential [5]. Many of these tools involve detecting performed sign language gestures and outputting a useful response. Some applications which use this type of technology include gloves which can translate sign language gestures into audio and text, Dialogue systems which use image recognition, and educational tools using gesture recognition in VR [3, 10, 23]. The following sections outline some of the more notable tools that have been created with the aim of facilitating education and communication within the DHH community.

2.1.2.1 ASL Translator Glove

Just as phone apps exist to translate spoken words into other languages, it is also useful to translate sign gestures into their textual and auditory representation. UCLA researchers successfully developed a glove which achieves this at a low cost. Figure 2.2 shows an image of the glove translating an ASL gesture. The glove is able to translate

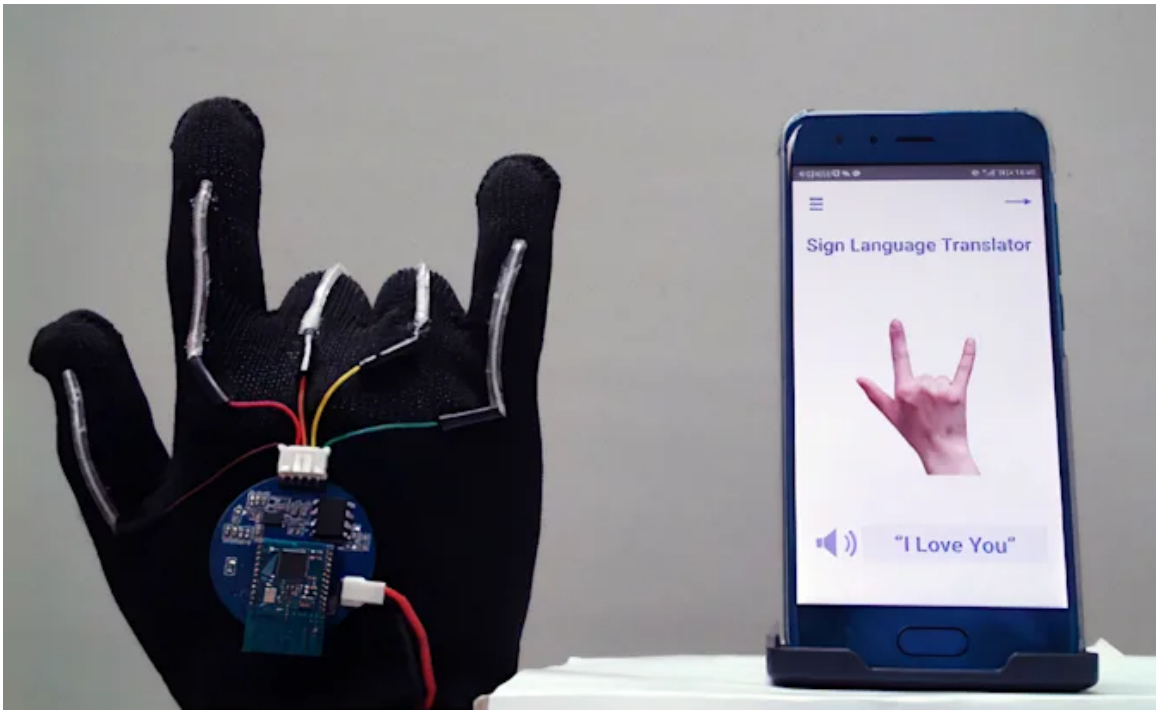


Figure 2.2: Motion sensor glove developed by researchers at UCLA which can translate signs into images and text [3].

660 ASL gestures in real time at 98.63 accuracy and at a rate of one word per second. The researchers who created this glove also tested capturing data from face sensors to detect facial expressions that are a part of ASL signs.

While the glove is able to translate gestures fairly efficiently, the response rate is a bit too slow to be used as an effective communication tool for experienced signers. Critics in the DHH community have stated that the glove would instead be useful in helping more people learn sign language on their own. As such, this glove is an example of a tool useful for sign language education.

2.1.2.2 Multi Modal Sign Language Kiosk

Some researchers have chosen to focus more on User Interfaces that would be comfortable for those in the DHH community to use. One such project is a multi-modal dialogue system that can be used within information kiosks for the deaf [10]. Developed by researchers in Switzerland, Turkey, and the Czech Republic, this application

uses cameras to recognize hand, facial, and body gestures, and respond with an animated interface as illustrated in Figure 2.3. As another input option, the interface also consists of a touch screen.

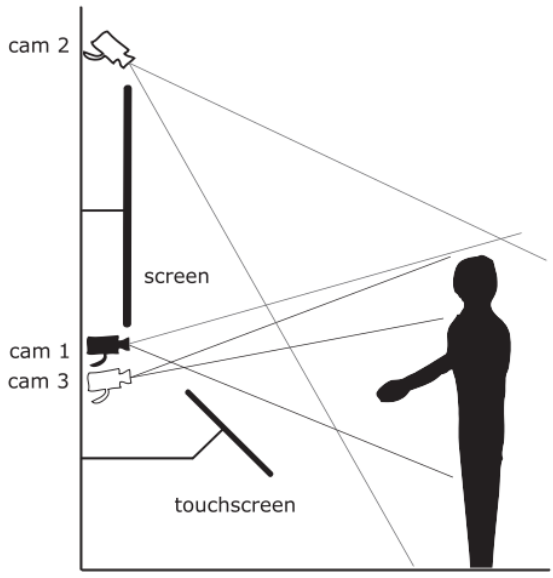


Figure 2.3: Setup for an informational kiosk capable of sign language recognition [10].

One major benefit of this application is that it does not involve the user to be equipped with any sensors or other hardware in order to use it. Encumbrances such as these can reduce freedom of movement and make for a less comfortable user experience. Also, this application gathers enough data to capture full body movements and facial gestures, which are an essential part of sign language. However, as with other gesture recognition methods that use image processing, there can be several restrictions based on lighting, colors and textures present, and occlusion. However, this project is a good example of a tool designed to ease communication difficulties in the DHH community.

2.1.2.3 ASL Educational VR App using LEAP Motion

While some research projects have shown to have potential use as an educational tool, such as the translating gloves, others have chosen to directly target an audience

in sign language education due to the significant improvements they could provide. One such project is an ASL educational VR app developed by researchers at the University of South Florida [23]. Using the LEAP motion sensor in conjunction with the HTC Vive VR headset, the application successfully detects every letter in the ASL alphabet with an accuracy of 98.33 percent. To address educational concerns, the UI of the application places users in a virtual classroom-like environment that enables the user to select which ASL gesture they would like to learn and presents a poster outlining how to perform each gesture as shown in Figure 2.4.

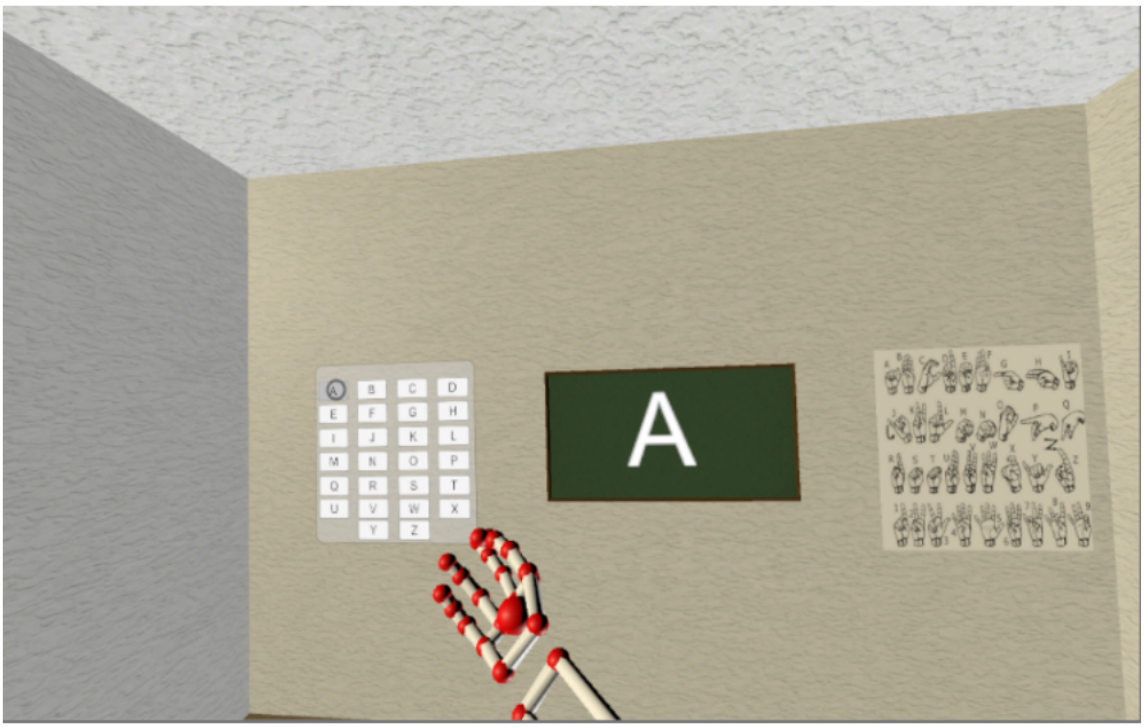


Figure 2.4: ASL Educational VR App which utilizes a classroom-like atmosphere in the UI [23].

While this application does limit the necessary peripheral devices to just a headset and sensor, the lack of data coming from the fingers does limit the total gestures that can be performed due to occlusion. For this same reason, gestures which are similar to each other (such as “M” and “T”) were incorrectly classified in this application.

2.2 Virtual Reality

2.2.1 Overview

As stated in Chapter 1, Virtual Reality (VR) works by replacing certain sensory stimuli of the user with artificial ones generated through software and hardware. The sensory stimuli that is most commonly replaced through VR is that of vision, but there are many instances in which other senses are manipulated as well. By replacing or simulating human senses, VR can effectively "trick" the senses of a user, which can allow them to experience a variety of different things that they could not in the real world. While these experiences may be simulated, they are sometimes close enough to reality such that they can they influence a user's behavior in the real world as well.

One of the most common ways to experience VR is through a VR headset such as HTC Vive (shown in Figure 2.5). A VR headset displays virtual world data from a computer or smartphone to the user, and tracks head movements so that the user can navigate through the virtual environment. Headsets are sometimes paired with controllers as well to enable the user to further interact with the virtual world. While VR is often associated with gaming, there are several other types of applications that have been developed using it, such as those involved in cinema, healthcare, socialization, prototyping, psychology, and robotics [14].

2.2.2 Application Development

2.2.2.1 Overview

In regard to software, the heart of any modern VR application is the Virtual world generator or (VWG) [14]. As the name implies, the main role of the VWG is to take in input data from the real world, and use it to create a virtual world which the user can interact with intuitively. The diagram in Figure 2.6 shows how a standard VWG system operates [14]. Inputs into a VWG system could include the motion tracking sensors within a VR headset, Controller inputs, and other peripheral device inputs. The output is a rendered world which can be interacted with using one or

Figure 2.5: The HTC Vive is one of many VR headsets available commercially.

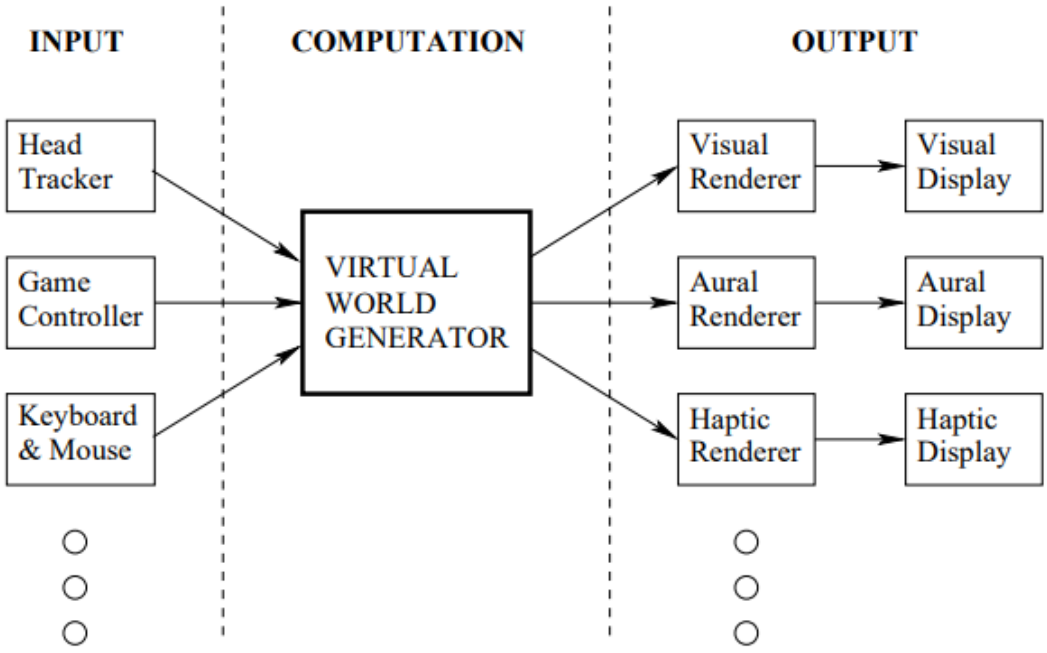


Figure 2.6: The VWG takes in input data from the real world, and uses it to create and display a virtual world that the user can interact with intuitively [14].

multiple artificial sensory stimuli. The type of world generated by the VWG can vary greatly based on user preferences and the intended audience of the application being developed. For example, the virtual world can be completely synthetic, meaning all elements of the virtual world are computer generated, or it can be a duplicate of the real world generated using various cameras and computer vision techniques. Also, the laws of physics within the virtual world can either be more closely related to those of the real world, or can be a warped version allowing the user to accomplish things that would normally be impossible.

Naturally, in order to enable the user to effectively interact with the world generated by the VWG, the appropriate input hardware must also be selected. Many VR headsets come equipped with controllers to interface with the virtual environment. However, there are a wide range of other options to replace or expand upon this including motion tracking gloves (explained further in Section 2.3), treadmills, or simply data from motion tracking tools such as the Microsoft Kinect [14]. Some have even developed custom input mechanisms such as with the Bicycle VR app shown in Figure 2.7. Depending on the intentions of the application it is important to select the appropriate hardware and software tools in order to create a user interface that will be acceptable by the audience.

2.2.2.2 Software Development Tools

There are a number of toolsets available for developing VR applications that come equipped with a VWG and an intuitive interface to handle concerns such as physics, interaction, lighting, graphical rendering, etc. Tools such as these are very popular among indie VR developers since all the tools they need are readily available to facilitate the development process. On the other hand, developers also have the choice to build their VWG from scratch using low-level SDKs. This approach does give the developer more power to modify and optimize various parts of the VR application as they see fit, but it can also greatly increase the implementation time and resources[14]. Many game development studios choose this route by developing their own proprietary

Figure 2.7: A custom VR system connecting a bicycle to a VR headset [14].

game engines [35]. In regard to this thesis, the former approach of VR specific toolsets is more applicable. Two of the most popular tools which fall under this category are Unity and Unreal Engine. The application presented in this thesis is developed using the Unity engine as it has a default compatibility with the API which accompanies the motion tracking gloves. Further information regarding both development engines is presented in the following sections.

Unity Unity is currently the leading platform in developing real-time 3D and 2D applications. In 2020 over 50 percent mobile, PC, and console games were made with Unity [39]. Founded in 2004, the creators of unity had a goal of making game development more universally accessible to potential users. In achieving this goal, developers have seen Unity to be more intuitive and easy to use then other development engines. Part of the reason for this is the large number of online tutorials that are available regarding Unity. Unity also supports a larger number of platforms than other engines, enabling it's applications to reach a wider array of audiences. While Unity does surpass others in regards to intuitiveness and universal compatibility, it's

offerings regarding high quality graphics can be seen as inferior when compared to some competitors. However, this is also an area in which unity is improving. Unity uses C sharp as it's primary programming language for development [35].

Unreal Engine Unreal Engine was founded in 1998 by Tim Sweeney, who also founded the popular game development company Epic Games [35]. One of the most well known and popular games made using Unreal Engine is Fortnite, which generated a revenue of one billion dollars as of January 2019, and had an audience of 200 million users. A major benefit of Unreal Engine over it's competitors is the high quality of graphics it offers. Unreal engine also provides a number of tools by default, which would instead be installed as extra plugins in other game engines such as Unity. A trade off for this, however, is limited offerings regarding possible platforms to develop on, and also a more complex development process. Unreal Engine also offers all of it's development features for free without any limitations, unlike some other development engines. Developers using Unreal Engine can develop games using either C++ or Blueprint Visual Scripting as their programming language [35].

2.2.2.3 Hardware Components

Almost any modern VR system on the market today will include a VR Headset and a set of controllers as a form of user input. However, as mentioned in Section 2.2.2.1, there can be several variations to this, which introduces other types of input hardware that can be used. Given the variety of input hardware that can come available with a VR system, it is most useful to describe the fundamental hardware components that make up any modern VR setup. These can be categorized as Displays, Sensors, and Computers, and are presented in the following sections [14]. The image displayed in Figure 2.8 also shows the disassembly of a popular VR headset called the HTC Vive, and can be used as a visual reference for the components which the following sections explain.

Displays: As mentioned in Section 2.2.1 VR systems involve providing artificial stimuli to certain human senses in an effort to replace stimuli provided by the real world. The devices which provide this artificial stimuli are called displays. While the term "display" may suggest a purely visual stimuli, this term can also be used to describe stimuli for the other senses such as auditory displays or haptic displays. However, in most modern VR systems, visual displays are the most important.

A common way of creating a visual display in VR is by combining a smartphone with a compatible headset which brings the phone screen close to the eyes, and uses magnifying lenses to bring the screen into focus. Several VR companies also use custom displays which use the latest LED display technology incorporated into smartphones. More advanced VR applications, such as CAVE systems, use large-panel displays or a combination of digital projectors and mirrors. Haptic displays can be achieved through methods such as vibratory feedback or resistance in response to user input. Auditory displays can be achieved through general speakers emitting sound or through more advanced methods such as bone conduction in which the skull is vibrated propagating waves to the inner ear [14]. In Figure 2.8, the visual displays of the HTC vive headset are made up of two AMOLED panels and two Fresnel lenses which are shown in the bottom-center of the image. Some versions of the HTC Vive also feature a pair of headphones, but the version presented in this image does not [17].

Sensors: Sensors are devices which are used by VR systems to extract information from the real world. In order to successfully apply visual and auditory stimuli, it is important for VR headsets to track both the position and orientation of the head. Orientation tracking is achieved through the use of sensors known as IMUs or inertial measurement units. Using a gyroscope combined with an accelerometer and occasionally a magnetormeter, IMUs measure their own rate of rotation (also known as angular velocity) and integrate them over time in order to estimate the change in orientation.

Cameras are used to track position of the head in most VR systems, and in some

Figure 2.8: The individual components of an HTC Vive headset displayed after a disassembly [17].

cases the positions of the eyes, hands, and other parts of the body. This is achieved by using features in an image as reference points to narrow down the possible positions for moving objects. Sometimes special markers or objects are placed on users or around a scene in order to obtain more reliable tracking data, but efforts are being made to try and reduce this need. Depth cameras can also be used, which project infrared lights on a scene and measure the distance between the lights and an infrared sensor. One well known device that uses this method is the Microsoft Kinect.

In Figure 2.8, the sensors of the HTC vive headset are shown on the right hand side within a network of orange ribbon cables. Connected to these cables are a number of photodiodes, which are sensors used to receive infrared light from base stations mounted in the play area. These enable a connected computer to calculate

the headsets position and orientation during operation [17]. Section 2.3 goes into more detail regarding how various sensors are used to perform motion tracking.

Computers: As explained in Section 2.2.1, the core of modern VR applications is the virtual world generator or VWG. The primary role of the computer in a VR setup is to execute the VWG. Sometimes an external PC may be used as this computer, and be connected to the headset through either a wired setup or, in some newer cases, a wireless one. Other times, the computer may just be a smartphone that is placed within the VR headset. The necessary costs and complexity of the VWG are important considerations when selecting the type of computer that will be part of the VR system. Given the potential computing power and modularity of PCs vs. smartphones, these are typically the better choice for more advanced VR systems. Advancements continue to be made in Graphical processing units (GPUs) which can be an important foundation for VR as performance demands increase. Headsets also commonly contain microcontrollers which gather information from the sensors of a VR system and send them to the computer, and display interface chips which enable the conversion of input video to display commands [14]. In Figure 2.8 a microcontroller which performs these operations along with several other functions is shown in the lower center of the image [17].

2.2.2.4 Development Challenges

VR Sickness: VR sickness has been and continues to be one of the most major hindrances towards widespread adoption of VR technology. VR sickness can be defined as a form of visually induced motion sickness that occurs when using VR systems, or any other uncomfortable experiences that occur when taking part in VR. Visually induced motion sickness refers to a type of motion sickness that results from being exposed to a stimuli that tricks the brain into thinking movement is occurring when it actually is not. This stimuli exposure is known as apparent motion, or vection when caused by simulators or VR. Vection occurs as the result of a negative sensory conflict

in which the vision sense detects that the body is acclerating while the balance sense (or vestibular sense) detects that the body is motionless. This can occur in VR when the user moves in the virtual environment using a controller instead of moving in the real world [14].

A simple way to reduce or remove vection is to reduce or remove visual stimuli that may indicate that the user is moving when they are not. This can be done by placing the user within a virtual container, such as a cockpit of a plane, which would block most outside movement, or just limiting any movement within the virtual environment. If movement must be performed in a virtual environment, there are also a number of other methods to reduce vection such as reducing the amount of time in which the visual stimuli is exposed, making the visual stimuli more realistic to help convince the brain that it is actually moving, and incorporating distractions which would help the user avoid the visual stimuli such as shooting enemies. Repeated practice is also another way to reduce sensitivity towards vection [14].

Aside from vection, there could be a number of other causes to VR sickness that are difficult to determine. Some of the causes may be due to previous activities which would induce such experiences, such as staying up too late or not eating enough. There could also be some users of VR which are more susceptible to VR sickness than others which causes varied results. Excessive latency being present in a VR system can also be a cause of VR sickness, as this can cause delays in perceived movement vs. actual movement, resulting in a sensory conflict. However, most modern VR systems do not have excessive latency issues thanks to advancements in motion tracking, graphical processing, and displays. Determining the root cause of VR sickness symptoms is an effort that continues to be performed by researchers in the field [14].

Uncanny Valley: While realistic VR experiences are desirable in many cases, excessive realism when applied to virtual representations of human beings, or avatars, can invoke feelings of discomfort and uneasiness among observers. This phenomenon is known as the uncanny valley. Because of this, developers try to avoid making

avatars look exactly as people do in the real world, and instead try to find an ideal balance between realistic and unrealistic appearances [14].

There are a few ways to bypass the difficulties of the uncanny valley. One popular way is to use animation to create avatars with unique appearances that are close enough to humans to at least be somewhat relatable. This can be done in subtle ways such as enlarging the eyes of characters, or through more exaggerated methods such as created characters that are dramatically different from humans. It is possible to create avatars that are so realistic that they bypass the effects of the uncanny valley, but to do so would likely cost an excessively large amount of time and resources that would seem infeasible to many development teams [28].

2.2.3 Accessible User Interfaces in VR

2.2.3.1 Overview

When designing software applications that could likely be used by those with a physical or mental disability, such as the application presented in this thesis, it is essential to understand the considerations that must be made when targeting such users, and what can be done to make the their experience as comfortable and intuitive as possible. Not only must the developer consider how the user is able to provide input to the interface, but they must also ensure that any output is sufficiently understood by the user given their limitations. VR software is an ideal choice for developing such applications, since it can be designed to accept nearly any ordinary human sense as an input whether it be sight, sound, or touch. Because of this, it is no surprise that research continues to be performed on effective interfaces in virtual environments for those with a wide range of disabilities. The following section presents a summary of various VR applications that have been created using this research.

2.2.3.2 Accessible VR Applications

One particularly unique research project regarding accessibility in VR is the Canetroller created by Microsoft [41]. Designed for the visually impaired, the Canetroller is a type

of VR interface device used with the HTC Vive headset which provides haptic and auditory feedback to the user, enabling them to navigate an environment containing virtual obstacles. Figure 2.9 demonstrates and example of its usage. The Canetroller was meant to provide a virtual version of a white cane, which is a popular tool used by the blind and visually impaired to navigate environments in the real world. While users of the Canetroller indicated that the multi-modal feedback of the device was useful for determining virtual boundaries, many also thought that the auditory feedback was not specific enough, making it difficult to determine what types of objects they were encountering [41]. Microsoft also developed an application in this area focusing on an interface using hand gestures as input [32]. While this application does not focus on a specific disability, it is apparent how this could be useful for those who use hand gestures as a primary means of communication, such as the hearing-impaired. As with many hand-tracking applications, Microsoft's tool utilized machine learning in order to achieve detailed poses during the hand tracking process, and also incorporated improvements to make the hand-tracking process more accurate and efficient. This application utilized both a VR headset and a Kinect device in order to track hand movements, and had issues dealing with occlusions due to the un-encumbered nature of the hand-tracking [32].

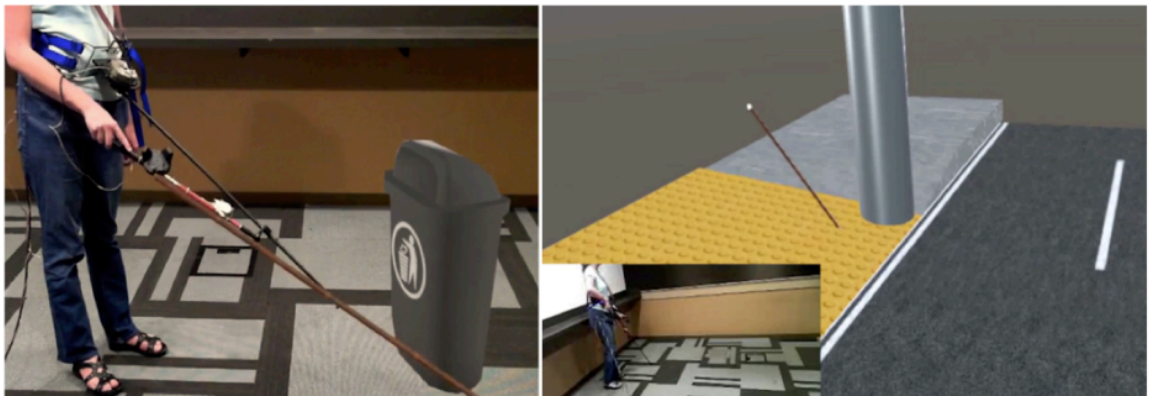


Figure 2.9: A demonstration of how the Canetroller apparatus is used. The left image shows a user interacting with a virtual trash can, and the the right image shows a user interacting with a virtual street crossing. When certain encounters are made with virtual objects, vibratory feedback is provided to the user [41].

While there have been many VR applications directed towards physical disabilities, there are also several applications directed towards mental limitations. The University of Texas at Dallas has researched various ways to utilize VR for addressing social anxiety and aggression in teenagers afflicted with disorders such as Autism Spectrum Disorder (ASD) and Asperger's Syndrome [34]. In their VR applications, Teens socialize with avatars and participate in several different activities to improve their social behavior. Teenagers taking part in this program have reported that the virtual world is less threatening than the real one, and allows them to calmly address issues as they need to [34]. Researchers at the University of Florida have developed a similar application that focuses on assisting those afflicted with mental disabilities in their effort to find jobs in a professional environment [1]. This software is called "a virtual reality system for vocational rehabilitation of individuals with disabilities" or "VR4VR" and aims to teach those with ASD one of six vocational skills including cleaning, loading, money management, shelving, environmental awareness, and social skills. Many of the individuals tested found the software beneficial, though some thought that it was not very realistic, and the fact that they knew it was not real made tasks too easy [1].

2.2.3.3 Challenges

Based on research performed on already present VR applications directed towards the mentally/physically disabled, the current major challenges with these types of applications are the following:

- **Providing Adequate Feedback to the User** - As shown with Microsoft's "Canetroller" tool and hand-tracking application, having insufficient feedback to the user can be highly detrimental in applications directed towards the disabled. In order to achieve an immersive interface, it is important to target as many sensory stimuli as possible, and when doing so, ensure that the output is what one would expect given the situation.

- **Providing a Realistic Interface** - As shown with the VR4VR application,

not having a believable virtual environment can greatly limit the experience of the user, especially if the goal is to simulate real-world scenarios. Having a more realistic interface can also serve to compensate for the limited senses of the user by still providing an immersive experience. However, as explained in Section 2.2.2.4, one must also beware of the uncanny when attempting to design realistic VR applications.

- **Reducing Setup Effort** - Having an overwhelming number of tools present in order to generate a VR experience can decrease the overall accessibility of the application and make users with limitations less likely to participate. VR4VR had a significant setup effort which may have also contributed to the lack of realism within the application.

2.3 Motion Tracking

2.3.1 Overview

Motion tracking can be defined as the process of converting physical movements into digital data which can be interpreted and analyzed by computer software in real time [22]. Motion tracking is often used interchangeably with the term *motion capture*, and in many ways they are similar, but there are some key differences between the two which allow for them to be considered separate entities [18]. Perhaps the most significant difference between the two is that motion capture involves acquiring movement data to be used in post-processing, while motion tracking involves processing movement data in real time. Motion Capture is often used to construct animated movies with the aim of creating synthetic characters that have realistic movements. Motion tracking has also been used in a number of different applications including medical technology, robotics, and virtual reality [12, 13].

In order for Motion tracking to be successful, it is necessary to track every possible direction in which a target can move within a three-dimensional space. Each of these directions is called a degree of freedom of DOF, and any rigid object moving in 3D

space has six of them [14]. These six DOFs are shown in Figure 2.10 which uses a basic VR headset as an example motion tracker. Three of the DOFs refer to translational movements (front to back, left to right, up and down) while the other three refer to rotational movements (also known as yaw, roll, and pitch) [14]. Section 2.2 touched on some of the hardware used regarding motion tracking, but the following sections will go into motion tracking in more detail including the types of tracking that exist along with current related technologies other than VR headsets. Given the various considerations that must be made regarding motion tracking, particularly in relation to sign language gesture recognition, providing a more in-depth understanding of this process is necessary.

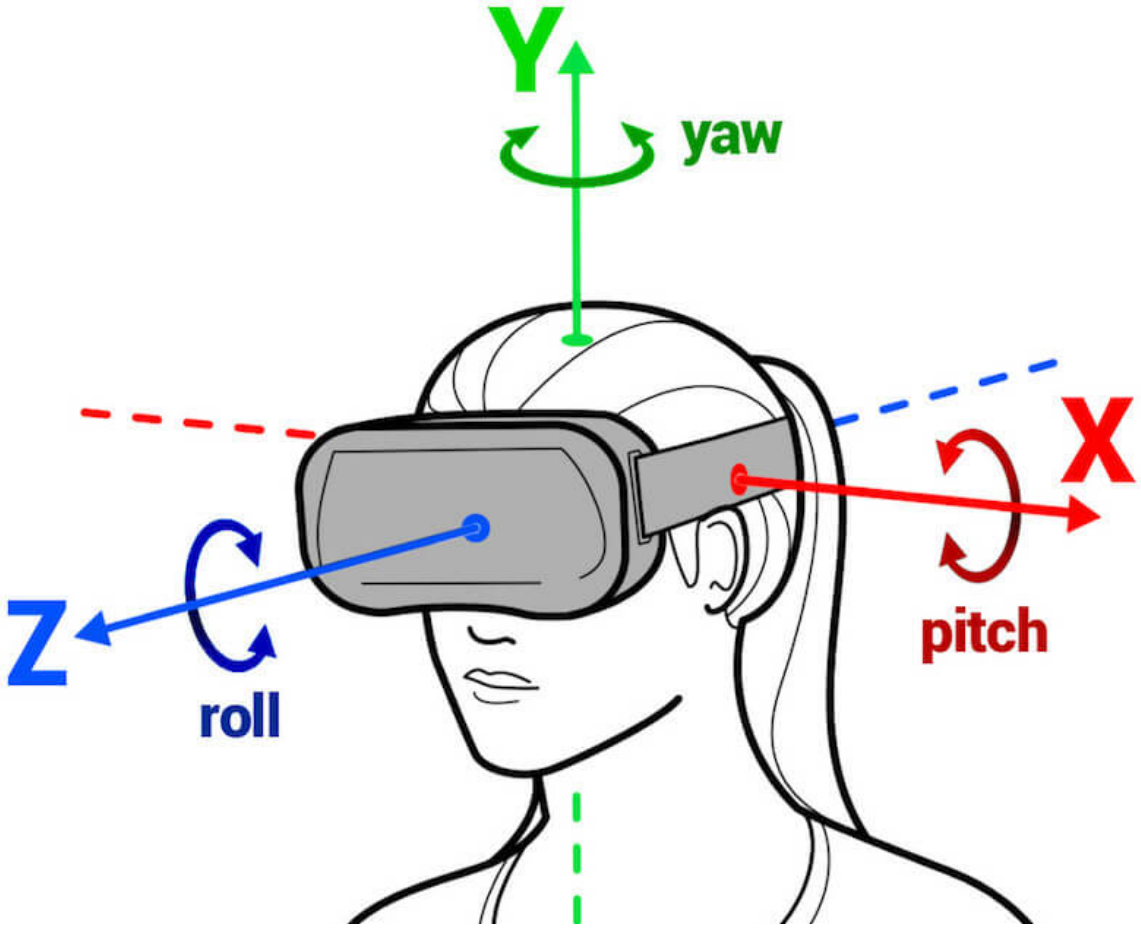


Figure 2.10: An illustration of the six degrees of freedom that exist with any object in 3D space, using the human head being tracked by a VR headset as an example [22].

2.3.2 Types of motion tracking

2.3.2.1 Optical Tracking

Optical tracking is a type of motion tracking which uses imaging devices to track motion. [22]. Many VR based optical motion tracking systems use depth sensing cameras in conjunction with reflective markers to perform motion tracking. These markers are captured by depth cameras and mapped to a 3D virtual space. The cameras which perform this tracking are often not restricted to visible light, but utilize infrared light instead so as to no let natural light distract the user experience [26]. The markers used in the process can be either passive or active [22]. Active markers feature computer controlled LEDs while passive markers do not. As such, active markers have a higher tracking accuracy compared to passive markers. However, active markers also require a power source or a computer connection unlike passive markers, which can reduce the immersive qualities of a VR experience.

