NTNU
Norwegian University of
Science and Technology

Gesture Recognition Using Template Matching in VR

Christer Peltoperæ Somby

Applied Computer Science
Submission date: June 2018
Supervisor: Simon McCallum, IDI
Co-supervisor: Rune Hjelsvold, IDI

Norwegian University of Science and Technology
Department of Computer Science

Preface

This is a master thesis in Applied Computer Science at NTNU Gjøvik. The master thesis was written during the spring semester of 2018. The project came about due to a suggestion from my supervisor. The project was originally going to be about gestures for controlling voice chat, but the scope of that turned out to be too large so it was downscoped. My interest in motion capture and gesture recognition also had a big role to play in the choice of project. The interest sparked during the different courses that was done during the course of the master program. Doing it in virtual reality was simply a perfect choice as the field is new and this is the time when the really good papers start coming out. The master thesis is written with the assumption that the reader is a programmer, but does no necessarily have to have indepth knowledge about image processing techniques.

01-06-2018

Acknowledgment

I would like to thank my father for always being a person that gives me a push when I need it. He was also my motivation for finishing the project.

I would like to thank my supervisor Simon McCallum for his indispensable help during the course of the master thesis. Every time I was lost as to what to do, he would be able to point me in the right direction no matter what it was that I asked him. Without him, I would not have been able to complete this project.

I would like to thank my co-supervisor Rune Hjelsvold. I did not utilize him as much as I should have.

I would like to thank Mariusz Nowostawski for being an upbeat lecturer that helped me improve my writing.

I would also like to thank my classmate Bjørn Fuglestad for helping with testing the experiments and generally being a good guy to talk to.

And lastly I would like to thank all the participants who participated in my experiments.

C.P.S.

Abstract

The goal of this master project was to implement and run experiments on gesture recognition using template matching and particle filtering in virtual reality. Experiments were done using the gesture recognition software written to test certain research questions. The main observation from the research is that users do not have the same mindset when forming gestures and that users can indeed take simple instructions and form gestures. Thresholds are not specific to users, but they are specific to gestures, unless you improve the system as mentioned in future work. And finally, it is fully possible to create a real-time gesture recognition system that works quite well with high accuracy.

Contents

List of Figures

Listings

1 Introduction

1.1 Virtual Reality

We are currently in a time period where a lot of interesting developments are happening. One of the particulars is the proliferation of Virtual Reality(VR). In 2016 Oculus Rift and HTC Vive paved the way for VR to truly proliferate to the masses. With the rush to push out products, developers use methods that are used for 3D games to make sure that people that try VR are not lost as to what to do. User Interfaces(UI) are currently mostly created this way. But there are other ways to interact in VR environments.

1.2 What is Virtual Reality?

In the context of this paper, Virtual Reality(VR) refers to a computer generated world that is interacted with through the use of a Head-Mounted Display(HMD). The HMD uses stereoscopy to trick the brain into perceiving 3D depth. The HMD uses two screens to achieve the effect through showing half of the the virtual environment in each eye. The HMD has a wide range of sensors such as gyroscopes and accelerometers. Most high-end HMD use base stations with infrared lasers to improve the tracking system for increased accuracy of position. The earlier HMD such as the Oculus Rift DK1 only had the capability of stationary tracking, meaning one could only use the HMD to turn their heads without any movement. The newer Oculus Rift and HTC Vive each have base stations that allow for movement around a confined space defined by the distance that their base stations can track

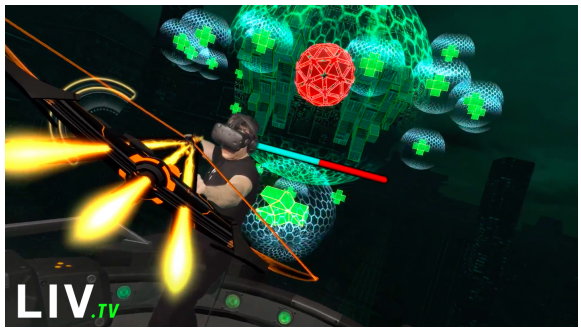

Figure 1: Mixed reality play in the HTC Vive [1]

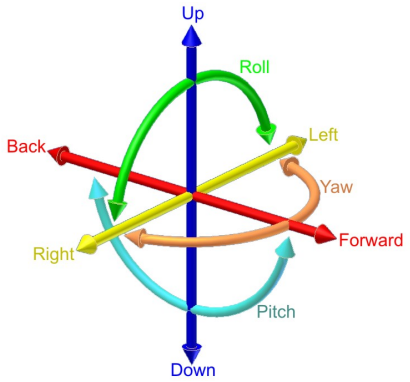

Figure 2: Six degree of freedom [2]

and the size of the room used. The HTC Vive comes with controllers that are virtual replacements for your hands, allowing for greater interaction with the VR system. An example of someone in VR can be seen in Figure 1.

1.3 Gestures in Virtual Reality

Gestures in VR are in an interesting place at the moment. There is not much in terms of actual working implementations for it. There are still some that are available[3][4], however the methods use requires training of their systems. The best way for someone to create a gesture would be to keybind them for themselves, by only doing it one time. Neural networks on their own might struggle with this, but if it is combined with other methods such as template matching these can do just that. Customization of gestures is probably preferable for people as people have different mindsets when thinking of gestures.

1.4 Challenges of Virtual Reality Gesture Detection

Gesture detection can be implemented in many ways, but there are still some fundamental challenges that pose themselves when considering it. Teaching people what gestures to use can be a challenge as there is no standardized gesture library specifically for VR. Of course that does not mean that there are no databases which could be adapted to this[5].

Another part to consider is that within VR a person has six degrees of freedom(6DoF). 6DoF refers to the three positional and three rotational axis available in VR, as seen in figure 2. Each of these axis have to be considered when both making a gesture and trying to detect them. A gesture will be vastly different if you have a controller turned upside down and perform it. The same goes for the HMD's rotation, a gesture is different when you perform a gesture while looking forwards

compared to having your head tilted to the left.

1.5 Research Questions

1. When given tasks to achieve with no instruction on gesture formation or type, do participants form similar gestures
2. Can users form gestures given simple instructions
   2.1. Is the threshold specific per user
   2.2. Is the threshold specific per gesture
3. Can gestures be recorded and detected in real-time

1.5.1 Hypotheses

Some of the research questions have null hypothesis that were formed prior to the start of the experiments.

- $H1$: Participants make similar gestures when given an action to perform.
- $H2$: Gestures need individual thresholds

2 Background

Gestures as a means to replace user interfaces is a field with a lot of research. But there is a lack of it when it comes to replacing it in a VR environment using standard equipment. Adding additional hardware requirements might cause people to shy away from using such implementations. The HTC Vive and Oculus rift were both released in 2016. As the VR field is fairly new, there are still areas that are left mostly untouched. Most of the academic literature will be about other devices used for gesture recognition. Gesture recognition is generally done using neural networks and other learning methods.

The search for research papers was done mostly through use of Google Scholar. For some of the more complex papers, the search was spread to IEEE and ACM. The organization of the references was done with JabRef.

2.1 Virtual Reality Tracking Accuracy

Usually 6DoF HMD have two different ways of handling positional and rotational data. One of these is internal accelerometers and gyroscopes, the second way is using external tracking with fixed location trackers. High-end systems use both systems for higher degrees of accuracy, such as the HTC Vive. Systems only using accelerometers and gyroscopes for VR, such as mobile phones, will suffer from drift. Drift in mobile systems occur almost as soon as the device is in use. Generally this drift is reduced by using different filters to reduce the drift, but it can still not eliminate it. That is why purely accelerometer and gyroscoping systems are unreliable when trying to implement a VR system using them. But there is still research into how to reduce this drift with HMD [6], but it does not remove it.

Using fixed position external trackers increases accuracy by a huge amount, effectively eliminating drift. The HTC Vive uses Lighthouses. These Lighthouses emit pulsed infrared lasers that effectively enables submillimeter precision. This differs from reality though as there is a lot of ambient infrared radiation and reflective surfaces that can interfere with this making it a bit less reliable. The internal sensors of the HMD and controllers also experience drift, which is cancelled out by the Lighthouses, but this is something that might need to be taking into account if doing high precision recording. The accuracy that can be expected is within the range of about 2 millimeters [7].

