Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Master in Computer Science

Documentation

Using Leap Motion for Gesture-Based Querying of Cultural Assets.

Candidate: Edgar Obare

Supervisor: Sven Helmer

April 5, 2018

Using Leap Motion for Gesture-Based Querying of Cultural Assets.

Edgar Obare

April 5, 2018
Version: Final

Edgar Obare
Using Leap Motion for Gesture-Based Querying of Cultural Assets.
Documentation, April 5, 2018

Supervisor: Sven Helmer

Free University of Bozen Bolzano

Master in Computer Science
Faculty of Computer Science
Universitätsplatz 1 - Piazza Università, 1
39100 , Bolzano

Abstract

Take a moment and imagine trying to describe to someone in words how to uncork a wine bottle using a cork screw, its a daunting task, but its much easier to just demonstrate the process [6]. This project came about from the need to find a better way to retrieve information for domains where using traditional means like textual based was a challenge. In the world of artifact preservation where objects have intangible properties and so are hard to represent and describe, we wanted to create a way in which to represent these objects in an intuitive way and so we use gestures. How we interact with our devices to access information has been changing over the years[9], gestures are a natural way in which humans communicate. This project focuses on hand held tools and their uses, with the aim of recording gestures useful to operate these tools and ultimately categorize them. Giving meaning and context to otherwise indescribable artifacts. We use hand tracking technology and specifically the leap motion device, which is comparatively an affordable and accessible motion device, together with our application allowing us to train and read gestures. A user would be able to search artifacts by just making a gesture with their hands. This project is undertaken in collaboration with the faculty of Design & Art and the faculty of Computer Science, as an interdisciplinary project using technology to preserve our cultural heritage.

Keywords: Information retrieval, gesture recognition, leap motion, human machine interaction, gesture based querying.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Project Goals . . . 2
- 1.3 Thesis Structure . . . 2

2 Background **5**
- 2.1 Leap Motion . . . 5
- 2.2 State of the Art . . . 9
  - 2.2.1 Machine Learning . . . 9
- 2.3 Related Work . . . 12
  - 2.3.1 Interpreting cultural artifacts . . . 12
  - 2.3.2 Similar Applications . . . 15

3 System **17**
- 3.1 Requirements . . . 17
  - 3.1.1 User Requirements . . . 17
  - 3.1.2 System Requirements . . . 21
- 3.2 Design . . . 21
  - 3.2.1 Interface . . . 21
  - 3.2.2 Architecture . . . 23
- 3.3 Integration . . . 25
  - 3.3.1 Using the $P Algorithm . . . 26
  - 3.3.2 Normalization . . . 26
  - 3.3.3 Gesture Based Query . . . 27
- 3.4 Implementation . . . 30
  - 3.4.1 Launch . . . 31
  - 3.4.2 Search . . . 32
  - 3.4.3 Train Gestures . . . 33

4 Evaluation **35**
- 4.1 Device Performance . . . 35
- 4.2 Recognition Performance . . . 36

5 Conclusion **41**
- 5.1 Future Work . . . 42

5.2 Final thoughts 44

Bibliography **47**

1 Introduction

1.1 Motivation

The focus of this work is in the context of preserving cultural and natural heritage which has been a priority for many countries since world war II, where we lost a lot of artifacts of significance. It still happens even to this day for example archives originating from the Ottoman empire were burned and destroyed as a result of the violent political protests in Bosnia and Herzegovina, Europe. Technology has been instrumental in the efforts of preservation of artifacts by digitizing, modelling and visualizing them to be more accessible to more people [8].

The project focuses on hand operated tools. Tools have been an important step for mankind in our evolution, allowing us to accomplish feats impossible without them. For example the Egyptians used copper tools as shown in figure 1.1 to carve the stones that built the pyramids.

Fig. 1.1: Egyptians traveled far to mine copper for the tools to cut the stones for the pyramids.

Tools have form and function and one is able to discern their use from their physical characteristics. We will look at ways in chapter 2.3.1 to interpret this artifacts so as to develop a structured and systematic method for understanding these artifacts for our application. Our goal is to make a collection of objects with historic significance accessible by describing how they are used with gestures.

Why Gestures?

Gestures are actions human beings do all the time, mapping directly to "user intent" without forcing a person to learn and remember commands or options [9]. Simple actions in non-verbal communication for example pointing, nodding, and sign language use gestures to communicate. A user would be able to describe using gestures, how the object is used with a device that has the capability of tracking their motion. Examples include data gloves, kinect devices or hand tracking technology like the leap motion controller [15] with sub-millimeter accuracy. For this project we mainly focused on the Leap motion controller as it was the best balance between cost, effectiveness and value.

1.2 Project Goals

The goals of this project can be summarized as follows:

- Analyzing a collection of hand operated tools to identify their attributes not just physical but also their cultural and historical significance.
- Investigate models to represent these tools based on how they are operated and represent them using hand gestures.
- Train and recognize these gestures using a leap motion controller.
- Querying a database and matching these gestures with digitized representations of these tools.

1.3 Thesis Structure

Chapter 2

This chapter gives background information on the project. We look at the technology we will use which is the leap motion controller, how it works. We discuss the state of

the art in this domain, specifically machine learning, similar projects that implement gesture recognition and finally ways to interpret cultural artifacts.

Chapter 3

In this chapter we cover all the aspects of the Leapy application system, from requirements analysis, to design of the system, technologies we integrated and finally the implementation where we get to see how it all pulled together and works.

Chapter 4

In this chapter we see the evaluation and testing of the system, and discuss the different aspects of the project and how they impact the overall results of this system.

Chapter 5

In this chapter we make our conclusions and discuss the general goals of this project as described in chapter 1.2, and whether they were achieved. We also cover future work to improve this application.

2 Background

2.1 Leap Motion

Fig. 2.1: Leap motion device

The Leap motion device as shown in figure 2.1, is a small rectangular box shaped sensor that is able to detect and track hand and tool motion. The data being tracked includes the palm, finger positions, direction and velocity all accessible through its SDK (Software Development Kit). It makes use of two monochrome infrared cameras and three infrared LED's at the top of the device, allowing it to generate 3D images from the 2D frames captured by both cameras. These sensors can track infrared light with a wavelength of 850nm and is claimed to have an accuracy of 0.7mm [15]. Similar motion devices like the Kinect which are more expensive have an accuracy on average of 1.5 cm.