The lighthouse system of the HTC Vive is an example of optical tracking which treats the headset and controllers as external markers [22]. In this motion tracking system, external lighthouses or base stations emit infrared (IR) pulses along with IR laser sweeps on the X/Y axis [16]. An array of IR-filtered photodiodes are also embedded within all items that require tracking, which in this case are the VR headset and controllers. When the photodiodes are exposed to the IR light from the lighthouses, their outputs are amplified and passed on to an ASIC (Application-specific Integrated Circuit) within the tracked devices which determines the location and orientation of the tracked devices within the area [16]. Some other optical tracking systems, such as the LEAP motion and Microsoft Kinect, do not require external markers to perform optical tracking, and instead use other techniques to achieve this [22]. More information regarding these motion tracking systems is presented in Section 2.3.3. A limitation with single camera optical tracking techniques is that there must always be a line of sight between the tracked object and the camera, which restricts movement [26]. As such multiple cameras are often integrated in systems which use optical

tracking [26].

2.3.2.2 Non-Optical Tracking

Non-optical tracking is a type of motion tracking which uses a collection of sensors rather than imaging devices [22]. Three well known sensors which have been widely used for motion tracking in cell phones and tablets are the gyroscope, accelerometer, and magnetometer [14]. As mentioned in Section 2.2, these three sensors combined make up what is known as an IMU or Inertial Measurement Unit, which is used to determine orientation in most modern VR systems. Modern technology has enabled IMUs to be developed as Micro Electro Mechanical Systems or MEMS for short. As a result the components of modern IMUs such as those in cellphones and VR Headsets are less than 1mm in length [14].

The gyroscope is the main component of the IMU [14]. This component works by measuring the angular velocity as we rotate about the X, Y, and Z axes. The sensing elements for each axis to be measured include two vibrating masses which are vibrating perpendicular to those axes [20]. If a rotation occurs on each axis, a force is exerted by the vibrating masses due to a scientific principle known as the Coriolis effect. These Coriolis forces are converted into electrical signals, which are then calibrated to produce an understandable output such as degrees or radians per second [14]. Figure 2.11 displays an example of a MEMS gyroscope. These gyroscopes provide angular velocity measurements at a certain frequency, every millisecond for example, and we can estimate the orientation of the object to be tracked by integrating these measurements.

Despite the accuracy of some MEMS gyroscopes, there is typically some discrepancy that exists between the actual orientation of a tracked object, and the orientation measured by the gyroscope [14]. This discrepancy is called drift error, and in order to perform optimal motion tracking, other sensors are typically needed to estimate and account for it. These other sensors are usually an accelerometer and magnetometer. The accelerometer handles the portion of drift error which corresponds to the and

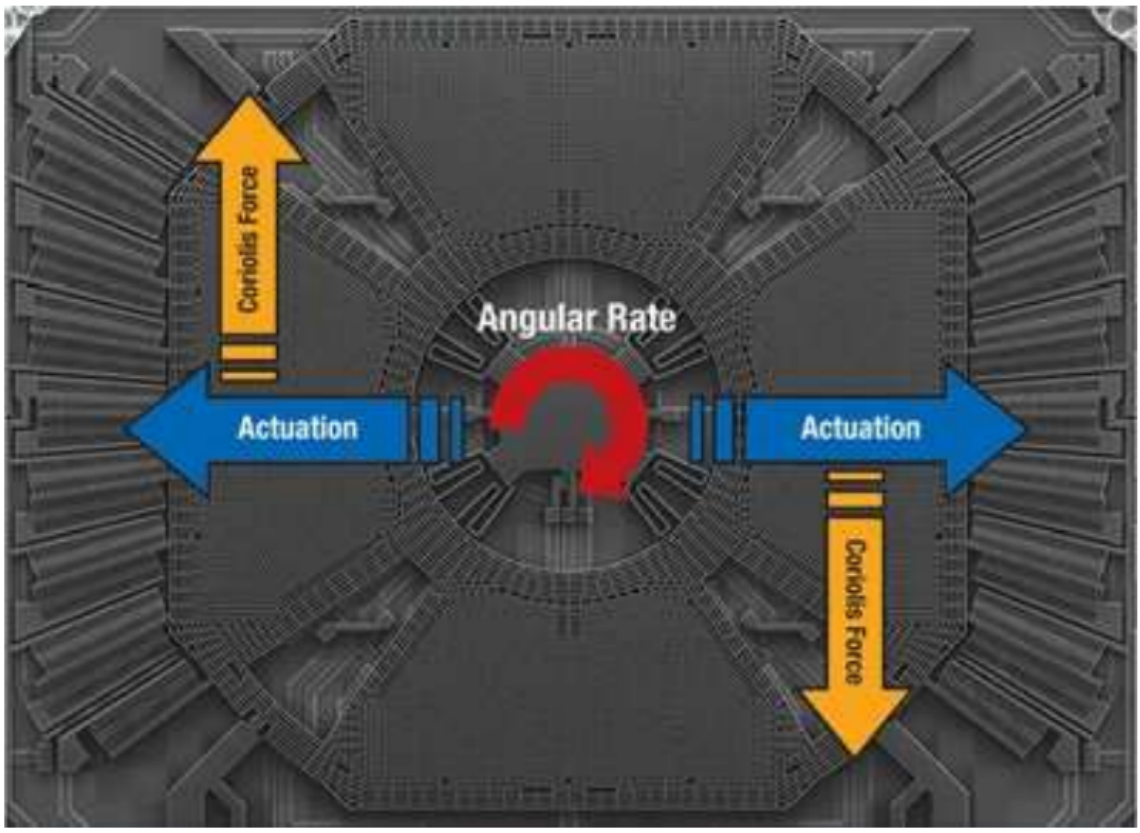


Figure 2.11: An image of a MEMS gyroscope with arrows overlayed to indicate how angular velocity is measured on a particular axis. Two masses on are vibrated on an axis to be measured, as indicated by the Actuation arrows, and if a rotation occurs across a perpendicular axis, Coriolis forces are exerted [14].

pitch and roll rotational axes (or the "floor" in a VR environment), which is also called tilt error. An accelerometer works by measuring linear acceleration along the X, Y, and Z axes. It typically consists of a mass attached to a spring which allows it to move along one axis as shown in Figure 2.12. A series of fixed plates surround the mass, and when acceleration occurs, the capacitance between the fixed plates and the moving mass changes. These capacitance values are then processed to gain a measurement of acceleration. Accelerometers help to minimize tilt error by helping to define where the "up" vector is located, or vector which is parallel with gravity. Since the up vector is always perpendicular with the pitch/roll axes (left/right vectors) in a world frame, it can be used as a reference point to determine where these axes are located in the actual world, which can they be used to help correct the sensor's interpretation of those axes when identifying it's own orientation. The accelerometer estimates the up vector by sensing the acceleration of gravity (9.8m/s2) and adding it to the acceleration of the body being tracked, which may be very low especially if the body is stationary [14].

While the accelerometer handles drift error for the pitch and roll axes, magnetometers handle drift error for the yaw axis, also known as yaw error [14]. This type of error can be described as a discrepancy between what direction the tracked object is facing in the real world and the direction detected by the motion tracker. The magnetometer works as a compass by measuring a 3D magnetic field vector that is used as a frame of reference similar to how the "up" vector is used with the accelerometer. A difference between a compass and a magnetometer is that the vectors the sensor outputs do not always point to true "North". This is essentially due to the fact that earth's magnetic field varies from area to area requiring certain offsets to be applied to establish what true north is. However, in regards to a motion tracking system, true north is typically not important, since we really just require a reference point to enable us to calculate our yaw error. Calibrations do need to be made though to ensure that the detected magnetic field lies on the horizontal plane, and to ensure that magnetic fields generated from other components such as buildings or circuitry

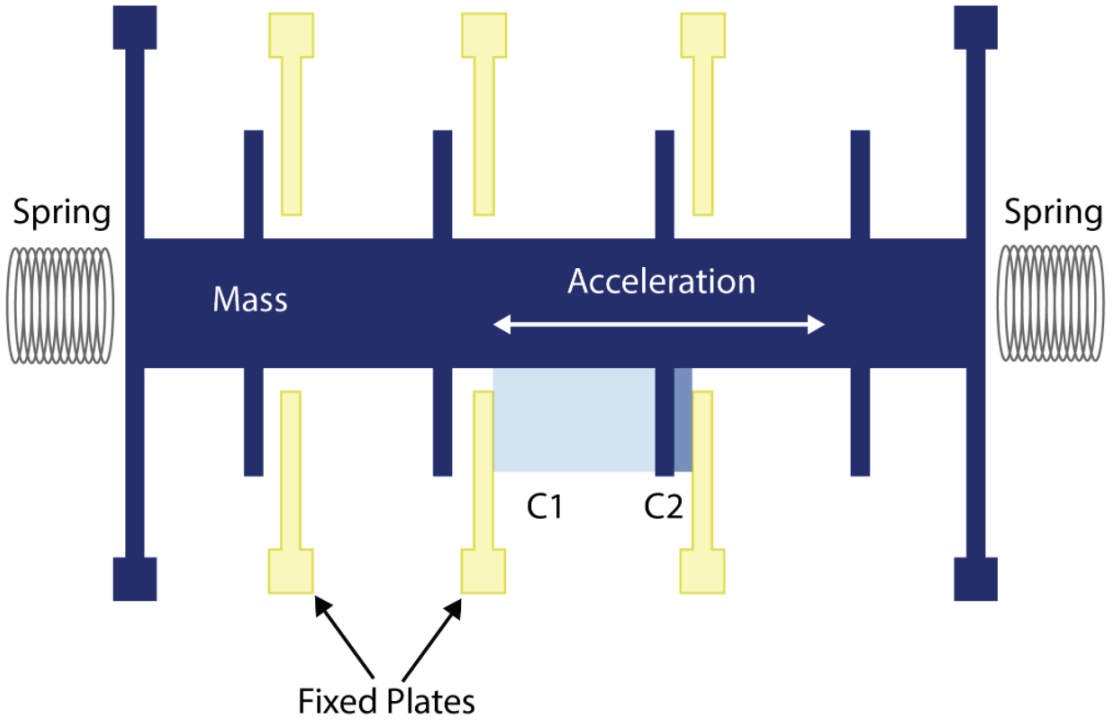


Figure 2.12: A diagram depicting the basic operation of a MEMS accelerometer. Two springs on opposite sides move a mass when acceleration occurs. Capacitance values are taken between the moving mass and a series of fixed plates, which directly correspond to the measurement of acceleration [15].

adjacent to the magnetometer do not negatively influence measurements [14].

While the IMU is a more popular approach to non-optical motion tracking, there are other approaches that have been experimented with as well. One such approach is known as acoustic tracking [18]. This type of motion tracking involves a series of emitters which emit sound that is received by one or more microphones at the location of the object to be tracked. The location of the tracked object is determined by measuring the time taken for the sound to travel from the emitters to the microphones, which then can give us the distance between the emitters and the microphones. By using multiple emitters and microphones, both position and orientation can be determined through this method. However, since temperature and air pressure can have an influence on the speed of sound, this type of equipment must also be calibrated before use or have built-in mechanisms to account for temperature and pressure during measurement. Mechanical tracking is another form of non-optical tracking that has been used. This type of motion tracking works by connecting to a reference point, such as a bone joint, and measuring the position of that point. Using trackers with haptic feedback, this type of tracking can measure the angles and distance between joints which can help determine position. Trackers like these have been used to construct full body suits which measure major joint positions as well as devices that fit around the fingers and gloves to track smaller scale movements [18].

2.3.3 Hand Motion Tracking

2.3.3.1 Overview

Motion tracking technology is often used for estimating the motion of multiple connected bodies relative to each other [14]. This may involve tracking eye movement relative to the head, head movement relative to the torso, or hand movement relative to the wrist. In fact, the entire human body can be described as a group of attached bodies which make up the body's skeleton. This group of attached bodies is called a multibody system, and the mathematical representation of the poses these bodies make relative to each other is called multibody kinematics [14]. Each body, or link,

in a multibody system is connected to its adjacent body by a joint, allowing one or more DOFs of motion between them. A common problem in motion tracking involves predicting the position and/or orientation of a link using only data from other adjacent links to which it is connected. For example, we may want to find the position of a fingertip using data from sensors on the adjacent bones of the finger, but not on the tip itself. This process is known as forward kinematics. A more difficult task is performing this process in reverse, in which we try to predict the positions of finger bones using only the finger tip, or the positions of an arm using only data from the hand. This process is known as inverse kinematics [14].

As discussed in Section 2.1, there are number of factors that go into sign language recognition, and hand tracking is certainly among the most important. The hand can be considered as a multibody system with each hand bone treated as a link in a chain of bodies to be tracked. There a number of different hand tracking devices available which each detect the position and orientation of hands in different ways. Some include only non-optical tracking by default, while others include only optical tracking, and some utilize a combination of both. The following sub section will provide details regarding a number of hand tracking options that are available along with a comparison between them. Each of these hand tracking options were researched in order to identify the option which would be most ideal for the sign language recognition application presented in this thesis. The option which was ultimately chosen was the VRfree hand tracking system by Sensoryx.

2.3.3.2 Comparison

LEAP Motion The LEAP Motion is an optical hand tracking device which consists of two infrared cameras paired with three infrared LEDs [36]. The LEDs shine infrared light upon the tracking area while the cameras pick up any light that is reflected back. Using this reflected light, and image of the hands can be generated. While the device itself has a small footprint, it provides a tracking window of approximately 2 feet away from the sensor with a field of view spanning a 140x120 degree area [37]. The

Figure 2.13: A user's hands being tracked by the LEAP motion device and the resulting skeletal figures provided by the accompanying SDK [37].

device can be used standalone or paired with a VR headset. As demonstrated in Figure 2.13, an accompanying SDK enables the generation a skeletal model based on the image captured by the device, which can be used in gesture recognition [37].

While the LEAP motion is capable of hand gesture recognition to an extent, and has even demonstrated limited sign language gesture recognition as discussed in Section 2.1, it still faces difficulties common among many optical trackers including occlusion and decreased reliability when in the presence of conflicting light sources [36]. Also, while the possible tracking area is large relative to the device, it is not large enough to allow for the wide range of movements that many gestures involve, especially those in sign language. A positive trait of the device, however, is that it provides optimal comfort to the user, as no tracking devices need to be worn by the user when performing tracking.

Vive Hand Tracking System The HTC Vive VR headset provides another method of optical tracking which utilizes its included cameras to capture images of a user's

Figure 2.14: An image showing ASL gesture recognition being performed using the Vive hand tracking SDK. The left-most image shows the skeletal model generated by the SDK while in an idle state. The center and right-most images show two ASL gestures being successfully recognized. (Figure by the author)

hands, similar to the Leap motion presented in the previous paragraphs [11]. Using the captured images, an accompanying SDK provides the capability to generate skeletal representations of the hands in a virtual environment. These tracking capabilities also extend across multiple headsets offered by HTC, and not just the original Vive headset. Depending on the number of cameras present on the headset, this solution also offers a larger tracking area and field of view than some competitors. The fact that no additional hardware is needed aside from the VR headset also gives the Vive Hand tracking system an edge regarding usability [11].

An early prototype of the application presented in this thesis utilized the Vive hand tracking system to recognize certain sign language gestures. Some results are shown in Figure 2.14. While the hand tracking did allow for the recognition of certain sign language gestures, the common issues of occlusion and light interference still inhibited this tracking method, limiting its sign language tracking capabilities. As with other optical tracking solutions, the lack of sensors that need to be equipped allow for a more comfortable user experience. Also, when testing this tracking method, the Vive cosmos headset was used which is unique in that it does not require external base stations for tracking. This helped to reduce necessary setup for the testing that was performed.

Hi5 Gloves The Hi5 Gloves are an example of non-optical motion tracking which uses 6 IMU sensors to track the orientation of the hand and fingers. The Hi5 gloves can also be combined with a optical motion tracker to enable location tracking using the HTC Vive Headset. A large advantage of these gloves over competitors is their ability to provide haptic feedback to the user. This can provide a much more immersive experience in certain scenarios, especially in VR. However, despite the gloves sensors being able to track some directions of finger movement, they do not fully track finger spreading, or the process creating and removing gaps between the fingers. This type of tracking is essential for sign language recognition since there are a number of gestures which rely on this capability. Also, the vive trackers which must be attached to the gloves to enable location tracking are rather bulky, causing more restrictive movement when forming gestures.

VRFree Gloves The VRFree Glove system by Sensoryx has a number of features which are ideal for hand motion tracking when compared to competitors. This tracking system consists of two gloves which each have multiple sensor types including multiple IMUs along with a head module which can attach to a VR headset [25] as shown in Figure 2.15. The head module communicates with wrist module sensors on the gloves to track their location in a virtual space while other sensors track the orientation of the fingers within the hands. So the VRfree gloves can track both position and orientation of the gloves without the need of external tools like third party optical trackers. This combination of sensors also present the common problem of occlusion which is present in most optical motion tracking solutions. The gloves also come with a comprehensive API which allows for granular tracking data to be retrieved from the sensors as needed. Because of these features, the VRfree gloves were chosen as the motion tracking solution for the sign language gesture recognition application presented in this thesis.

Of course, these gloves like several other motion tracking systems do come with flaws. The main flaw in relation to sign language gesture recognition is the more

Figure 2.15: An image demonstration of the tracking capabilities of the VRfree glove system. A user plays a virtual piano using the gloves which is one of many possible activities that can be performed in the demo application provided with the glove SDK [25].

restrictive movement and reduced comfort which comes with wearing gloves vs. not wearing gloves. Ideally comprehensive motion tracking could be achieved without the user needing to equip any hardware, but this is currently not a realistic expectation given the current state of motion tracking technology. The VRfree gloves also do not provide haptic feedback which may be desirable in certain scenarios, but not in the application presented in this thesis.

2.4 Machine Learning

2.4.1 Overview

Machine learning is a term used to describe any program or system which can generate predictions from unknown input data using a mathematical entity called a model, which defines the relationship between the input data and the possible predictions [8]. In the simplest case, a model could be a linear function, while in many cases it is a

complex algorithm consisting of several steps and parameters. By building, or training, a model using example data which has the same traits as the unknown data, the model can "learn" how to recognize these traits in future, unknown data sets. Machine learning can be separated into two categories: supervised learning and unsupervised learning. To train a model using supervised learning, we provide a set of labels which consist of possible predictions that can be made, and a set of features which are the traits the data has that can be used to help predict those labels. The model then learns which feature values best represent each label, and can then use new feature values to predict unknown labels within the same range. Unsupervised learning involves finding patterns in a dataset, usually without knowing any corresponding labels. The project presented in this thesis uses supervised learning, so this method will be the primary one discussed in this section.

Machine learning is a useful practice for a number of different reasons. For one, it reduces time spent programming and necessary code. This is because rather than programming a series of rules and restrictions that define an application, one can simply provide a a set of examples that abide by these rules, and feed them into a model which can learn these rules and restrictions based on features of the examples [8]. Also, machine learning makes applications easier to customize. For example, if an application needs to be changed to cater to a new group of users, a new data set could just be provided representing these user's needs with little or no changes being needed to the model [8]. Finally, and perhaps most importantly, machine learning helps us to solve problems which we do not know how to solve by hand, or would be excessively complex when solved by hand. Relating a set of features to a set of labels can result in rather complex formulas, but with machine learning, these formulas are generated by the model, and only require examples of inputs and outputs to do so [8]. The following sections will outline some key concepts regarding machine learning which were used to build the gesture recognition application presented in this thesis.

2.4.2 Training a Model

To provide more detail as to what a model is, it is easiest to start by looking at the simplest case, in other words a linear function. Figure 2.16 shows an chart taken from Google's machine learning crash course [8]. This graph shows a number of points plotted which relates the number of chirps per minute a cricket makes to the temperature of the environment in which the cricket is located. Using this data, we could create a model that takes the chirps per minute as an input feature, and outputs a predicted temperature, or vice versa. In this case, our model can just be a line which approximates the relationship between the input and the output, with the function being the basic equation for a line: $y = mx + b$. In this function y would represent the temperature (which we are trying to predict), m would be the slope of the line, x would be the chirps per minute (our input feature), and b would be the y-intercept. This works as a model because we could just plug in a value for our feature x along with the known slope and y intercept, and get a predicted output value. However, when applying machine learning conventions, this function would look like the following: $y' = b + w_1x_1$. In this case, y' would be our predicted label (the temperature value), b would be the bias (or y intercept), w_1 would be the weight (which can be related to the slope in the line formula) and x_1 would be the feature. This type of model is known as linear regression. A more complex version of this model could also have multiple features which each have separate weights. In this case the function for the model can be written as the following: $y' = b + w_1x_1 + w_2x_2 + w_3x3$. Training a model is essentially the process of finding the best values for the weights and bias in this function such that our prediction is accurate. Or in our example, finding the line that best fits with the trend of our input data so as to accurately predict new, unknown data which has the same features [8].

Most training processes for a model can be seen as an iterative trial and error process such as the one outlined in Figure 2.17. Each iteration starts by first plugging in one or more features into the model function along with starting values for the

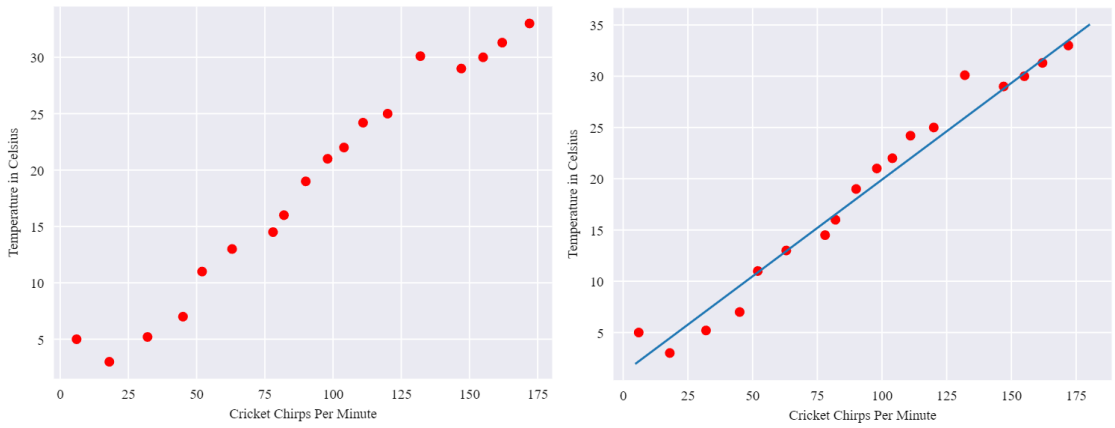


Figure 2.16: An example of a simple model to predict environmental temperature based on a number of cricket chirps. The left side shows the data without a model applied and the right side shows the data with the model. In this case the model is essentially just the function for a line: y=mx+b [8].

weight and bias parameters [8]. A prediction is then made using these values of what the corresponding label is. This predicted label along with the actual label are then plugged into a loss function. The purpose of this loss function is to determine how far off our predicted label is from the actual label which corresponds to the input features. Using the computed loss, new values for the weight and bias parameters will be generated which will reduce the loss during the next iteration. This process is repeated until parameter values are found which produce the lowest possible loss. For a linear regression model, a popular loss function used is the mean square error (MSE) but there are a variety of other loss functions that can be used depending on the complexity of the model [8]. The process of generating new parameter values to reduce loss relies on a few other values called hyperparameters which can be adjusted to optimize the learning process. Examples of hyperparameters include the learning rate, which essentially determines how large of an adjustment should be made to the model parameters during each iteration, and the number of epochs, which determines how many iterations should be executed during the training process. More information regarding the specific processes that occur in each stage of Figure 2.17 can be found in [8].

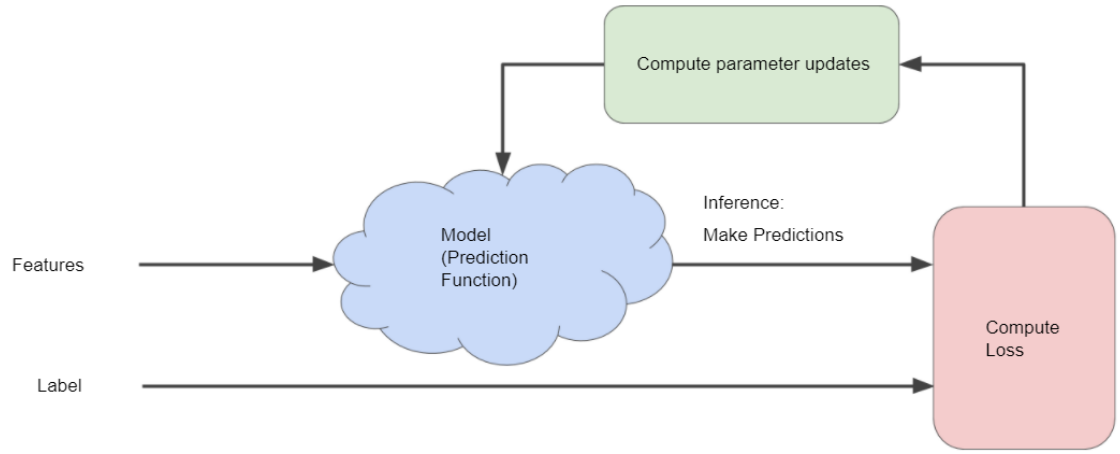


Figure 2.17: A diagram showing the typical process when training a supervised machine learning model [8].

Even if a model is trained successfully with a low loss, issues could still occur during the inference (prediction) phase due to a number of reasons. one possible cause for this is overfitting, which describes a scenario in which a model is unable to make accurate predictions on new data due to having parameters which specifically correspond to peculiarities within the training data [8]. The main reason this happens is because a model is more complex than it needs to be. By simplifying a model, we can reduce, or completely eliminate, any peculiarities which may not be present in unseen data, enabling the model to make better predictions. One way to simplify models is through a process called regularization. Regularization is essentially a way of minimizing the complexity of model by modifying parameters which contribute to that complexity. For example, L2 regularization encourages the values of weights to be close to zero, as weights with a large absolute value can contribute to the complexity of a model. Ensuring that appropriate features are chosen is also a good way to reduce complexity of a model. Machine learning developers generally spend a great deal of time performing feature engineering, which is the process of converting raw feature data into a format which can be easily manipulated and understood by a model. Also, testing the model against the same data set that was used in training can cause unexpected results when testing against unseen data. It is best to use a

test data set that contains different data from the training data set when evaluating the performance of a model [8].

2.4.3 Neural Networks

A common machine learning problem is known as classification, in which the labels can be seen as the set of classes we are trying to predict based on given input data [8]. For example, the ASL hand gesture for the letter "A" would be considered one of these classes (or labels). The features would be various characteristics of each class which distinctly separate them from the other classes. For hand gestures, these could be finger locations/rotations, movement parameters, etc. Because of this, gesture recognition can be seen as a classification problem. Sometimes a classification problem can be solved using a linear model, but in many cases the input features and output labels do not have a linear relationship, and other methods need to be utilized. One method to resolve nonlinear relationships such as these is by combining multiple features into one synthetic feature known as a feature cross, but this still cannot solve some of the more difficult nonlinear classification problems. In the most difficult cases, a common solution is using neural networks [8].

A neural network is a type of model used in machine learning which is composed of layers which consist of simple connected units, or neurons [8]. As the name suggests, this type of model is inspired by the network of neurons located within the brain of an organism. Figure 2.18 shows the basic structure of a neural network. The structure starts with an input layer of neurons, which is used to pass input data to be classified, or in other words, to be used to make predictions. Then, there are one or more hidden layers present. Each of these hidden layers will transform the data in such a way so as to better relate it to the output labels. In some cases, each of the neurons in these hidden layers will simply contain a weighted sum of the input features of the same format as the function for a basic linear model presented in the overview of this chapter: $y' = b + w_1x_1 + w_2x_2 + w_3x3$. However, in order to resolve non-linear relationships between the inputs and outputs, it is also necessary to transform the

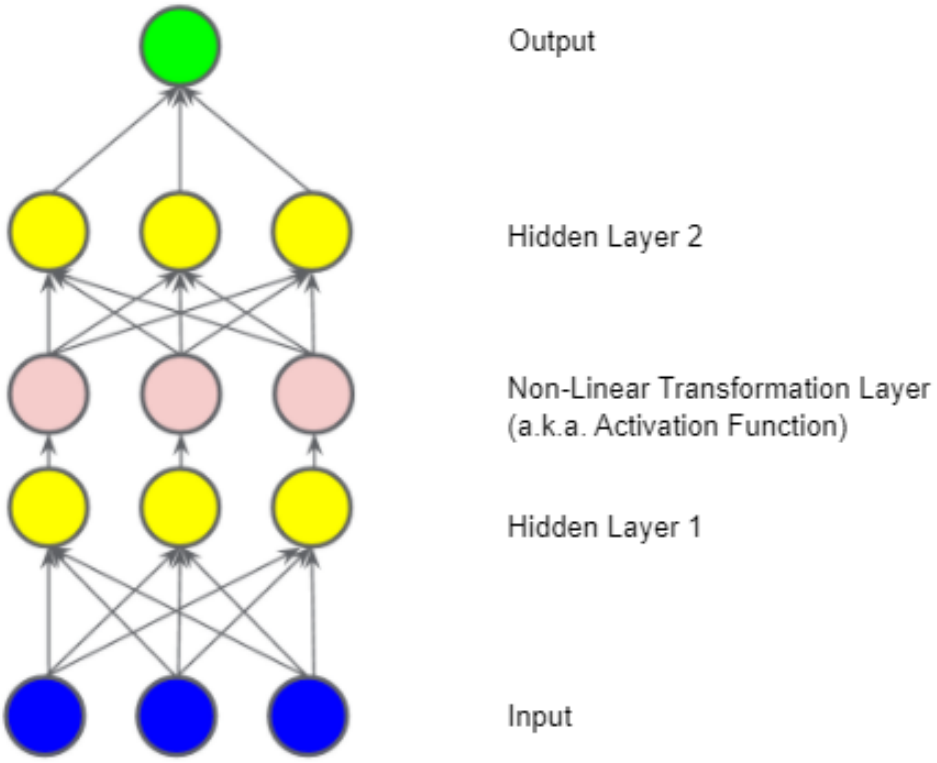


Figure 2.18: A diagram showing the basic structure of a neural network. This type of model consists of an input layer, multiple hidden layers containing linear transformations (weighted sum), a non-linear transformation layer containing an activation function, and an output layer [8].

data in a non-linear way. So along with these linear layers, we also have layers which pass the outputs of the linear transformations through a non-linear transformation called an activation function. There are a number of possible activation functions that can be used, with two of the more popular ones being Sigmoid and ReLU [8]. Including these non-linear transformations enables us to model much more complex relationships between the input and output values. Finally, after the hidden layers, there is an output layer, which will contain the results of the predictions made [8]. The training process for a neural network has some similarities to the process outlined in Section 2.4.2, however there are a number of other functions applied to each node of the neural network making the process a bit more complex. The most common training algorithm for neural networks is a process called backpropagation, and a good visual explanation of the process can be found in [6].

The structure presented in Figure 2.18 contains a single output node, which can be useful in any case in which we just have two mutually exclusive classes [8]. For example, we may want a neural network to determine which emails being received are spam or not spam. Such problems as this are known as binary classification problems. However, if we want to determine if the input features correspond to one of many classes, it would be more efficient to use a slightly different structure known as a multi-class neural network. Figure 2.19 shows an example of this. Also, instead of having an output value of true or false to distinguish between multiple classes, we can assign a probability value showing how likely the input features correspond to each class. This is achieved by applying a function before the output layer known as softmax. Using probabilities not only expedites the training process, but can also provide useful insights regarding how different feature values can influence the classification, such as how different variants of a sign language letter can affect the probability of being classified as that gesture [8].

The neural networks which have been presented in the examples so far are known as feed forward neural networks or FNNs. FNNs have input data travel through each layer once until they reach the output layer [9]. This type of neural network is the

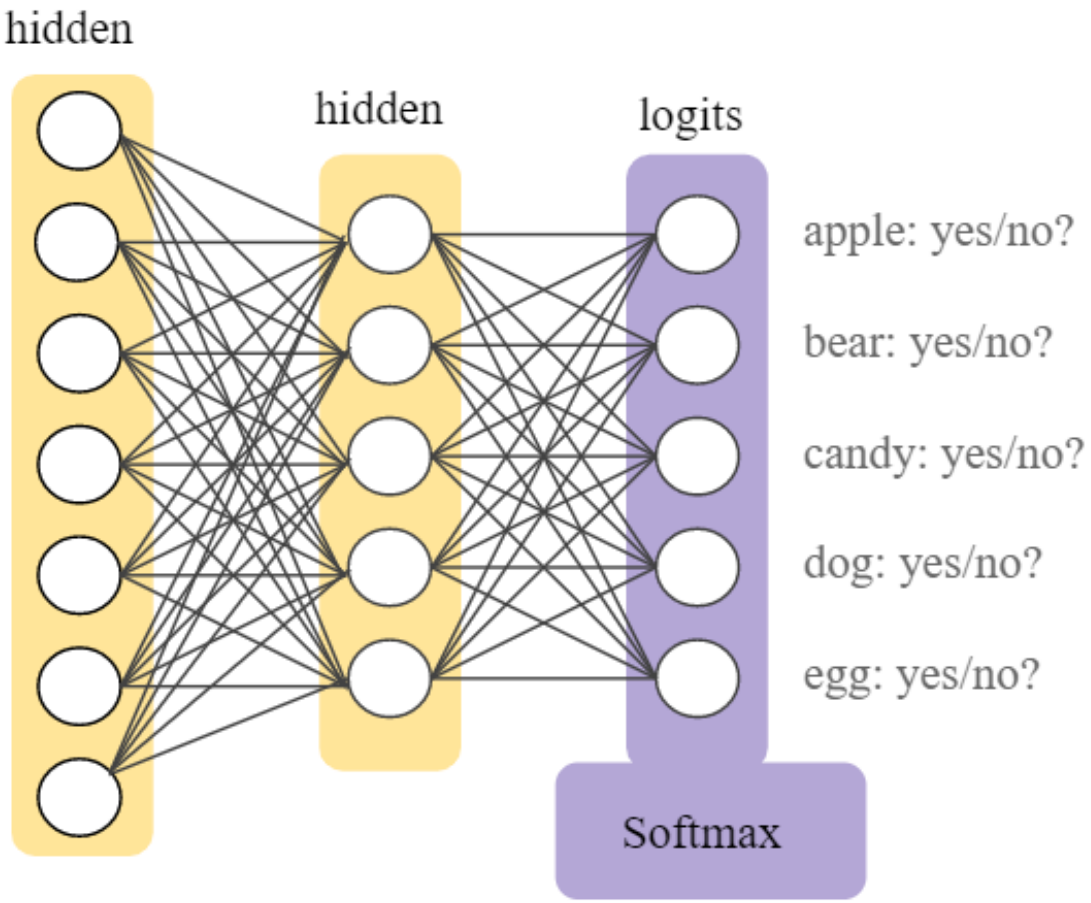


Figure 2.19: A diagram showing an example of a multi class neural network. As opposed to a standard neural network, this one has multiple output predictions, with each prediction corresponding to a different class. A logit is a raw prediction value before it undergoes some kind of transformation, such as a regularization function or a softmax function. [8]

same one which was used to develop the project presented in this thesis. There are a number of other neural network types with more complex structures than FNNs which are used for a variety of different purposes. One of these is the Recurrent Neural Network or RNN. An RNN differs from an FNN in that it is run multiple times in order to generate a desired predicted output based on a combination of multiple inputs [9]. This is because the hidden layers from a previous run of an RNN pass on parts of their used inputs to the hidden layers in each subsequent run. Because of this, RNNs are often used to predict sequences of data, such as bodies of text. For example, if we wanted to predict a desired song based on single lyric, we could split the song lyric into parts and pass each part into an RNN. The RNN would maintain a memory of the previous lyric portion with each run, enabling it to combine them together to make a more accurate prediction of what the desired song will be. Other types of neural networks are explained in the glossary of google's machine learning crash course which can be found in [9]. Neural networks can also further be characterized as being deep or wide. A deep neural network is one which contains multiple hidden layers while a wide neural network is one which has many inputs mapping directly to the output layer with no hidden layers in-between. Because of this, wide neural networks cannot express non-linearities through hidden layers, but they can mimic nonlinear behavior through other methods such as feature crosses [9].