2.2 Natural Gestures

One often hears talk about something being natural or intuitive, within computer interaction this would imply it being something natural or intuitive to a person who already knows the overarching system. Therefore there should be natural gestures that could be used in VR. Schmidt en al. [8] suggest that user interaction should be invisible when interacting with a system, at least to an extent where the person can focus on performing the task, without worrying about the technology itself. Seeking intuitive and natural gestures is a pursuit that might be hard without the system assisting the user.

2.2.1 Gesture Language

When people speak or explain things, they have a tendency of moving their hands around to further emphasize a point or add to their descriptions. This is something that accompanies spoken language and is mostly spontaneous. These gestures are categorized into four different types [9][10].

- Iconic gestures use concrete aspects of the same scene that the speech is representing.
- Metaphoric gestures are representational, they do not represent any physical form, it is instead a form of gesture that comes from common metaphors.
- Deictics gestures uses the space in between the user and the person or object the person is speaking to. The space is a metaphorical representation of the subject being talked about.
- Beat gestures are usually accompanied by a persons mistake or discontinuity in speech.

2.3 Gesture Recognition

Gesture recognition is motion made by humans that in turn can be interpreted by other humans to mean a specific thing [11]. This interpretation can also be performed by computers, which in turn can give feedback to the people using that computer as to whether or not that gesture was then recognized. In computer science there are many ways of interpreting gestures, there are two types of gesture types that are used as a high-end classifier [12]. These two are:

- Online gestures: Refers to gestures that activate as you are performing actions, such as picking up or rotating an item. These are types of gestures are direct manipulation gestures.
- Offline gestures: Refers to gestures that activate after a certain gesture is done. An example would be a circle being drawn, and as soon as it is drawn a menu pops up.

2.3.1 Gesture Detection Devices

There are several types of devices that are currently in widespread use for gesture detection. These are generally camera based and movement-sensor based.

Camera based Gesture detection

There are cameras that are specifically made for capturing gestures, but there is a lot of research that has been done and is still being done on gesture recognition using 2D images taken through a 2D camera. There are challenges in doing this in real-time, but there are still many smart people out there that continue pursuing ways that circumvent the possible challenges [13].

As for the specific camares for gesture detection, there are two types of 3D gesture detection devices generally used for camera based 3D gesture detection.

- Depth camera: Uses depth data gathered to detect gestures at short ranges. It uses structured light or time-of-flight to generate the depth data.
- Stereoscopic cameras: Uses two cameras that know of each others positions and rotations to generate a 3D representation of the data gathered. This is turn can be used to detect gestures.

Movement-sensor Based Gesture Detection

There are many types of devices in use that can be utilized for detecting gestures. While some are specifically made for this purpose, there are other devices that can be utilized that are seen as consumer grade. Many devices utilized for both personal use and gaming can be used for detecting gestures. One of the most common types would be a smart phone. These devices have touch screens that can capture touches that could be made into a pattern that would then be used to detect a gesture. They also have inbuilt sensors such as accelerometers, gyroscopes, compasses, magnetometers, GPS-receivers and cameras that could be utilized for the purpose of gesture detection [14] [15] [16] [17] [18] [19].

2.3.2 Template Matching

Template matching has a high degree of accuracy that ensures that you find what you are looking for [20] [21]. Template matching can be prone to giving false results if the obfuscation of the templates or the obfuscation of the data compared to the template is great. The reason it works so well is that 100% accuracy is not a necessity. Template matching can, with additional work, look for similarities rather than absolutes [22]. Template matching can work together with many different types of filters to get to the result. One with good expandability is using template matching together with particle filters [23].

2.3.3 Particle filter

The use of particle filtering [24] is a good match for template matching where a lot of online data is streamed in. The use of particle filtering together with template matching is not widely used as template matching is an image processing technique. But when extending it outside of processing images it can be used to great effect [23] [25]. Using particle filtering will reduce the computational need of template matching as more and more of the data being matches against the template are eliminated. This can be especially true as the number of templates being matched and the size of templates increases.

2.4 Related Work

2.4.1 Air Gestures and Camera Based Gestures

Air gestures are gestures that are performed without having to physically interact with a system. The gestures are usually captured through a 2D camera or depth camera. An example of air gestures is a deck operator on an offshore platform. They need to communicate with ships in order to guide them in the operation they are required to do. One implementation of this is located in Aalesund using a Deck Operation simulator [26]. The implementation uses a Kinect camera for capturing the gestures done by the user. Another example of a air gesture approach is an in progress study on gestures in VR [27]. Their approach is going to be using Unity3D with the plugin VR Infinite Gesture [4]. VR infinite gesture uses a neural network to train in detecting gestures done while interacting with the system with an HTC vive controller. Ruppert et al. [28] shows how effective and vital a touchless air gesture approach can be for surgeons and medical personnel. During surgery a surgeon would not like to contaminate their hands, screen or mouse of a computer system, so a gestural system is the perfect solution for them when needing to look up information. Leap motion is often used for gesture recognition. Clark et al. [29] created a system where a Leap Motion Controller was attached to an Oculus Rift DK2 to detect hand gestures for a sign language study. Mahbub et al. [30] use a template matching approach for one-shot-learning gestures. The one-shot-learning in their case would be for the system to learn the gesture by only using one video clip. There is an extremely interesting review [5] that goes through many vision-based hand gesture recognition methods and databases that were done between 1998-2014.

The upcoming Android P will have navigation gestures to control their phones without having to use any physical buttons [31]. Both Android and Apple will in the future have air gestures that will let you control your phone without actually touching the screen [32]. The gesture that has so far been described is that you can unlock your phone by just raising it up to your face. This has seen limited use

for Android previously as well, where you could wave your hand over the screen without touching it to answer the phone or flip through web pages.

2.4.2 Movement-sensor Based Gestures

Deyou [33] presents a hand gesture recognition system using a data glove. This system is used for training people to drive an SPG. The system in itself is old as of 2018, but the neural network that they use shows that neural networks are very capable of detecting gestures. Huette et al. [34] present a system for training users with practical skills by capturing experienced users and then getting the new users to replicate their actions. There is a novel approach called Action Capture by Slyper et al.[14]. The approach uses motion capture data from accelerometers and matches the data to a database of motion capture data to then display an animation.

The WiiMote [35] is frequently used in research as it is a cheap controller with many sensors. There is however a flaw in them that is a flaw that also comes from any device that only uses accelerometers and gyroscopes for their movement detection. That flaw is that there is a possibility of cheating [36]. The cheating comes from utilizing the fact that accelerometers pick up a lot of fast movement when flicking the controller. This usually affects larger gestures [37]. For example, a swinging of a sword can be replaced by quick flicks of the wrist [38].

3 Methods

3.1 Overview and Choice of VR HMD

3.1.1 HTC Vive

The VR HMD HTC Vive is a gaming focused VR HMD. It is made in collaboration between HTC and Valve. The HTC Vive HMD has a resolution of 2160x1200 pixels combined, that means that there are two screens, one for each eye. These screens each have 1080x1200 pixel resolution for each eye [39]. The HMD weighs around half a kilogram, making it a fairly lightweight system. The base HTC Vive package contains a HMD, two controllers and two base stations.

3.1.2 Oculus Rift

The Oculus Rift VR HMD did not originally come package with stationary infrared sensors and controllers. This made it so that people usually only had one choice if they wanted a more true VR experience, which was in the form of the HTC Vive. Oculus were however the company that sparked the wave of VR HMD as they showed how much interest there was through doing their kickstarter campaign [40]. The Oculus Rift HMD has the same resolution as the HTC Vive. The base Oculus Rift package contains a HMD, two controllers and two stationary infrared sensors.

3.1.3 Target Device

Both the Oculus Rift and the HTC Vive were considered for the project. The HTC Vive came on top because of its excellent accuracy, see chapter 2.1, and availability. The availability made it so that it was almost no contest, however if there was a Oculus Rift on campus that would also be a target to test gestures. The VR lab at NTNU Gjøvik has four HTC Vives as well as four workstations that can run those VR systems. The accurcacy of the system is important for gesture recognition as random movement errors would cause the recognition to be wrong.

With Unity3D, SteamVR [41] plugin for Unity3D and VRTK [42] plugin for Unity3D the target of the project can more easily be supported by other VR systems. The only stipulation for this is that the system has a HMD and two controllers that can be tracked by position and rotation.