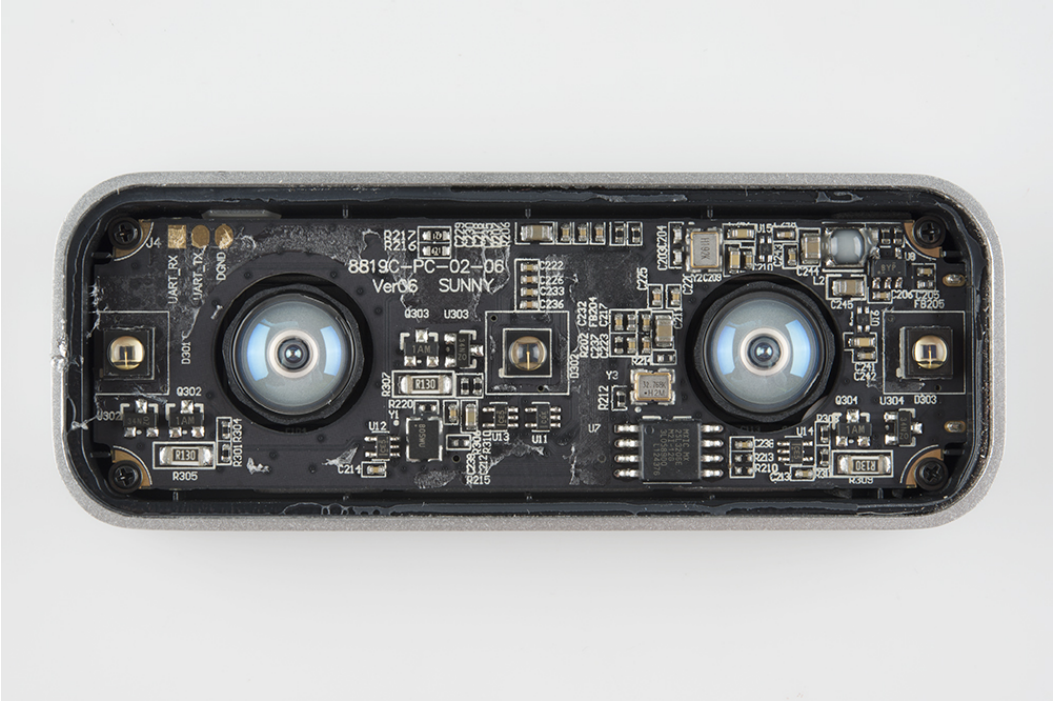


Fig. 2.2: Top PCB of a leap motion device

It is USB powered device designed to be used on a flat desk or mounted on a virtual reality headset. Its field of view is a conical shape ranging from 2.5 cm to 60 cm above it as shown in figure 2.3. "It is a popular misconception that the leap motion device generates a depth map, but instead applies advanced mathematical algorithms to the raw sensor data" [4].

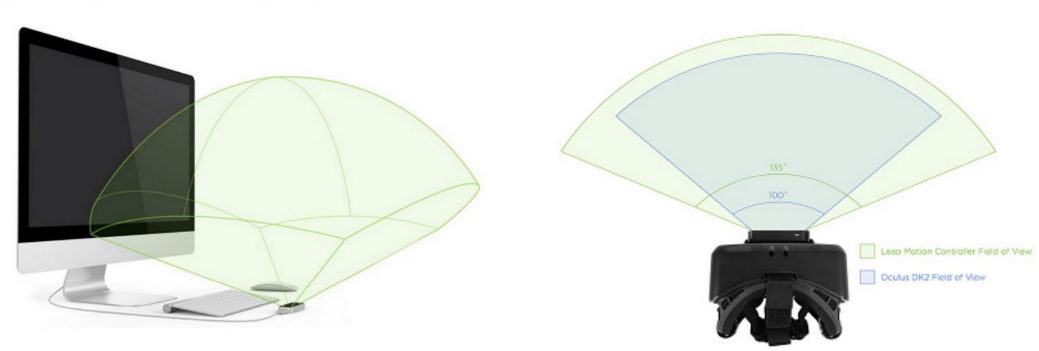


Fig. 2.3: Leap motion field of view

How does it work?

The device records data in a series of frames, where each frame contains positions of detected fingers and hands. The leap motion provides developers prepossessed data through their API (Application Programming Interface), which can be accessed

using frame objects. These frame objects contain several features that distinguish gestures from each other as illustrated in figures 2.4 & 2.5 and which include:

- Palm position P_{pos}, velocity P_v and direction.
- Hand direction P_D.
- Finger position F_{pos}, direction F_D and velocity F_V.

(a) Palm position and direction.

(b) Fingers position and direction.

Fig. 2.4: Hand features.

- Metacarpal position: this is the bone inside the hand connecting the finger to the wrist (except the thumb)
- Proximal phalanges position: is the bone at the base of the finger, connected to the palm
- Distal phalanges position: is the terminal bone at the end of the finger
- Intermediate phalanges position: the middle bone of the finger, between the tip and the base

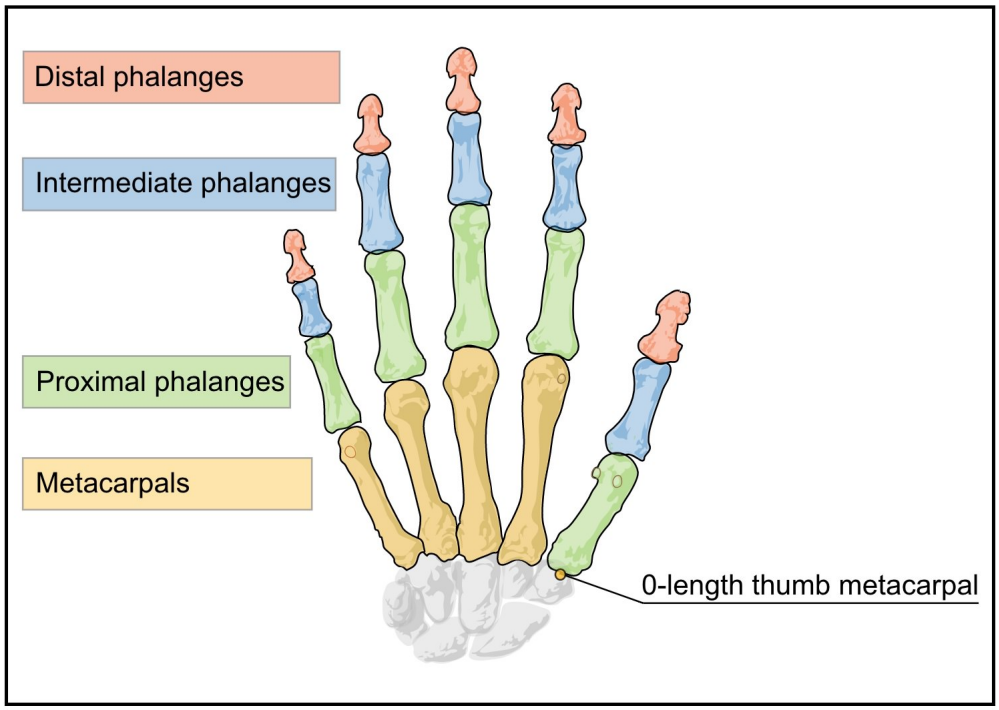


Fig. 2.5: Bones in a hand

" The leap motion service processes the images captured by the cameras while compensating for background objects, and the light environment to reconstruct a 3D representation of what the device is seeing. The tracking layer then matches the data and extracts tracking information of objects i.e fingers or tools, while the system tracking algorithms interpret the 3D data and infer positions of occluded objects. Some filtering techniques are used to achieve smooth temporal coherence, then the service feeds back the results, i.e a series of frames with all the tracking data, to a transport protocol. Where the service talks to the leap motion control panel as well as native & web client libraries through a local socket connection. A client library organizes this data into an object oriented API structure, manages frame history, providing helper functions and classes. The application logic then ties into the leap motion controller input, enabling the motion controlled interactive experience " [4].

The API also provides basic predefined gestures as shown in figure 2.6 to be used by developers, however they are not sufficient for the goals of this project and we are not provided baked in tools for creating our own gestures.