2.4.4 Supporting Tools

As the usage of machine learning has become more prevalent, a number of software tools have been created and utilized in order to facilitate machine learning development, and make the process more intuitive. Some of these tools, which were also used to develop the project presented in this thesis, are explained below:

Tensorflow: Tensorflow is an open source platform used to develop machine learning applications [7]. While it includes a number of tools and resources to both build and deploy machine learning based software, the area of focus regarding this the-

sis is the upper level APIs used to build and train models, and make predictions using those models. The majority of this functionality is present in an API called `tf.keras`, which is a Tensorflow variation of another open source API named `Keras`. The `tf.data` API is also useful for handling large amounts of data of different types through an abstraction called a dataset. Many of the operations in Tensorflow revolve around the use of another abstraction known as a tensor, which is essentially a specialized multidimensional array. Tensors are used as a primary data structures during both the training and the inferrence (or prediction) phase of development in tensorflow [7].

Pandas and NumPy: Pandas and Numpy are two powerful data handling libraries which accompany the Python programming language. Numpy provides a number of useful data structures such as multidimensional arrays and matrices, along with several functions for performing complex operations on these structures quickly [21]. Pandas provides a very useful data structure used extensively in machine learning applications called a dataframe [33]. A dataframe is similar to a spreadsheet or a table within a database. One of the dataframe's more useful qualities is that it can contain multiple different data types, which exceeds the capabilities of data structures in several other languages. Pandas not only includes a number of functions to perform various operations on dataframes, but also includes functions to display them in intuitive ways for research purposes. Pandas also features a number of functions for converting between Numpy structures and its own, making the two a very useful combination [33].

Spyder: Spyder is an open source development environment written in Python and used primarily for developing applications related to data science [29]. Some features of Spyder include advanced data visualization tools, a variable explorer enabling developers to view and edit variables at runtime, and an embedded console window which can be used to run code separately or view outputs. The variable explorer is

very useful in viewing large dataframes as it also color coats each cell based on the differences between its contents and the surrounding data, making it easy to observe patterns within the dataframe [29].

Barracuda and ONNX: Barracuda is neural network inferrence library used in the Unity development environment [38]. Once a neural network is trained through other tools such as tensorflow and a model file is created, this model can be imported into Unity code using functions available within the Barracuda API. In order for model files to be imported using Barracuda, they must be of the ONNX format. ONNX or Open Neural Network Exchange is a standard format for representing machine learning models, and is meant to be easily interchangeable between various software tools and frameworks. Tensorflow, along with a few other machine learning APIs, offer functions to easily convert their default model formats to the ONNX format so that they may be shared among other applications [38].

Chapter 3

ASL Gesture Recognition Application

3.1 Overview

This chapter explains the structure and implementation of the ASL Gesture Recognition Application. This application consists of two parts: A trainer and a recognizer. The trainer is responsible for generating training data for the machine learning models which perform the gesture recognition, executing the training process to generate those models, and converting those models to a format which can be used by the recognizer. The trainer would only need to be run when the set of gestures to be recognized changes. For example, if we would want to include more ASL words or phrases to be recognized than what is already included, the trainer would need to be run so that new models can be generated. The recognizer uses the models created by the trainer to classify incoming ASL gestures being performed by a user. The recognizer also consists of a UI that guides the user in performing each ASL gesture, and instructs them how to do so. The current state of the application includes a series of stages which the user can navigate through, which each prompt for a separate ASL gesture to be performed. However, this UI is fairly malleable, and can be reworked to suit a wide variety of ASL related applications.

Given the number of possible gestures that can be recognized within ASL, it was essential to design this application in such a way that it could be easily expanded upon as necessary. In this initial implementation of the application, the models are

trained to recognize 34 ASL gestures, including the 26 letters of the alphabet as well as 8 words/phrases. Each ASL gesture is categorized based on the features of it which are used for recognition. These categories include static gestures, multi gestures, and positional gestures. Static gestures are essentially any ASL gesture that involves a single hand pose with one or both hands while Multi Gestures involve a series of poses. Positional gestures are static gestures or multi gestures that have a more strict reliance on the position in which they are performed. These categories are explained further in Section 3.4. Gestures are performed using the VRFree motion tracking gloves developed by Sensoryx, which are used in conjunction with the Vive Cosmos VR Headset. The VR gloves were chosen as input due to their ability to fully track individual finger movement and robust API as explained in Section 2.3.3. The Vive Cosmos was chosen as the VR hardware due to the low number of peripheral hardware components required, such as optical tracking lighthouses, as well as it's easy setup process. On the software side, tensorflow combined with python tools are used in the training process, while Unity and C-Sharp tools are used for both training data generation and performing the entire gesture recognition process.

The remainder of this chapter is outlined as follows: Section 3.2 goes over the functional and non-functional requirements for the ASL gesture recognition application. Section 3.3 presents a use case diagram for the gesture recognition application along with detailed explanations of the use cases on both the training and recognition side. Section 3.4 outlines the architecture and implementation of the application, explaining the trainer and architecture modules separately. A class diagram is also presented which explains the Unity class structure in more detail, as this makes up most of the application's structure. Finally Section 3.5 goes over the User Interface of the application developed using Unity.

3.2 Software Requirements

The requirements of the ASL gesture recognition application are outlined in this section. These include both functional and non-functional requirements. Functional

requirements describe what services the specified system should perform in order to meet it's user needs [27]. Non-functional requirements are those which do not describe services which are provided directly for user, but rather traits or constraints of the software which enable it to provide these services [27].

3.2.1 Functional Requirements

Table 3.1 outlines the functional requirements for the ASL Gesture recognition application. Each functional requirement is assigned a priority between 1 and 3. Priority 1 requirements are those that are considered essential for the application to perform its base functionality. Priority 2 requirements are not essential, but provide additional functionality to improve the functionality of the application. Priority 3 requirements are the lowest priority, and correspond to features which could be added later to further improve the functionality of the application. All priority 1 requirements were satisfied in the development of the application along with a few priority 2 requirements.

Table 3.1: List of Functional Requirements

Name	**Priority**	**Description**
FR1	1	The application will recognize trained ASL gestures performed by a user in real time.
FR2	1	The application will provide feedback to the user when a trained ASL gesture is successfully recognized.
FR3	1	The application will provide visual aids to help teach a user how to perform various trained ASL gestures and calibration gestures.
FR4	1	The application will display virtual hands which will mimic any movement performed by the user while equipped with the VR headset and motion tracking gloves.
FR5	1	The application will enable the user to calibrate the gloves using a calibration routine provided by the glove API.
FR6	1	The application will only start tracking moving ASL gestures (or multi gestures) when a starting gesture is performed indicating the start of the moving gesture.

Continued on next page

Table 3.1 – continued from previous page

Name	Priority	Description
FR7	1	The application will only start tracking gestures which rely on position (or positional gestures) when a starting gesture is performed indicating the start of the positional gesture.
FR8	1	The application will enable a user to generate training data for new static ASL gestures in the form of JSON files.
FR9	1	The application will enable a user to generate training data for new moving ASL gestures (multi gestures) in the form of JSON files.
FR10	2	The application will enable the user to adjust the threshold for gesture recognition in order to make gestures more/less strict as necessary.
FR11	2	The user will be able to start the application from an executable file.
FR12	2	The application will feature an advanced data panel which displays which gesture is being detected and more gesture detection details.
FR13	2	The application will enable the user to execute advanced calibration procedures, such as head and neck calibration.
FR14	2	The application will display a count of how many attempts each gestures takes and how long each gesture takes.
FR15	2	A free mode to perform any asl gesture and form random sentences, etc.
FR16	2	The application will log anytime a gesture is recognized along with an accompanying timestamp.
FR17	3	The user will be able to switch between different modes of the application through a menu.
FR18	3	The user will be able to interact with the application through a virtual touch interface.
FR19	3	The user will have the ability to replay gestures by loading json files of previous gestures performed.
FR20	3	The user will be able to chat with another user using ASL through a network connection.
FR21	3	The user will be able to execute the training process from unity rather than through separate python execution.

Continued on next page

Table 3.1 – continued from previous page

Name	Priority	Description
FR22	3	The user will be able to communicate with virtual NPCs in a gamefied environment.

3.2.2 Non-Functional Requirements

Table 3.2 outlines the non-functional requirements for the ASL Gesture recognition application.

Table 3.2: List of Non-Functional Requirements

Non-Functional Requirements	
Name	Description
NFR1	The inference process and UI shall be executed in separate threads.
NFR2	The inference code and uI code shall be written in C Sharp.
NFR3	The model training code shall be written in python.
NFR4	There shall be three different neural networks implemented for ASL gesture recognition: One for static gestures, one for multi gestures, one for positional gestures.
NFR5	The model files shall be converted to ONNX format for use in Unity.
NFR6	The barracuda framework shall be used to interpret model files.
NFR7	The inferrence code shall be populated with gesture and label data using CSV files.

3.3 Use Case Modeling

This section outlines the various use cases of the ASL gesture recognition application. Use cases are described for each module of the application: the model training system (trainer) and the gesture recognition system (recognizer). Each of these systems also has a separate actor. The User interacts with the use cases of the gesture recognition system. This can be considered the front end of the application, and involves the use cases which would most often be performed. The Developer interacts with the use cases of the model training system. These use cases describe the process for both

generating training data and training the gesture recognition models that will be used by the application. There are also a number of included use cases triggered by each system when the user or developer performs certain actions.

The use case diagram displayed in Figure 3.1 illustrates all use cases that are involved in both systems. Table 3.3 provides descriptions for each use case.

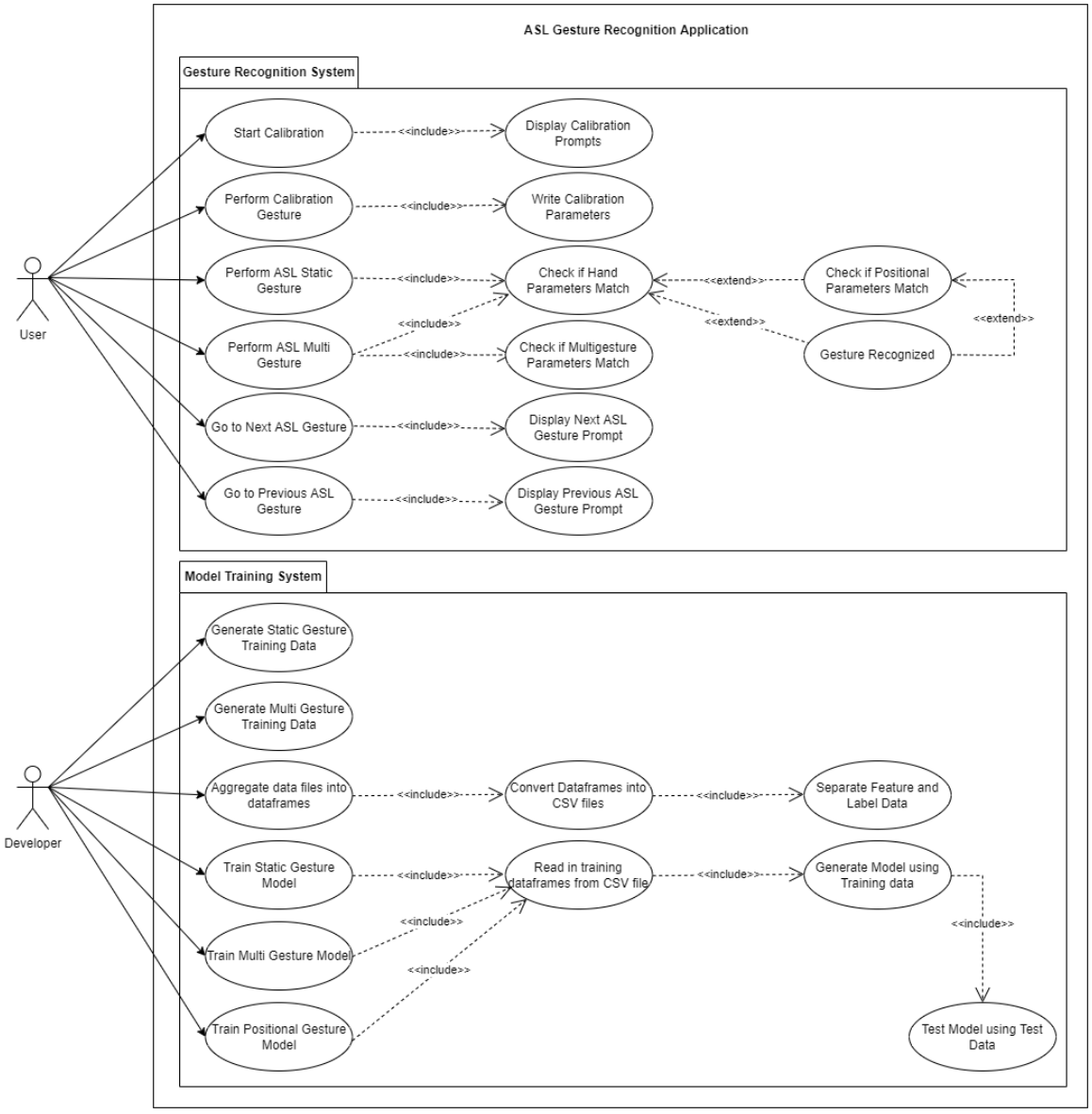


Figure 3.1: Use case diagram for the ASL gesture recognition application.

Table 3.3: Use Case Descriptions

Name	**Description**
Start Calibration	The user begins the calibration routine for the gloves. This involves holding three distinct poses for about three seconds each. This calibration routine is defaultly available with the glove API, and is typically performed if there is a significant mismatch between the locations of the user's hands and the locations of their virtual counterparts.
Display Calibration Prompts	After the user starts the calibration process, the system will display an image for each calibration gesture to assist with the process. After the user completes a calibration gesture, an image showing the next image to perform will appear.
Perform Calibration Gesture	After the calibration routine has started the user will be prompted to perform each calibration gesture. Three gestures are required, and each gesture must be held for a few seconds each.
Write Calibration Parameters	As the calibration process is performed the system will write relevant calibration parameters to a series of text files which are used by the glove display system to synchronize the virtual hands with the real ones.
Perform ASL Static Gesture	Users will be able to perform static ASL gestures, which are ASL gestures that involve a single hand gesture without any motion involved. These include all letters of the alphabet except for "J" and "Z".
Perform ASL Multi Gesture	Users will be able to perform multi ASL gestures, which are ASL gestures that involve one or more hand/arm movements and possible more than one distinct hand gesture. These include the ASL letters "J" and "Z", along with a number of words and phrases.
Check if Hand Parameters Match	Each time a possible ASL gesture is performed by the user, the system will check if the hand parameters match those of ASL gestures that have been trained by the application. This is done by plugging the hand parameters into the machine learning model that was trained using them as features. Hand parameters consist of joint locations for each of the fingers.

Continued on next page

Table 3.3 – continued from previous page

Name	Description
Check if Multi gesture Parameters Match	Each time a possible ASL multi gesture is performed by the user, the system will check if the multi gesture parameters match those of ASL gestures that have been trained by the application. This is done by plugging the multi gesture parameters into the machine learning model that was trained using them as features. multi gesture parameters relate to the pattern of movement which is performed by the multi gesture, the distances between each waypoint of this movement, and some others.
Check if Positional Parameters Match	Each time a possible ASL positional gesture is performed by the user, the system will check if the positional parameters match those of ASL gestures that have been trained by the application. This is done by plugging the positional parameters into the machine learning model that was trained using them as features. Positional parameters relate to the location of the gloves relative to the headset.
Gesture Recognized	Assuming all necessary gesture parameters match those of a gesture that has been trained by the application, a gesture is labeled as recognized by the application, which will display an appropriate prompt to the user indicating this, such as the virtual hand changing colors.
Go to next ASL gesture	The user will be able to navigate between various ASL gestures of focus which can be performed within the application. This is meant to provide a tutorial-like atmosphere to help facilitate ASL education. Navigation is performed through the use of arrows which are activated by being looked at in the VR environment.
Display next ASL gesture Prompt	When the user navigates to a different ASL gesture to perform, an image and/or video will be displayed which is meant to help guide the user in performing that gesture.
Go to previous ASL gesture	The user will be able to navigate between various ASL gestures of focus which can be performed within the application. This is meant to provide a tutorial-like atmosphere to help facilitate ASL education. Navigation is performed through the use of arrows which are activated by being looked at in the VR environment.
Display previous ASL gesture Prompt	When the user navigates to a different ASL gesture to perform, an image and/or video will be displayed which is meant to help guide the user in performing that gesture.

Continued on next page

Table 3.3 – continued from previous page

Name	Description
Generate Static Gesture Training Data	The developer will be able to generate training data for static ASL gestures through the ASL Gesture Recognition Application. This is done by performing the gesture to be trained as many times as needed while wearing the motion tracking gloves and headset, and generating the data for those gestures in the form of JSON files. This training data also includes features for positional gestures.
Generate Multi Gesture Training Data	The developer will be able to generate training data for ASL multi gestures through the trainer module of the ASL Gesture Recognition Application. This is done by performing the gesture to be trained as many times as needed while wearing the motion tracking gloves and headset, and generating the data for those gestures in the form of JSON files. This training data also includes features for positional gestures.
Aggregate data files into dataframes	The developer will be able to group the training data JSON files into dataframes through the use of a pre-processing script. Dataframes are a table-like data structure available within the Pandas framework. These dataframes are then used in the training process at a later point.
Convert dataframes into CSV files	After the developer creates the training dataframes, they will automatically be converted into CSV files by the system. This is so that these dataframes will not need to be re-generated at another point, in case optimization needs to be performed during the training process.
Seperate Feature and Label Data	When the dataframes are converted into CSV files, the system will also separate the feature and label data into distinct columns. The label data corresponds to the name of the trained letter,word, or phrase, while the feature data corresponds to the various parameters of each of these.
Train Static Gesture Model	The developer will be able to train the machine learning model for Static ASL gestures. This is done by running the appropriate training script which references the dataframe containing the necessary training data.
Train Multi Gesture Model	The developer will be able to train the machine learning model for Multi ASL gestures. This is done by running the appropriate training script which references the dataframe containing the necessary training data.

Continued on next page

Table 3.3 – continued from previous page

Name	Description
Train Positional Gesture Model	The developer will be able to train the machine learning model for Positional ASL gestures. This is done by running the appropriate training script which references the dataframe containing the necessary training data.
Read in training dataframes from CSV file	When the developer runs any training script, the system will automatically read in the CSV file that was creating when aggregating the training data JSON files.
Generate Model using Training Data	After the training data has been read in and parsed by the system, the data will be automatically passed into a training function to generate an associated neural network.
Test Model using Test Data	After a model has been generated by the training process, the system will then automatically test the model against a separate set of Test data, and notify the developer of the results.

3.4 Architecture and Implementation

These sections provide a high-level view of the architecture and implementation of the ASL gesture recognition application. The training module and recognition module of the application are described separately. A class diagram is also presented in this section which describes the c-sharp classes used by the application and the relationships between them.

3.4.1 ASL Gesture Representation

This section provides more details regarding the three gesture categories used to identify ASL gestures: static gestures, multi gestures, and positional gestures.

3.4.1.1 Static Gestures

Static gestures are any ASL gesture which involves performing a single hand pose with one or both hands. Once the hand pose is performed successfully, the gesture is immediately identified as successful by the application as well. See Figure 3.2 for examples of static gestures. Some gestures which fall under this category include all letters of the alphabet except for “J” and “Z”, and words or phrases such as

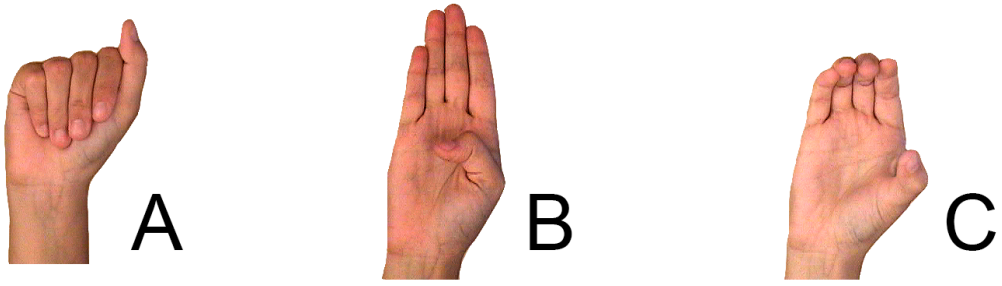


Figure 3.2: Examples of ASL gestures that are seen as static gestures by the application [30].

"LOVE", "I AM", and "MY". Static gestures are essentially the base of all ASL gestures recognized by the application. For example, a multi gesture is seen as a series of static gestures, and a static positional gesture is simply an ASL static gesture that is more reliant on position. The static gesture representation was adopted from a default gesture representation of the same name which came available with the VRfree API. Both data types essentially serve the same purpose, though the hand parameters are handled differently with each.

Another important note regarding static gestures is that they can be repeated among different ASL gestures. For example, the gesture for the letter "I" is the same as the starting gesture for the letter "J". Because of this, certain static gestures are grouped under a category called base gestures so as to allow for more distinction between each set of gesture parameters, which also makes the corresponding machine learning model easier to train.

3.4.1.2 Multi Gestures

Multi gestures are any ASL gesture which involves performing one or more hand poses, possibly at different locations around the body. The first pose of a multi gesture is called the starting pose, and once this is performed, the application begins to check any subsequent gestures performed to see if they are also a segment of the complete multi gesture. If no subsequent gestures are detected within a given time

frame, then the application identifies that multi gesture as a failure, and will require the user to start over by re-performing the starting pose. Some examples of multi gestures include the letters "J" and "Z" (Figure 3.3), and words or phrases such as "LEARNING", "SIGN LANGUAGE", and "NAME IS". Multi gestures can be either one handed or two handed.

The multi gesture representation was adopted from a default gesture representation of the same name which came available with the VRfree API. However, the original representation did not consider the position of the hands when recognizing gestures. The representation was updated to include both positions of the hands and distances between each subsequent hand pose, or multi gesture segment. Hand rotation was also incorporated as a parameter to account for hand gestures which had the same overall finger placement but were oriented at a different rotation. Multi gestures are essentially seen by the application as a sequence of static gestures which

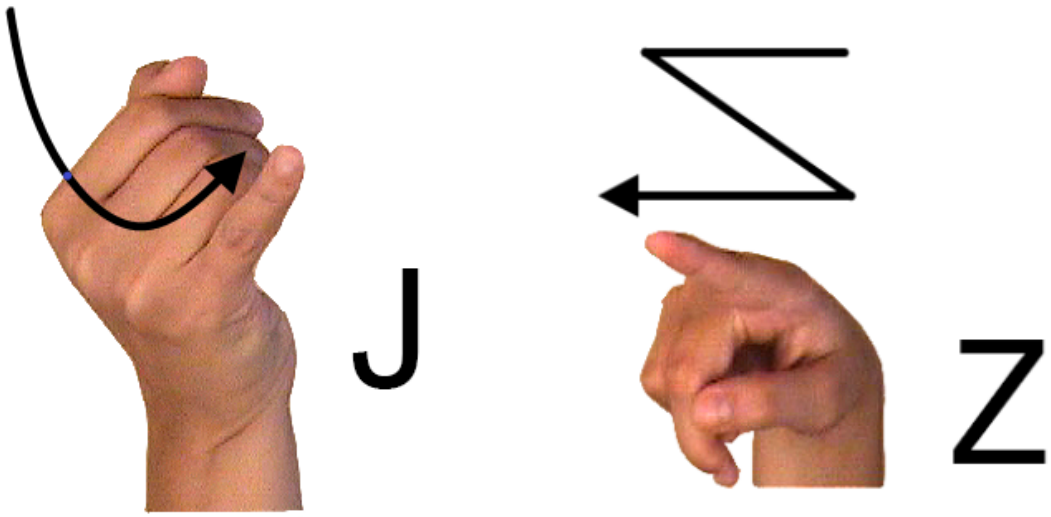


Figure 3.3: Examples of ASL gestures that are seen as multi gestures by the application [30].

are reliant on position, rotation, distance, and the time in which they are performed.

3.4.1.3 Positional Gestures

Positional gestures are static gestures or multi gestures that require a stricter adherence to the positions of the hands relative to the body when being performed. For example, some ASL gestures need to start directly adjacent to the head, such as the word "HELLO". The words/phrases "MY" and "I AM" are other examples as they need to be performed close to the chest. To reinforce the positioning constraints of these types of gestures, the application uses the distance and direction of the gloves from the VR headset as a gesture parameter. If positional gestures fall out of an expected distance/direction threshold, they are not detected by the application. Positional gestures are not seen as an explicit data type within this application, but rather a trait of both the static gesture and multi gesture categories. As such Certain static gestures and multi gestures trigger this additional positional tracking when performed.

3.4.2 Trainer Architecture and Implementation

The architecture of the training system for the ASL Gesture Recognition Application is described in Figure 3.4. This architecture is broken up into five modules which are described in the following sections:

3.4.2.1 Hardware Input

The trainer uses both the VRfree gloves and Vive Cosmos VR Headset as input hardware. This hardware is used to generate training data through the Unity software. This is done by performing the necessary gesture to be trained as many times as necessary while equipped with the headset and gloves.

3.4.2.2 Input Data

After the gestures to be trained are performed using the input hardware, a JSON file is generated for each time the gesture was performed. This JSON file contains

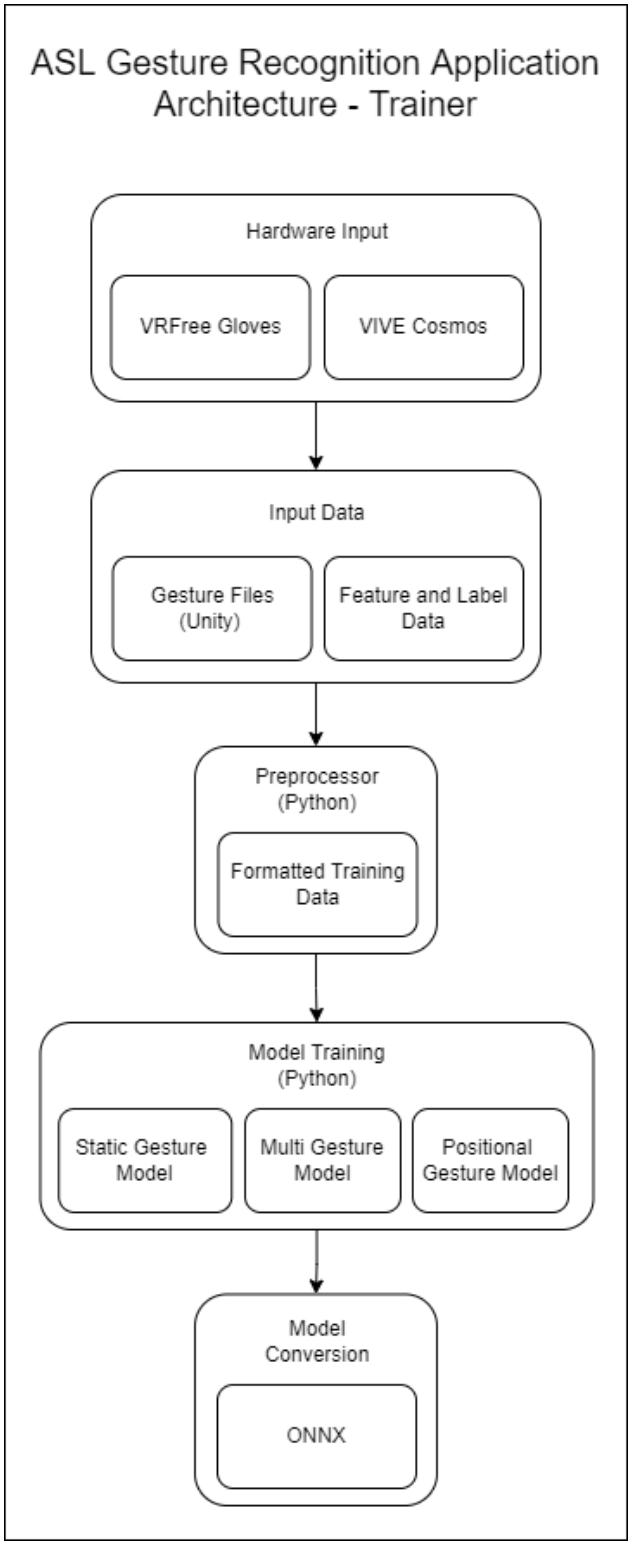


Figure 3.4: Architecture Diagram for the training portion of the ASL gesture recognition application.

necessary parameter data which will eventually be used as features within the machine learning models at a later stage. Along with this JSON file, a separate file containing the feature and label data for each model is also compiled which includes names, and flags to assist extracting data from the JSON file. This feature and label data file would only need to be updated when new gestures need to be trained that have not already been trained previously. Both of these data sets are passed into the next module of the trainer known as the Preprocessor.

3.4.2.3 Preprocessor

The Preprocessor module involves reformatting the JSON files from the Input module into a format which can be passed into the model training functions. This module also involves reading in the feature and label data from from the corresponding file defined within the Input module, and using those to assist reformatting the JSON data. This reformatted data will then be passed into the model training functions in the next module. The Preprocessor is executed through a single python script.

3.4.2.4 Model Training

The Model Training module handles the training of each model used when performing recognition in the ASL Gesture recognition application. This module is responsible for taking in preprocessed data and passing it to the model training functions. There are three models which need to be trained for the ASL gesture recognition application: A static gesture model, a multi gesture model, and a positional gesture model. Models are generated using tools from the Tensorflow framework along with general python and numpy data structures and operations. More information regarding the models used with this application is present in Section 3.4.5.

3.4.2.5 Model Conversion

After the models are trained, they are then converted into the ONNX format so that they can be utilized within the Barracuda machine learning framework. Once these models are converted, they can then be passed into the gesture recognition process.

3.4.3 Recognizer Architecture and Implementation

The architecture of the recognition system for the ASL Gesture Recognition Application is described in Figure 3.5. This architecture is broken up into five modules which are described in the following sections:

3.4.3.1 Hardware Input

Just as with the trainer, the Recognizer uses both the VRfree gloves and Vive Cosmos VR Headset as input hardware. Data from the gloves and headset are transferred to the gesture recognition application by the user as they perform gestures. The user will also perform calibration routines using the gloves combined with the headset.

3.4.3.2 Input Data

Rather than taking external gesture data files as input, the gesture recognizer takes sensor data directly from the gloves in real time. Rotational and positional data from

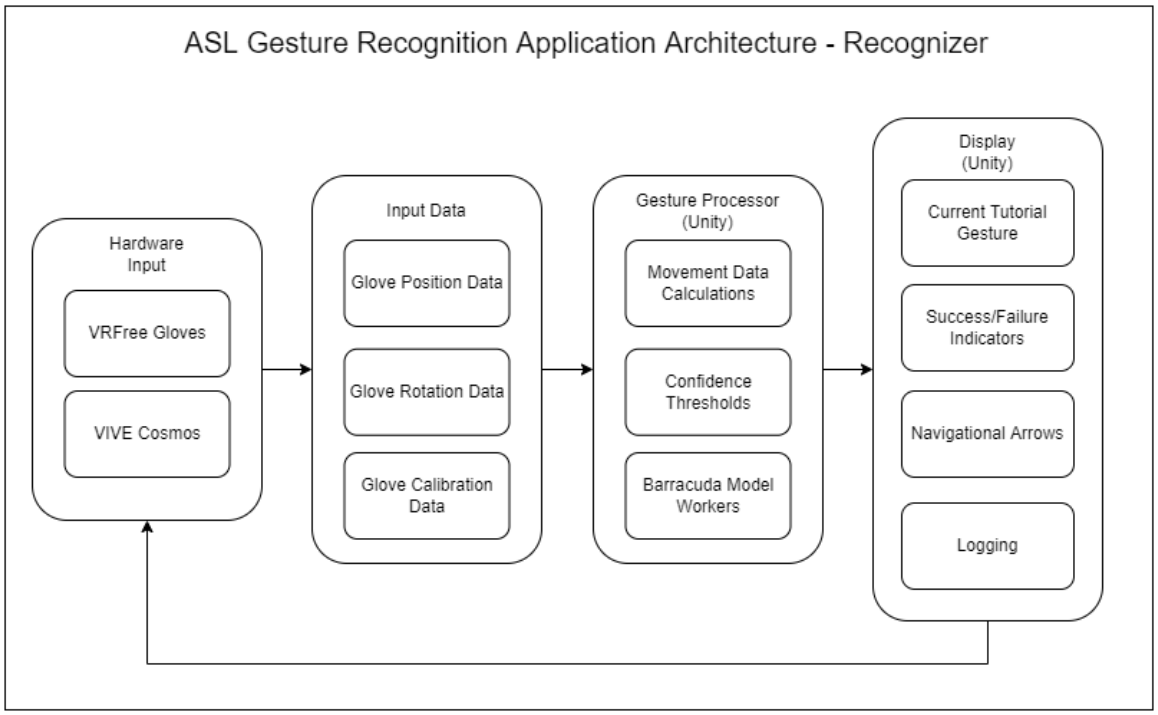


Figure 3.5: Architecture Diagram for the inference portion of the ASL gesture recognition application.

the gloves are read in by the corresponding API to be processed through Unity. This includes data from each joint of each finger along with the wrist. Calibration files are also read in through the API to make the virtual representation of the gloves more accurately match their real-world counterparts.