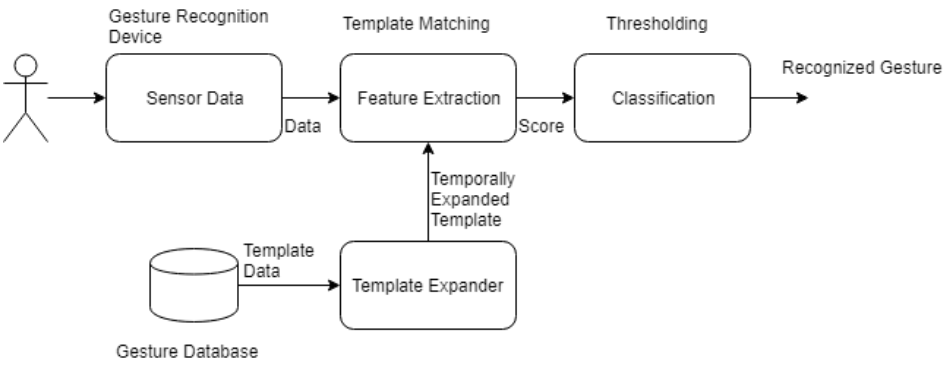

Figure 3: Gesture Recognition Pipeline

3.2 Gesture Recognition

Gesture recognition follows certain rules for most implementations. In this particular project the pipeline is as pictured in Figure 3. The first thing that happens in the pipeline is that the gesture database sends the original template data to the template expander. The template expander then uses linear interpolation to temporally expand the templates. The temporally expanded templates are then sent into the feature extractor to be used as a comparator for the sensor data. The sensor data comes from the HTC Vive HMD and controllers. The data is then processed, see chapter 4.1, before it is sent over to the feature extractor. The feature extractor uses template matching to calculate the scores, see chapter 4.3, from the gesture template and sensor data. The score is sent over to the classifier where it uses thresholding to see whether or not a gesture is detected.

3.3 Particle Filter

A particle filter is an online process that approximates the marginal distribution as observations become available [24]. The particle filtering for the project was slightly dumbed down, this resulted in particles not being destroyed until the particle was the size of the longest template. The infrastructure is there, but the problem is that without any form of optimization work, it just turned out to be way too slow.

3.4 Template Matching

Template matching is a digital image processing technique that uses a template to match to other data to find matches. The way template matching is used in the project is that instead of the templates being in 2D, as you would with template matching when matching images, it uses 1D arrays. This simplifies the process

as one less dimension means that the computational requirement is dropped by a dimension. Matching templates in a 3D environment is a challenge as well, because there are considerations that need to be made in whether or not to employ absolute positions or use normalized magnitudes with distance calculations to do template matching. The absolute position matching ended up being what was used for the gesture activation experiment. The normalized magnitude calculations ended up having issues before the test that resulted in bad sensitivity and selectivity, see chapter 5.2.1 for explanation on sensitivity and selectivity.

3.5 Experiments

3.5.1 Spontaneous Gesture Creation Experiment

The Experiment

The purpose of the experiment was to see what was to see what gesture participants would make when presented with a word or a phrase. The participants were given a short description of what they had to do, before being placed in a VR environment. None of the participants were allowed to observe previous participants. This was done intentionally for fear of contaminating their way of thinking about a gesture. The experiment was also done purely in English for the purpose of not having any different interpretations that may arise from using another language. The participants were then given HTC Vive controllers to make their gestures. These controllers were described as being a metaphor for their own hands. They were then told to wave, as this is a gesture that most people are familiar with. This gives them a pointer as to what a gesture would be. They were then told the passage as seen in the following textbox.

> Have you tried VR before? What I want you to do is do gestures for me when I tell you something to do. What I want you to do is think of what motion would best represent the word that I give. You just need to do the first thing that comes to your mind. For example, what would you do if you were to wave? Everyone knows how to wave, people do it differently but the concept is the same. This test is done without using any buttons on the controllers and only gestures that you do. These are what enable you to do the actions.

The question on whether or not they have tried VR before was there in case someone who had not tried VR needed introduction to how VR works. This was also to determine whether or not a participant would be affected by motion sickness, in that case they were told that they could end the experiment at any time in case they felt that they were unable to continue. The participants were then told to perform 12 different gestures in sequence. Some of the gestures had additional explanations to them, the speak gesture had an explanation just to make sure that they understood what they had to do. The rest of the gestures with explanations

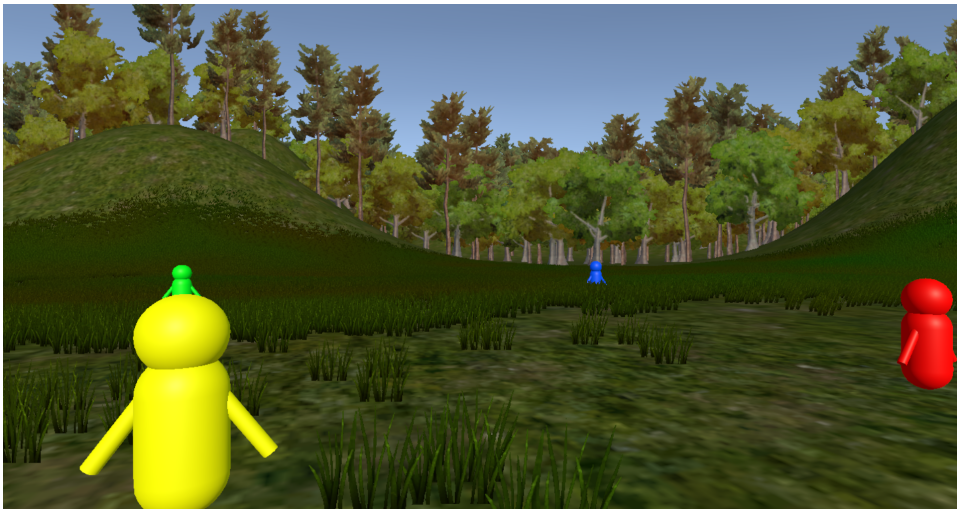

Figure 4: Characters to interact with

had specific characters that they had to perform the gestures towards, as can be seen in Figure 4.

- Speak (if you are muted, how would you enable speaking)
- Shout
- Whisper (To only speak to the closest people to you)
- Whisper to a specific person (Whisper to the red person, whisper to one further away, green character)
- Listen (This is to drown out the other users in the vicinity. Listen to a specific character, red character)
- Starting a private conversation (with yellow character)
- Virtual mobile phone
- Mute yourself
- Unmute yourself
- Mute someone (yellow character)
- Increase volume
- Decrease Volume

Data Gathered

The data that was gathered for this test was only data from the devices themselves. The data was positional, rotational and deltatime. The data was from each of the devices, the HMD and the two controllers. The data was stored in txt files with the format seen in Listing 3.1.

Listing 3.1: Data Format for experiment one

```
Time: the current time.
Position data: x,y,z positions in line with line breaks
    after each entry.
Rotational data: x,y,z euler rotation in line with line
    breaks after each entry.
Delta time: deltatime in line with line breaks after
    each entry.
```

The files were split up so that there was one file for the HMD data, left controller data and right controller data.

3.5.2 Gesture Activation Experiment

The Experiment

The purpose of the experiment was to gather data from participants that see how accurate they are when doing gestures they are told to do. The participants were told to watch a video describing what gestures are to be used in the test. The video link can be found under appendix C.1. After being shown the video, the participant was then set up for VR. When in VR, the participant was asked to perform each gesture in the sequence shown in the video as the tester said the name of the gestures. This was to make sure that the tester remembered the gestures. The gestures the participant had to do were as follows.

- Speak
- Swipe
- Drag
- Sheath
- Circle

After the tester has made sure that the participant understands the gestures, the testing can commence. This would entail the participant doing each gesture three times in a row.

Data Gathered

The data gathered for experiment two was the scores generated when the participants were making gestures. The scores that were gathered were stored in txt files. Each of the five gestures were compared every frame and then stored in the txt file. This data was stored in the format as seen in Listing 3.2. The format used for the txt file is the same format that is used for GNUPlot.

In addition to that, the same data as in Listing 3.1 was gathered just in case of additional processing being needed.

Listing 3.2: Score data Format for experiment two

```
"Gesture1"
score
score
...


"Gesture2"
score
score
...
```

3.6 VR Development

Making an application in VR is challenging in its own, but when considering the prospect of having to implement it from basic libraries makes this a challenge outside of the scope of this project. A game engine is therefore needed to take this burden. This allows the developer to only focus on the application itself rather than all the backend. There are two relevant choices for this, one being Unity3D and the other being Unreal Engine.