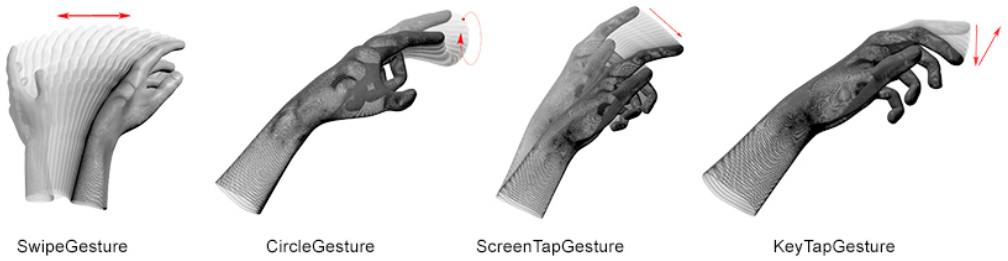


Fig. 2.6: Predefined Gestures

2.2 State of the Art

Today cultural objects are categorized by physical attributes e.g size, material, form and their descriptions are usually in the form of images and text. Systems built to reference and search these objects use keywords and simple image matching [6]. These kind of systems have been proven to work better with literature but is less suitable for three dimensional objects like the focus of this project cultural assets i.e hand tools which their functional aspects are missing or underrepresented [6].

Gesture recognition is the bulk of the work for this project, hand data is transmitted in real time and the application should be able to recognize them and retrieve the relevant results. A suitable approach is using an algorithm that can calculate and match results on the fly, remaining invisible to the user and without causing delays. As we worked on the project it became clear that the best approach would be to use some sort of machine learning on the data, to be able to train gestures and their many different variations.

2.2.1 Machine Learning

Machine learning is an application of artificial intelligence to provide a system the ability to learn and improve automatically without being explicitly programmed. These applications are provided data and use it to learn for themselves. We researched a few approaches for our system that we will discuss in the following sections, but we ultimately went with the $P algorithm.

Artificial Neural Networks

This is a machine learning algorithm that resembles a human brain in that its composed of a series of simple processing units, like neurons, which have a determined

weight and are interconnected. This network learns if given test data,and calculates specific outputs provided certain inputs were given. This has been used mainly in multi-layer perception neural network (MLP) for mapping the Arabic sign language (ArSL)[10].

Hidden Markov models

A Markov model is composed of a network of states, each connected with a specific weight of probability. The systems based on this model, are characterized by the future state of a node being only dependant on the current state and probability of the states that are linked to it. However, a Hidden Markov model (HMM) is different in that its states are partially obscured [3]. For example in a speech recognition application like Siri we can see the waveform of speech but what is actually being spoken is hidden. Comparatively in a gesture system, we can see the hand motion but the intended gesture is hidden. HMM are widely used in pattern matching algorithms for speech recognition and also type prediction.

Point-Cloud Recognizer ($P)

The $ P recognizer is a 2-D gesture recognizer that was designed for rapid prototyping of gesture based user interfaces giving developers access to fast, simple, and accurate recognition approaches [14].It works by overcoming the complexities of matching user gestures by instead treating them as groups or a cloud of points and evaluating them separately in turn. User gestures can be interpreted in many different ways depending on the properties of its strokes. i.e beginning and end points, direction and order. To give an example:

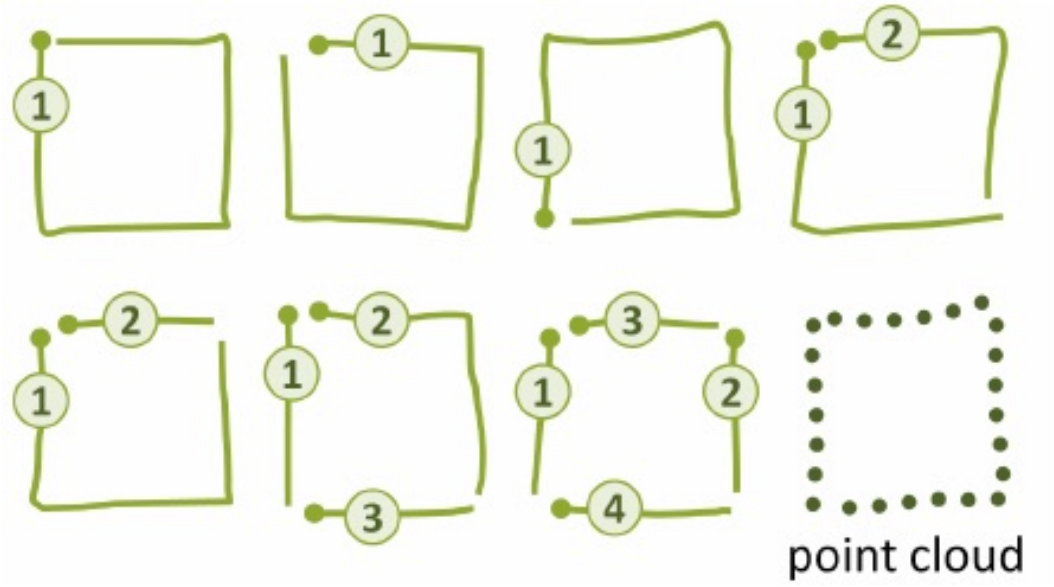


Fig. 2.7: Drawing a Square

Figure 2.7 illustrates the many ways to draw a square, by following strokes 1,2,3, or 4 in different orders and directions (442 scenarios). Vatavu [14] argues that instead we look at the gestures as a cloud of points, hiding away the order, directions, time labels which are not essential for recognizing a gesture and even just complicates the structure of the recognizer rather than helping in classification [1]. Using this algorithm helps remove ambiguity and simplifies comparison and recognition with 99% accuracy [14].

The Leapy application uses the $P algorithm to perform gesture recognition, which is based on machine learning. This algorithm was originally developed for recognition of handwriting gestures (2D). It represents a 2D gesture as a series of evenly spaced points on a flat plane, each point is assigned a stroke ID. For our purposes we need it to represent 3D gestures, so we need to include the z-axis and hand features going beyond just representing 2D which are finger tips equivalent to pen tips if in the context of handwriting.

" The $P algorithm is an instance-based nearest neighbour classifier with an Euclidean scoring function. It selects the appropriate category for an object, given a sample of objects, by calculating the euclidean difference in positions between the objects and finds the closest and most likely match " [14]. These objects can be grouped based on their features into categories. Features are what makes an object unique compared to other objects. These features as shown in figures 2.4 and 2.5 help us add a third dimension to make it a 3D gesture and distinguish gestures apart. We adapted this algorithm for our application and its discussed further in chapter 3.3.1, specifically how its useful to this project.

2.3 Related Work

2.3.1 Interpreting cultural artifacts

This method looks at three levels which include evident features (physical), immediate reasoning, and the deep cultural influential factors[16].

Physical evident attributes artifacts can be analyzed by style/decoration, size, color, material/texture and from these physical characteristics one can discern its function, who is the user and how it is used.

Deeper reasoning gives us context of the tools which requires research and weigh in by experts of the domain. It gives us more information about the artifacts background like symbolic meanings, aesthetics and ergonomics of a tool and the technology being used.

Influential factors give us information to understand the artifact, its S.E.T.I.G (social, economic, technological, geographical) factors that define the historical period from when it was used and also the traditions surrounding it.