3.4.3.3 Gesture Processor

After the glove data is retrieved, it is then parsed and passed through a series of functions within the Unity application as a part of the gesture recognition process. All data is ultimately passed into one of three model workers which each correspond to a model that was trained to recognize a certain type of ASL gesture. These model workers are a part of the barracuda machine learning framework. After the data is passed through each model, the predicted ASL gesture ID which has the highest probability of matching the supplied data is output. This corresponding probability is then compared to a confidence threshold which would cause the application to either accept or reject the model's prediction based on the probability that has seemed most accurate in practice.

Based on the ASL gesture type, the corresponding data may undergo different transformations before being passed to the model workers. Appropriate static gesture features are extracted directly from the base hand data structure from the API, and then passed to the static gesture model worker. If the hand data corresponds to a starting gesture that is part of a larger multi gesture, then movement data calculations are performed on both this gesture and subsequent potential multi gesture segments as more hand gestures are performed. Certain features are then extracted from both this movement data and the individual gesture segments, and then passed into the multi gesture model worker. In both cases, the positional data of the static gesture or starting gesture is also extracted, and passed on to the positional model worker if the ASL gesture is also one that has a strict reliance on position.

3.4.3.4 Display

The display module is responsible for both notifying the user of the success and failure of corresponding gestures performed, as well as providing an user interface in which the user can navigate through different gestures to be performed and observe visual guides to assist them with performing each gesture. The default mode of the application has users focus on an individual ASL gesture (or tutorial gestures) to be performed based on a visual guide. The user navigates between these gestures using virtual arrows which take the user to the next or previous gesture to be performed. Along with visual cues in the VR environment to indicate gesture success/failure, logs are also generated to provide this data along with associated timestamps for development purposes.

3.4.4 Class Diagram

A class diagram for the gesture recognition application is outlined in Figure 3.6. Since some of the classes used have a large amount of member variables and functions, this diagram only includes the connections between each class along with their names for clarity. Specific details regarding each class are specified in the following sections, though only public member variables are displayed along with the member functions, as this are sufficient to the scope of this thesis. Note that there are several public variables present in some classes because certain public variables in Unity can be adjusted during runtime, which helped to facilitate the adjustment of gesture recognition parameters during the user study as needed. In more finalized versions of this application, most of these variables would be private. Some Unity specific functions present within some of the classes include `Awake()`, `Start()`, and `Update()`.`Awake()` and `Start()` essentially perform initialization procedures on the classes before the application starts. `Update()` performs operations during every frame while the application is executing. A frame is essentially one of several snapshots that make up the visuals displayed to the user, similar to how multiple images make up a video. As such, frames can take place in a fraction of a second, enabling `Update()` to be called

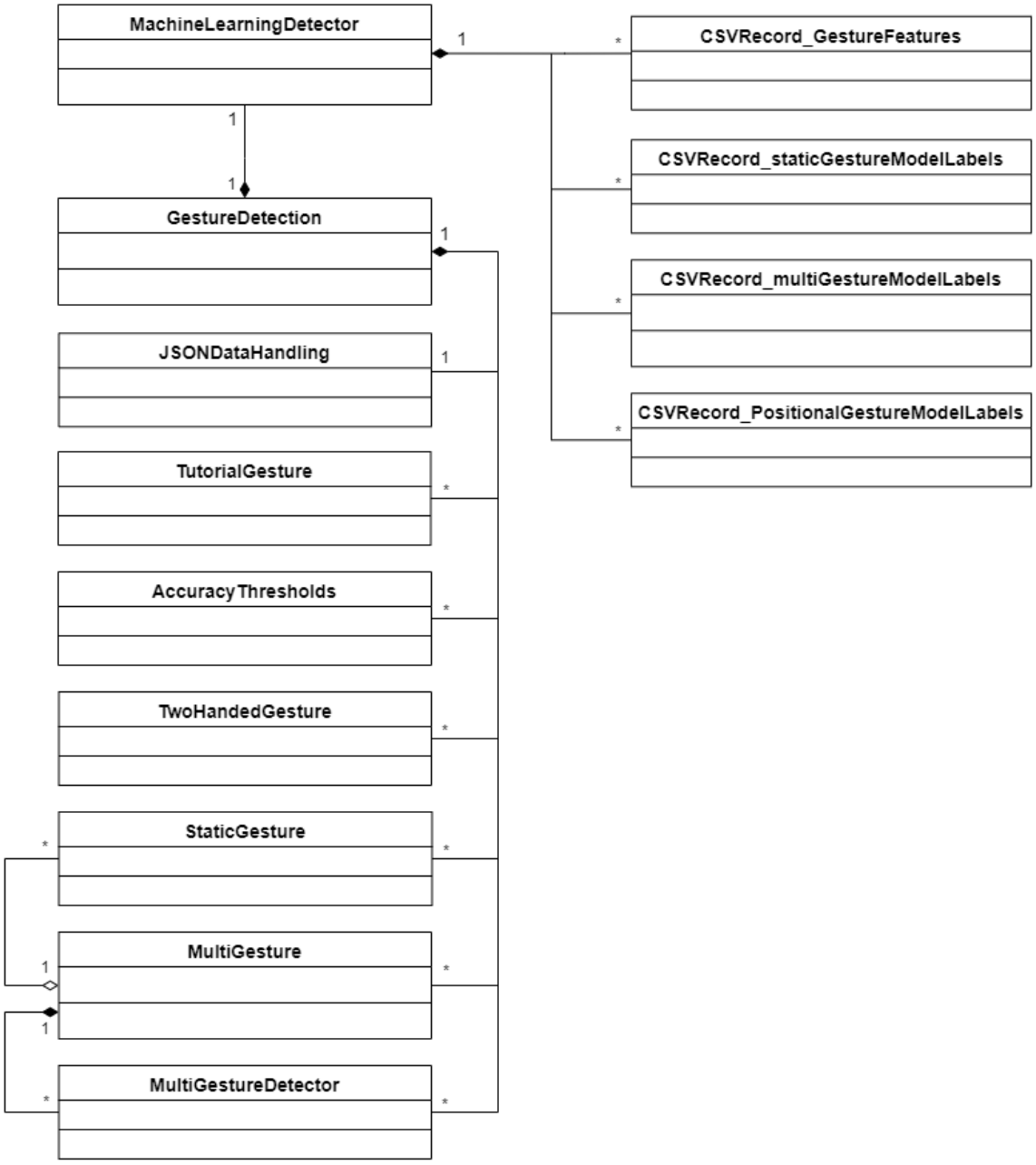


Figure 3.6: Class Diagram for the ASL gesture recognition application.

at a very high frequency.

The classes for this application were all developed within Unity, and as such involve functionality related to the VR portion of the application. This includes all functionality in the recognizer, and a smaller portion of functionality within the trainer involved in data generation. Most of the trainer functionality is handled within separate python scripts used for building the machine learning models which are used by the application. Some of these classes presented in this section originally came with the VRFree API, and were customized for the needs of the application. Others were created specifically for this application, such as those more involved in the machine learning process.

3.4.4.1 `MachineLearningDetector`

The `MachineLearningDetector` class is shown in Figure 3.7. This class is responsible for both initializing the models before they are used by the application, reading in the feature and label data to be used with those models, and making predictions on incoming gestures that are performed while using the application. This class uses a number of `CVSRecord` functions to read in the feature and label data from corresponding CSV files that were generated during the training process. This class also implements a recognition function for each model that passes the appropriate feature data into each model to gain a predicted label. These functions are `detectGestureStatic()`, `detectGestureMulti()`, and `detectGesturePositional()`.

The member variables with the type `NNModel` correspond to objects representing the models that were trained to perform recognition of various types of ASL gestures. The various `CSVRecord` objects correspond to the rows of CSV files that contain lable and feature data which are read in to facilitate the prediction process. The `detectedGestureID` variable corresponds to the current predicted gesture based on paramters fed in through the motion tracking gloves and VR headset. Member functions include those for parsing CSV files containing label and feature data, and detection functions which perform gesture predictions based on data that is read in.

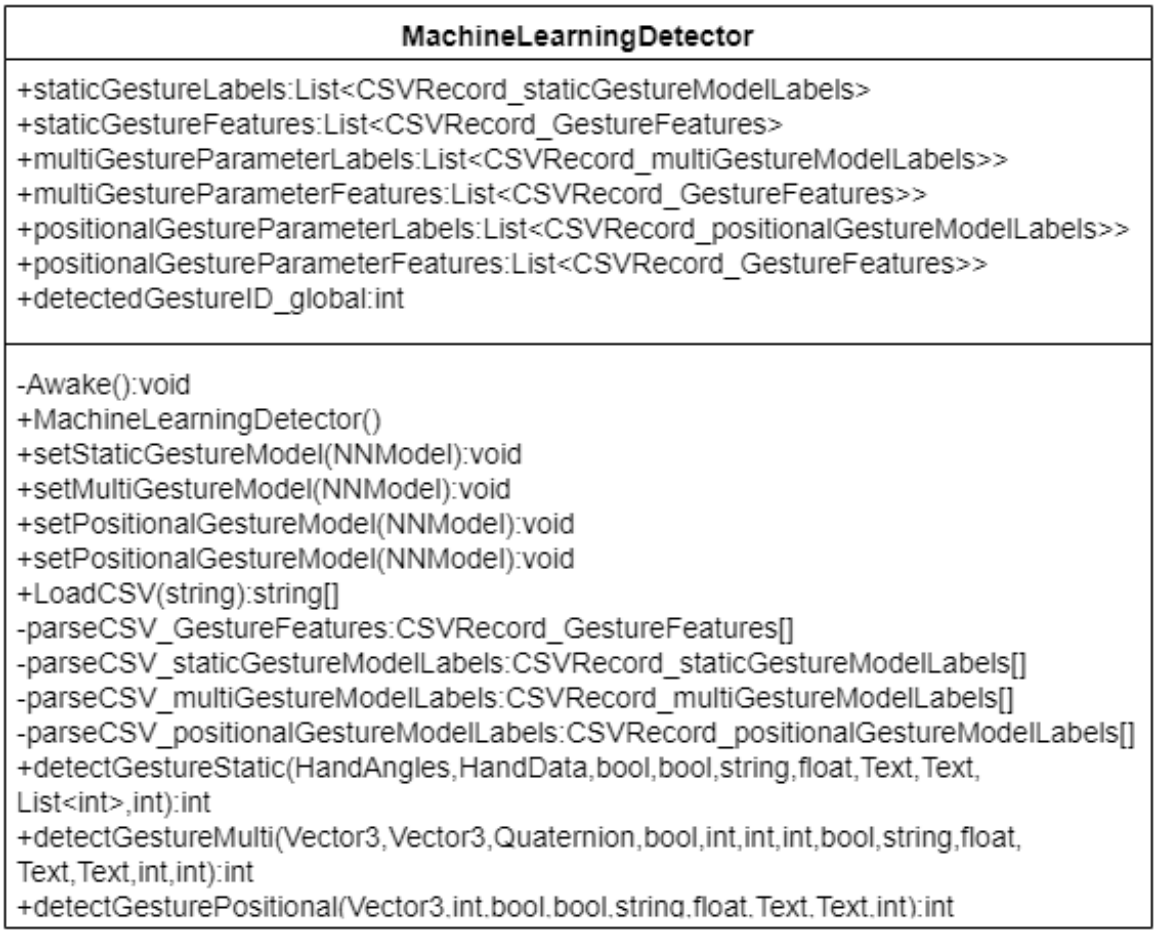


Figure 3.7: `MachineLearningDetector` class

3.4.4.2 `GestureDectection`

This class has the most tasks of any other class which include the following: Extracting and calculating additional feature data that is not defaultly available through the glove API such as multi gesture data and positional data; confirming which feature data should be passed to which models, and passing them to these models using a `MachineLearningDetector` class object; Displaying the primary UI including tutorial prompts and navigational arrows; Instantiating all gesture objects along with their corresponding confidence thresholds and data structures; and facilitating the training data generation process along with generating these corresponding data files. To assist with these processes, this class invokes a number of instances of other classes which have more specific tasks.

The `GestureDectection` class is shown in Figure 3.8. Three variables are dedicated to defining the usage mode of the application. The variable `trainingMode` corresponds to the usage of the application which involves training gestures. The variable `tutorialMode` corresponds to the front end of the application which involves performing gestures based on a series of tutorials. The variable `inferrenceMode` corresponds to a usage of the application which involves testing recently trained gestures without any specific tutorial prompt. This mode is mainly used on the developer side. Model objects are also defined here which are passed into the `MachineLearningDetector` class. Aside from this, various variables are defined to help facilitate the training and prediction process. Functions include those to help navigate through the tutorial process, such as `NextTutorialGesture()` and `PreviousTutorialGesture()` as well as those to assist with generating training data, such as `addStaticGestures()` and `addMultiGestures()`. Training data is generated within the `Update()` function of this class. The function `checkPoseCoroutine()` is by far the most important as it executes the inference process on the models by calling detection functions present in the `MachineLearningDetector` class. This process is run in a separate thread so as to not inhibit the UI being presented to the user. Functions which are run in a separate thread in Unity, or coroutines, are represented by the `IEnumerator` data

Figure 3.8: `GestureDetection` class

type. `checkPoseCoroutine()` is called at a similar frequency to the update function with calls taking place every 0.02 seconds.

3.4.4.3 `CSVRecord`

There are a number of CSV Record classes which have nearly identical functionality but different parameters. Due to their similarities, they are all explained within this section. These classes include `CSVRecord_GestureFeatures`, `CSVRecord_StaticGestureModelLabels`, `CSVRecord_MultiGestureModelLabels`, and `CSVRecord_PositionalGestureModelLabels`. Each class is meant to hold records of a CSV file containing associated features and labels to be processed by the `MachineLearningDetector` class. Each row of the CSV file is read into a CSV Record object. These objects are iterated through in order to determine which ASL gestures to check and and display.

Each of the CSVRecord classes are shown in Figure 3.9. `CSVRecord_GestureFeatures` is the simplest class as it only contains a single member variable used for holding a string representing a feature of a given model. `CSVRecord_StaticGestureModelLab`

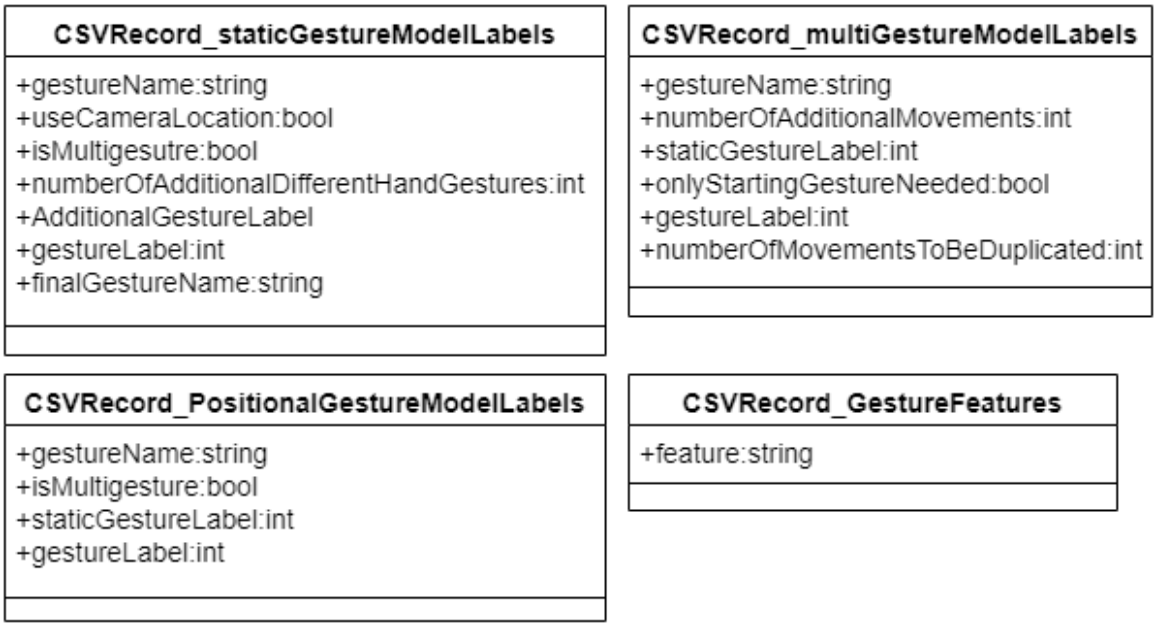


Figure 3.9: CSVRecord classes

els contains flags to help identify if the static gesture is part of a larger multigesture or positional gesture so as to treat the gesture differently if necessary. The variable numberOfMovementsToBeDuplicated corresponds to the number of times this particular trained gesture is used in other ASL gestures. CSVRecord_MultiGestureModelLabels contains similar flags to help identify they type of multigesture that is being performed. CSVRecord_PositionalGestureModelLabels contains basic traits regarding the corresponding position gesture such as whether or not it is a multi gesture and the underlying static gesture used.

3.4.4.4 JSONDataHandling

This class is responsible for writing gesture data to JSON files to be used in the training process of the application. This class was adapted from already present JSON handling functions with the VRFree API. After all training gestures are saved locally through the GestureDetection class, they are then exported using a JSONDataHandling object. There are separate functions for saving both static gestures and multi gestures. This class was adapted from an existing class within the VRFree API, and updated to accommodate new data parameters deemed necessary for ASL gesture recognition.

The JSONDataHandling class is shown in Figure 3.10. The only member variable of the class is a flag named max_num_gestures which corresponds to the maximum number of gesture files that can be generated at one time using the trainer. This is here to prevent file generation from blocking the training process for a significant period of time. All of the member functions of this class are meant to facilitate the gesture file generation process performed by this class. For example, AddStaticGesture() and AddMultiGesture() essentially stage corresponding static gesture and multi gesture objects to later be serialized into JSON files. Various "save" functions such as SaveStaticGestures() and SaveMultiGestures() actually generate these JSON files.

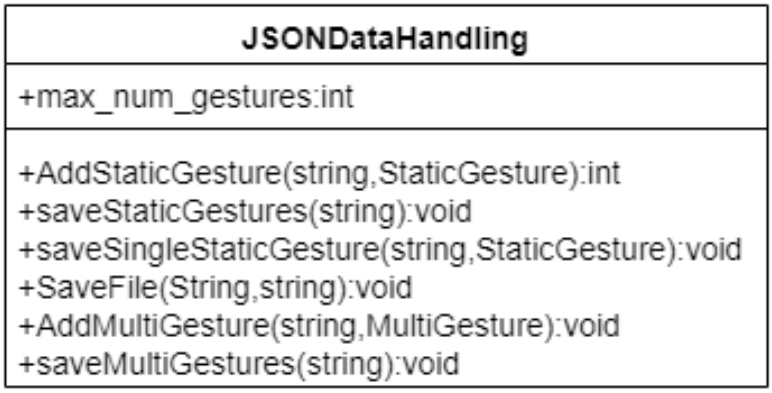


Figure 3.10: `JSONDataHandling` class

3.4.4.5 `StaticGesture`

This class is used as a representation for static ASL gestures within the application. An object of this class is created for each static ASL gesture in both the training and recognition process. During training, new instances of this class are created for each saved training gesture, which are serialized into JSON files. During the recognition process, static gesture objects are created for any incoming gestures by default, and removed if they do not match any ASL gesture currently supported by the application. An array of static gesture objects is also created for each multi gesture, since there may be a different static gestures involved for each multi gesture pose. This class was adapted from an existing class within the VRFree API, and updated to accommodate new data parameters deemed necessary for ASL gesture recognition.

The `StaticGesture` class is shown in Figure 3.11. The member variable `centerpose` corresponds to data read in from the gloves which is used to populate the feature data required by the static gesture model. The variable `handLocation` and `cameraLocation` are also used for positional and multi gestures. The member functions `poseSatisfiesGesture` and `poseSatisfiesPositionalConstraints` are prediction functions which use the `MachineLearningDetector` class to match the gesture performed with the static gesture model and positional gesture model respectively.

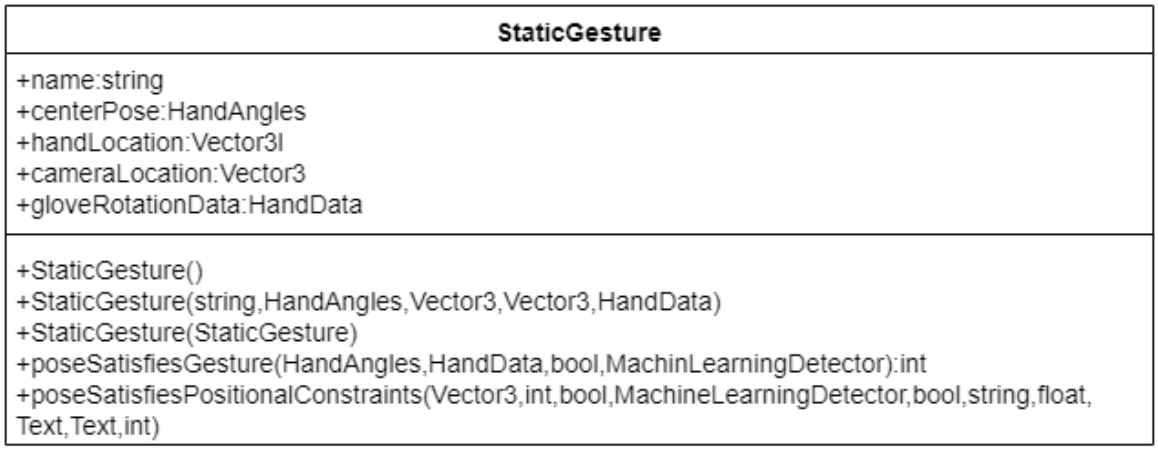


Figure 3.11: `StaticGesture` class

3.4.4.6 `MultiGesture`

This class is used as a representation for ASL multi gestures within the application. An object of this class is created for each multi ASL gesture in both the training and recognition process. During training, new instances of this class are created for each saved training gesture, which are serialized into JSON files. During the recognition process, multi gesture objects are created for incoming static gestures that are recognized as also being the starting gesture for a multi gesture. They are then deleted once processing is finished or if the multi gesture detected is not currently supported by the application. This class was adapted from an existing class within the VRFree API, and updated to accommodate new data parameters deemed necessary for ASL gesture recognition.

The `MultiGesture` class is shown in Figure 3.12. The member variables prefaced with `gestureSuccession` are used to hold data relating to each segment of the multi gesture that is being performed. The list `timeIntervals` defines the maximum amount of time allowed to perform each segment of the multi gesture. The lists `intermediateGestureDistances`, `handLocations`, and `intermediateGestureDirections` are used to calculate the distance and direction between each gesture and hold the resulting values. The lists `handRotations`, `handRotationDifferences`, and `handRotationInverses` are used to calculate the rotational differences between each

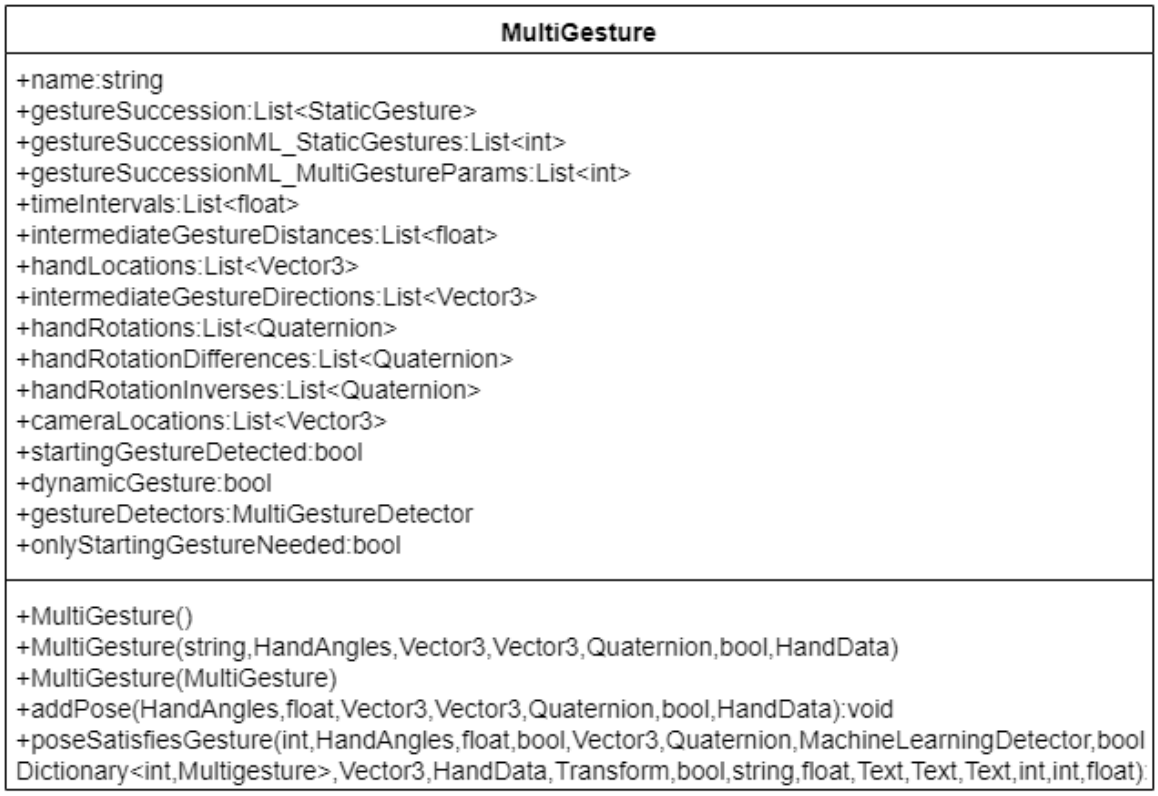


Figure 3.12: `MultiGesture` class

gesture segment. The list `cameraLocations` is used to define the locations of each gesture segment with respect to the camera location for positional gesture tracking. The flag `startingGestureDetected` indicates when to start tracking the segments of a multigesture by signaling when the starting gesture has been performed. The list `gestureDetectors` corresponds to a list of `MultiGestureDetector` objects which perform the bulk of the gesture segment tracking as explained with the next class presented. Just as with the `StaticGesture` class, there is a member function named `poseSatisfiesGesture` which passes along multi gesture parameters to a corresponding model to perform predictions. There is also a member function named `addPose` which is used during the training process to add more poses or gesture segments to a multi gesture to be trained.

3.4.4.7 `MultiGestureDetector`

This class performs the bulk of the gesture recognition process regarding multi gestures. Once a static gesture is performed that is also the start of a potential multi gesture, a `MultiGestureDetector` object is created for that potential multi gesture. This object will then track any subsequent multi gesture segments to determine if all necessary segments within the multi gesture are detected within a certain timeframe. If all segments are detected within that time frame, then the parent `MultiGesture` object is seen as detected. Otherwise, if a non-matching gesture segment is detected, or the detection process goes past the necessary time frame, the parent `MultiGesture` is seen as not detected. This class was adapted from an existing class within the VR-Free API, and updated to accommodate new data parameters deemed necessary for ASL gesture recognition.

The `MultiGestureDetector` class is shown in Figure 3.13. The variable `successionStatus` indicates which segment of a multi gesture object to analyze during the detection process. For example if the `successionstatus` is 2 then the second multi gesture segment will be analyzed. The flag `done` indicates that the detection process is finished for this particular multi gesture, regardless of success or failure, and the flag `detected` indicates whether or not the detection process was successful. The variable `firstDetectedStaticGestureID` corresponds to the id of the static gesture which corresponds to the first segment of the multi gesture. All gesture segments are

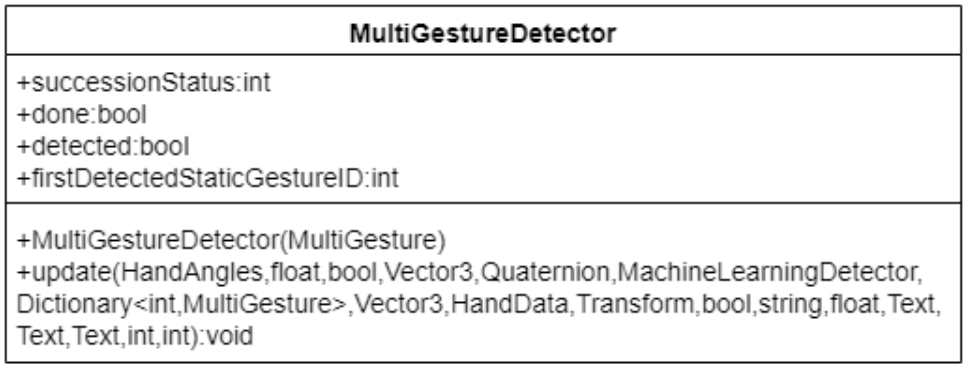


Figure 3.13: `MultiGestureDetector` class

checked within the `Update()` function of this class.

3.4.4.8 `TwoHandedGesture`

This class provides functionality for processing two handed ASL gestures. The class features detection flags for both the left hand and right hand so that they can be handled in parallel. `TwoHandedGesture` objects are created when both training two handed gestures as well as performing recognition on two handed gestures.

The `TwoHandedGesture` class is shown in Figure 3.14. The variables `leftHandGestureName` and `rightHandGestureName` represent the identifiers of each hand's gesture used by the models during the prediction process. The flags `leftHandDetected` and `rightHandDetected` are used to determine whether or not each hand's gesture has been detected. No member functions aside from a constructor are present since the main purpose of this class is to hold flags related to two handed gestures.

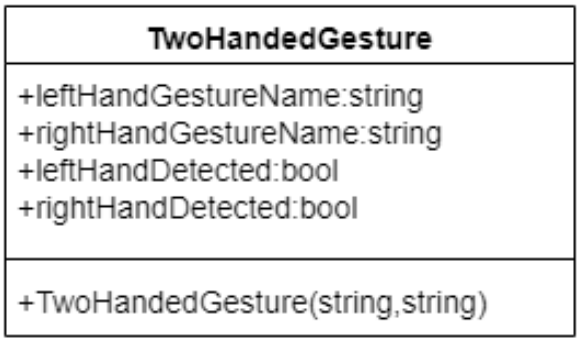


Figure 3.14: `TwoHandedGesture` class

3.4.4.9 `TutorialGesture`

This class is used to store data related to the current gesture being recognized when using the application in tutorial mode. When starting the application in tutorial mode, a list of `TutorialGesture` objects is created corresponding to any gestures that must be performed as part of the tutorial. Any incoming gesture data is compared against the current `TutorialGesture` object in this list when determining recognition.

The `TutorialGesture` class is shown in Figure 3.15. Basic identifiers for the class are present as member variables such as the gesture ID and flags to indicate whether or the gesture is a multigesture or positional gesture. There is also a constructor present to build the `TutorialGesture` object.

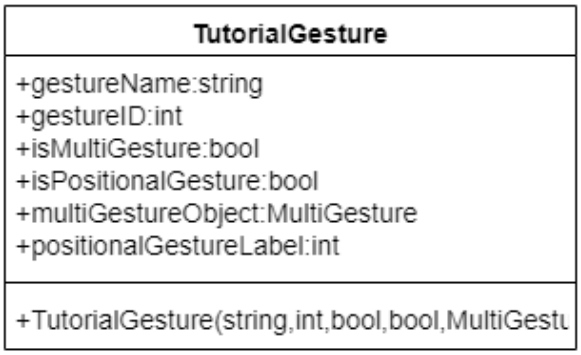


Figure 3.15: `TutorialGesture` class

3.4.4.10 `AccuracyThresholds`

This class is used to store data related to the accuracyThresholds (or confidence thresholds) associated with ASL gestures that can be recognize by the application. These thresholds correspond to the minimum recognition probability that must be met in order for an ASL gesture to be labeled as recognized by the application. This recognition probability is the probability output by each model which states how likely the given gesture parameters correspond to each gesture. The values which are used for accuracy/confidence thresholds for each gesture were determined through testing during the development of the application. Though they also be adjusted at runtime if needed. The `AccuracyThresholds` class is shown in Figure 3.16. Along with the constructor there are just three member variables corresponding to the accuracy thresholds for each gesture model present.

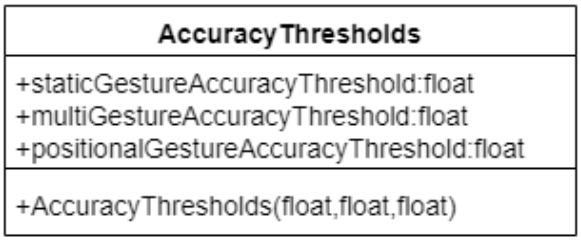


Figure 3.16: `AccuracyThresholds` class

3.4.5 Models

3.4.5.1 Overview

There are three models used on the machine learning side of the application: A static gesture model, a multi gesture model, and a positional gesture model. Each model is implemented through python using the tensorflow framework for model generation and a combination of the numPy and Pandas libraries for data manipulation. Each model has an associated python script that is used to execute the training process. A preprocessor python script is also used to reformat the raw JSON training data into dataframes that could be used easily with the models.

All models used are feed forward neural networks with a softmax function applied to the output layer to force each output value to be a probability. Each output node corresponds to a different ASL gesture, so this probability represents the likelihood that the input features correspond to that particular ASL gesture. Each neural network has an input layer, output layer, and one intermediate layer with a "relu" activation function to handle non-linearities. Each model was trained using 900 to 1000 samples per label, which were generated by three different people. Along with the 34 ASL gestures that were trained to be recognized by the application, 3 unused gestures were also trained which represented false positives that were mistakenly recognized as actual ASL gestures during development. By having these false positive gestures trained, and ignoring them during the inference process, the number of false positives during gesture recognition was reduced. The following sections will outline more specific details regarding each model.

3.4.5.2 Static Gesture model

The static gesture model takes a total of twenty-one features corresponding to parameters of the gloves which are available through the VRFree API. The parameters used in this case are the angles of gaps between various joints of the hand, which are calculated using data from the sensors of the gloves. Each angle of interest has a corresponding data field that is part of a parent "HandAngles" class. Each of the data fields which are used in this application, along with the corresponding hand angles they represent, are explained below:

- `fingerAngles0close` - This data field corresponds to the angle of opening between the first joints of each of the five fingers of the hand. As such, there is one feature value for each finger corresponding to this field, for a total of five.
- `fingerAngles0side` - This data field corresponds to the sideways movement of the first joints of each of the five fingers of the hand. As such, there is one feature value for each finger corresponding to this field, for a total of five.
- `fingerAngles1close` - This data field corresponds to the angle of opening between the second joints of each of the five fingers of the hand. As such, there is one feature value for each finger corresponding to this field, for a total of five.
- `fingerAngles2close` - This data field corresponds to the angle of opening between the third joints of each of the five fingers of the hand. As such, there is one feature value for each finger corresponding to this field, for a total of five.
- `thumbAngle1Side` - This data field corresponds to the sideways movement of the second joint of the thumb. As such there is only one feature value for this particular data parameter.