3.6.1 Unreal Engine

Unreal Engine was initially released in 1998, making it the older of the two game engines. The engine has been improving ever since then. The engine is written in C++ and supports C++ development. The game engine is free to anyone and the source code is also available to anyone. Unreal Engine also has visual scripting in the form of blueprints. Blueprints makes it easy for rapid development and easy to learn, an entire game can be made without even touching any code. Unreal Engine has an extreme drawback, which is the fact that the documentation for the engine is extremely lackluster, making it hard for people to learn how to delve deeper into the engine. The engine is also used by large scale developers, but a lot of the things they do to improve the engine usually stays in-house without contributing to the community at large.

3.6.2 Unity3D

Unity3D was initially released in 2005 by a Danish company. It was initially only supported Apple OS X, but currently in 2018 it supports 27 platforms. It is often used for rapid prototyping as it is fairly easy to use and learn. It is generally favored by smaller companies and due to the fact that it has excellent documentation. Unity3D offers scripting in both C# and Unityscript/Javascript, however the support for Unityscript/Javascript has begun phasing out as of 2017.

3.6.3 Why Unity

Both Unity and Unreal were considered for the project as they both have their merits. One of the reasons that Unreal was not chosen was that the documentation is too limited. Another reason is that I have little to no experience using the engine. Unity was chosen due to several key factors. I have a lot of experience working with Unity, this experience comes from using it during the entirety of the master program as well as during commercial work as part of a startup company. This also means that I had access to scripts that I have previously written that made the development faster. Unity has excellent documentation. As for the scripting language C#, it really is not my strongest language, but the similarities it has with C++ makes it so that both using and learning the language is fairly easy.

4 Implementation

4.1 Data processing

The processing of the data is done in different steps. It starts with getting the position and rotation data from the HMD and controllers. The next step is to then reset the Y-axis rotation to make it easier to have a consistent area for calculations to take place.

Listing 4.1: Reseting Y-axis.

```
rightControllerData.transform.RotateAround(headData.
    transform.position, Vector3.up, -headData.
    transform.eulerAngles.y);
leftControllerData.transform.RotateAround(headData.
    transform.position, Vector3.up, -headData.
    transform.eulerAngles.y);
headData.transform.eulerAngles = new Vector3(
    headData.transform.eulerAngles.x, 0f, headData.
    transform.eulerAngles.z);
```

Then the HMD position is set back to origin, with the controllers being moved together with the HMD position.

Listing 4.2: Reseting position to origin.

```
rightControllerData.transform.position =
    rightControllerData.transform.position - headData
    .transform.position;
leftControllerData.transform.position =
    leftControllerData.transform.position - headData.
    transform.position;
headData.transform.position = Vector3.zero;
```

Lastly, the position data is forced into being positive.

Listing 4.3: Reseting position to origin.

```
Vector3 tempVec3 = new Vector3(1f, 1f, 1f);
leftControllerGameObject.transform.position = (
    leftControllerGameObject.transform.position +
    tempVec3) / 2;
```

4.2 Particle

A datapoint is gathered every frame, this data is then run through the data processing to be sent to the template class to be stored in a List for template matching.

The data is stored in a data storage class called RecordedData. The RecordedData class stores a List of Vector3s. It also has the calculations for the scores in it a described in Listing 4.4.

4.3 Templates

The templates are generated from raw positional and rotation data that are loaded from file. The data is then processed in the same way as described in chapter 4.1. As the data is processed, they are then stored in lists of Vector3's. To check the template, a function is run to calculate the scores. What this function does first is to check which particle list has the same number of data points as a template. If one such check passes, the data is then processed into scores.

Listing 4.4: Calculating score

```
public List<Vector3> computeScore(List<Vector3>
    templateData)
    {
        List<Vector3> score = new List<Vector3>();
        for (int i = 0; i < dataVector.Count; i++)
        {
            score.Add(powVec3((dataVector[i] -
                templateData[i]), 2));
        }
        return score;
    }
```

The RecordedData class has a function called computeScore that takes in a template to calculate against the data the class has gathered. The function makes a temporary Vector3 list that stores the scores to be returned. The score is calculated with this formula: `score[n] = {(data[n] - template[n])^2}`. As a Vector3 has three components a separate function was needed to calculate the final value.

The generation of a template can be done during run-time and used immediately. This is a side effect that happened as the robustness of the methods made it so that generating templates during run-time was possible without much work.

4.4 Thresholding

The thresholding is fairly simple when the scores have been sent to be checked. The threshold code is in essence an if statement that checks whether or not a score

is below the threshold.

5 Results

5.1 Spontaneous Gesture Creation Experiment

The experiment had 22 participants who each made gestures for 12 different gestures when given the task through verbal communication and it was run over one day, with each participant taking roughly 20 minutes each to run through the entire experiment. The experiment was very interesting and unexpected. The $H1$ for this experiment was that the gestures that the participants ended up making would be fairly similar across every participant. This however turned out to be false. There was only one participant that did the same type of gesture envisioned by me for many of the gestures, many of those gestures were gestures discussed with others prior to the experiment to get an idea of what type of action each should be. This created a bias towards creating gestures in similar ways with the people that discussed this, as they see it as being the most natural thing since they already know what it is. The purpose of other participants not being allowed to observe was exactly done for this reason, and the results of this shows. The gestures that were done were very different for each participant. The initial assumption for the Speak gesture what that people would use a metaphor for a microphone to create the gesture as seen in figure 5. The way that the gestures were so different for each other made it hard to actually extract any meaningful data, it was only possible to do manual visual observation from the gesture data that was gathered.

Figure 5: Left: Holding microphone. Right: Doing a metaphoric gesture for speak

5.2 Gesture Activation Experiment

The second experiment went as expected. The explanation of what people were going to do went pretty well with only a few people messing up the gesture before the start of the experiment. As the experiment had pretty strict gestures this was to be expected. The experiment had 10 participants that each formed 3 of each of the 5 gestures each, giving 15 data files per participant. This also equates to comparisons between 750 gesture score points as each of the data files contains gesture data on all 5 gestures running concurrently and being detected.

5.2.1 Sensitivity and Selectivity

Sensitivity and selectivity are statistical measures of performance of a binary classification test. These values are used to find the crossover points of the two measures in graphs. This crossover point will show the ideal threshold that should be used for the gestures that they test. As the threshold gets higher the higher likelihood there is for sensitivity and selectivity to crossover. Sensitivity tests whether or not a gesture is correctly detected.

$$Sensitivity = \frac{numberoftruepositives}{numberoftruepositives + numberoffalsenegatives} \tag{5.1}$$

Selectivity tests whether or not a gesture is correctly rejected.

$$Selectivity = \frac{numberoftruenegatives}{numberoftruenegatives + numberoffalsepositives} \tag{5.2}$$

The sensitivity and selectivity for each of the gestures generally follow a similar pattern.

Speak

The optimal threshold for the gesture Speak as seen in Figure 6 turns out to be 1.5 with a sensitivity and selectivity of 0.95. For this gesture to get a higher crossover, the Drag gesture needs to either have a different threshold or be removed. Th reason for why Drag turns out to give false positives is due to the endpoint position of Speak and Drag are very similar.

Swipe

The optimal threshold for the gesture Swipe as seen in Figure 7 turns out to be 2 with a sensitivity and selectivity of 0.92.

Drag

The optimal threshold for the gesture Drag as seen in Figure 8 turns out to be around 1.4 with a sensitivity and selectivity of 0.84. Drag has the same issue with its crossover in the same way as Speak, just that Speak is the one affecting Drag.