Figure 2.8 shows the classification of these levels to help us interpret cultural artifacts.

Name of an Artifact
- background information

		Evident Attributes - Usage - Design	Deeper Reasoning - Hierarchical Rules - Symbolic Meanings - Aesthetics / Ergonomics - Technology	Influential Factors - SETIG Factors - Philosophic Foundation - Traditions
Usage	function			
	user			
	context of use			
Design	style / decoration			
	shape / size			
	color			
	material / texture			
	technology			

Fig. 2.8: Interpreting Cultural Artifacts.

This strategy is to help us categorize the tools properly and be able to create the right gestures that are intuitive to a user to be able to use of for search in our application. Especially with artifacts that are difficult to discern their function, breaking it down and classifying it may help us arrive at its purpose.

For example chisel tools, as shown in table 2.1 that were used to build the great pyramids. Physical attributes are that its made out of copper, the shape of the tip i.e flat top or round top for cutting and flattening surfaces respectively, symbolic significance as it was used to carve hieroglyphs and sacred symbols which was a tradition and how the ancient Egyptians passed on knowledge with this ancient writing technique. Looking at all these factors we get a sense of how the tool was used and can design ways to represent it in the system using gestures for example mimicking with our hands how it was held or used or the shape if its unique.

CHISEL TOOL				
		Evident Attributes	Deeper reasoning	Influential factors
Usage	Function	Carving rocks or wood	Hard material, could dent objects.	Better than chisels made out of rocks
	User	Builders or Carpenters	Required skilled artists who had experience.	Good with their hands.
	Context of use	Building sites	Useful as they were precise cutting tools.	Building pyramids
Design	Style / decoration	Jagged grooves	Easier to hold and handle	
	shape/size	Flat/round tip with varied shapes.	Flat top for cutting, round top for flattening surfaces. Can fit in a man's hand.	Made to be used by people, and designed to be hand held.
	color	Red brown	Natural color of the metal.	
	material/texture	Copper alloy, rough texture	Rough texture to be held securely by builder.	Easily sourced from the ground, natural resource.
	technology	Sharp object	Easier to cut through objects	Held in the hand and struck with wooden mallets.

Tab. 2.1: Interpreting a chisel tool

2.3.2 Similar Applications

To date there hasn't been much research on our specific topic goals, retrieving cultural assets using gestures, but as the technology is the same using a leap motion controller, other projects have successfully implemented this technology leveraging hand motion tracking to read sign language, control thermostat interfaces, controlling car system dashboards, enthusiast projects to replace the mice & keyboard, playing music, and of course games. Some project examples are as follows:

Leap Trainer

This is a web based program built with JavaScript that provides a framework to train and record gestures by Robert O'Leary [11]. It allows developers to train gestures and export data in Json format for use in their applications. It takes advantage of neural networks, cross correlation and geometric matching algorithms.

Leap Learn

This is another web based application that allows a user to define their own custom gestures using a leap motion controller developed by Jesús García [5]. It takes advantage of machine learning to recognize these gestures. This work was useful in creating our own application to train gestures.

Sign Language Converter

This application was developed by Fazlur Khan [7], it converts the American sign language to text. Using a leap motion device it interfaces hand movements and matches it with internal models of the American sign language.

Space Mouse

Human and computer interaction has been an important part of the evolution of technology and so the development of input devices to better interact with systems have had breakthroughs in the last few years. Space mouse is a contact less three dimensional pointing device that replaces the traditional mouse, making use of leap motion tracking technology. Applications for this technology would include

environments where environmental cleanliness is required like medical applications, or driving situations where the user have their hands already occupied and need an easy way to press a button. It can also be fun for kids working with an application like Paint, to draw.

Bartender System

This application was developed by Uppsala University [13] for making orders in a bar where the customers would be able to gesture what they wanted. It would be a different way of communicating requests, allowing the customer to order food and drinks with the automated system and pay for it. Ethical concerns and challenges were evident for this kind of system, and they include:

- Different gestures mean different things to different people with different backgrounds e.g A thumbs up is a positive sign in the west but in Iraq is regarded as an insult [2].

- Selling to minors or intoxicated adults, the system wouldn't be able to make such judgments the way a human being would be better suited.

- Application assumes the user has all their 10 fingers, discriminates against people with disabilities.

3 System

3.1 Requirements

Now that we understand more of the domain we do a requirement analysis to determine those tasks that will determine the needs or conditions to be met for our application to satisfy all the stakeholders involved.

3.1.1 User Requirements

A user for this application is assumed to be a normal user of the system, who wants to search for an artifact in the system, performs gestures with their hands and the system returns matching results. A moderator is a special type of user with more privileges as they are able to train the system and record gestures and specify the artifacts that they correspond to in the system. An Admin is a different type of user who has all privileges to the system and manages the system.

Use Cases

We use a use case to show these relationships within the system and make obvious their functions as shown in figure 3.1. We also present scenarios for some use cases as shown in tables 3.1, 3.2 and 3.3.