3.4.5.3 Multi Gesture model

The multi gesture model takes a total of thirteen input features corresponding to the differences between each gesture segment of the multi gesture as well as flags related

to the type of gesture being performed. Each segment of a multi gesture is predicted using this model. For example, the letter "Z" has four poses or gesture segments, so there are four output predictions that correspond to a full multi gesture of "Z" being successfully performed. Each of these gesture segments has a set of feature values which correspond to it's difference from the previous gesture segment that was performed. The differences between each segment of the multi gesture are quantified in three different ways:

- **Direction and distance from previous pose** - This parameter corresponds to the distance vector going from each gesture segment to each subsequent gesture segment. The sign of the corresponding coordinates of this vector can also give us the direction of each gesture segment when compared to the previous one. Each coordinate value of this vector (X, Y, and Z) is isolated into it's own feature, making three features total for this parameter.

- **Direction and distance from starting gesture** - This parameter corresponds to the vector going from the starting gesture to each subsequent gesture segment. The sign of the corresponding coordinates of this vector can also give us the direction of each gesture segment when compared to the previous one. Each coordinate value of this vector (X, Y, and Z) is isolated into it's own feature, making three features total for this parameter.

- **Rotational difference to previous gesture** - This parameter corresponds to the rotational difference between each gesture segment and each subsequent gesture segment. This is done by multiplying the rotation of the previous gesture segment to the inverse of the rotation of the next gesture segment. Each coordinate value of this rotation or quaternion (X, Y, Z, and W) is isolated into it's own feature, making four features total for this parameter.

Other flags are also included as features to make each multi gesture more distinct from others. These include a static gesture id corresponding to the starting gesture

that is performed, a two handed gesture flag to indicate if the gesture is two-handed or one-handed, and a camera location usage flag to determine if the multi gesture is also one that relies on position.

3.4.5.4 Positional Gesture model

The positional gesture model takes in a total of four input features corresponding to the distance and direction of the ASL gesture in reference to the VR headset, and also an id for the static gesture used within the ASL gesture. The distance and direction in this case is gained by the distance vector between the gloves and headset. Using these parameters, the model can predict if a positional gesture is accurate. For example, when the hand is next to the head, it will have a different distance and direction to the headset if it is instead located at the chest.

3.5 User Interface

The UI of the application is a virtual environment generated through Unity. The UI is displayed through the Vive Cosmos VR headset and also through an additional screen on the associated PC for development purposes. The following sections outline various parts of the UI along with screenshots.

3.5.1 Calibration UI

When users first start the application, they are prompted to calibrate the gloves by looking up at a calibrate button as shown in Figure 3.17. When they look up, they will see the calibrate button as shown in Figure 3.18.

Once they look at the calibration button for about one second, an image will appear to perform the first calibration gesture as shown in Figure 3.19. Once this calibration pose is performed two additional images will appear prompting the user to perform the next calibration gestures as shown in Figure 3.20 and Figure 3.21. After the calibration process is complete, the corresponding calibration data is written to a series of files which are used by the VRFree framework to display gloves accurately

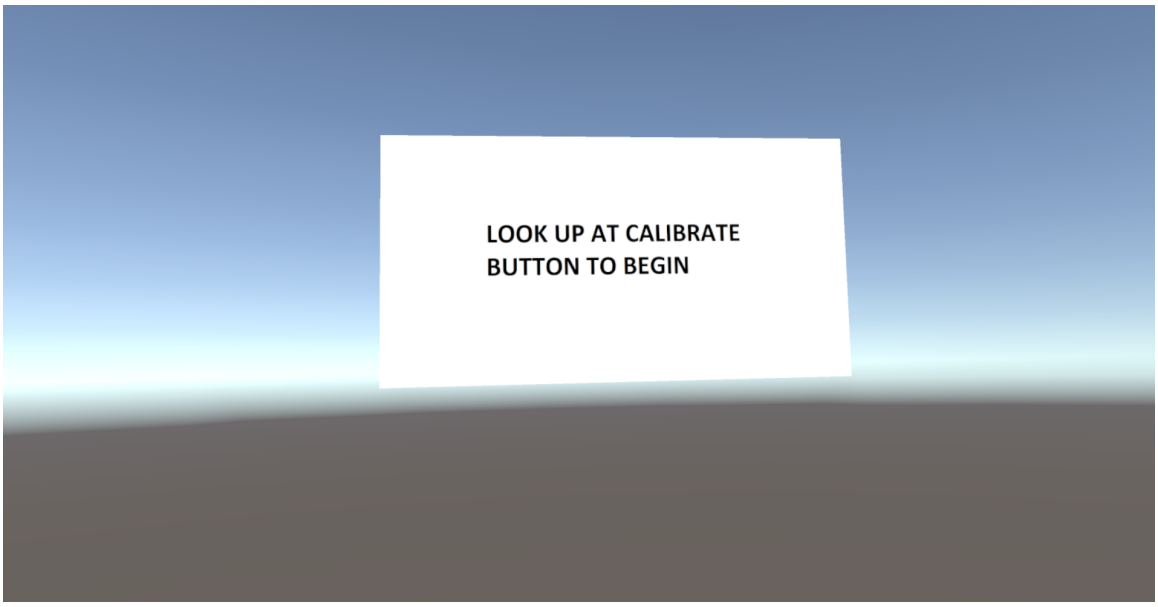


Figure 3.17: The initial dialog the user sees in the application which prompts them to calibrate the gloves.

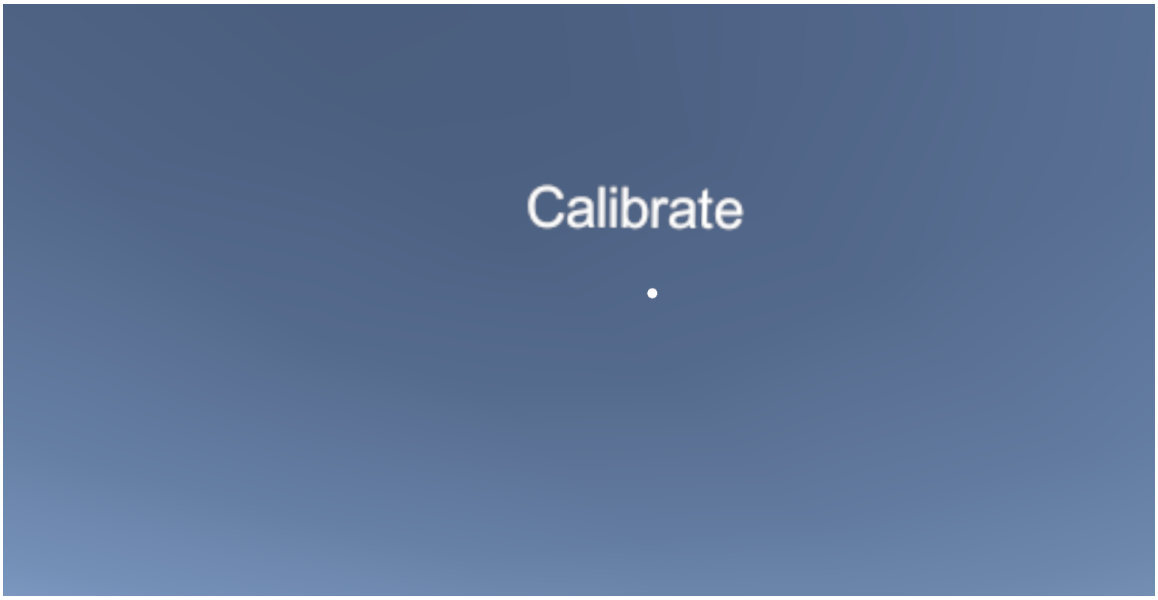


Figure 3.18: The calibration button which the user looks at to start the calibration process.

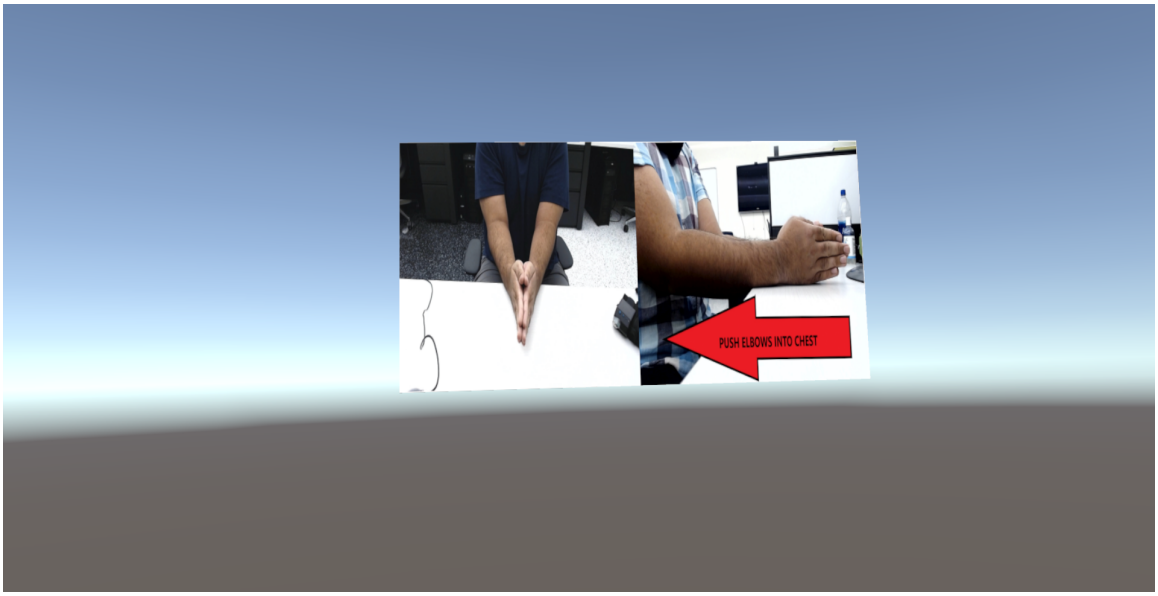


Figure 3.19: The first calibration gesture that is performed in the calibration process.

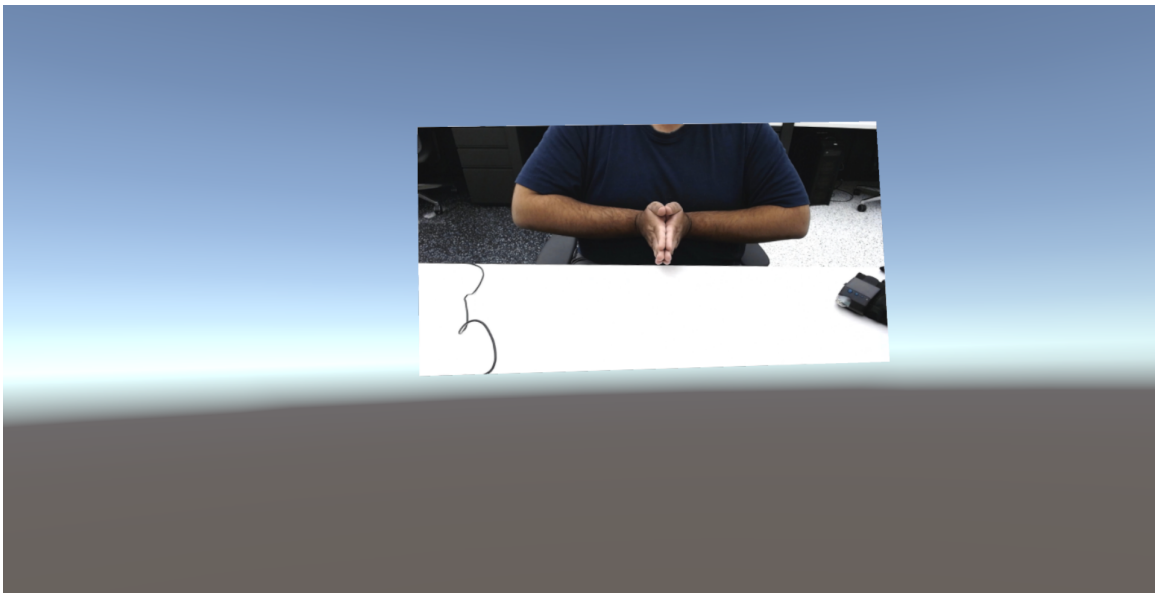


Figure 3.20: The second calibration gesture that is performed in the calibration process.

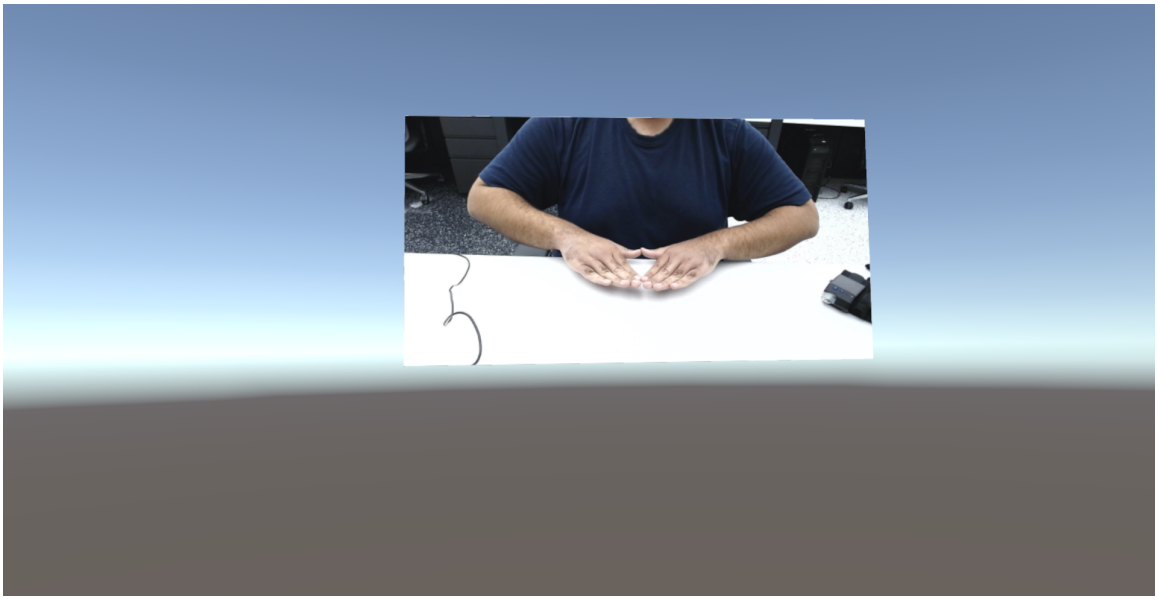


Figure 3.21: The third calibration gesture that is performed in the calibration process.

to the user.

3.5.2 Tutorial UI

After calibration is performed, the user is prompted to perform an ASL gesture by being shown a display of the gesture to perform as shown in Figure 3.22. For simpler ASL gestures, such as most letters of the alphabet, a single image is shown to the user which mirrors the gesture they are intended to perform. Once the gesture is performed successfully, the corresponding letter, word, or phrase is displayed to the user and the virtual hand(s) the gesture is being performed with turns green as shown in Figure 3.23. Each gesture prompt will also have two arrows enabling the user to navigate between different gestures to perform.

More complex gestures, such as multi gestures, have multiple images along with a video to show the user how to perform the corresponding movement. This display is shown in Figure 3.24. When the starting pose of a multi gesture is performed, green text will appear to the user indicating that multi gesture will be tracked on the next movement, as shown in Figure 3.25.

After the multi gesture has started, the green text will change to red and indicate

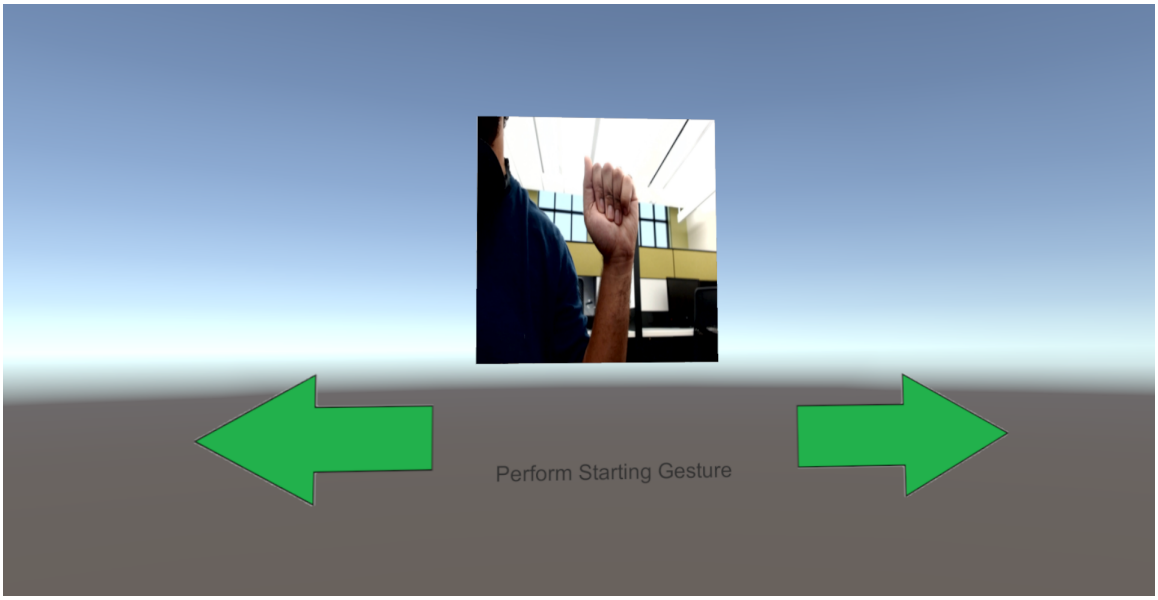


Figure 3.22: The first stage of the tutorial in which the user is prompted to perform the ASL gesture for the letter “A”.

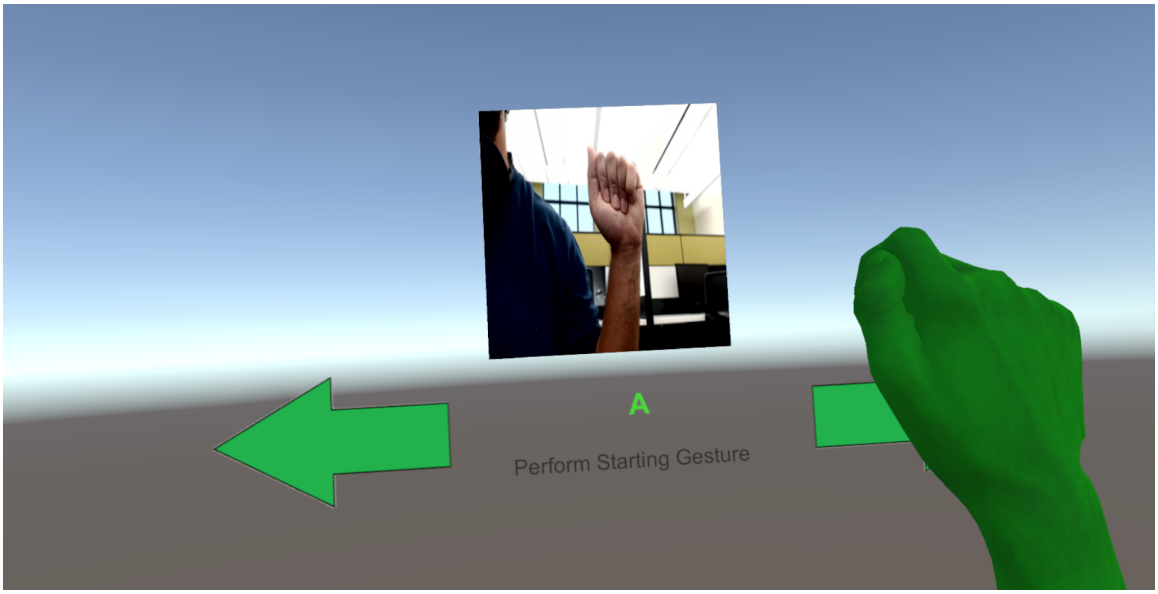


Figure 3.23: The display after performing the first stage of the tutorial successfully.

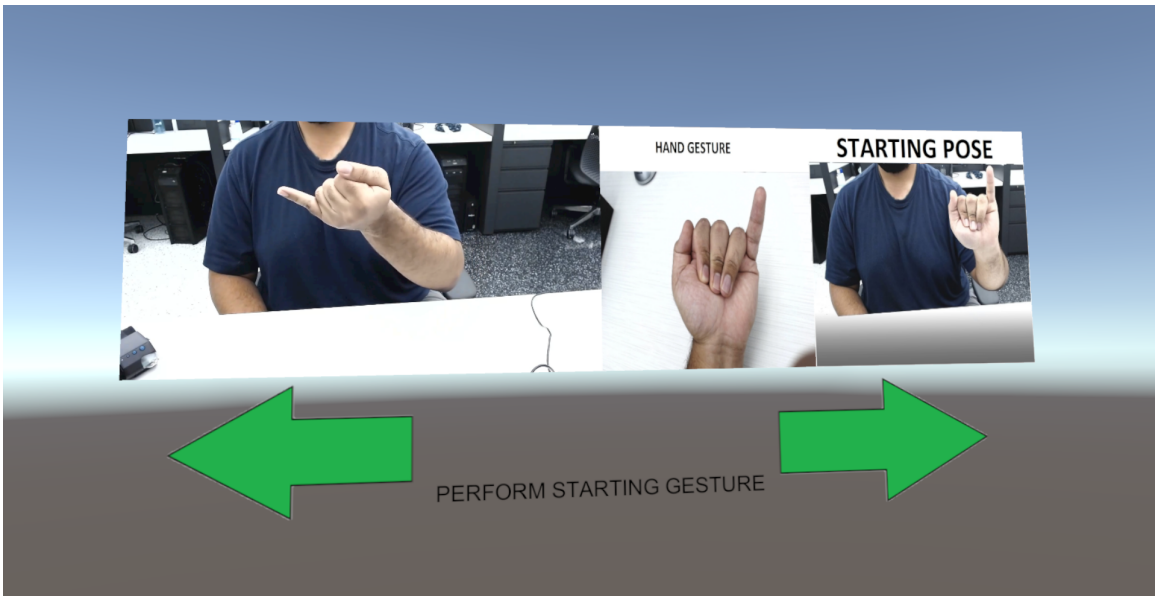


Figure 3.24: A stage of the tutorial in which the user is prompted to perform the ASL gesture for the letter "J".

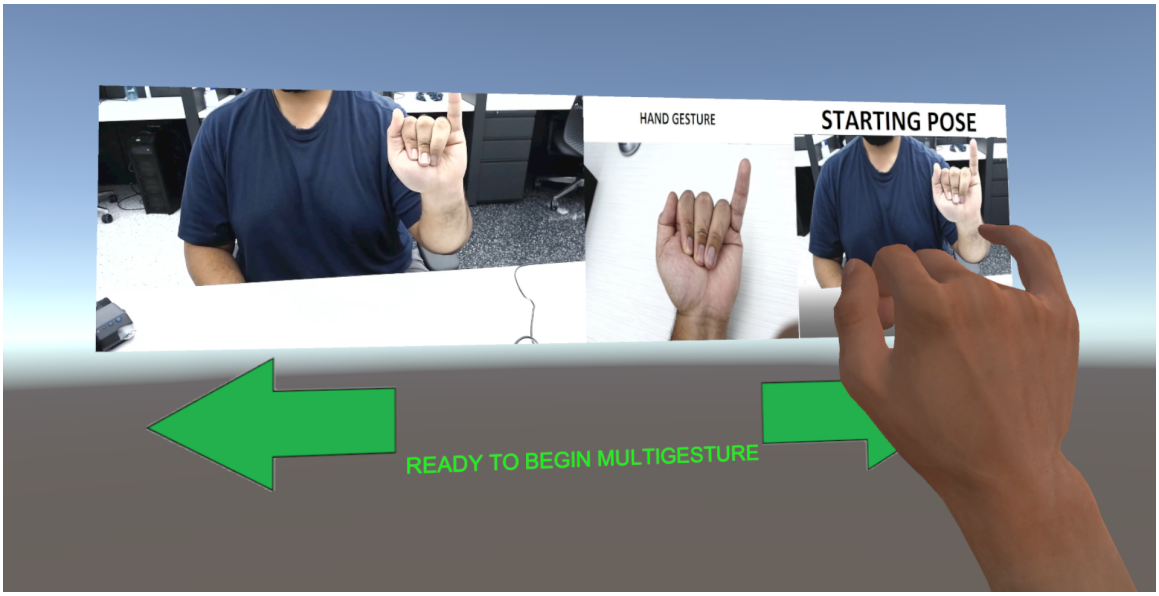


Figure 3.25: A multi gesture prompt in the tutorial after a user makes the starting pose for the gesture.

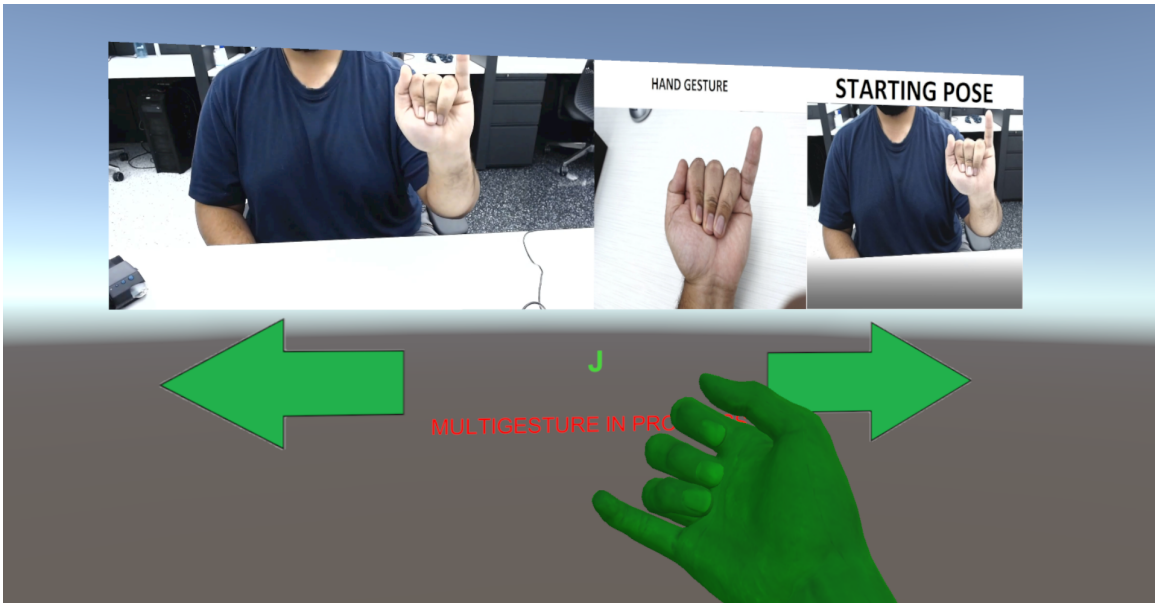


Figure 3.26: A tutorial dialog after the user successfully performs an ASL multi gesture.

that the multi gesture is currently in progress. From that point the user will have a limited amount of time to finish the multi gesture. This is shown in Figure 3.26.

3.5.3 Training UI

The training UI of the application is meant for developers, and as such, features more configurable options than the front-end UI. Screenshots of this UI are shown in Figure 3.27 and Figure 3.28. The training UI incorporates the inspector window of the Unity software, which has a number of fields to configure the training parameters. These include the number of gestures to be performed within multi gestures, the name of the gesture being trained, and others. For two handed gestures, the developer can set for training records to be generated automatically based on a timer and maximum sample size since both hands will be used to perform the gesture. For single hand multi and static gestures, training records are generated through a key-press.

As the training process continues, the colors of the hands change based on the current stage within the training process. For example, when a multi gesture is being trained, and a new gesture segment within that multi gesture is defined, the

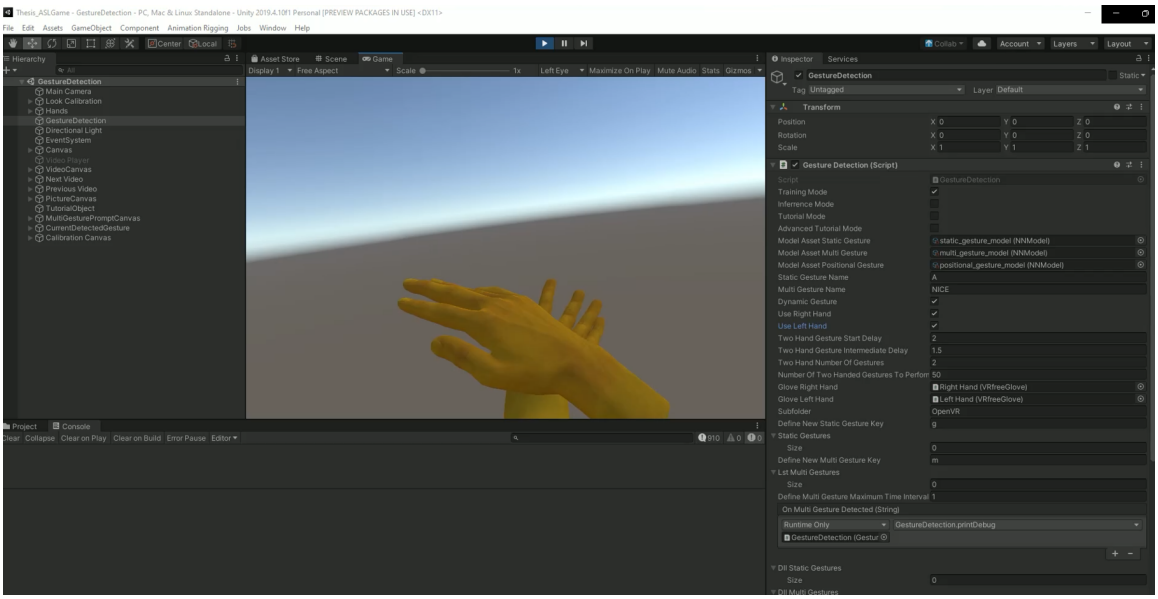


Figure 3.27: Screenshot of the training UI in which a segment in a two-handed multi gesture is being defined.

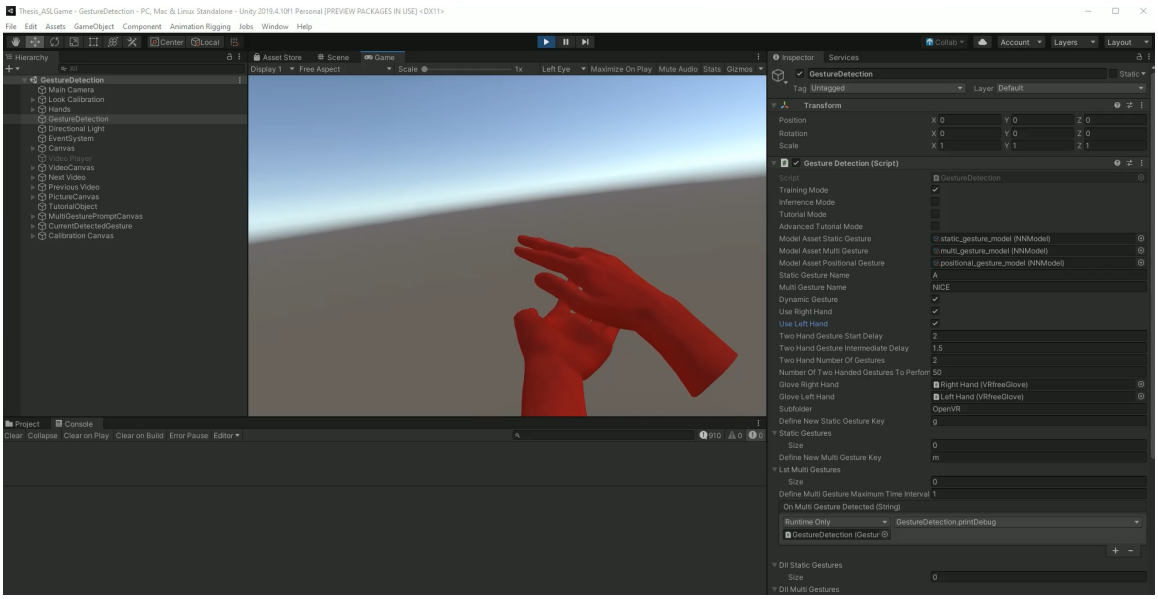


Figure 3.28: Screenshot of the training UI in which the last segment of a two-handed multi gesture is being defined.

hands turn blue. When a two handed gesture segment is being performed, the hands turn yellow, as shown in Figure 3.27. When the entire two-handed gesture has been performed, the hands turn red as shown in Figure 3.28.

Chapter 4

User Study

4.1 Overview

When developing software applications that involve a direct interaction with one or more users, it is crucial to understand the needs of those users and how the software application meets those needs [40]. A large part of this process is gathering data from prospective users, which can be done through a few different ways such as through questionnaires or through observational studies. This type of research was also used to evaluate the ASL gesture recognition application presented in this thesis, to ensure that it adequately met the user needs it is intended to fulfill. The front-end of the application, or the Recognizer, was the primary component evaluated through this type of research, since it is accessed much more frequently than the trainer, and would target a more general user group while the trainer targets those more on the development side.

Prior to performing research using human test subjects, a few steps also need to be performed. The first step is receiving a certification to perform the relevant research through the Collaborative Institutional Training initiative (CITI). This training provides an in-depth look into both the history of perform research using human test subjects, as well as the necessary protocols in order to ensure the safety of users and integrity of the research. Once this is done, the next step is to gain approval to perform the research from the Institutional Review Board (IRB). This involves submitting the CITI training certification along with a number of other documents

related to the research to be performed for the IRB's review. These documents can be found in the Appendices of this thesis.

The remainder of this chapter will go over the details of the user study that was conducted for this thesis. Section 4.2 goes over the participants that were involved in the user study along with the recruitment process. Section 4.3 goes over the tools used to perform the study and the setup process. Section 4.4 outlines the procedure for performing the user study including the necessary steps taken and location. Section 4.5 outlines the tasks users were asked to perform in order to evaluate the functionality and performance of the ASL gesture recognition application. Section 4.6 reviews the metrics that were involved as a part of the user study and their relevance.

4.2 Participants

The participants of the user study varied among different ages, backgrounds, and familiarity with VR and ASL. They included mostly students at the University of Nevada, Reno along with family and friends. To recruit participants of the user study, the recruitment email shown in Appendix F was used along with word of mouth. A total of 21 participants were recruited in the study. In order to participate, users were also required to have a hand size that fit appropriately within the VRFree gloves to be used during the study. This was determined through a hand sizing guide that was provided to each participant. Users were also required to be right-handed to match the gesture types that were generated by the trainers of the application.