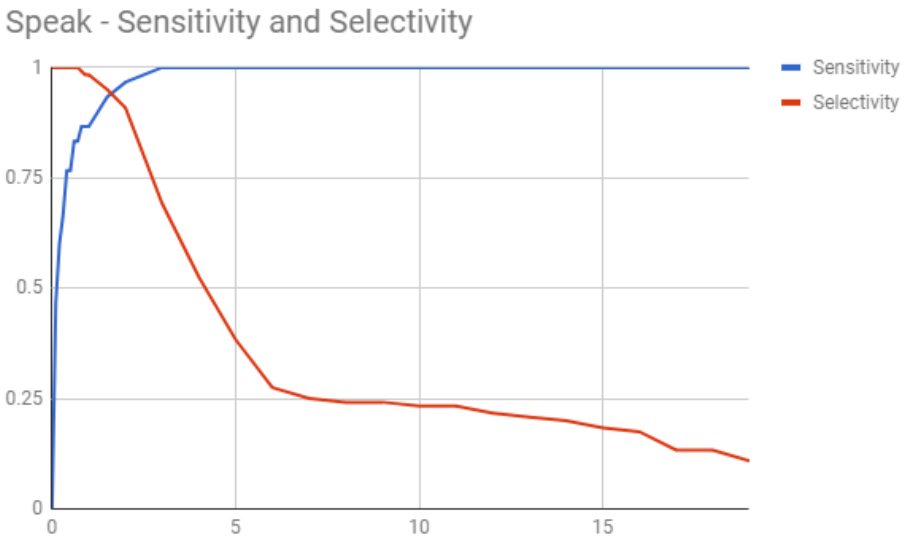

Figure 6: Speak: Sensitivity and Selectivity

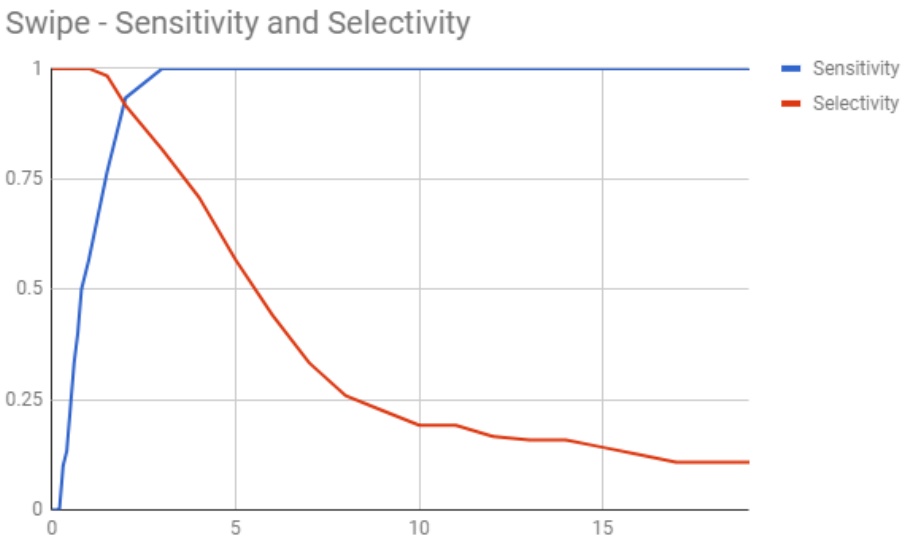

Figure 7: Swipe: Sensitivity and Selectivity

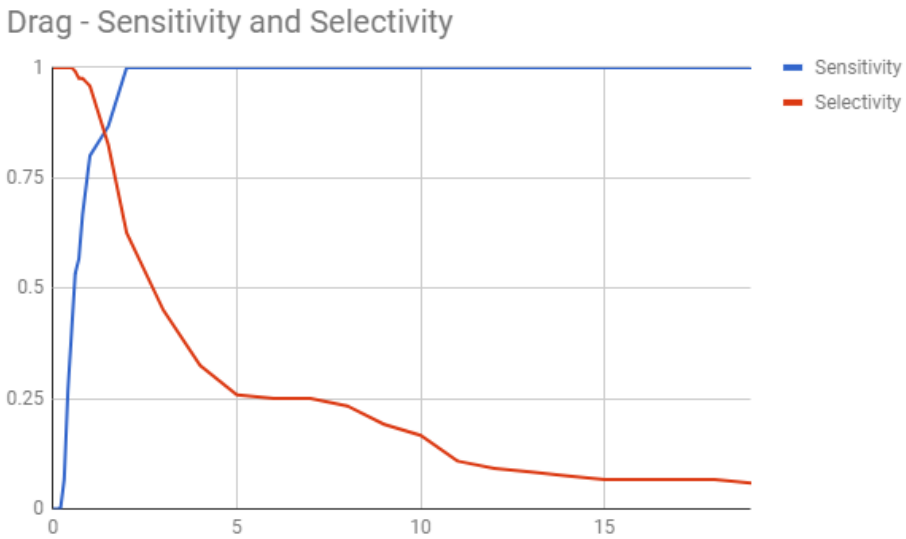

Figure 8: Drag: Sensitivity and Selectivity

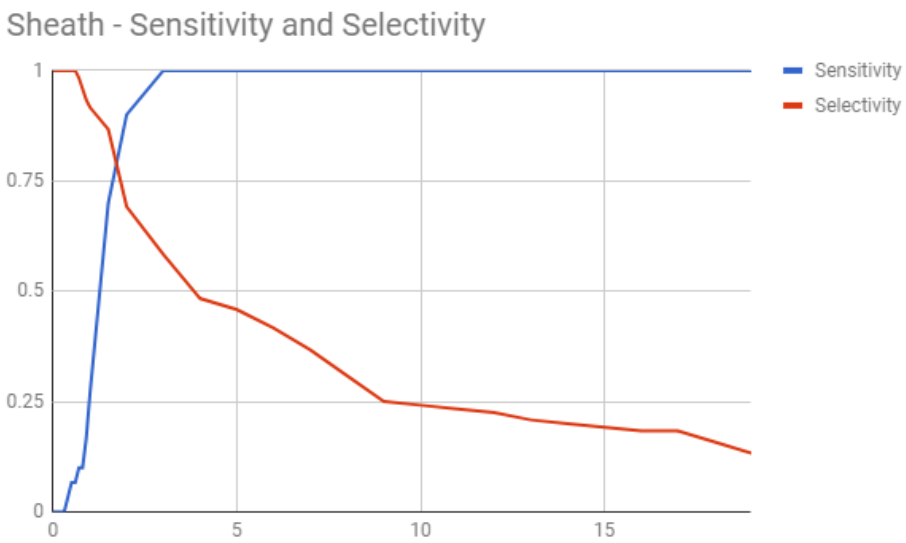

Figure 9: Sheath: Sensitivity and Selectivity

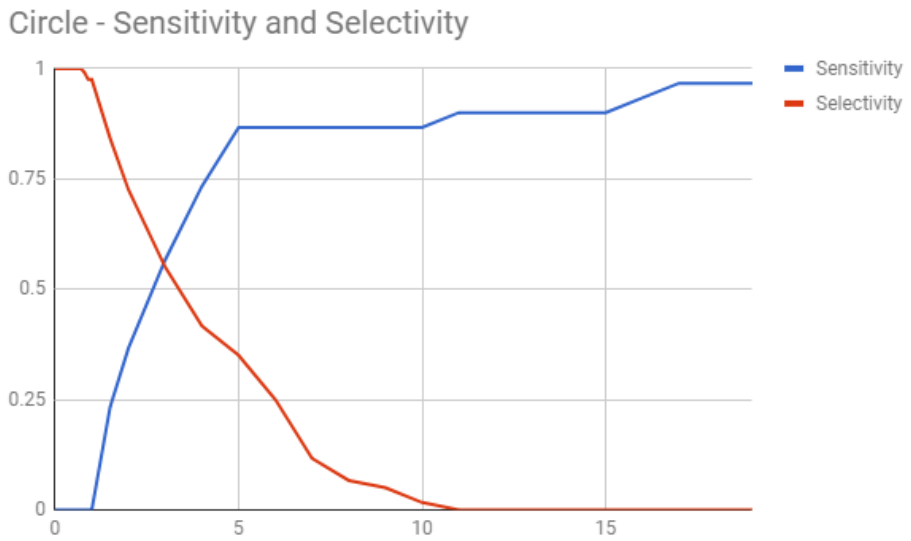

Figure 10: Circle: Sensitivity and Selectivity

Sheath

The optimal threshold for the gesture Sheath as seen in Figure 9 turns out to be around 1.75 with a sensitivity and selectivity of 0.8.

Circle

The optimal threshold for the gesture Circle as seen in Figure 10 turns out to be around 3 with a sensitivity and selectivity of 0.55.

5.2.2 Individual Gestures

These are examples of the score data gathered from one of the participants performing gestures during the experiment. The Figures 11-15 are graphs depicting the scores of all five gestures running concurrently and shows how the scores get closer to zero. When discussing this data, the threshold will be assumed to be 1, as this is what was used when testing the gestures in real-time.

Speak

The Speak gesture score as seen in Figure 11 is extremely close to zero. Speak was the gesture that got closest to zero in experiment two with the score being 0.04. This is almost perfect in consideration of the template.