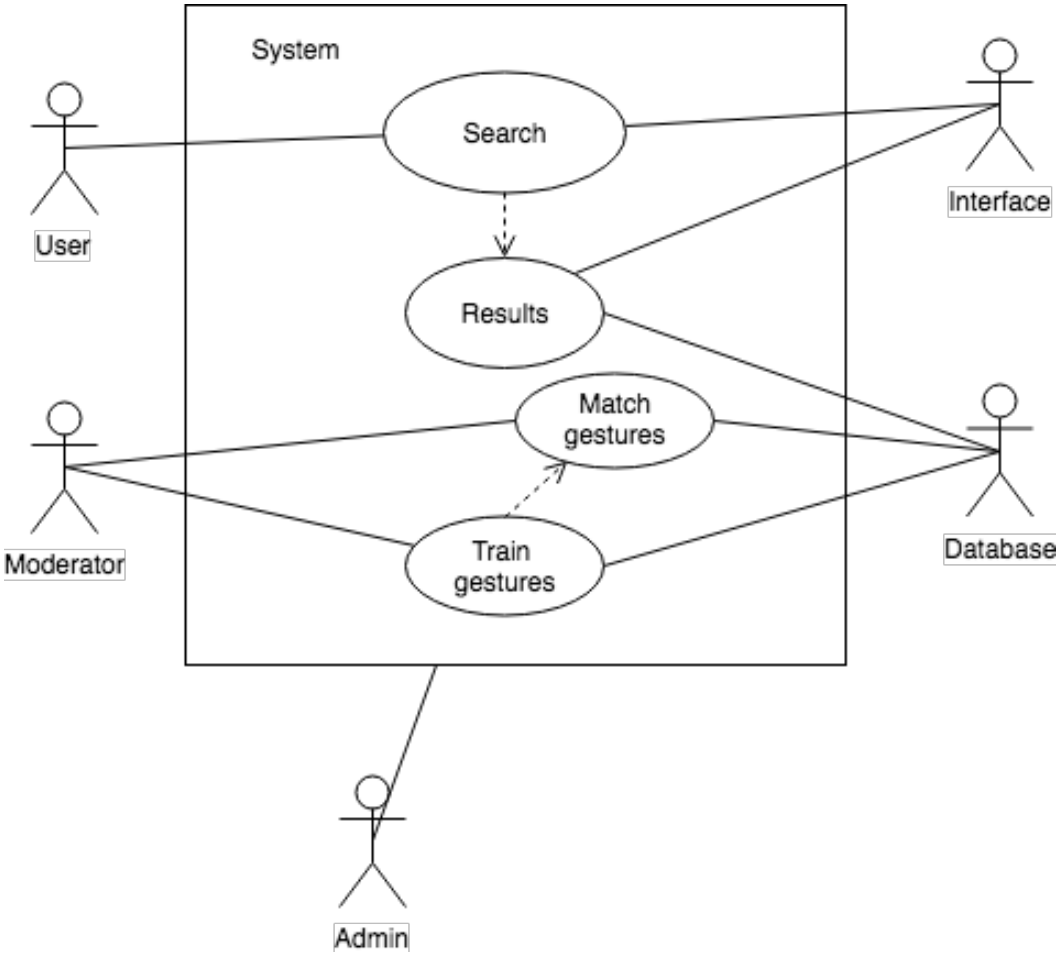


Fig. 3.1: Use case diagram

Use case	Search
Scope	System
Actors	User, Interface
Summary	A user wants to gesture their query with hands and its displayed within the application.
Preconditions	Leap motion device is connected and enabled.
Scenario	1. The user moves hands into the device's field of view. 2. Device recognizes the image data frames 3. The interface displays the image data, and draws the hands. 4. Interface adjusts hand representation position live according to the data. 5. New hand positions are redrawn on interface.
Exceptions	The user moves away from the field of view of the device, hand data isn't available and interface doesn't draw the hands.
Post conditions	The user observes representation of their hands on the interface.

Tab. 3.1: Scenario for the "Search" use case

Use case	Train Gestures
Scope	System
Actors	Moderator, Interface, Database
Summary	A moderator wants to train and record gestures to the application
Preconditions	Leap motion device is connected and enabled.
Scenario	1. The user moves hands into the device's field of view. 2. Device recognizes the image data frames 3. The moderator types the gesture name and presses a record button 4. Moves hands above a velocity threshold and recording is initiated. 5. The moderators hand velocity drops below threshold and recording stops. 6. New gesture is saved in to the database.
Exceptions	The moderator moves away from the field of view of the device, hand data isn't available and recording is stopped and data discarded
Post conditions	The moderator trains some gestures to the application

Tab. 3.2: Scenario for the "Train Gesture" use case

Use case	Match Gestures
Scope	System
Actors	User, Moderator, Interface, Database
Summary	Users want perform gestures and the application should interpret the input
Preconditions	Leap motion device is connected and enabled.
Scenario	1. The user moves hands into the device's field of view. 2. Device recognizes the image data frames 3. The user moves hands above a velocity threshold and recognition is initiated. 4. The users hand velocity drops below threshold and recognition stops. 5. The gesture is compared to known gestures in database and closest match is shown on interface.
Exceptions	The user moves away from the field of view of the device, hand data isn't available and matching is stopped and data discarded
Post conditions	The user observes the system interpret the gestures.

Tab. 3.3: Scenario for the "Match Gesture" use case

Functional Requirements

R1. The system will display the user's hand movements in real time.

R2. A moderator should be able to train and record their own gestures.

R3. The system should match and recognize gestures provided by the user.

R4. The system will return results based on the user gestures, matching them with known gestures stored within the application.

Non-Functional Requirements

R5. The system will be intuitive and simple to use, the interface should be self evident and clear instructions provided.

R6. The system should render the user's hands with latency of 5ms or below.

R7. The matching of gestures should be accurate.

R8. The application should be responsive, prompting the user and giving clear signs of what is going on.

3.1.2 System Requirements

SR1: The product shall be compatible with popular operating systems, including open source alternatives.

SR2: The product shall use appropriate database technologies to store large stores of data and follow best practices towards data management.

SR3: The product shall provide for user authentication and access to the system

SR4: Intrusion detection should be available within a reasonable time frame.

SR5: Periodic back ups of the system and data integrity should be ensured.

SR6: Recognition of gestures should be accurate to at least 80% to be sufficient.

3.2 Design

3.2.1 Interface

A lot of consideration was given to the user interface design of this application to be both functional and simplistic. The leap motion guidelines give suggestions on how to design interfaces to be used with the device and they are geared for a gesture based experience, with large buttons and interface elements. Since the device isn't as accurate as a keyboard and mouse, we wanted an interface that's a mix of both these human-machine interaction input methods. When a user wants to search an artifact, they can gesture their query with the leap motion controller and get a result without any other input devices, when they want to train and record gestures then we make available the use of the keyboard and mouse to perform these advanced tasks.

The system will be a a web based application with a simplistic design, intuitive enough for a user to interact with. The intent is that intuitive gestures should map onto the application without sending users to menus, through help screens or having to read system manuals. In our interface design we give a large interaction area for the drawing of the hands, since its a big part of the application and with the user

requirements we have, this data should be available and rendered in real time. On the left area we have a list of gestures already programmed into the system, and a way to create a gesture and record it. On the right area we have an informational panel that gives us context as to whats happening on the main interaction gesture area at the centre. A wire frame of the application is as shown in figure 3.2.

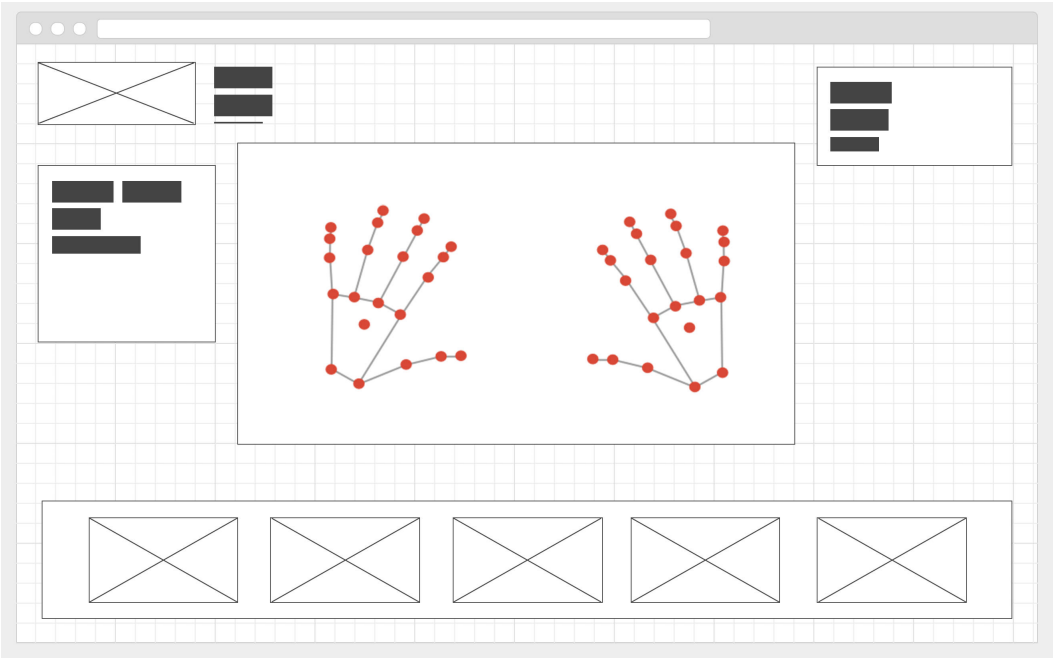


Fig. 3.2: Wire frame diagram

Just below the interaction area we have an image carousel of the tools available in the database, when a user searches and successfully matches a gesture to known data in the system, the correct result which is an artifact will be shown to the user as shown in figure 3.3.