4.3 Apparatus

The apparatus for the user study included a Windows 10 desktop computer, a Vive Cosmos VR headset, and a pair of the VRFree motion tracking gloves by sensoryx. A user equipped with the VR headset and gloves is shown in Figure 4.1 The application was run through the Unity IDE. A view of the VR environment, a logging window, and a inspector panel were displayed through unity during the user study process.

Figure 4.1: A user equipped with the two primary components of the apparatus: the Vive Cosmos VR Headset, and the VRFree Gloves.

A view of the VR environment was necessary so that the researcher leading the user study could see what the user was seeing while the headset was equipped and perform guidance as necessary. The logging window was used to observe quantitative metrics during to the process to assist with troubleshooting if needed and observe important trends. The inspector panel was used to make adjustments to the accuracy/confidence thresholds of gestures to be recognized as necessary. These adjustments were made if participants were unable to perform gestures within a significant amount of time, even with manual assistance by the researcher.

The user was seated throughout the entire user study process. Before the user study began, a horizontal and vertical alignment was performed to maintain a consistent placement among all users that performed the test. Horizontal alignment was done using a line of tape on the floor to indicate how far the user should be from the computer. A vertical alignment was achieved using a mirror with a marking to indicated the adequate height of the user's eyeline. Questionnaires/Surveys were provided to the user through PDF Documents located on the computer as well as hard paper copies.

4.4 Procedure

After participants agreed to take part in the study, they were informed in detail on what the study involves. The participants then were provided with a pre-test survey which asked them about their demographics, familiarity with virtual reality, familiarity with sign language, familiarity with motion tracking hardware/software, and familiarity with electronic entertainment such as video games. Participants were then provided with a simulator sickness questionnaire which asked them if they are currently experiencing any symptoms associated with simulator sickness, and how severe those symptoms are. The participant was then given a demonstration of the tasks that needed to be performed during the study. Once the demonstration was over, the participant began performing the tasks themselves with assistance being provided from the researcher as needed.

The tasks of the study involve performing a series of American Sign Language (ASL) gestures as prompted by the VR application while equipped with the VRfree Sensoryx Motion Tracking gloves and the Vive Cosmos VR Headset. After putting on the headset and gloves, the process began by powering on the gloves and starting the VR application from the PC. The gloves were then calibrated through a routine in the application which involved the participant performing three poses. Sometimes multiple calibrations were needed if the 3D hand models in the virtual environment did not accurately match the locations of the hands in the real world. Once calibration had been performed successfully, the participants then followed a series of prompts within the virtual environment which each instructed them on how to perform a specific ASL gesture. This instruction was conveyed through both images and video. When the user was able to perform the ASL gesture successfully, they were asked to move on to the next one by looking at a button in the virtual environment.

If participants were not able to perform a gesture successfully from observing the prompt alone, the researcher would assist the participant either by providing verbal instruction or manually manipulating the user's hands. If the participant was still unable to perform the gesture successfully, the researcher would either prompt the user to re-calibrate the gloves, or adjust confidence thresholds used by the recognition application. This process was repeated for a total of 34 ASL gestures. Throughout this process, data from three quantitative metrics was gathered including the time necessary to perform each gesture successfully, the number of attempts needed to perform a multi gesture successfully, and the number of times a re-calibration was needed. This data was collected through logs generated within the VR application as well as observation by the researcher. Once all gestures had been performed successfully, the participants then removed the VR headset and gloves with the researcher's assistance.

The study concluded with a post-test survey and a second simulator sickness questionnaire meant to determine if any symptoms were being experienced after going through the study. The post-test survey asked the participant some questions

regarding their overall experience while performing the tasks. These included overall comfort while performing the tasks, difficulty performing each gesture, intuitiveness of the user interface, usefulness of the gloves in teaching ASL hand gestures, potential improvements that can be made, other possible applications for the gloves, and general feedback along with any other general comments and observations.

4.5 Tasks

More detail regarding each of the tasks user study participants were asked to perform are outlined below:

Calibrate Gloves: When the application first begins, the user is required to calibrate the gloves to accommodate their hand size and body type. This involves performing three different calibration gestures that have been defined by the VRFree API. The user begins the calibration by looking up at the calibration button within the virtual environment for couple of seconds which then makes the first calibration gesture prompt appear to the user. When the researcher has determined each calibration gesture is being performed accurately, the user is prompted to hold the gesture for a few seconds as part of the calibration process before moving on to the next. After the first calibration process is complete, the user could optionally perform another calibration at any other time during the user study as an effort to troubleshoot issues when prompted by the researcher.

Observe Gesture Prompts: When a calibration is not being performed, the user is prompted to perform 1 of 34 ASL gestures that have been trained for the application. Each prompt appears as a visual cue of which gesture to perform in the form of an image, a video, or both. Each visual cue displays the researcher performing the corresponding gesture in a mirrored view. In some cases, gestures are also displayed in a first-person view for clarity. If the user encounters issues when performing gestures, they are able to observe these visual cues to verify there are no issues with their

form or movement. If the user continues to encounter difficulties after attempting to match the visual cue, they are assisted by the researcher.

Perform ASL gestures: Once the user feels confident that they can match the visual cue, they are asked to attempt to perform the corresponding gesture. This involves attempting to match both the finger and hand placement depicted in the prompt, and in the case of multi gestures, matching the movement performed in the displayed video. The gestures the user is prompted to perform include all 26 letters of the ASL alphabet along with 8 words and phrases. Some gestures involve just using a single hand while others involve two hands. If the user continues to encounter difficulties after multiple attempts, the researcher helps to guide them, and in severe cases, helps physically manipulate their hands and fingers to be in the appropriate position.

Navigate between gestures: Whenever the user is prompted to perform an ASL gesture, there are also two arrows displayed to the user: one pointing to the left and one pointing to the right. When the user completes a gesture successfully, they are prompted to advance to the next gesture by looking at the arrow to the right. If the user accidentally navigates to the next gesture before previous one is performed successfully, they are prompted to go back to the previous gesture by looking at the left arrow before continuing.

4.6 Design

Three quantitative metrics are taken during each user study session. These include the amount of time necessary to perform each gesture, the number of attempts in performing each multi gesture, and the number of calibrations performed. These metrics were evaluated to determine capabilities of the gloves and application in recognizing ASL gestures, and any limitations when performing gesture recognition on multiple users. Qualitative metrics were gained through a post-survey which gained

input from users regarding the components of the application along with it's usefulness if different areas.

The user study was designed using the within-subjects method which involves each participant testing all interfaces of the application under the same conditions [2]. This is opposed to a between-subjects design which involves each person testing a different interface of the application. The dependent variables of the application include the three quantitative metrics introduced previously. The ASL gesture recognition application works as the single independent variable of the study.

Chapter 5

Results and Data Analysis

The following sections outline the results that were obtained during the user study. Section 5.1 reviews data gathered through a pre-questionnaire before the tasks involved in the user study were executed. The data gathered here includes the participant's backgrounds and their familiarity with related concepts. Section 5.2 reviews the quantitative results of the measurements obtained from each task. Section 5.3 details the data obtained from the post-questionnaire outlining feedback from the participants regarding the application. This section also reviews the results of a simulator sickness questionnaire that was used to determine if usage of the application exacerbated any symptoms of simulator or VR sickness. Finally, Section 5.4 presents a discussion regarding the significance of the results and the possible reasons for the data distribution.

5.1 Data From Pre Questionnaire

Figure 5.1 shows a graph outlining the ages of all participants who took part in the study. As the graph shows, most participants were in their 20's or 30's with a few being younger and older. Most of the participants were either current or graduated UNR students with some having education from other institutions.

Along with age and educational background, the pre-questionnaire also asked participants to rate their familiarity regarding different technologies and concepts on a scale of 1 to 5, with 1 being not familiar, and 5 being very familiar. First the user

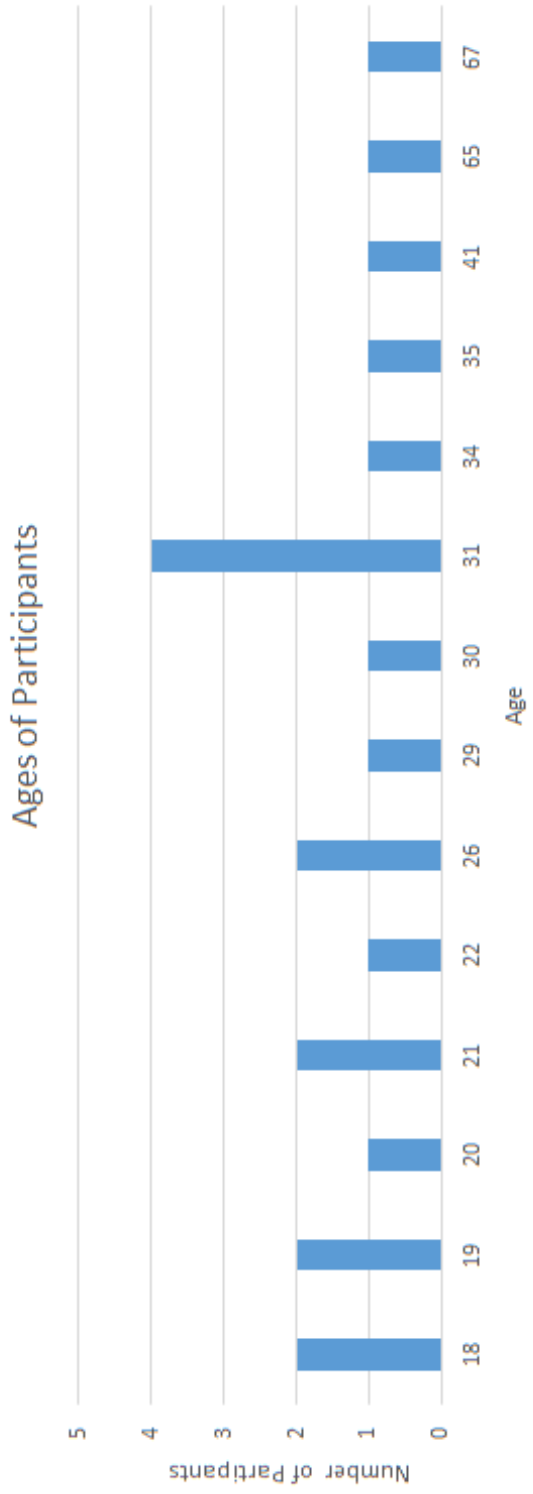


Figure 5.1: Ages of user study participants

was asked how familiar they are with virtual reality. This results of this question are shown in Figure 5.2. Four users stated they were not familiar with VR while six stated they were very familiar. The majority of others rated their familiarity between a two and three.

Next the participants were asked to rate their familiarity with American Sign Language (ASL). The results of this question are displayed in Figure 5.3. Here the majority of users were not familiar with ASL or only slightly familiar. Two participants felt they were more familiar with ASL, rating themselves at a four.

Participants were also asked to rate their familiarity with motion tracking software and hardware as well as forms of electronic entertainment such as video games. Regarding motion tracking hardware and software, the majority of the participants stated that they were not familiar with a rating of one. The others were split fairly evenly between two and five. Regarding familiarity with electronic entertainment, the majority of participants felt that they were very familiar providing a rating of five while most others provided a rating of three or four. One participant provided a

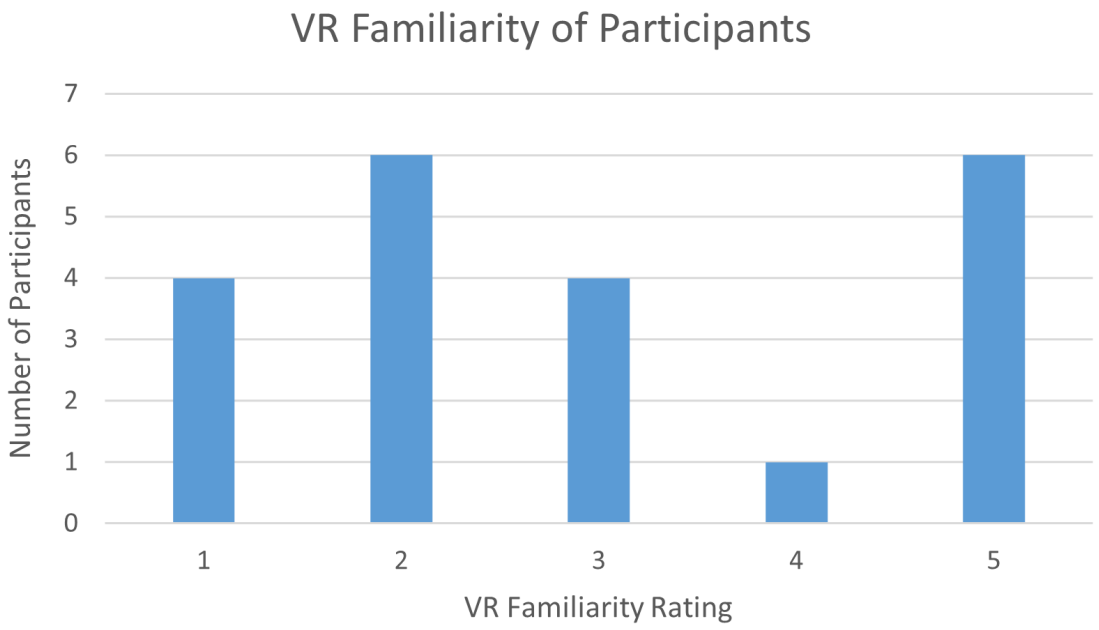


Figure 5.2: VR familiarity of user study participants

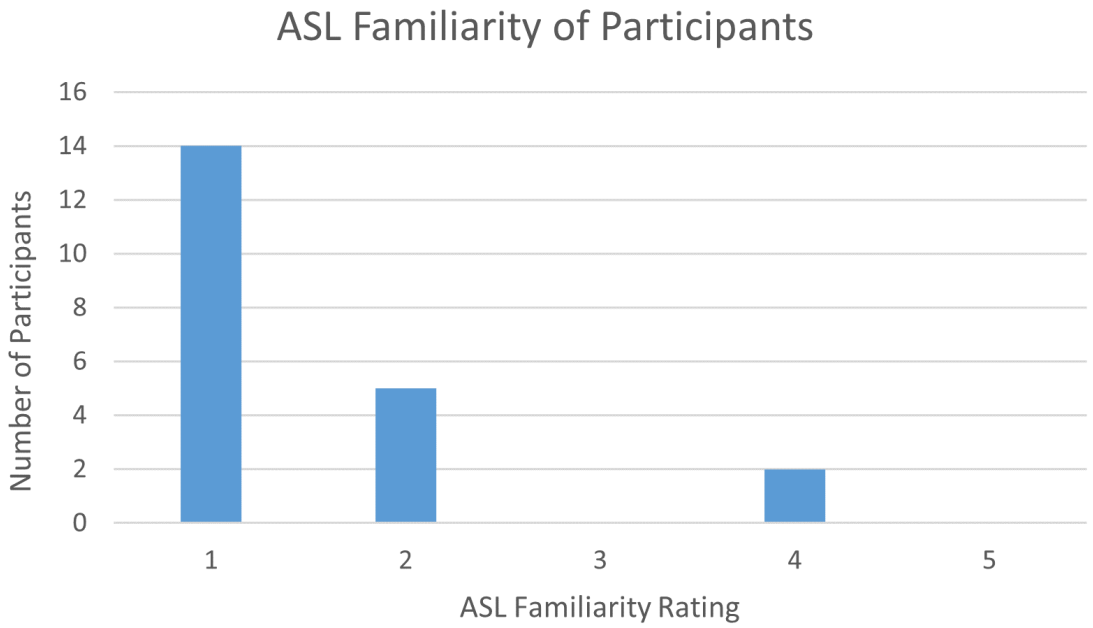


Figure 5.3: ASL familiarity of user study participants

rating of two indicating they were not as familiar with electronic entertainment such as video games.

5.2 Data Analysis

Each participant was asked to perform thirty-four ASL hand gestures while having them successfully recognized by the application. In many cases, multiple attempts were needed to perform each gesture and have them successfully recognized. This was due to a few different reasons including gestures not being performed correctly, the motion tracking gloves needing to be re-calibrated, or confidence thresholds used in the recognition process needing to be adjusted. Because of this, data was gathered to evaluate the following three metrics: The amount of time needed to perform each gesture, the number of attempts need to perform multi gestures, and the number of calibrations performed by each participant. Each of these resulting data sets were aggregated into tables which are presented in this section. The amount of time needed to perform each gesture along with the attempts needed to perform each multi gesture

are also aggregated into box plots showing the analysis of variance (ANOVA) of these metrics for each gesture.

Tables 5.1 and 5.2 show the amount of time needed to perform each letter of the ASL alphabet. The tables also show the mean completion time along with standard deviation for each gesture. Table 5.3 shows these same measurements for the eight ASL words and phrases that were also measured as a part of the study. Each row of each table corresponds to a participant of the user study, who is identified by an ID number between 1 and 21.

Graphs depicting box plots for each of these data sets are shown in Figures 5.4, 5.5, and 5.6. Box plots, also known as Box and Whisker plots, are a standard way of presenting five key components relating to the distribution of a dataset: the minimum (excluding outliers), the first quartile, the median, the third quartile, and the maximum (excluding outliers). The lower whisker of the box plot shows the minimum value while the upper whisker shows the maximum value. The line within each "box" shows the median of the data set. The bottom half of the box depicts the first quartile of the dataset and the top half depicts the second quartile. In the graphs presented, dots are also display indicating each data point including outliers. An "x" symbol is used to represent the mean of each gesture. Extreme outliers are indicated through boxes with arrows pointing towards the top of the graph.

As shown in Figures 5.4 and 5.5, the majority of performance times for letters A-Z range between 1 to 10 seconds indicating that most ASL gestures were performed and recognized by the application quickly. Letters that had significantly higher performance times include "A", "G", "J", "P", "Q", and "Z". Since the letter "A" was the first gesture performed by participants, it is likely that participants were still getting comfortable with using the application at this time, which may contribute to the higher variance in this case. The letters "J" and "Z" have the highest variance among the alphabet, showing that participants had more difficulty with multi gestures rather than static gestures. This was partly due to these gestures being more complex than static gestures, and also due to the application not capturing all

Table 5.1: Performance Times in Seconds for Letters A-Q

Performance Times in Seconds for Letters A-Q																	
ID	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**
1	27	3	2	2	2	2	3	6	2	23	2	2	2	2	1	3	2
2	4	2	2	4	5	3	18	7	5	43	4	3	6	8	2	7	2
3	19	4	6	5	9	3	17	6	6	8	3	2	4	2	2	6	3
4	15	10	9	10	9	4	6	4	5	24	69	11	42	20	5	132	25
5	11	3	5	3	8	3	14	31	3	9	11	3	4	4	3	3	3
6	8	5	12	4	25	4	7	11	17	29	9	5	6	4	6	15	7
7	5	3	9	4	9	2	33	17	5	44	12	3	6	4	4	4	5
8	4	2	2	6	2	2	18	2	2	4	14	2	3	3	4	4	2
9	10	4	3	7	1	3	14	3	2	6	3	2	2	7	3	6	3
10	16	7	4	6	35	5	15	7	5	25	5	2	7	10	3	4	7
11	3	2	3	3	2	2	8	2	5	4	2	2	2	2	4	4	2
12	10	4	4	5	15	4	5	4	4	13	5	4	10	5	7	10	7
13	12	3	3	2	33	3	9	1	2	8	5	2	5	3	2	13	7
14	8	2	2	2	6	3	19	3	3	18	10	3	4	3	3	4	5
15	11	4	3	4	5	3	4	4	42	151	8	3	3	4	3	3	7
16	12	2	2	2	3	2	12	2	2	9	2	2	2	3	2	2	1
17	4	2	1	2	2	2	9	3	4	9	10	2	2	3	2	2	3
18	9	3	24	15	3	3	7	2	86	268	8	2	3	7	3	4	7
19	38	22	5	8	4	10	13	6	9	15	11	5	24	8	5	26	14
20	18	4	4	7	4	3	8	8	8	17	11	3	4	4	4	25	5
21	13	4	9	3	4	3	8	8	6	36	17	4	3	6	7	13	4
Mean	12.24	4.52	5.43	4.95	8.86	3.29	11.76	6.52	10.62	36.33	10.52	3.19	6.86	5.33	3.57	13.81	5.76
SD	8.12	4.33	5.05	3.12	9.74	1.69	6.73	6.56	18.89	60.16	13.73	1.99	9.14	3.94	1.62	27.29	5.17

Table 5.2: Performance Times in Seconds for Letters R-Z

Performance Times in Seconds for Letters R-Z									
ID	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
1	3	1	2	2	2	2	2	2	7
2	53	2	3	2	3	3	4	2	14
3	11	28	5	21	12	4	4	3	13
4	28	21	34	6	3	4	24	11	25
5	3	3	3	3	2	3	2	2	6
6	9	7	27	6	6	4	5	4	27
7	23	4	3	3	3	3	2	2	27
8	7	3	5	2	1	2	2	2	20
9	2	3	2	1	1	2	2	2	8
10	3	5	3	3	2	2	5	2	30
11	4	37	3	4	2	2	2	2	6
12	17	5	37	14	4	5	4	3	8
13	12	2	2	3	2	2	3	2	22
14	5	3	4	4	3	2	3	4	15
15	7	4	4	3	4	3	3	3	96
16	31	2	1	1	2	2	2	2	108
17	10	2	2	3	16	7	3	2	24
18	14	3	3	4	4	3	3	4	11
19	6	6	9	4	8	6	6	5	8
20	20	4	13	4	7	4	5	6	53
21	7	11	3	10	3	3	5	3	53
Mean	13.1	7.43	8	4.9	4.29	3.24	4.33	3.24	27.67
SD	12.09	9.26	10.52	4.64	3.67	1.38	4.57	2.07	27.51

Table 5.3: Performance times in Seconds for ASL words and phrases

Performance times in Seconds for ASL words and phrases								
ID	**HELLO**	**MY**	**NAME IS**	**I AM**	**LEARNING**	**SIGN LANGUAGE**	**NICE**	**TO MEET YOU**
1	8	15	12	3	11	17	6	6
2	10	3	35	4	31	12	5	8
3	5	3	61	7	93	22	6	2
4	10	6	301	6	283	20	11	18
5	8	4	11	6	5	13	9	11
6	13	6	159	13	28	21	11	23
7	10	3	8	7	36	114	5	14
8	7	2	21	3	41	51	6	16
9	9	6	16	5	50	15	6	21
10	21	2	31	4	21	9	3	17
11	5	2	11	6	183	19	6	34
12	11	34	14	10	14	40	13	66
13	33	13	18	7	30	27	8	15
14	8	3	10	8	192	5	6	39
15	5	7	5	4	46	11	6	4
16	22	3	12	5	67	35	14	20
17	5	3	20	3	28	57	6	21
18	10	3	8	6	8	12	5	7
19	10	5	12	6	1	4	10	23
20	7	11	37	14	23	23	16	26
21	12	5	22	4	17	13	8	13
Mean	10.9	6.62	39.24	6.24	57.52	25.71	7.9	19.24
SD	6.63	7.07	66.85	2.93	71.31	24.02	3.32	13.85

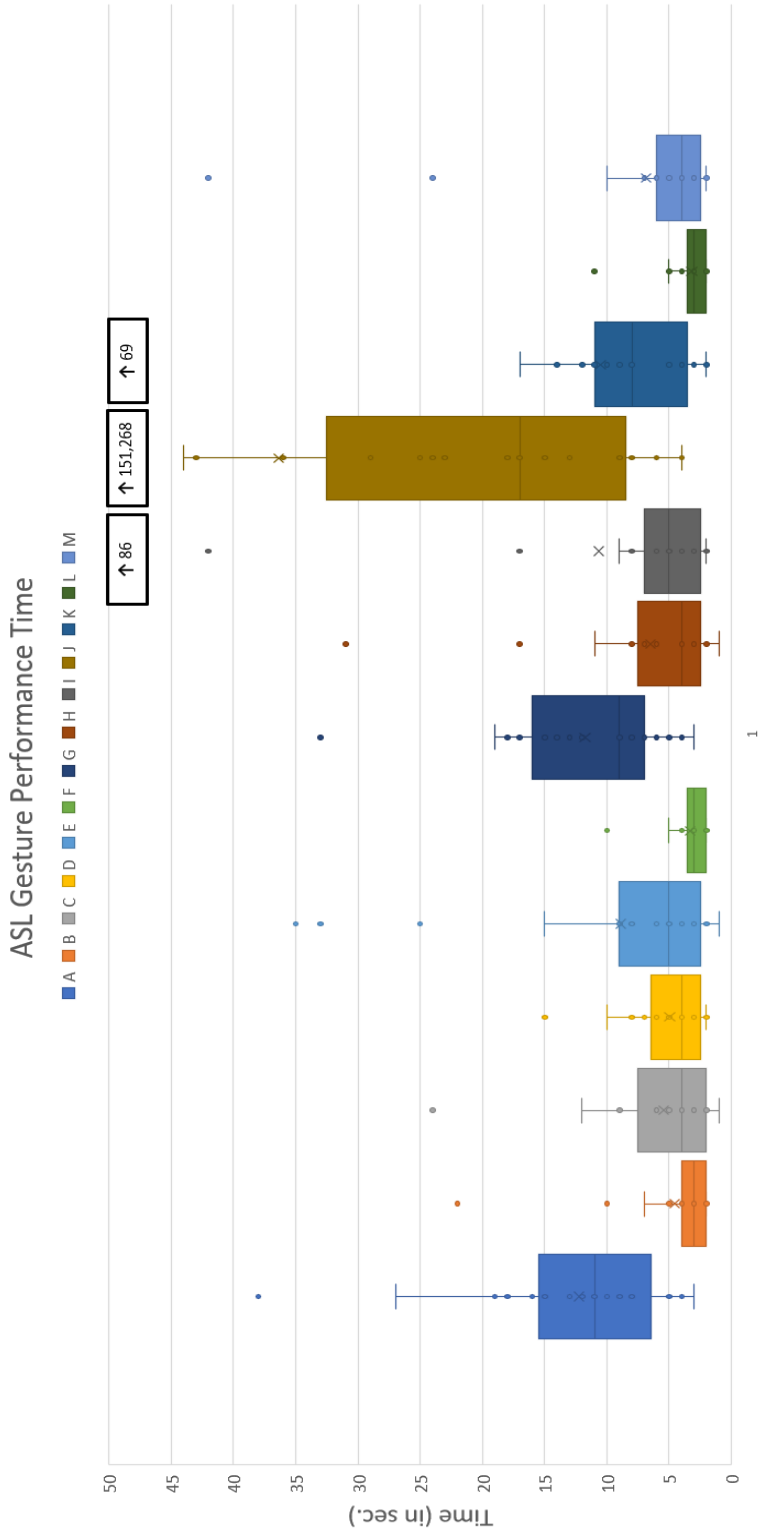


Figure 5.4: Box and whisker plot depicting performance times for letters A-M.

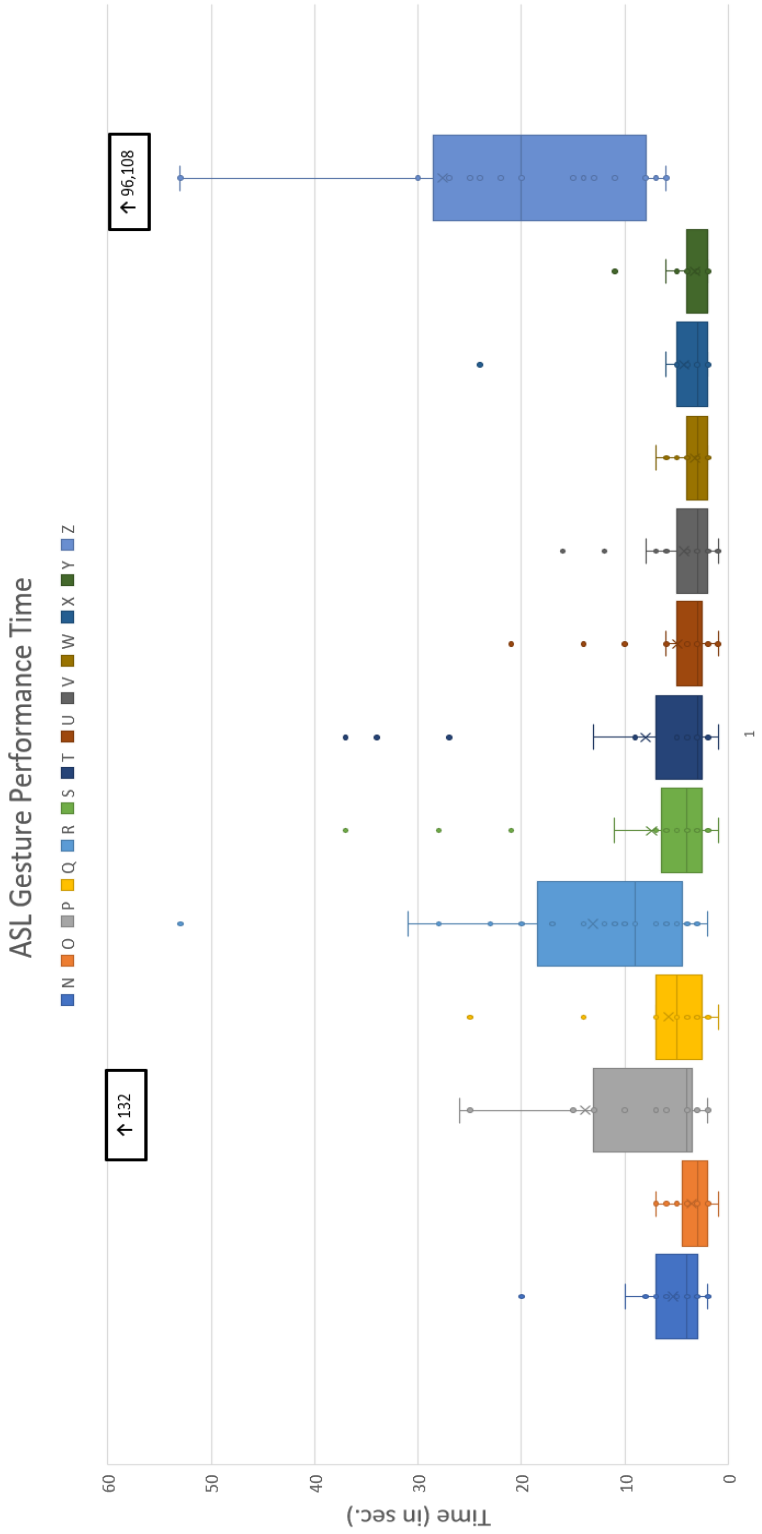


Figure 5.5: Box and whisker plot depicting performance times for letters N-Z.

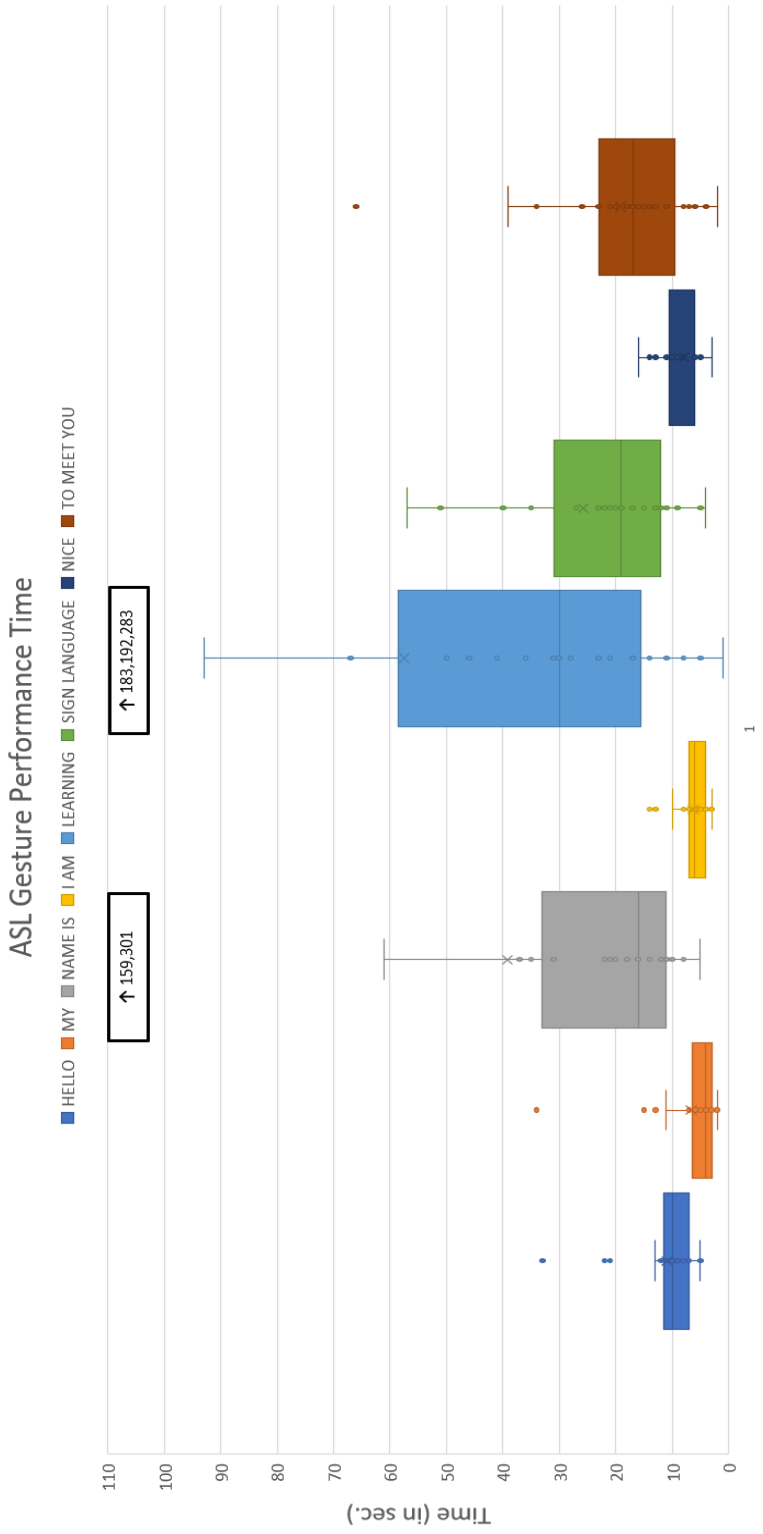


Figure 5.6: Box and whisker plot depicting performance times for ASL words and phrases.

possible variations of the gestures that can be performed. There are also a number of outliers in each case with extreme outliers being present with the letters “I”, “J”, “K”, and “Z”. While some of these outliers are due to the same aforementioned difficulties, the more extreme cases may also be due to participants having bodily features that did not adhere closely enough with the those of the researchers who trained the models used for recognition. Such features include hand size, height, arm length, and others. While the study attempted to constrain hand size to match those that could fit well within the motion tracking gloves, there was still some variability in this and other body features that could be contributing factors to outliers. On this same note, the calibration process was limited to only a basic glove calibration in order to facilitate the setup process and confirm that this type of calibration was sufficient. However, other calibrations, such as headset calibration and more advanced glove calibrations may be needed in order to maintain consistent recognition. These same trends are noticed in Figure 5.6 with ASL words and phrases, with the phrases “NAME IS” and “LEARNING” being those with the highest variance. However, with some of these gestures being two handed, the recognition process becomes more complex exacerbating the difficulties previously presented.