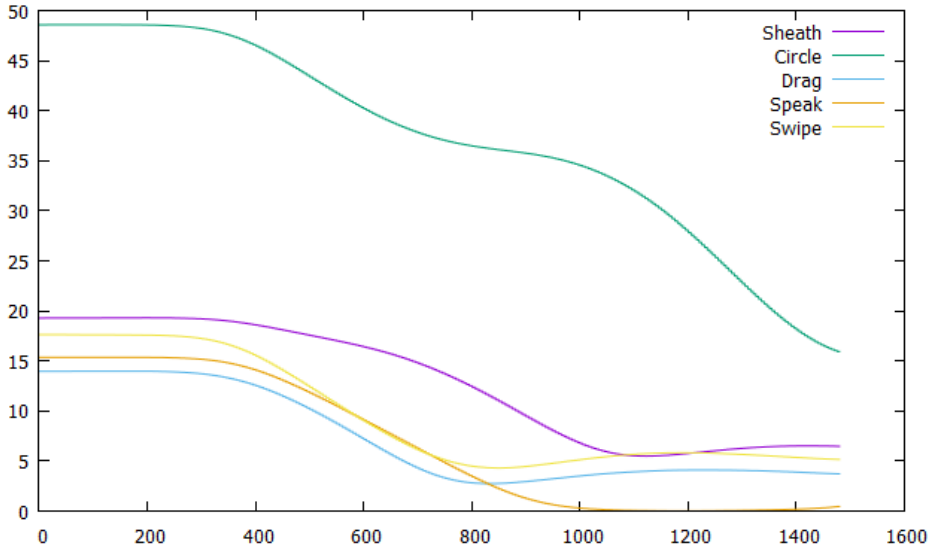

Figure 11: Speak

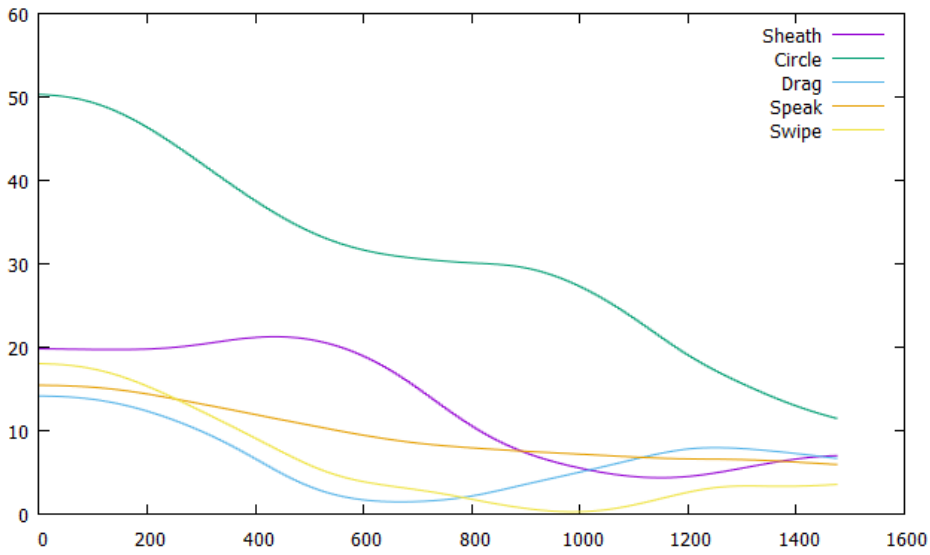

Figure 12: Swipe

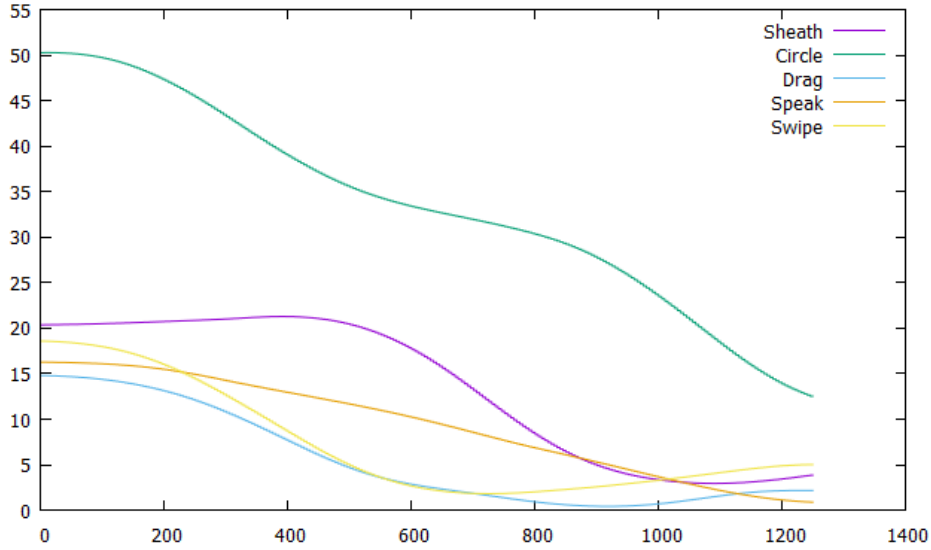

Figure 13: Drag

Swipe

The Swipe gesture score as seen in Figure 12 is fairly close to zero as well, but this being perfect is hard as the gesture is based on a person having a certain arm length. This differs from person to person meaning that there will be differences.

Drag

The Drag gesture score as seen in Figure 13 is fairly close to zero. It is executed quite well, but there is another gesture that is approaching zero as well, that being the Speak gesture. The Speak gesture triggered a false positive to occur towards the end.

Sheath

The Sheath gesture score as seen in Figure 14 does get close to zero. The Swipe gesture throws a false positive before the Sheath gesture throws a true negative. This comes about due to improper training, as it is hard to get people to do gestures the exact way they are supposed to when metaphors are used to describe the gestures. The Sheath does also throw a true positive.

Circle

The Circle gesture score as seen in Figure 15 is not that close to zero. It gets very close to the threshold, but the complexity of the gesture makes it hard for users other than the creator to actual do the gesture. The Drag gesture does not throw a

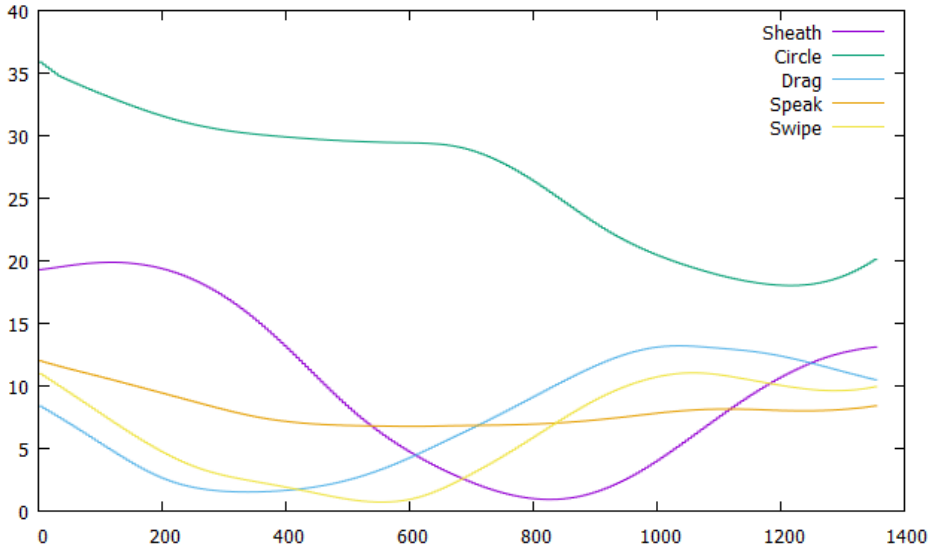

Figure 14: Sheath

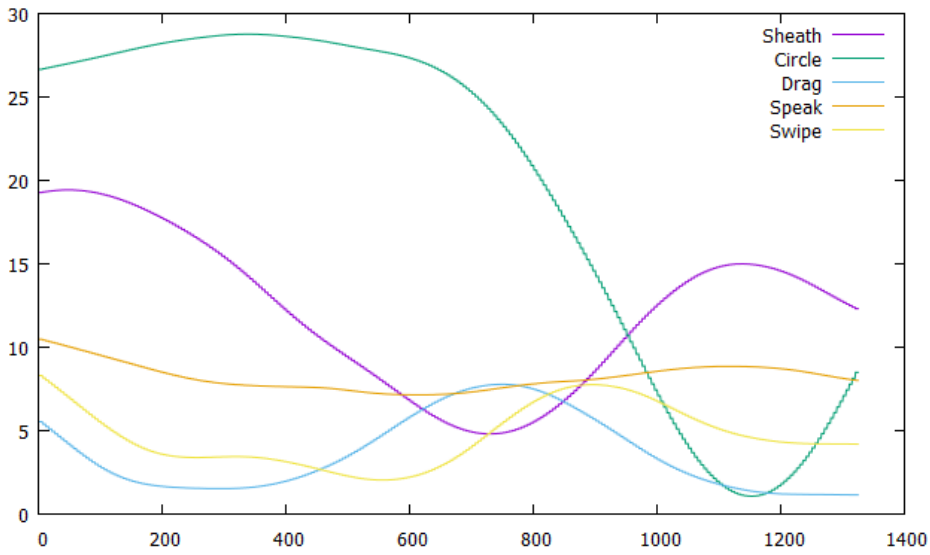

Figure 15: Circle

false positive in the graph.