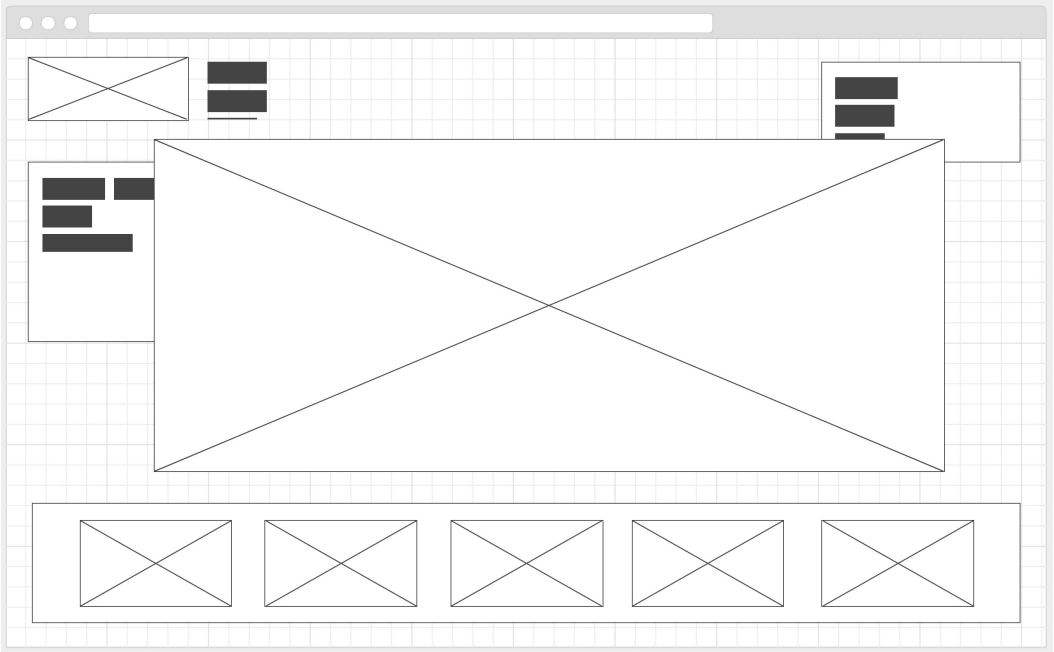


Fig. 3.3: Search result diagram

3.2.2 Architecture

For our application we design a system architecture to break down the system into components and show how they work together. The main components of this system are:

- **Gesture Manager**: This component handles all the processes and services involved in using gestures from training and recording to recognition and matching known gestures.

- **Content Manager**: This component deals with search and retrieving results from the database, as well as handling all the artifact information.

- **Presentation Manager**: This component handles the interface and communication between the system and user, its presents all the information necessary for a user to complete their tasks.

- **Data Manager**: This component serves as a DBMS and handles all the data from the application and appropriately storing it in the database.

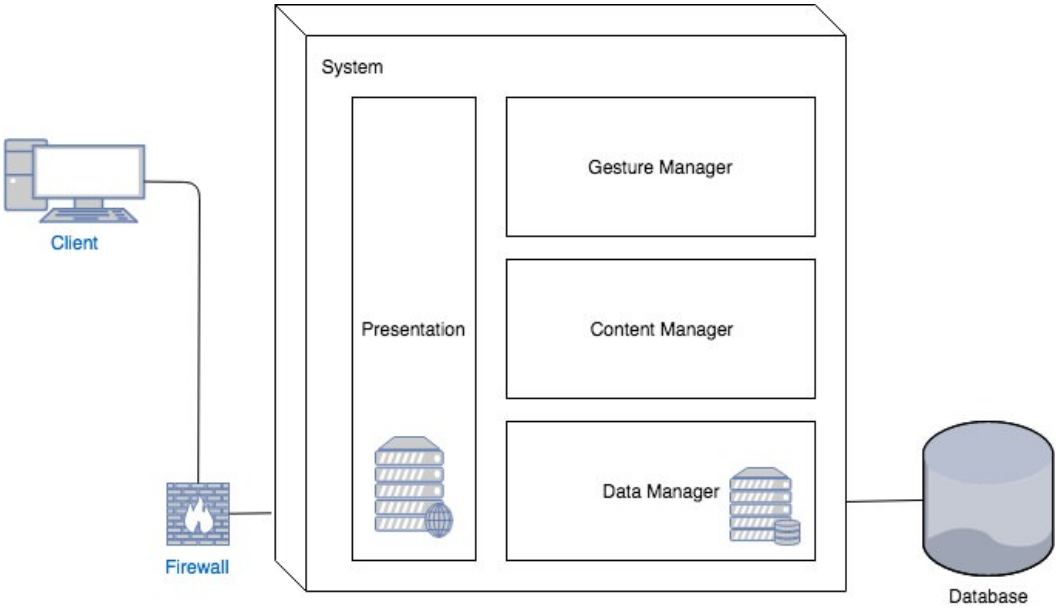


Fig. 3.4: System Architecture

Sequence Diagrams

For us to get a better understanding of the system, we use sequence diagrams to show the interaction logic between different objects encapsulated in the gesture manager in the time order that these events take place.