Table 5.4 outlines the number of attempts needed to perform each multi gesture for each participant. This data is also displayed through a box plot in Figure 5.7.

The number of attempts was measured for multi gestures and not static gestures because multi gestures have distinct starting and ending points which can more easily be tracked by the application vs. the starting and ending points of static gestures. Most multi gestures attempts ranged between 1 and 10, indicating most multi gestures took minimal attempts to perform successfully. The number of attempts for the multi gestures “LEARNING” and “SIGN LANGUAGE” had the highest data range excluding outliers. The gestures “J” and “NAME IS” had extreme outliers which coincide with their significant performance times shown in the previous figures. As mentioned previously, limited training data was likely a contributing factor to these outliers, as some participants bodily features did not match those of the researchers

Table 5.4: Number of attempts to perform ASL Multi Gestures

Number of attempts to perform ASL Multi Gestures								
ID	**J**	**Z**	**HELLO**	**NAME IS**	**LEARNING**	**SIGN LANGUAGE**	**NICE**	**TO MEET YOU**
1	2	1	3	2	2	2	2	4
2	9	4	4	6	2	3	4	5
3	3	4	2	12	6	3	3	2
4	6	4	2	55	28	5	3	5
5	2	2	4	3	1	2	4	3
6	6	5	2	29	3	3	2	7
7	9	2	3	2	7	22	2	4
8	2	7	3	3	4	10	4	7
9	2	3	5	5	4	3	2	8
10	8	10	6	4	2	2	2	8
11	2	2	2	4	24	5	3	6
12	3	2	2	2	3	8	6	9
13	2	5	6	3	2	5	3	5
14	5	4	2	1	30	21	3	8
15	18	30	2	1	13	1	2	2
16	2	29	5	4	11	7	5	4
17	2	6	2	5	2	12	3	3
18	59	3	4	2	2	2	2	3
19	2	2	4	3	2	13	2	5
20	4	19	2	4	3	6	8	8
21	4	20	4	3	2	2	4	7
Mean	7.24	7.81	3.29	7.29	7.29	6.52	3.25	5.3
SD	12.18	8.62	1.35	12.17	8.76	5.88	1.55	2.15

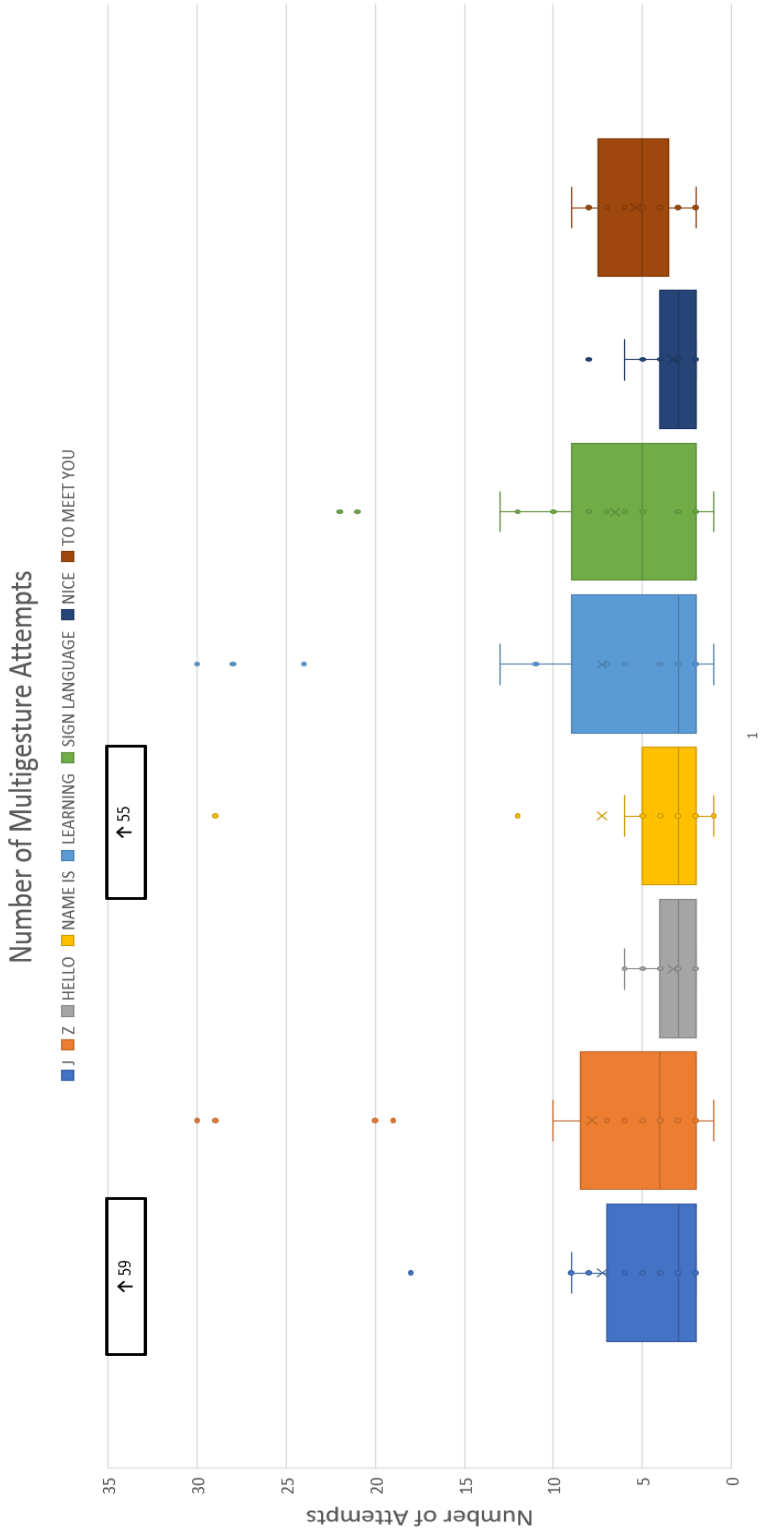


Figure 5.7: Box and whisker plot depicting number of attempts to perform ASL multi gestures

who initially trained the application. More advanced calibration methods may have also been needed in order to keep the number of necessary attempts consistent among multi gestures.

Table 5.5 outlines the number of calibrations necessary for each participant who took part in the user study. Every participant performed the calibration process at least once before any ASL gesture recognition took place. Subsequent calibrations were prompted if a noticeable mismatch was present between the virtual representation of the hands and their real-world counterparts. While most participants only needed a single calibration, one user required three calibrations while two others required 2. This shows in most cases there was an accurate representation of the gloves within the virtual environment. Those that did not likely had bodily features that did not correspond with the training data, such as hand size, arm length, and height. Also as mentioned previously, only a basic calibration occurred, so it is possible that with more advanced calibration procedures, fewer calibrations overall would be needed.

5.3 Data From Post Questionnaire and Simulator Sickness Questionnaire

This section outlines data from the post-questionnaire and simulator sickness questionnaire that were distributed as a part of the user study. The post-questionnaire asked details regarding the participant's opinion of the application after using it as well as their thoughts about the overall process. First users were asked to rate how comfortable they were while using the application. The results of this question are shown in Figure 5.8. Most participants answered with a rating of 5 indicating that they were very comfortable while one participant answered with a rating of 3 indicating they were only somewhat comfortable. Next, participants were asked to rate how easy it was to perform each gesture with 1 being very difficult, and 5 being very easy. The results of this question are shown in Figure 5.9. Most participants provided a rating of 4 indicating that it was easy, but not very easy. The remainder of the participants were split between a rating of 3 and 5. Since most participants were not

Table 5.5: Number of Calibrations Performed for each Participant

ID	**Number of Calibrations Performed**
1	1
2	1
3	1
4	3
5	1
6	1
7	1
8	1
9	1
10	1
11	2
12	1
13	1
14	1
15	1
16	1
17	1
18	2
19	1
20	1
21	1
Mean	1.19
SD	0.5

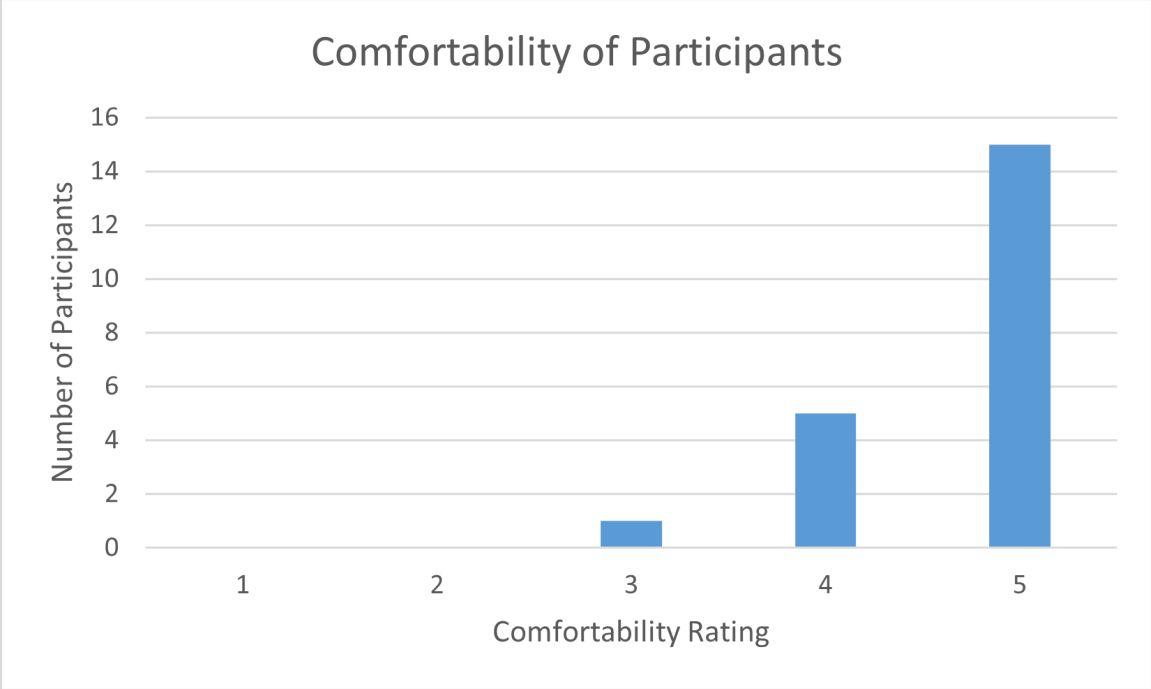


Figure 5.8: Comfort of user study participants

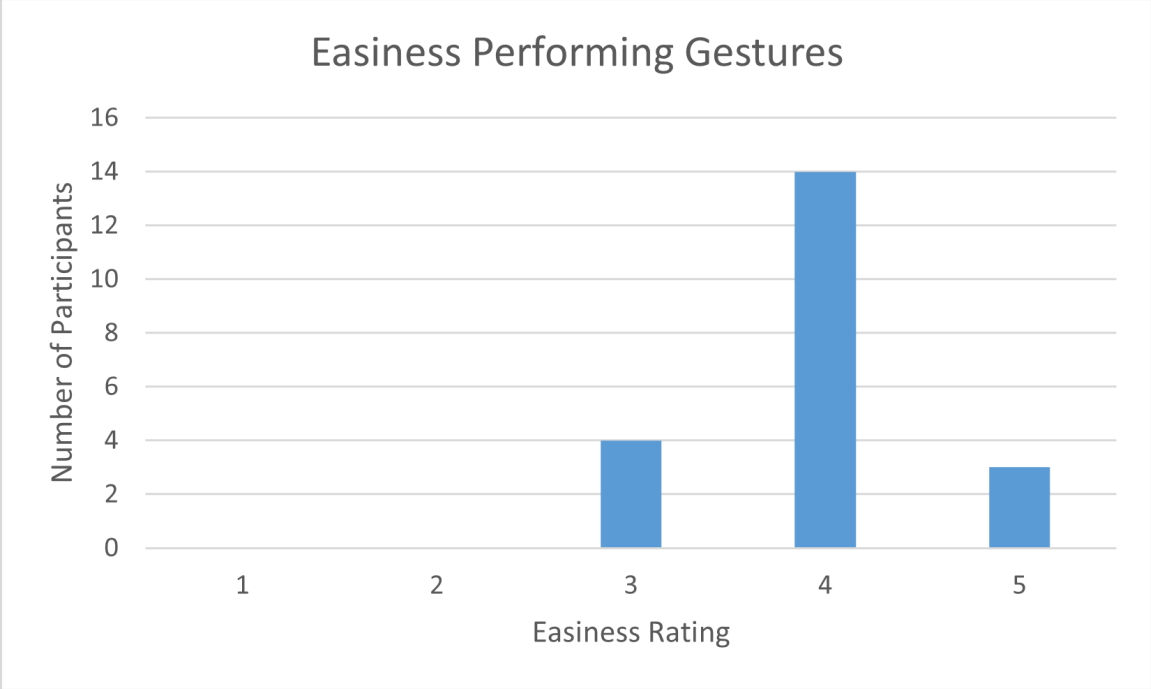


Figure 5.9: Easiness in performing gestures

familiar with ASL prior to using the application, it makes sense that most would not find the application very easy to use, though limitations an training data did also make the application more difficult to use in certain cases.

Next, participants were asked to rate how intuitive they thought the UI of the application was. The UI includes the images and videos which work as gesture prompts, navigational arrows, and calibration prompts. The results of this question are shown in Figure 5.10. Most participants answered this question with a rating between 4 and 5 indicating that they thought the UI was intuitive. Participants were then asked how useful they felt the application was in teaching ASL to potential users. While the application is not necessarily geared towards this specific purpose, the hope is that it could be expanded to become an application which does perform ASL education, as it could be expanded to be other ASL related tools as well. The results of this question are shown in Figure 5.11. Most participants provided a rating of 5 here indicating that they thought the application would be very useful in teaching ASL while others were split between a rating of 1 and 4. Participants were then asked

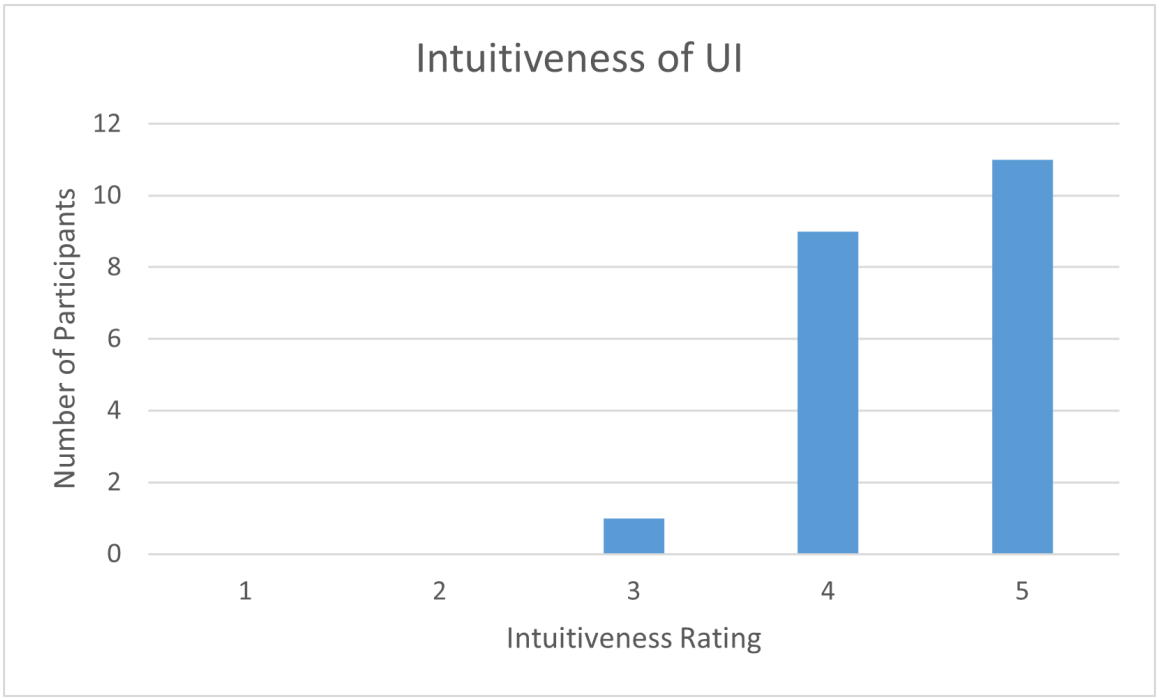


Figure 5.10: Intuitiveness of UI

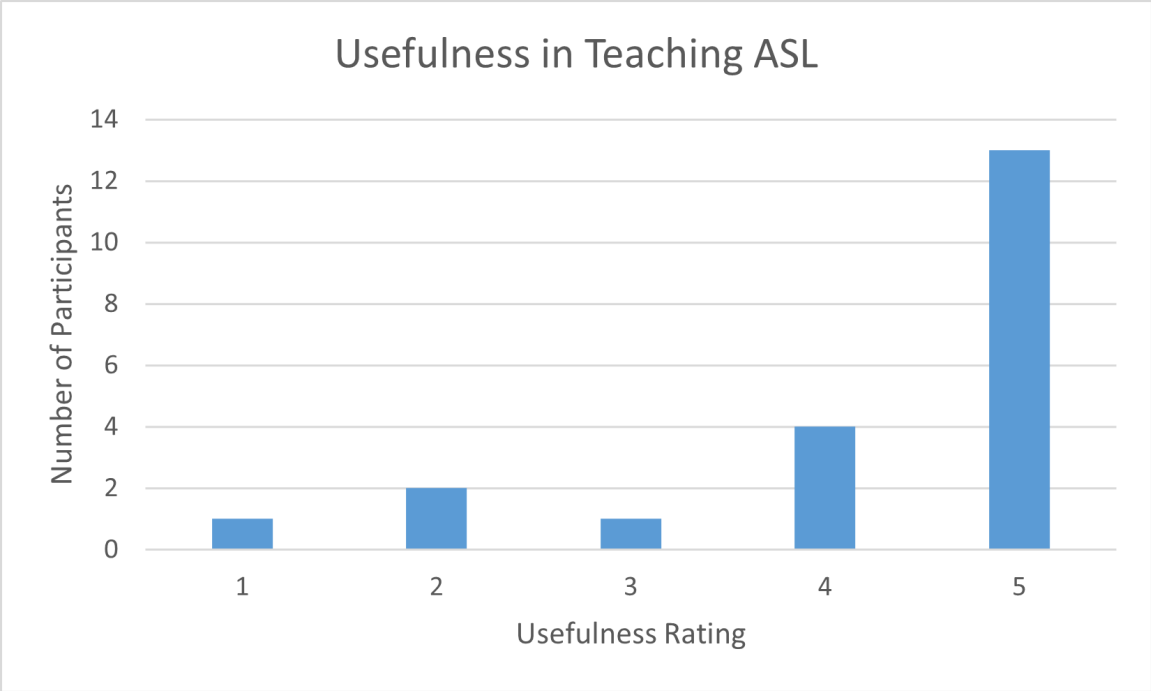


Figure 5.11: Usefulness in Teaching ASL

to rate their overall experience when using the application. These results are shown in Figure 5.12. Move participants provided a rating here between 4 and 5 indicating their overall experience was positive.

Participants were also asked to provide feedback regarding possible improvements that could be made to the application and other general observations. One participant believed that the application could better handle variations of ASL gestures that are performed by different users. This is because not every ASL user signs the same way, and as such, different variations should be caught and recognized through the application in order to increase it's usability. Another participant thought that it would be useful if the application provided more notifications as to what is wrong with a gesture when it is being performed incorrectly, so as to help users more easily correct the issue themselves. Another participant thought that more instruction could be provided aside from the videos and images. Overall, most participants proposed changes to make the UI clearer and have the application provide more guidance regarding how to perform gestures correctly.

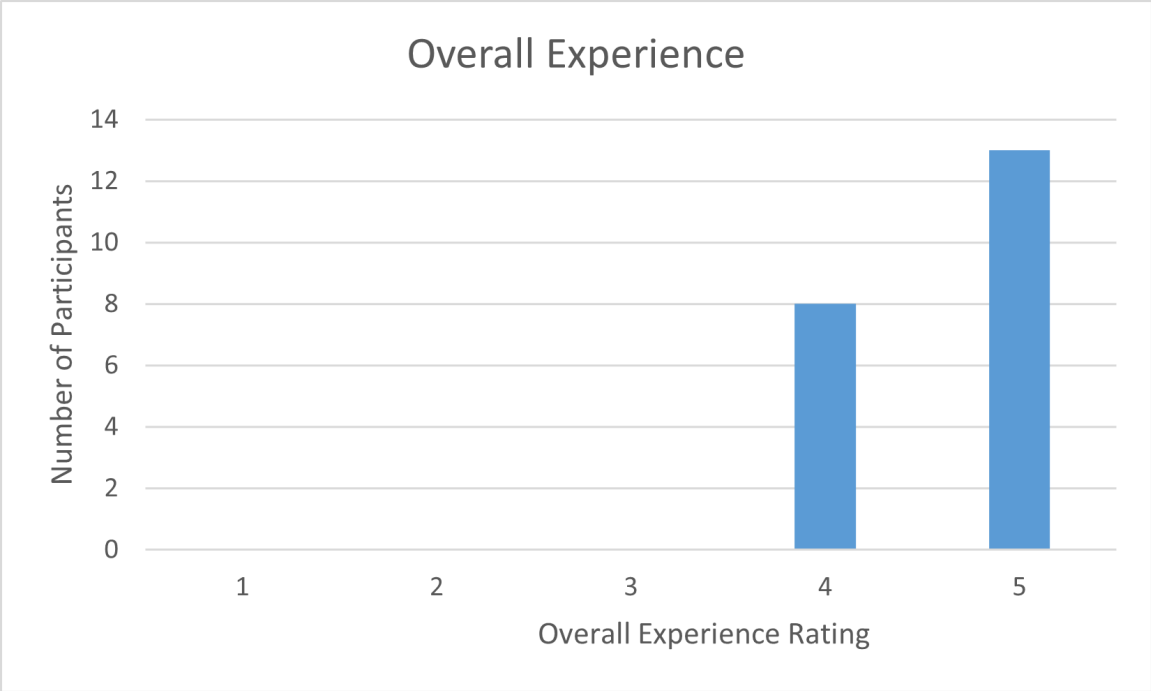


Figure 5.12: Overall User Experience

Participants were also given a simulator sickness questionnaire before and after using the application. Each questionnaire asked the participant to rate the frequency in which they were feeling a number of symptoms related to simulator or VR sickness. The first questionnaire was meant to establish a baseline of how users felt before using the application, so as to determine if any changes occurred after going through it. The second questionnaire outlined any changes in symptoms that occurred after using the application. The majority of the participants had no change in symptoms indicating that the application did not cause or exacerbate symptoms of VR sickness in most cases. A total of 7 participants did report a change in symptoms. The largest change reported by a participant was a moderate feeling of eye strain when there was none previously. Other participants reported a slight increased feeling of nausea and sweating. Other reported symptoms throughout the 7 participants include slight increased feelings of general discomfort,fullness of the head,dizziness,headache, and burping. Two participants also reported decreases in symptoms they were initially feeling before the study began. For example, one participant initially reported a

difficulty focusing and concentrating which went away after going through the application. Another participant reported they had a decreased feeling of fullness in their head after participating in the application.

5.4 Discussion

Overall based on the results gathered, the majority of participants thought that the application was a useful tool and had a positive experience when taking part in the study. From a quantitative perspective, most static ASL gestures were recognized within a shorter timespan of 1 to 10 seconds, while most multi gestures required a timespan of 1 to 60 seconds. Most multi gestures were recognized successfully after 1 to 10 attempts by the participants. At most, three glove calibrations were required by a user in order to resolve a mismatch between the participant's hands and their virtual counterparts.

Regarding outliers and deviations to the trends of the majority, there are a few factors which contributed to this. These include certain participants finding it more difficult to match the ASL gestures presented in the UI prompts than others. This could be resolved by providing clearer instruction within the UI, as suggested by certain participants in the post-questionnaire. Another contributing factor which was mentioned previously is a variation in bodily features that was present among the participants which also deviates more significantly from those of the researchers that trained the application. The features in this case which have been shown to be most influential are hand size, arm length, and height. While efforts were made to constrain hand size as much as possible, deviations did still occur. More training data from people with various body types and hand sizes would help resolve this issue. Finally, more advanced calibration procedures aside from the basic calibration that was performed with the application may provide more consistent recognition results.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

This thesis has presented a novel approach to performing ASL gesture recognition using motion tracking gloves in conjunction with a VR headset. This approach has been implemented into a usable application and tested among various participants. By utilizing machine learning, the application in it's current state successfully recognizes 34 ASL gestures as they are performed by a user, including 26 letters of the alphabet, and 8 words/phrases. The front-end of the application incorporates a tutorial-like interface, enabling users to cycle between various gestures to be performed while receiving guidance in the form of images and videos. The application was designed such that it could easily be expanded into various ASL related VR applications which could be used for education, general communication, or a number of other purposes.

To simplify the development and recognition process, gestures were categorized as static, multi, and positional gestures based on their characteristics as specified by the syntax of ASL. The application was also split into two separate modules: A trainer and a recognizer. The trainer facilitated the process of generating training data for ASL gestures to be used to generated a neural network for each gesture category. These neural networks are passed to the recognizer, which uses them to perform recognition on incoming hand gestures which are performed in a VR environment. Gestures were performed using the VRFree motion tracking gloves by Sensoryx, and

data handling was achieved through the API associated with the gloves. The VR Headset used was the Vive Cosmos.

The testing process involved 21 participants performing all 34 ASL gestures while having them recognized by the application and while being monitored by a researcher. Feedback from each participant regarding the application was gained through the use of questionnaires. Three quantitative metrics were measured during each user study including the amount of time taken to perform each the gesture, the number of attempts needed to perform multi gestures, and the number of glove calibrations needed. It was found that the majority of static gestures were recognized within 1-10 seconds and most multi gestures within 1-60 seconds. The majority of multi gestures were also performed using 1-10 attempts. Most participants required a single glove calibration, while at most 3 calibrations were required.

6.2 Future Work

While the majority of the results presented in this thesis were positive, there are a number of improvements that can be made to reduce outliers that occurred and to better guide users in performing gestures correctly. The most important improvement would be to include more training data. Only three trainers were utilized in developing the application, so by increasing this number to a larger amount, and also ensuring that trainers have varied body types and hand sizes, the performance of the application would improve. Also, while it may require a longer setup process, incorporating more advanced calibration procedures that are part of both the VR-Free and Vive APIs may be ideal to ensure consistent recognition of the hands and optimal comfort of the user. Also to reduce the need for manual adjustment during application execution, it may be useful to automatically adjust confidence thresholds that influence successful recognition based on the performance of users. This would enable the application to find optimal confidence thresholds for specific users that maximize recognition capabilities while also minimizing false positives.

In regard to possible expansions that could be made to this application, one ad-

dition would be a conversational interface in which users could make any ASL hand gesture and have it recognized immediately. This differs from the current implementation of the application which has users move through specific stages and focus on a single gesture during each stage. While the current interface may be beneficial for those first learning ASL, a conversational interface would be an ideal environment for practicing gestures, especially if dialog capabilities were included to enable chats with other users over a network. Also, since there are multiple different types of glove sizes distributed by VRFree, another possible expansion would be to incorporate usage of these different glove sizes to make the application more accessible to different users. Also it would be useful to automate the training process further so as to not require users to manually execute python scripts and convert the models to a usable format. Both the trainer and recognizer functionalities could likely be incorporated into a single executable that a user can run.

References

[1] Lal Bozgeyikli, Evren Bozgeyikli, Andrew Raij, Redwan Alqasemi, Srinivas Katkoori, and Rajiv Dubey. Vocational rehabilitation of individuals with autism spectrum disorder with virtual reality. *ACM Trans. Access. Comput.*, 10(2), April 2017. ISSN: 1936-7228. DOI: 10.1145/3046786. URL: https://doi.org/10.1145/3046786.

[2] Raluca Budiu. Between-subjects vs. within-subjects study design. URL: https://www.nngroup.com/articles/between-within-subjects/ (visited on 04/30/2022).

[3] Steve Dent. A high-tech glove can quickly translate sign language into speech — engadget, February 2020. URL: https://www.engadget.com/ucla-glove-sign-language-translator-134846711.html (visited on 04/27/2021).

[4] Sergio Escalera, Isabelle Guyon, and Vassilis Athitsos. *Gesture Recognition*. The Springer Series on Challenges in Machine Learning. Springer International Publishing, 2017. ISBN: 978-3319570228. URL: https://link.springer.com/book/10.1007/978-3-319-57021-1.

[5] Abraham Glasser, Vaishnavi Mande, and Matt Huenerfauth. Accessibility for deaf and hard of hearing users: sign language conversational user interfaces. In *Proceedings of the 2nd Conference on Conversational User Interfaces, Bilbao, Spain*, CUI '20, New York, NY, USA. Association for Computing Machinery, 2020. ISBN: 9781450375443. DOI: 10.1145/3405755.3406158. URL: https://doi.org/10.1145/3405755.3406158.

[6] Google Developers. Google developers: backpropagation visual explanation. URL: https://developers-dot-devsite-v2-prod.appspot.com/machine-learning/crash-course/backprop-scroll (visited on 04/26/2022).

[7] Google Developers. Guide: tensorflow core. URL: https://www.tensorflow.org/guide (visited on 04/27/2022).

[8] Google Developers. Machine learning crash course: google developers. URL: https://developers.google.com/machine-learning/crash-course (visited on 04/27/2022).

[9] Google Developers. Machine learning glossary : google developers. URL: https://developers.google.com/machine-learning/glossary (visited on 04/27/2022).

[10] Marek Hrúz, Pavel Campr, Zdenek Krňoul, Milos Železný, Oya Aran, and Pinar Santemiz. Multi-modal dialogue system with sign language capabilities. In New York, NY, USA. Association for Computing Machinery, 2011. ISBN: 9781450309202. DOI: 10.1145/2049536.2049599. URL: https://doi.org/10.1145/2049536.2049599.

[11] HTC Corporation. Vive hand tracking sdk overview - vive hand tracking sdk 1.0.0 documentation. URL: https://hub.vive.com/storage/tracking/overview/index.html (visited on 04/27/2022).

[12] Boyoon Jung and Gaurav S. Sukhatme. Real-time motion tracking from a mobile robot. *Int J of Soc Robotics*, 2:63–78, March 2010. DOI: https://doi.org/10.1007/s12369-009-0038-y.

[13] Timothy D. Keiper, An Tai, Chen Xinfeng, Eric Paulson, Fabienne Lathuilière, Silvain Bériault, François Hébert, David T. Cooper, Martin Lachaine, and X. Allen Li. Feasibility of real-time motion tracking using cine mri during mr-guided radiation therapy for abdominal targets. *Medical Physics*, 47:3554–3566, 8, August 2020. DOI: https://doi.org/10.1002/mp.14230.

[14] Steven M. LaValle. *Virtual Reality*. Cambridge University Press, 2020. URL: http://lavalle.pl/vr/ (visited on 04/27/2022).

[15] Level Developments, Ltd. Mems accelerometer — level developments. URL: https://www.leveldevelopments.com/2020/10/what-are-inclinometers/mems-accelerometer/ (visited on 04/27/2022).

[16] Allyn Malventano. Steamvr htc vive in-depth - lighthouse tracking system dissected and explored - pc perspective. URL: https://pcper.com/2016/04/steamvr-htc-vive-in-depth-lighthouse-tracking-system-dissected-and-explored/2/ (visited on 04/27/2022).

[17] Brittany McCrigler. HTC Vive Teardown - iFixit. URL: https://www.ifixit.com/Teardown/HTC+Vive+Teardown/62213 (visited on 04/27/2022).

[18] Karen McMenemy and Stuart Ferguson. *A Hitchhiker's Guide to Virtual Reality*. A.K. Peters, Ltd., Wellesley, Massachusetts, 2007. ISBN: 978-1568813035. URL: www.akpeters.com.

[19] Shamsul Anuar Mokhtar and Siti Mashitah Shamsul Anuar. Learning application for malaysian sign language: content design, user interface and usability. In *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, Bali, Indonesia*, IMCOM '15, New York, NY, USA. Association for Computing Machinery, 2015. ISBN: 9781450333771. DOI: 10.1145/2701126.2701142. URL: https://doi.org/10.1145/2701126.2701142.

[20] nanolearning. 3-axis mems gyroscope. URL: https://www.youtube.com/watch?v=5BWerr7rJmU (visited on 04/26/2022).

[21] NumPy Developers. Numpy documentation: numpy v1.22 manual. URL: https://numpy.org/doc/stable/ (visited on 04/27/2022).

[22] OnlineCmag Team. The present and future of motion tracking technology in virtual reality. URL: https://www.onlinecmag.com/virtual-reality-motion-tracking-technology/ (visited on 04/27/2022).

[23] Jacob Schioppo, Zachary Meyer, Diego Fabiano, and Shaun Canavan. Sign language recognition: learning american sign language in a virtual environment. In New York, NY, USA. Association for Computing Machinery, 2019. ISBN: 9781450359719. DOI: 10.1145/3290607.3313025. URL: https://doi.org/10.1145/3290607.3313025.

[24] Muhammed N. Sefer, Rawan A. Al-Rashid Agha, and Serkan Özbay. Comparison of neural network and hausdorff distance methods in american, british and turkish sign languages recognition. In *Proceedings of the First International Conference on Data Science, E-Learning and Information Systems, Madrid, Spain*, DATA '18, New York, NY, USA. Association for Computing Machinery, 2018. ISBN: 9781450365369. DOI: 10.1145/3279996.3280007. URL: https://doi.org/10.1145/3279996.3280007.

[25] Sensoryx AG. Sensoryx - VRfree® glove - intuitive VR interaction. URL: https://www.sensoryx.com/ (visited on 03/07/2022).

[26] William R. Sherman and Alan B. Craig. *Understanding Virtual Reality*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, 2nd edition, 2018. ISBN: 978-0-12-800965-9. DOI: https://doi.org/10.1016/C2013-0-18583-2. URL: https://www.sciencedirect.com/book/9780128009659/understanding-virtual-reality.