6 Discussion

6.1 Spontaneous Gesture Creation Experiment

The experiment had some interesting things that were visually observed about how humans think when interacting with a system. There was in fact a thirteenth gesture that was done for the first 6 participants, this gesture was on how you would talk into a Walkie-Talkie, but was dropped due to people not knowing what a Walkie-Talkie was.

6.1.1 Observable Patterns

There were observable patterns in how people made gestures in some cases, especially the cases of muting and unmuting yourself as well as increasing and decreasing volume. For the volume control the gestures usually started at some either arbitrary point or near the ear. The participants would then proceed to lift their arms higher or lower depending on whether or not their decreasing or increasing volume. The gesture was not always straight up or down, sometimes the participants did spirals while raising their hands. There were also participants who did gestures with the metaphor of turning volume knobs that you might see on audio mixing boards or loudspeakers. So there is no definite answer on what would be the best gesture for volume control. As for the muting and unmuting yourself gestures one might think that to mute youself you would make a gesture, and then make that same gesture again to unmute yourself. This was however not the case most of the time. Some of the patterns that followed would be to do the mute gesture and then to unmute the participant would do it in the opposite direction or use their other hand to do the gesture in the opposite way. The mute yourself gesture did not have any specific gesture that was thought to be good prior to the experiment, but there was for the unmute yourself gesture. The unmuting of yourself was originally thought to be the same as the speak gesture as that is what logic would say would be correct, but people are not computers.

6.1.2 Expected Gestures

Some of the gestures required to be created in the experiment were already predicted to have certain ways of doing them. These include virtual mobile phone and listen. The virtual mobile phone was expected to have the participant hold the HTC Vive controller as one would a phone, either an analog phone or smart phone, and bring it up to the ear and keep it there. There were some really interesting outliers

with an unexpected gesture that actually should make sense in VR. The gesture was actually to hold the controller in front of the face at a distance of roughly 30cm. This would symbolize video chatting, which would probably be a perfectly valid way of communicating in VR.

The listen gesture was expected to involve bringing either one or both of the controllers up to your ears. And that is exactly what happened for most of the participants.

6.1.3 Aggregate gestures

There were some gestures that were supposed to be aggregates, these gestures include whispering to someone specific, starting a private conversation and mute someone. Only the starting a private conversation gesture had both of the supposed aggregate parts as separate gestures, those being speak and listen. The way to start a conversation was predicted to be both the speak and listen gestures used at the same time. None of the participants did this though. Most of the participants did an aggregate of a microphone like gesture as seen in Figure 5 and pointing or looking towards a pawn. Muting someone and whispering to someone specific was an aggregate of first doing the gesture for muting or whispering and then pointing at the person you wish to act on.

6.1.4 Validity

The experiment would probably turn out much the same for most unless the participants already have preconceived notions on how to do gestures. Currently, as gestures are not a staple in VR, it turns out that getting data on how people do gestures is fairly easy. The validity of the project will be hard to predict if gesture recognition becomes standard in VR.

6.2 Gesture Activation Experiment

This project was pretty straight forward, the main thing in the experiment was to gather pure gesture data. There was also a secondary objective, and that was to see whether or not people could learn gestures in a short span of time with only a video to demonstrate how to do the gesture. And it turns out that learning five different gestures is not that hard, especially not if they are named well. The naming of the gestures contributed to people being able to remember and differentiate between the different gestures. There were a few instances where the participants did the gestures the opposite direction of what they were supposed to it in, specifically the swipe and circle gesture. The swipe gesture should probably work either way you swipe it, but for the experiment it was only in one specific way. The circle gesture did have a few participants who did it in the opposite direction during the explanation, but this was rectified before gathering data. The main problem

with the circle gesture is that the template was pretty rigid, meaning that the start point was very specific. The start point of the circle was to start from the right side of the circle and then draw it counter-clockwise. The issue however, even after checking that the participants did it correctly, was that the participants forgot what the start point was or started drawing it from their mental model of how they draw circles. This could be mitigated by simply making the template a rolling template. Meaning that you roll the data starting from different points on the template and see whether it fits somewhere.

The results from the data was better than expected for all the gestures except the Circle gesture. The data shows that the precision of the system is fairly high. If the gestures had individual thresholds instead of each of them having the same threshold, the sensitivity and selectivity values would improve by quite a bit. This is not the ideal solution though. The most ideal would be for users to create their own gestures, which is fully possible with what the project has done.

6.2.1 Weakness of the experiment

This experiment was supposed to have another experiment that was supposed to look at key binding of gestures instead of using pre-existing templates. Due to time constraints of the implementation, the key binding part of the experiment was not done. The key binding aspect is there, but doing it in real-time is not. Doing the experiment where you would need 30 minutes of set up with the participant there for an experiment that takes 10 minutes is not something that would be very suitable. Another weakness of the experiment is that instead of the tester starting and stopping the recording, the participant should have had the power to do that. As that is what would be how an actual implementation of a gesture recognition system in a game or application would do. Press and hold a button, perform the gesture, then release.

7 Conclusion

7.1 Research Questions

In this part, the research question will be discussed and partially answered, as research questions can never be fully answered.

7.1.1 RQ 1, Given tasks users will form similar gestures

From what was observed during the spontaneous gesture creation experiment participants will not form similar gestures, but they will usually follow a pattern. $H1$ was rejected as the gestures very wildly different from each other, this was found out through visual observation of the data as playing back the data is fully possible.

7.1.2 RQ 2, Can users form gestures given simple instructions

From what was observed in the gesture activation experiment, users are able to form gestures, but complex gestures are too hard to do if the gestures are premade based on a person that is not the user.

RQ 2.1 Is the threshold specific per user

The threshold should not be specific for each user. To calculate what the threshold would be for each user would be too hard to do programmatically. At least compared to doing it on a per-gesture basis.

RQ 2.2 Is the threshold specific per gesture

With the current implementation thresholds would need to be specific for each gesture to get adequate sensitivity and selectivity values for the gestures. This is especially true for complex gestures. With a more robust system with narrower detection using cross-correlation, the need for specific thresholds might not be necessary. The gesture activation experiment supported $H2$. It showed that each of the gestures formed in the experiment had different optimal thresholds. But the support for this hypothesis may be questioned with the aforementioned robust system.

7.1.3 RQ 3. Can gestures be recorded and detected in real-time

Yes they can, it works very well with template matching and particle filtering. The only issue with the current implementation is that it is too slow when many gestures are in the system. This limits the system to processing only 10 gestures, which is enough to do almost any action. However, if you want those temporally expanded, then the system needs optimization.

7.2 Future Work

There is plenty of future work that should be done. Many of these improvements have been mentioned throughout the thesis, but will be compiled here.

7.2.1 Implementation

A good place to start improving on would be the gesture recognition implementation. Getting the particle filter to work at full capacity would be an ideal way of improving the performance of the system. As for the template matching, the matching should be improved upon to give a more stable and possibly expandable gesture creation system. Another expansion for the template matching would be to get real-time key binding of gestures up and running, this would ensure that the accuracy of gesture recognition will improve a lot. Another thing that would be excellent to have when doing the template matching would be to actually incorporate the rotational values as well. For both template matching and particle filtering there should be GPU computation set up for them as this will increase the performance by many folds.

7.2.2 The Data

The data that was gathered has a lot of potential uses as it has 6DoF data and deltatimes recorded, so it is pure motion data. Some of the things that would be interesting to see from the data would be temporal invariance. Seeing the selectivity and sensitivity data using the rotational data as well would be interesting although it was not captured with that in mind. The Sheath gesture would be especially good to compare with rotational data added in as the motion is like sheathing a sword, meaning that a complete 180 degree turn in the x axis is done before the gesture is complete.