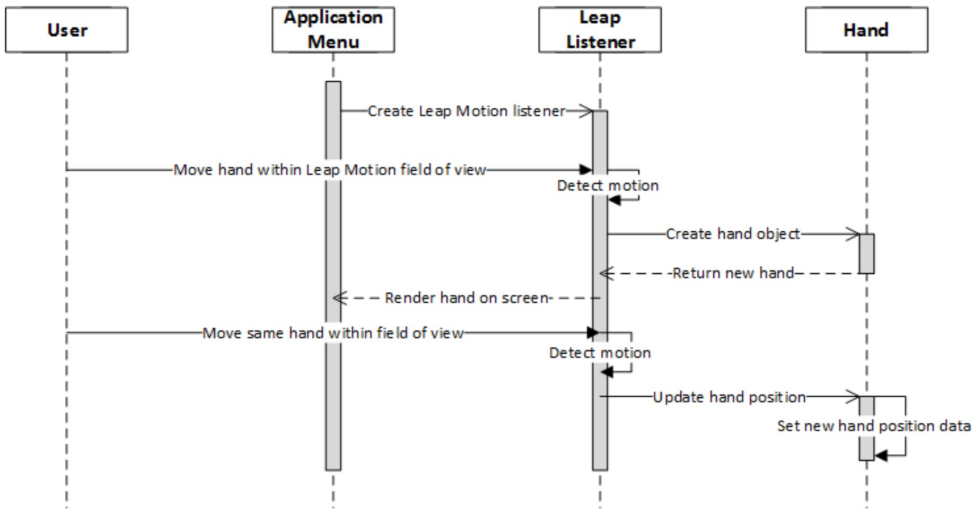


Fig. 3.5: Display a hand

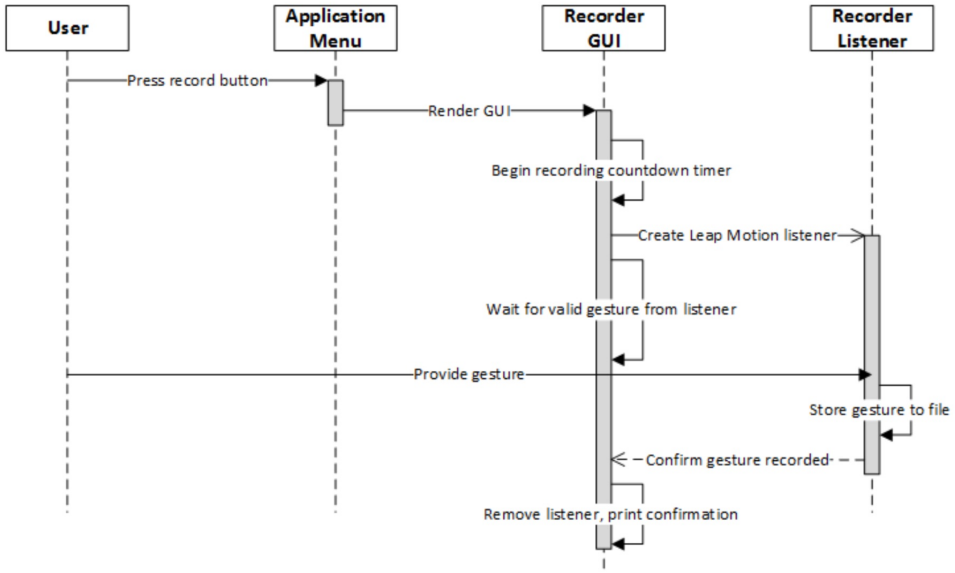


Fig. 3.6: Recording a Gesture

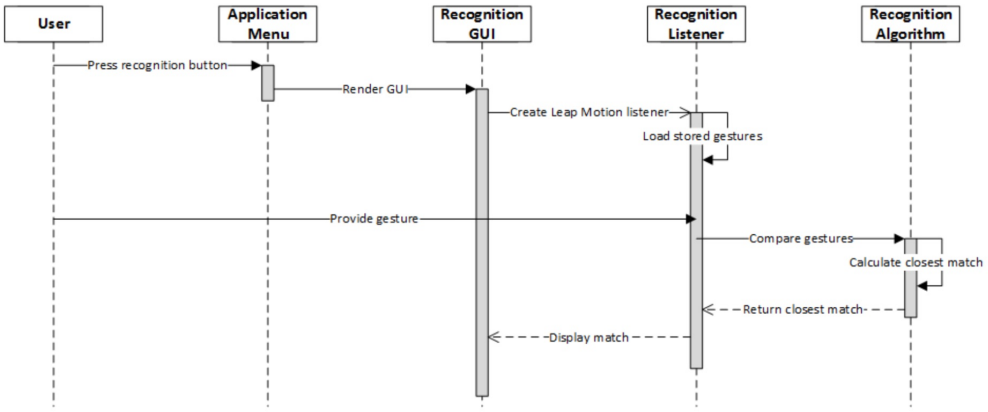


Fig. 3.7: Recognition/Matching of a Gesture

3.3 Integration

As we have seen earlier in chapter 2.1, the leap motion controller records tracking data using frames. Each frame object containing lists of tracked entities. These frames are made available to an application by using call back methods to event listeners. The application uses JavaScript and the we use listener and controller classes to access the leap motion controller. The controller class represents the device and the listener class defines call back methods to be ran depending on the device's status.

For gesture recognition, the application checks that the new frame data has the hand velocity above a certain threshold, and the listener will recognize the gesture anything below this and it stops. Also there is accommodation of poses of hands in the application, but first the user has to move their hands in the frame to start the recognition and then perform the pose gesture.

3.3.1 Using the $P Algorithm

When we began working on this project one of the challenges we faced was finding a way to recognize and program gestures to the system. The leap motion API documentation gave us the basic information of how to access the data and it came out in strings of numbers and point values for the x, y, & z axes. We could simply program the system to look for specific data points for example (x, y, z) = [100, 100, -100] and use these coordinates to tell the program if it reads these then do the following. But what if the user put their hands in a different way or had shorter hands then the coordinates would be different but the intent of the user would have been to perform the same gesture.

With the $P algorithm, we determine gestures by the summation of the euclidean distance between each of the feature points. When an unknown gesture is detected, the feature point distances are calculated against other known gesture points, until the closest points are determined. The gesture with the smallest overall point distance is declared to be the closest match to the unknown gesture. The system gets better over time and more accurate as we supply it new data. For example a swipe gesture with feature points (X, Y, Z). If we were to perform the same gesture we would provide the system with feature points closer to (X, Y, Z) than any other known gesture. If a different person were to perform a gesture with feature points (X-1, Y+1, Z) the gesture would be closer in point distances to (X,Y,Z) than any other gesture and so the system learns and is designed to give a percentage match score of how related they are. The more data we give the system the more accuracy of recognition, it improves greatly and over time its a better system.

3.3.2 Normalization

Gestures are normalized before they are matched, or checked against known gestures. Different people have different sized hands, can perform the same gesture in different ways, and may do these gestures at different motion speeds. So some variation is expected and its by normalizing this data that we ensure the results are accurate and the system has acceptable performance. To solve some of these issues we take on different approaches to mitigate the variations. Some people will perform some

gestures not always in the same position and so we translate the feature values always to a known origin, specifically (0,0,0). In the case that everyone has different hand sizes we scale the values to fit a specific range between [0-1]. It is also helpful to represent each gesture as a specific number of points, in the case where different users perform the same gesture differently and lets say lasts different intervals or lets say take more frames than others. Normalizing the data would help and improve the performance of the system.

3.3.3 Gesture Based Query

(a) Pliers **(b)** Scissors

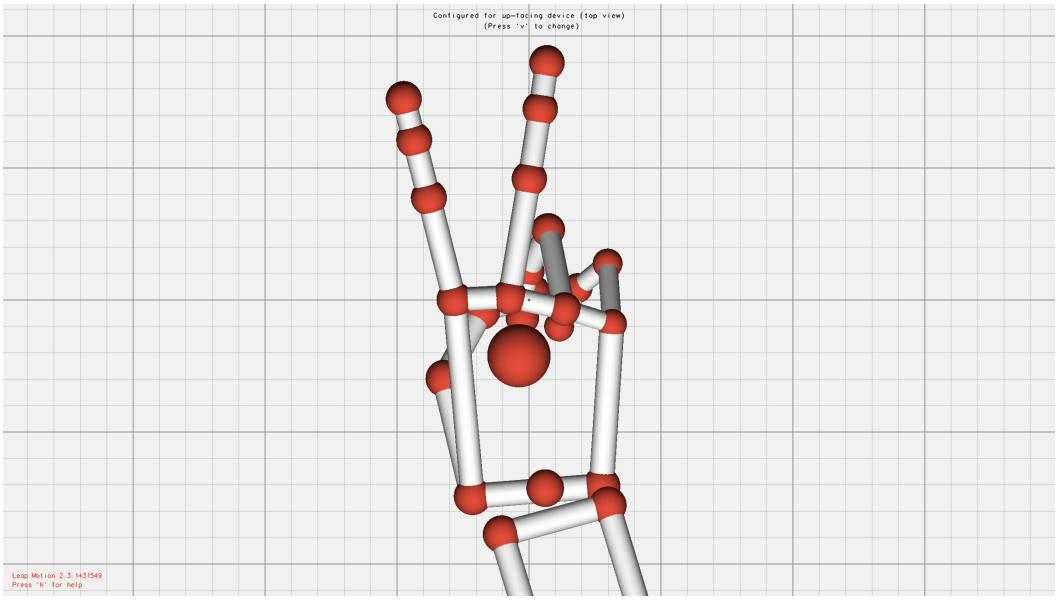
Fig. 3.8: Examples cultural artifacts used in this project