[27] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2016. ISBN: 978-0133943030. URL: https://iansommerville.com/software-engineering-book/.

[28] Matt Sparks. Metafocus: avoiding the uncanny valley in vr & serious games - learning solutions magazine. URL: https://learningsolutionsmag.com/articles/metafocus-avoiding-the-uncanny-valley-in-vr-serious-games (visited on 04/27/2022).

[29] Spyder Team. Home - spyder ide. URL: https://www.spyder-ide.org/ (visited on 04/27/2022).

[30] Start ASL. American sign language alphabet. URL: https://www.startasl.com/american-sign-language-alphabet/ (visited on 04/26/2022).

[31] David A. Stewart and Jennifer Stewart. *Barron's American Sign Language: A Comprehensive Guide to ASL 1 and 2 with Online Video Practice*. Barrons Educational Series, 2021, pages 1–9. ISBN: 9781506263823. URL: https://books.google.com/books?id=0nANEAAAQBAJ.

[32] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, Arran Topalian, Erroll Wood, Sameh Khamis, Pushmeet Kohli, Shahram Izadi, Richard Banks, Andrew Fitzgibbon, and Jamie Shotton. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Trans. Graph.*, 35(4), July 2016. ISSN: 0730-0301. DOI: 10.1145/2897824.2925965. URL: https://doi.org/10.1145/2897824.2925965.

[33] the pandas development team. 10 minutes to pandas: pandas 1.4.2 documentation. URL: https://pandas.pydata.org/docs/user_guide/10min.html (visited on 04/27/2022).

[34] The University of Texas at Dallas. The Center for Brain Health. URL: https://brainhealth.utdallas.edu/ (visited on 04/26/2022).

[35] Ben Tristem. Unity vs. unreal: which one is best for you? - udemy blog. URL: https://blog.udemy.com/unity-vs-unreal-which-game-engine-is-best-for-you/ (visited on 04/27/2022).

[36] ultraleap. Bite size: how to optimize your vr tracking. URL: https://www.youtube.com/watch?v=lJ4twi6jaSk\&list=PLZgjuTxMC0h0Prqc9UJPYfJdXzxywiF9y (visited on 04/27/2022).

[37] ultraleap. Digital worlds that feel human — ultraleap. URL: https://www.ultraleap.com (visited on 04/27/2022).

[38] Unity Technologies. Introduction to barracuda — barracuda — 1.0.4. URL: https://docs.unity3d.com/Packages/com.unity.barracuda@1.0/manual/index.html (visited on 04/26/2022).

[39] Unity Technologies. Unity real-time development platform — 3d, 2d vr & ar engine. URL: https://unity.com/ (visited on 04/27/2022).

[40] Nicholas Wang. What is user research? — interaction design foundation (ixdf). URL: https://www.interaction-design.org/literature/topics/user-research (visited on 04/27/2022).

[41] Yuhang Zhao, Cynthia L. Bennett, Hrvoje Benko, Edward Cutrell, Christian Holz, Meredith Ringel Morris, and Mike Sinclair. Enabling people with visual impairments to navigate virtual reality with a haptic and auditory cane simulation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal QC, Canada*, CHI '18, 1–14, New York, NY, USA. Association for Computing Machinery, 2018. ISBN: 9781450356206. DOI: 10.1145/3173574.3173690. URL: https://doi.org/10.1145/3173574.3173690.

Appendix A

UNR IRB Application Cover Sheet

University of Nevada, Reno
Institutional Review Board
Part I, Cover Sheet

Last edited by: Justice Colby

Last edited on: November 29, 2021

[click for checklist]

[1842535-1] American Sign Language Gesture Recognition using Motion Tracking Gloves in VR

Answer all questions on this form completely, include attachments and obtain signatures of Principal Investigator and the Responsible official prior to final submission on IRBNet.

I. Principal Investigator

Name: Sergiu Dascalu

Institution:

Department: College of Engineering

Telephone: (775) 784-4613 **Email:** dascalus@cse.unr.edu

Address: 1664 N Virginia St MS-171 Reno, NV 89557-0171, USA

COI Disclosure: Any financial interests related to this study?

- ☐ Yes
- ☑ No

If yes, **COI Disclosure Explanation:**

Could any external entity benefit financially from the results of this study?

- ☐ Yes
- ☑ No

II. Co-Investigator(s) or Research Team Member(s) N/A ☑

Name:

Department:

Institution:

Telephone: **Email:**

COI Disclosure: Any financial interests related to this study?

- ☐ Yes
- ☐ No

If yes, **COI Disclosure Explanation:**

III. Student Investigator(s) N/A ☐

Name: Justice Colby

Institution:

Telephone: (775) 219-5629 **Email:** jcolby@nevada.unr.edu

COI Disclosure: Any financial interests related to this study?

- ☐ Yes
- ☑ No

If yes, **COI Disclosure Explanation:**

IV. Project Information

Research Type:

- ☐ Biomedical
- ☑ Social Behavioral/Educational

Photographing or Video Recording:

- ☐ Yes
  Upload the Photo Release form.
- ☑ No

Identifiable Information from Education Records:
Note that accessing education records for research purposes invokes FERPA regulations. In the protocol, address how records are accessed, whether researchers will access directory information only, and how written permission will be collected from students or parents for minor students. For more information on FERPA, see IRB Policy 76.

- ☐ Yes
- ☑ No

Research Location:

- ☐ VA Sierra Nevada Healthcare System
- ☐ Saint Mary's Regional Medical Center
- ☐ Renown Health
- ☑ UNR Campus
- ☐ Other -

International Research:

- ☐ Yes
  For international research, reference IRB Policy 575 for details to address in the protocol.
- ☑ No

If yes, **specify the countries:**

Renown Health Research Locations:

- ☐ Renown Regional
- ☐ Renown Pharmacy

- ☐ Renown South Meadows
- ☐ Renown Pregnancy Center
- ☐ Renown Outpatient Clinic
- ☐ Renown Urgent Care
- ☐ Renown Imaging
- ☐ Renown Lab
- ☐ Renown Emergency Room
- ☐ Renown Skilled Nursing
- ☐ Renown Hospice Care
- ☐ Renown Home Health
- ☐ Renown Rehabilitation

Requested Review Path:

- ☐ Expedited IRB Review
  Complete Protocol - Social Behavioral Educational Research and Records Research or Protocol - Biomedical Research
- ☐ Full Board Review
  Complete Protocol - Social Behavioral Educational Research and Records Research or Protocol - Biomedical Research
- ☑ Exempt Review
  Complete Protocol - Social Behavioral Educational Research and Records Research
- ☐ Requesting a determination about whether a project is human research
  Complete Request for Human Research Determination
- ☐ Requesting authorization to use an external IRB
  Complete Request to Use an External IRB
- ☐ Reporting emergency use of an FDA-regulated drug or device
  Complete Emergency Use Investigational Drug or Device
- ☐ Review of a Humanitarian Use Device
  Complete Protocol - Humanitarian Use Device for Treatment or Diagnosis
- ☐ Research involving existing records or specimens
  Complete Protocol - Social Behavioral Educational Research and Records Research

Risk Level:

- ☑ Minimal risk
- ☐ Greater than minimal risk (requires full board review)
- ☐ No known risk

Involvement of Vulnerable Populations:

- ☑ N/A, research will not involve vulnerable populations
- ☐ Pregnant women and fetuses
  For research with pregnant women and fetuses, reference IRB Policy 210 and 211 for details to address in the protocol.
- ☐ Prisoners
  Complete Research with Prisoners
- ☐ Children (persons under 18 years of age)
  For research with children, reference IRB Policy 230 for details to address in the protocol.
- ☐ Adults with impaired decision-making capacity
  For research with adults with impaired decision-making capacity, reference IRB Policy 240 for details to address in the protocol.
- ☐ People who do not speak English

For research with people who do not speak English, reference IRB Policy 250 for details to address in the protocol.

V. Funding Information N/A ☑

Sponsor Type:

- ☐ Federal Government
- ☐ Industry Sponsor
- ☐ Departmental
- ☐ Other:
- ☐ Other Government (State/Local)
- ☐ Other Private Funds
- ☐ Subcontract

Sponsor Name:

Grant/Contract Title and Number:

VI. Federal Agencies with Additional Requirements to Protect Human Participants

Please see the "Instructions to Researchers" section at the end of this form for a list of required supplemental forms and relevant policies.

- ☐ DoD
- ☐ DoE
- ☐ DoEd
- ☐ DoJ or NIJ
- ☐ EPA
- ☐ NSF
- ☐ VA
- ☑ N/A

VII. FDA-Regulated Research

- ☑ N/A, research does not involve drugs or devices
- ☐ Drug research

Trade Name	Generic Name

- ☐ Device research

Name of Device	Device Manufacturer

VIII. External Committee Approvals

- ☑ Thesis or Dissertation Committee
- ☐ Radiation Safety Committee

- ☐ Biosafety Committee
- ☐ Other:
- ☐ N/A

INSTRUCTIONS TO RESEARCHERS

[top]

You have completed Part I of the application process. **Preview** Part I and correct if needed. Print the last page so you have the list of the researcher forms and additional regulatory requirements expected for this research. Click **Save and Exit**. **Add** the remaining required documents (listed below or referenced in the researcher forms/applications), **address** the necessary regulatory and policy requirements in the protocol and other project documents, and then the PI should electronically **Sign** and **Submit** the project. Make sure to upload training documentation for all researchers listed on this form.
If you have any questions, refer to the IRBNet pages of the Research Integrity website.

Additional required researcher forms and policies/regulations:

- Complete Protocol - Social Behavioral Educational Research and Records Research

Appendix B

UNR IRB Application

University of Nevada, Reno
Research & Innovation

Research Integrity
Tel: (775) 327-2368
Web: unr.edu/research-integrity

Protocol – Social Behavioral Educational Research and Record Research

IRBNet ID: 1842535-1, PI: Sergiu Dascalu, Project Title: American Sign Language Gesture Recognition using Motion Tracking Gloves in VR

Background:

Gesture recognition refers to the process of detecting and interpreting human gestures by acquiring and analyzing related data [i]. There has been abundant research in the field of gesture recognition, with related applications spanning a wide range of technologies and disciplines. Much of this research has been dedicated towards hand gesture recognition, with sign language gestures being a particular area of interest. Sign Language is a method of communication used primarily within the Deaf community which incorporates hand gestures representing letters and words [ii]. Some applications which have focused on sign language gesture recognition include gloves which can translate sign language gestures into audio and text, Dialogue systems which use image recognition, and various educational tools [iii] [iv] [v]. Despite the successes of gesture recognition applications, they also face several challenges which can limit their overall usability. One common issue with gesture recognition is occlusions, or obstructions within the area being detected, which can be caused by overlapping fingers, or forming fists. Many gesture recognition applications also lack the ability to detect moving gestures as well as gestures that involve parts of the body that are outside of a typical field of view.

Virtual Reality (VR) is another area of research which has seen substantial contributions in various disciplines. VR encompasses the creation and utilization of artificial simulations meant to replace one or more sensory stimuli provided by the real world [vi]. A common technology used to experience VR is a headset which provides visual and sometimes auditory simulations to a user. Since VR involves immersing the user into a virtual world with nearly endless possibilities, the field has massive potential to provide unique and intuitive user interfaces, especially when combined with different types of peripheral hardware. One approach to designing VR user interfaces involves providing a simulated version of a user's hands which can replicate real-world movements within the virtual environment. There are a number of tools which achieve this in different ways, with each one having its own benefits and trade-offs. Some tools include infrared sensor technologies such as the LEAP motion, as well as motion tracking gloves [v].

The purpose of this research is to develop and test a novel approach to performing ASL hand gesture recognition within a VR environment. This approach has been implemented into a software application which successfully recognizes ASL gestures being performed by a user. The hardware used with the application includes the Vive Cosmos VR Headset along with the VRFree Sensoryx motion capture gloves. The gesture recognition technique employed by this application has the capability to detect moving gestures as well as gestures which must be performed at specific parts of the body, which has been a limitation with many previous related applications. The VRFree gloves also overcome the occlusion issue of hand gesture recognition by providing distinct sensor data from each joint of each finger. In order to effectively leverage the VR portion of the application, an intuitive interface is created for the user which displays examples of gestures to be performed, and enables the user to perform gestures one at a time until each one is performed successfully.

Study Aims/Objectives:

- Determine the overall performance of the proposed gesture recognition approach when tested by different users
- Determine what improvements can be made to this approach to improve its performance and usability
- Determine the intuitiveness of the developed user interface and the overall satisfaction of users while testing the developed application.

Study Population:
The participant population will include people at least 18 years old, with any gender or ethnic background. Participants must have a hand size that at least comes close to the size specifications of the motion tracking VR gloves that will be used during the study.

Vulnerable Populations:
No vulnerable populations will be included in this research.

Sample Size:
The sample size will be 20 participants. This number was determined to be suitable after reviewing previous similar studies to the one I am performing. The number of participants in these studies were either close to or exactly 20, and it appears the researchers of these studies were able to extract sufficient data in these cases. Also, after discussing the sample size with more experienced members of my research group, they have stated that 20 is typically an adequate sample size for the type of study I am performing.

Recruitment Process:
Participants will be invited to participate in the study through word of mouth or email contact. This will be done as needed during various time periods and in various places until all participants are gathered. During this process, the prospective participant will be given a brief description of the study. A full overview of the study will be provided once the participant agrees to it.

Screening Procedures:
The first screening process for potential participants will be regarding hand size. To participate in the study, participants must have a hand size that at least comes close to the size specifications of the motion tracking VR gloves that will be used during the study. To confirm this, participants will place their hands on a sizing guide provided by the vendor of the gloves (this sizing guide has been attached a supporting document with my submission).

The second screening process will be regarding age. The participants will state their age on a pre survey before the study begins. If they 18 years old or older, they can participate in the study.

Informed Consent Process:
When first inviting the participant to take part in the study, the researcher will verbally explain the study to them. The participant will then be invited to the lab in which the study will be taking place if they are still interested in taking part in the study. Once in the lab, the participant will be provided a consent form to fill out which includes all the necessary details regarding the study. The researcher will then demonstrate the tasks that will need to be done during the study by performing them. After this, the researcher will ask the participant again if they would still like to participate in the study. The participant is free to opt-out of the study at any point in time. After the study has been completed, the participant will be given a copy of the consent form along with the researcher's contact information.

Data Collection Procedures:
After the participant agrees to take part in the study, they will be informed in detail what the study involves. The participant will then be provided with a pre-test survey which asks participants about their demographics, familiarity with virtual reality, familiarity with sign language, familiarity with motion tracking hardware/software, and familiarity with electronic entertainment such as video games. The participant will then be provided with a simulator sickness questionnaire which asks participants if they are experiencing any symptoms associated with simulator sickness, and how severe the symptoms are. The participant will then be given a demonstration of the tasks that need to be performed during the study. Once the demonstration is over, the participant will begin performing the tasks themselves with assistance being provided from the researcher as needed.

The tasks of the study involve performing a series of American Sign Language (ASL) gestures as prompted by a VR application while equipped with the VRfree Sensoryx Motion Tracking gloves and the Vive Cosmos VR Headset. After putting on the headset and gloves, the process begins by powering on the gloves and starting the VR application from the PC. The gloves are then calibrated through a routine in the software which involves the participant performing three poses. Sometimes multiple calibrations may be needed if the 3D hand models in the virtual environment do not accurately match the locations of the hands in the real world. Once calibration has been performed successfully, the participant will then follow a series of prompts within the virtual environment which each instruct the participant on how to perform a specific ASL gesture. This instruction will be conveyed through images and in some cases video. When the user is able to perform the ASL gesture successfully, they will be asked to move on to the next one by looking at a button in the virtual environment. This process will be repeated for a total of 34 ASL gestures. Throughout this process, data from three quantitative metrics will be gathered including the time necessary to perform each gesture successfully, the number of attempts needed to perform a moving gesture successfully, and the number of times a recalibration is needed. This data will be collected through logs generated within the VR application as well as observation by the researcher. Once all gestures have been performed successfully, the participant will then remove the VR headset and gloves with the researcher's assistance.

The study will then conclude with a post-test survey and a second simulator sickness questionnaire meant to determine if any symptoms are being experienced after going through the study. The post-test survey will ask the participant some questions regarding their overall experience while performing the tasks. These include overall comfortability while performing the tasks, difficulty performing each gesture, intuitiveness of the user interface, usefulness of the gloves in teaching ASL hand gestures, potential improvements that can be made, other possible applications for the gloves, and general experience along with any other general comments and observations.

Study Duration/ Study Timeline:
Only one meeting will be needed for each participant in the study. Each meeting should take approximately 1 hour to complete. The expected approximate end date of the study will be February 28th 2022.

Study Locations:
The study location will be Room 436 in the William Pennington Engineering Building at the University of Nevada, Reno.

International Research:
This research will take place in the U.S. and will not involve external international locations. This research involves American Sign Language (ASL) which is a primary method of communication for the deaf in both the United States and Canada.

Participant Compensation:
No compensation will be provided to participants in this study.

Risk to Participants:
Motion sickness may be experienced for some participants of the study who have not used virtual reality applications often or at all. Before the study begins, participants will be notified of this possibility and will be asked to remove the virtual reality headset if they begin to feel sick at any point in time. Should participants feel sick during the study, they can choose to either continue the study after a period of rest or stop the study altogether.

Benefits to Participants:
This research does not present any direct benefit to the participants. However, the research provides an opportunity to gain a better understanding of gesture recognition in general as well as its possible applications within a virtual reality environment.

Privacy of Participants:
Participants will be recruited individually, and no record will be kept of contact. The only record of a person's participation that will be kept in the study is the survey responses. No additional records will be kept. The participant will be interacting with the researcher in a research lab within the William Pennington Engineering Building at the University of Nevada, Reno.

Data Management and Confidentiality:
A randomized ID will be given to each participant which will have no identifiable information. All collected data will be stored under the ID numbers and saved on a computer which requires specific login credentials. Also, the computer in which the data will be saved is located in a lab that requires key card access.

Approach to Analysis:
A standard ANOVA method will be used for statistical analysis along with other common transformations and methods.

References:

[i] S. Escalera, I. Guyon, and V. Athitsos.Gesture Recognition. The Springer Se-ries on Challenges in Machine Learning. Springer International Publishing,2017.isbn: 9783319570211.url:https://books.google.com/books?id=rYotDwAAQBAJ.

[ii] D.A. Stewart and J. Stewart.Barron's American Sign Language: A Compre-hensive Guide to ASL 1 and 2 with Online Video Practice. Barrons EducationalSeries, 2021, pages 1–9.isbn: 9781506263823.url:https://books.google.com/books?id=0nANEAAAQBAJ.

[iii] endgadget. A high-tech glove can translate sign language with 99-percent ac-curacy.url:https://www.engadget.com/ucla-glove-sign-language-translator-134846711.html(visited on 04/15/2021).

[iv] Marek Hrúz, Pavel Campr, Zdenek Krňoul, Milosˇ Zelezn´y, Oya Aran, andPinar Santemiz. Multi-modal dialogue system with sign language capabilities.In New York, NY, USA. Association for Computing Machinery, 2011.isbn:9781450309202.doi:10.1145/2049536.2049599.url:https://doi.org/10.1145/2049536.2049599.

[v] Jacob Schioppo, Zachary Meyer, Diego Fabiano, and Shaun Canavan. Sign lan-guage recognition: learning american sign language in a virtual environment.In New York, NY, USA. Association for Computing Machinery, 2019.isbn:9781450359719.doi:10.1145/3290607.3313025.url:https://doi.org/10.1145/3290607.3313025.

[vi] Steven M. LaValle.Virtual Reality. Cambridge University Press, 2020, pages 1–3.url:http://lavalle.pl/vr/.

Appendix C

UNR IRB Consent Form

University of Nevada, Reno
Social Behavioral or Educational Research Consent Form

Title of Study:	**American Sign Language Gesture Recognition using Motion Tracking Gloves in VR**
Principal Investigator:	**Sergiu Dascalu, PhD**
Co-Investigators /	**Justice S Colby**
Study Contact:	**Justice S Colby**
Study ID Number:	**1842535-1**
Sponsor:	**N/A**

SUMMARY OF KEY ELEMENTS:

Introduction

You are being invited to participate in a research study. Before you agree to be in the study, read this form carefully. It explains why we are doing the study; and the procedures, risks, discomforts, benefits and precautions involved.

At any time, you may ask one of the researchers to explain anything about the study that you do not understand.

You do not have to be in this study. Your participation is voluntary. If you do not agree to participate, you will receive the care/education you would have received if the study was not taking place.

Take as much time as you need to decide. If you agree now but change your mind, you may quit the study at any time. Just let one of the researchers know you do not want to continue.

Why are we doing this study?

We are doing this study to determine the overall performance and usability of a novel approach to sign language recognition in virtual reality, while also examining possible improvements that can be made.

Why are we asking you to be in this study?

We are asking you to be in this study because you are an adult who can speak English and also has a hand size that can fit within the motion tracking gloves being used in the study.

How many people will be in this study?

We expect to enroll 20 participants at the University of Nevada, Reno.

What will you be asked to do if you agree to be in the study?

If you agree to be in this study you will be asked to complete a pre-test survey, post-test survey, and a simulator sickness questionnaire at the beginning and end of the experiment along with a series of tasks which help to assess overall performance and usability of the application.

The pre-test survey will ask you questions regarding your age, gender, highest completed level of education, and various questions relating to your familiarity with key components of the application. The simulator sickness questionnaire will ask you if you are experiencing any symptoms associated with simulator sickness and how severely they are affecting you.

The tasks of the study will be performed while wearing a VR headset and a pair of motion tracking gloves. These tasks include calibrating the gloves, and performing a number of ASL gestures while having them successfully recognized by the gesture recognition algorithm of the application.

The post-test survey will ask questions regarding the participant's thoughts on usability and design of the application along with any improvements that can be made.

How long will you be in the study?

The study will take about an hour of your time.

What are your choices if you do not volunteer to be in this research study?

If you decide not to be in the study, tell the investigator and you will be allowed to leave.

What if you agree to be in the study now, but change your mind later?

You do not have to stay in the study. You may withdraw from the study at any time by leaving the room after informing the investigator.

What if the study changes while you are in it?

If anything about the study changes or if we want to use your information in a different way, we will tell you and ask if you want to stay in the study. We will also tell you about any important new information that may affect your willingness to stay in the study.

Is there any way being in this study could be bad for you?

Some users of virtual reality experience discomfort, nausea, headaches, and eye strain. If at any point in time you begin to experience any of these symptoms, please remove the virtual reality headset. We can continue testing or stop altogether.

What happens if you become injured because of your participation in the study?

In the event that this research activity results in an injury, treatment will be available. This includes first aid, emergency treatment, and follow-up care as needed. Care for such injuries will be billed in the ordinary manner to you or your health insurance carrier.

Will being in this study help you in any way?

Probably not, except it will add to your experience with using VR applications.

Who will pay for the costs of your participation in this research study?

No costs are associated with participation in this study.

Will you be paid for being in this study?

You will not receive any payment for being this study.

Who will know that you are in in this study and who will have access to the information we collect about you?

The researchers, the University of Nevada, and the Reno Institutional Review Board will have access to your study records.

How will we protect your private information and the information we collect about you?

We will treat your identity with professional standards of confidentiality and protect your private information to the extent allowed by law.

We will not use your name or other information that could identify you in any reports or publications that result from this study.

Do the researchers have monetary interests tied to this study?

The researchers and/or their families have no monetary interests tied to this study.

Whom can you contact if you have questions about the study or want to report an injury?

At any time, if you have questions about this study or wish to report an injury that may be related to your participation in this study, contact Sergiu Dascalu at (775) 784-4613 and/or Justice Colby at (775) 219-5629.

Whom can you contact if you want to discuss a problem or complaint about the research or ask about your rights as a research participant?

You may discuss a problem or complaint or ask about your rights as a research participant by calling the University of Nevada, Reno Research Integrity Office at (775) 327-2368. You may also use the online *Contact the Research Integrity Office* form available from the *Contact Us* page of the University's Research Integrity Office website.

Agreement to be in study

If you agree to participate in this study, you must sign this consent form. We will give you a copy of the form to keep.

Participant's Name Printed	
Signature of Participant	Date
Signature of Person Obtaining Consent	Date

Appendix D

UNR IRB Pre-Test Survey

University of Nevada, Reno
Social Behavioral Research

Title of Study:	**American Sign Language Gesture Recognition using Motion Tracking Gloves in VR**
Principal Investigator:	**Sergiu Dascalu, PhD**
Co-Investigators /	**Justice S Colby**
Study Contact:	**Justice S Colby**
Study ID Number:	**1842535-1**
Sponsor:	**N/A**

Pre Test Survey

Participant ID #:_____

Please Enter Your Age:_____

What is your gender? _____

What is your highest level of completed education? _____

Please rate your familiarity with Virtual Reality (VR).
(1 – Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Please rate your familiarity with American Sign Language (ASL).
(1 – Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Please rate your familiarity with Motion Tracking Hardware and Software.
(1 – Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Please rate your familiarity with Video Games or Similar Forms of Electronic Entertainment.
(1 – Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Appendix E

UNR IRB Post-Test Survey

University of Nevada, Reno
Social Behavioral Research

Title of Study:	**American Sign Language Gesture Recognition using Motion Tracking Gloves in VR**
Principal Investigator:	**Sergiu Dascalu, PhD**
Co-Investigators /	**Justice S Colby**
Study Contact:	**Justice S Colby**
Study ID Number:	**1842535-1**
Sponsor:	**N/A**

Post Test Survey

Please rate how comfortable you were during the test.
(1 – Very Uncomfortable, 5 – Very Comfortable)

Very Uncomfortable	1	2	3	4	5	Very Comfortable

Please rate how easy it was to perform each gesture.
(1 – Very Difficult, 5 - Very Easy)

Very Difficult	1	2	3	4	5	Very Easy

Please rate the intuitiveness of the user interface.
(1 – Not Intuitive, 5 - Very Intuitive)

Not Intuitive	1	2	3	4	5	Very Intuitive

Please rate the usefulness you feel these gloves have in teaching ASL hand gestures.
(1 – Not Useful, 5 - Very Useful)

Not Useful	1	2	3	4	5	Very Useful

Please rate your overall user experience.
(1 – Very Bad, 5 - Very Good)

Very Bad	1	2	3	4	5	Very Good

What potential improvements, if any, would make the application more useful or easy to use?

_____
_____
_____
_____
_____
_____
_____
_____

Based on your experience, what are some other applications in which you feel these gloves would be useful?

_____
_____
_____
_____
_____
_____
_____
_____

Please write any other comments or observations that you might have.

_____
_____
_____
_____
_____
_____
_____
_____

Appendix F

UNR IRB Simulator Sickness Questionnaire

No_____ Date_____

SIMULATOR SICKNESS QUESTIONNAIRE

Kennedy, Lane, Berbaum, & Lilienthal (1993)***

Instructions : Circle how much each symptom below is affecting you right now.

1. General discomfort	None	Slight	Moderate	Severe
2. Fatigue	None	Slight	Moderate	Severe
3. Headache	None	Slight	Moderate	Severe
4. Eye strain	None	Slight	Moderate	Severe
5. Difficulty focusing	None	Slight	Moderate	Severe
6. Salivation increasing	None	Slight	Moderate	Severe
7. Sweating	None	Slight	Moderate	Severe
8. Nausea	None	Slight	Moderate	Severe
9. Difficulty concentrating	None	Slight	Moderate	Severe
10. « Fullness of the Head »	None	Slight	Moderate	Severe
11. Blurred vision	None	Slight	Moderate	Severe
12. Dizziness with eyes open	None	Slight	Moderate	Severe
13. Dizziness with eyes closed	None	Slight	Moderate	Severe
14. *Vertigo	None	Slight	Moderate	Severe
15. **Stomach awareness	None	Slight	Moderate	Severe
16. Burping	None	Slight	Moderate	Severe

* Vertigo is experienced as loss of orientation with respect to vertical upright.

** Stomach awareness is usually used to indicate a feeling of discomfort which is just short of nausea.

Last version : March 2013

***Original version : Kennedy, R.S., Lane, N.E., Berbaum, K.S., & Lilienthal, M.G. (1993). Simulator Sickness Questionnaire: An enhanced method for quantifying simulator sickness. *International Journal of Aviation Psychology, 3*(3), 203-220.

Simulator Sickness Questionnaire***

Kennedy, Lane, Berbaum, & Lilienthal (1993)***

Validation of the French-Canadian version of the SSQ developed by the UQO Cyberpsychology Lab :

- Total : items 1 to 16 (scale of 0 to 3).
  - « *Nausea* » : items 1 + 6 + 7 + 8 + 12 + 13 + 14 + 15 + 16.
  - « *Oculo-motor* »: items 2 + 3 + 4 + 5 + 9 + 10 + 11.

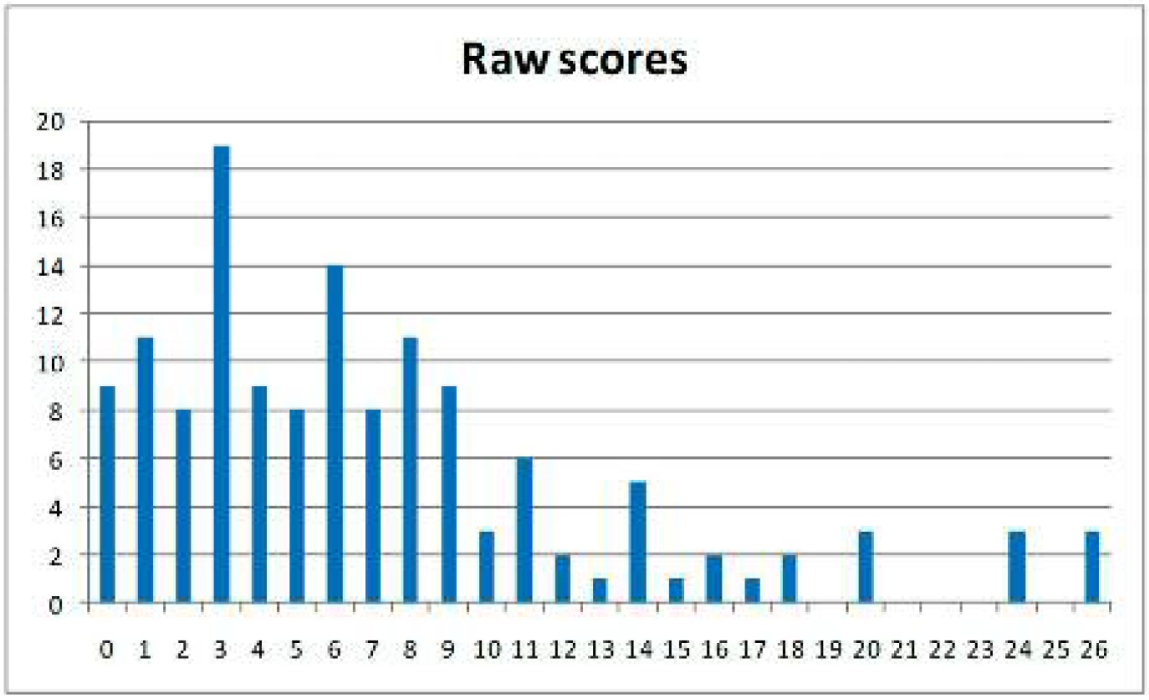
Please refer to the following articles for more information about the French-Canadian validated version :

BOUCHARD, S., Robillard, & Renaud, P. (2007). Revising the factor structure of the Simulator Sickness Questionnaire. Acte de colloque du *Annual Review of CyberTherapy and Telemedicine*, *5*, 117-122.

BOUCHARD, S., St-Jacques, J., Renaud, P., & Wiederhold, B.K. (2009). Side effects of immersions in virtual reality for people suffering from anxiety disorders. *Journal of Cybertherapy and Rehabilitation*, 2(2), 127-137.

BOUCHARD, S. Robillard, G., Renaud, P., & Bernier, F. (2011). Exploring new dimensions in the assessment of virtual reality induced side-effects. *Journal of Computer and Information Technology*, *1*(3), 20-32.

Based on results from Bouchard, St-Jacques, Renaud, & Wiederhold (2009), below are the mean scores reported:



Note. For the original scoring version, consult : Kennedy, R.S., Lane, N.E., Berbaum, K.S., & Lilienthal, M.G. (1993). Simulator Sickness Questionnaire: An enhanced method for quantifying simulator sickness. *International Journal of Aviation Psychology*, *3*(3), 203-220.

Appendix G

UNR IRB Participant Recruitment Email

Recruitment Email Script

General Recruitment Email:

Good [morning, afternoon, etc.],

I am conducting a research study to assess the performance and usability of a virtual reality (VR) application designed to recognize American Sign Language (ASL) gestures being performed by a user. I am currently looking for 20 participants to conduct a user study to advance this research. Participants in the study will be asked to perform a number of tasks while wearing both a VR headset and a pair of motion tracking gloves. These tasks include performing an initial calibration of the gloves and also performing 34 ASL hand gestures as prompted within the VR environment. The entire process should take approximately an hour to complete. The user study will be performed in WPEB Lab Room 436. If you are interested in participating in this user study, please email Justice Colby at jcolby@nevada.unr.edu to set up a time that works best for you to complete the study.

Sincerely,

Justice Colby

Personal Recruitment Email:

Dear [participant],

I am conducting a research study to assess the performance and usability of a virtual reality (VR) application designed to recognize American Sign Language (ASL) gestures being performed by a user. I am currently looking for 20 participants to conduct a user study to advance this research. Participants in the study will be asked to perform a number of tasks while wearing both a VR headset and a pair of motion tracking gloves. These tasks include performing an initial calibration of the gloves and also performing 34 ASL hand gestures as prompted within the VR environment. The entire process should take approximately an hour to complete. The user study will be performed in WPEB Lab Room 436. If you are interested in participating in this user study, please email Justice Colby at jcolby@nevada.unr.edu to set up a time that works best for you to complete the study.

Sincerely,

Justice Colby

Appendix H

UNR IRB User Study Flyer

Seeking Participants for VR Based User Study!

Overview

This study will assess the functionality of an application designed to recognize American Sign Language (ASL) gestures using motion tracking gloves and a virtual reality headset as input. Participants will be asked to perform a series of ASL gestures while wearing the headset and gloves.

How long will it take?

The study should take approximately 1 hour to complete.

Where is it located?

The William Pennington Engineering Building at UNR, Room WPEB 436.

How do I sign up?

Email jcolby@nevada.unr.edu
Or call/text 775-219-5629

Who can participate?

To participate you must:

- Be at least 18 years old
- Have a hand size that fits within the motion tracking gloves to be used in the study. A glove sizing guide is on the other side of this flyer, and can also be emailed to you upon request.

You do not need to know Sign Language to Participate!

VR FREE
by sensoryx

Large