7.2.3 Further Studies and Experiments

An experiment to check whether or not people are capable of using key binding successfully with the gestures would be incredibly useful. There should be a study on whether or not the gestures created in the thesis are actually what people would find intuitive.

Bibliography

[1] File:mixed reality with a virtual reality headset.png. Accessed 27. May 2018. URL: https://commons.wikimedia.org/wiki/File:Mixed_Reality_with_a_Virtual_Reality_Headset.png.

[2] File:6dof en.jpg - wikimedia commons. Accessed 27. May 2018. URL: https://commons.wikimedia.org/wiki/File:6DOF_en.jpg.

[3] Virtual reality pawn and components plugin. Accessed 30. May 2018. URL: https://www.unrealengine.com/marketplace/vr-pawn-components-plugin.

[4] Edwon Studio. 2017. Vr infinite gesture. Accessed 18. December 2017. URL: http://edwonstudio.com/vr-gesture.

[5] Pisharady, P. K. & Saerbeck, M. 2015. Recent methods and databases in vision-based hand gesture recognition: A review. *Computer Vision and Image Understanding*, 141, 152–165.

[6] LaValle, S. M., Yershova, A., Katsev, M., & Antonov, M. 2014. Head tracking for the oculus rift. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 187–194. IEEE.

[7] Kreylos, O. May 2016. Lighthouse tracking examined. Accessed 7. May 2018. URL: http://doc-ok.org/?p=1478.

[8] Schmidt, A., Pfleging, B., Alt, F., Sahami, A., & Fitzpatrick, G. 2012. Interacting with 21st-century computers. *IEEE Pervasive Computing*, 11(1), 22–31.

[9] McNeill, D. & Pedelty, L. L. 1995. *Language, gesture, and space*. Lawrence Erlbaum Associates.

[10] Cassell, J. 1998. A framework for gesture generation and interpretation. *Computer vision in human-machine interaction*, 191–215.

[11] Mitra, S. & Acharya, T. 2007. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3), 311–324.

[12] Kammer, D., Keck, M., Freitag, G., & Wacker, M. 2010. Taxonomy and overview of multi-touch frameworks: Architecture, scope and features. In *Workshop on Engineering Patterns for Multitouch Interfaces*.

[13] Yang, M.-H., Ahuja, N., & Tabb, M. 2002. Extraction of 2d motion trajectories and its application to hand gesture recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 24(8), 1061–1074.

[14] Slyper, R. & Hodgins, J. K. 2008. Action capture with accelerometers. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 193–199. Eurographics Association.

[15] Farella, E., Benini, L., Ricco, B., & Acquaviva, A. 2007. Moca: A low-power, low-cost motion capture system based on integrated accelerometers. *Advances in Multimedia*, 2007(1), 1–1.

[16] Fontaine, D., David, D., & Caritu, Y. 2003. Sourceless human body motion capture. In *Proceedings of Smart Objects Conference (SOC'03), Grenoble, France*.

[17] Sama, M., Pacella, V., Farella, E., Benini, L., & Riccó, B. 2006. 3did: a low-power, low-cost hand motion capture device. In *Proceedings of the conference on Design, automation and test in Europe: Designers' forum*, 136–141. European Design and Automation Association.

[18] Pascu, T., White, M., & Patoli, Z. 2013. Motion capture and activity tracking using smartphone-driven body sensor networks. In *Innovative Computing Technology (INTECH), 2013 Third International Conference on*, 456–462. IEEE.

[19] Seifert, K. & Camacho, O. 2007. Implementing positioning algorithms using accelerometers. *Freescale Semiconductor*, 1–13.

[20] Lewis, J. P. 1995. Fast template matching. In *Vision interface*, volume 95, 15–19.

[21] Briechle, K. & Hanebeck, U. D. 2001. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, 95–103. International Society for Optics and Photonics.

[22] Brunelli, R. 2009. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons.

[23] Li, H., Duan, H.-B., & Zhang, X.-Y. 2010. A novel image template matching based on particle filtering optimization. *Pattern Recognition Letters*, 31(13), 1825–1832.

[24] Turner, L. & Sherlock, C. 2013. An introduction to particle filtering.

[25] Shan, C., Tan, T., & Wei, Y. 2007. Real-time hand tracking using a mean shift embedded particle filter. *Pattern recognition*, 40(7), 1958–1970.

[26] Strazdins, G., Pedersen, B. S., Zhang, H., & Major, P. 2017. Virtual reality using gesture recognition for deck operation training. In *OCEANS 2017-Aberdeen*, 1–6. IEEE.

[27] Park, H., Jeong, S., Kim, T., Youn, D., & Kim, K. 2017. Visual representation of gesture interaction feedback in virtual reality games. In *Ubiquitous Virtual Reality (ISUVR), 2017 International Symposium on*, 20–23. IEEE.

[28] Ruppert, G. C. S., Reis, L. O., Amorim, P. H. J., de Moraes, T. F., & da Silva, J. V. L. 2012. Touchless gesture user interface for interactive image visualization in urological surgery. *World journal of urology*, 30(5), 687–691.

[29] Clark, A. & Moodley, D. 2016. A system for a hand gesture-manipulated virtual reality environment. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 10. ACM.

[30] Mahbub, U., Imtiaz, H., Roy, T., Rahman, M. S., & Ahad, M. A. R. 2013. A template matching approach of one-shot-learning gesture recognition. *Pattern Recognition Letters*, 34(15), 1780–1788.

[31] Wright, A. May 2018. Everything new in android p developer preview 2. Retrieved 31. May 2018 from. URL: https://www.xda-developers.com/everything-new-android-p-developer-preview-2/.

[32] Gurman, M. April 2018. Apple working on touchless control and curved iphone screen. Retrieved 31. May 2018 from. URL: https://www.bloomberg.com/news/articles/2018-04-04/apple-is-said-to-work-on-touchless-control-curved-iphone-screen.

[33] Xu, D. 2006. A neural network approach for hand gesture recognition in virtual reality driving training system of spg. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, 519–522. IEEE.

[34] Huette, S., Huang, Y., Kallmann, M., Matlock, T., & Matthews, J. L. 2011. Gesture variants and cognitive constraints for interactive virtual reality training systems. In *Proceedings of the 16th international conference on Intelligent user interfaces*, 351–354. ACM.

[35] Wii remote. Retrieved 31. May 2018 from. URL: https://en.wikipedia.org/wiki/Wii_Remote.

[36] Wingrave, C. A., Williamson, B., Varcholik, P. D., Rose, J., Miller, A., Charbonneau, E., Bott, J., & LaViola Jr, J. J. 2010. The wiimote and beyond: Spatially convenient devices for 3d user interfaces. *IEEE Computer Graphics and Applications*, 30(2), 71–85.

[37] Geurts, L., Van Woensel, A., & Abeele, V. V. 2012. No sweat, no fun: large-gesture recognition for computer games. In *Proceedings of the 4th International Conference on Fun and Games*, 109–112. ACM.

[38] Crampton, N., Fox, K., Johnston, H., & Whitehead, A. 2007. Dance, dance evolution: Accelerometer sensor networks as input to video games. In *Haptic, Audio and Visual Environments and Games, 2007. HAVE 2007. IEEE International Workshop on*, 107–112. IEEE.

[39] Vive virtual reality system. Accessed 31. May 2018. URL: https://www.vive.com/us/product/vive-virtual-reality-system/.

[40] Oculus rift: Step into the game. Accessed 31. May 2018. URL: https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game/description.

[41] Steamvr plugin. Accessed 31. May 2018. URL: https://assetstore.unity.com/packages/templates/systems/steamvr-plugin-32647.

[42] Vrtk - virtual reality toolkit. Accessed 31. May 2018. URL: https://vrtoolkit.readme.io/.

A Program Link

A.1 Link to Repository

Link to a repository containing the only the programs. The actual repositories used can not be shared a the repositories are private as they contain data from the experiments. https://github.com/Almightyquad/Gesture_recognition

B Experiment One

B.1 Experiment Data

https://drive.google.com/drive/folders/1dWjaFY_OLUU-GN1wHZIi7lQxQtj9XDID?usp=sharing

C Experiment Two

C.1 Video explaining gestures

https://drive.google.com/file/d/1FhINO5ewUfuIW1iqOVycL4BxCarq7pw6/view?usp=sharing

C.2 Experiment Data

https://drive.google.com/drive/folders/1TJa2Ljya0xzJcTe9pGgriM6mDOr5A2LE?usp=sharing