We mentioned earlier in chapter 2.3.1 about interpreting cultural artifacts so that we could be able to create intuitive gestures for them, we adapt this approach by Xiang [16] for our own application. This proposed approach of categorizing them into these different factors i.e evident physical features, immediate reasoning and deep cultural influential factors allow us to understand the artifacts better and classify them. So now we need to fit them with the right gestures in our system. The entire project is premised on users being able to use gestures to query a system. What are these gestures? And how do we create intuitive gestures that accurately depict our intent. Sign language has been around for quite some time, its a natural language with its own grammar and lexicon using hand movements and facial expressions to convey meaning. We draw inspiration from this as its already available and we can infer that gestures can communicate a world of information.

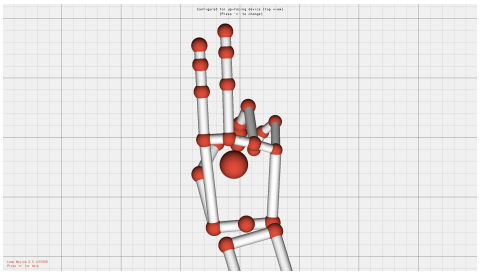
Gestures are categorized into either static or dynamic gestures and whats important for our system are their characteristics like the shape, finger flex angles, hand trajectory, scale and orientation. We can even use both or a mix of static and dynamic gestures one for each hand to create one gesture. With some tools we would need to mimic how the tool in operated in hand to represent the gesture associated with it, while others its gets a little more complicated and more creative ways need to be considered.

For example:

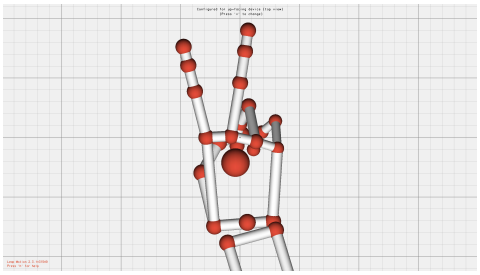
A scissor tool can be represented with a gesture as shown in figure 3.9. It would involve a motion going from (a) - (c) in succession repeatedly, representing a cutting motion.



(a)



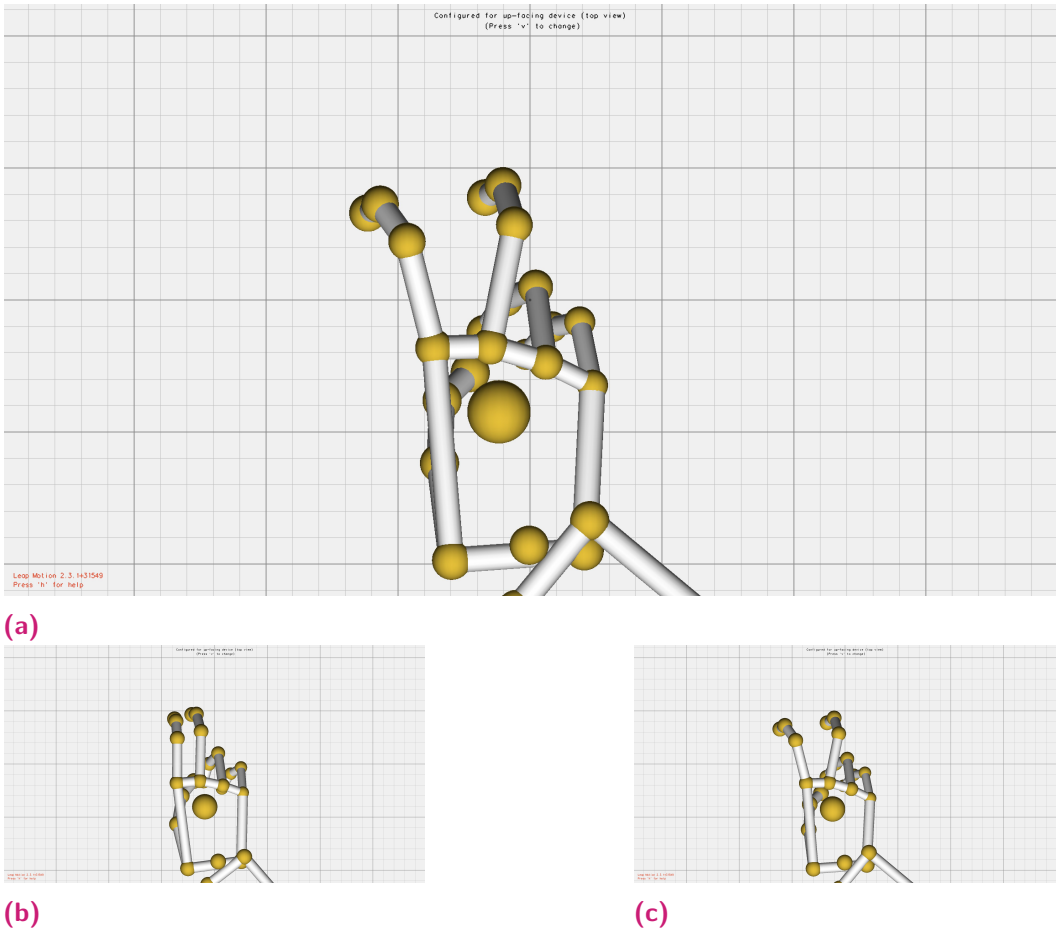
(b)



(c)

Fig. 3.9: Scissor gesture

But what happens when you have tools that have the appearance of similar functionality, as the tools in figure 3.8. A pliers and a scissor both cut objects with similar mechanics, how do you distinguish them from each other? A possible solution would be to gesture as shown in figure 3.10, by shortening the two fingers to show its a shorter cutter.



(a)

(b)

(c)

Fig. 3.10: Pliers gesture

Another solution would be to add another dimension and use a second hand to create a gesture. Use one hand for a dynamic gesture in motion and a second hand in a static i.e pose gesture. It can be useful to illustrate a tool with the object it operates on, for example to distinguish between sheers that cuts wool and scissors that cuts paper. These similar tools can even be categorized in levels of cutting power and we use the second hand to illustrate that, one finger for level 1, two fingers for level 2 and so on.

Static Gestures

Static gestures also referred to as poses are mainly gestures which we take position and direction as the main factors [12]. They can be calculated based on the palm and fingers relative distances. These include:

- Distance between finger tips F_{pos} and palm center P_{pos} denoted by D_i.

- Distance between two adjacent fingers.

Examples of static gestures are as shown in figure 3.11:

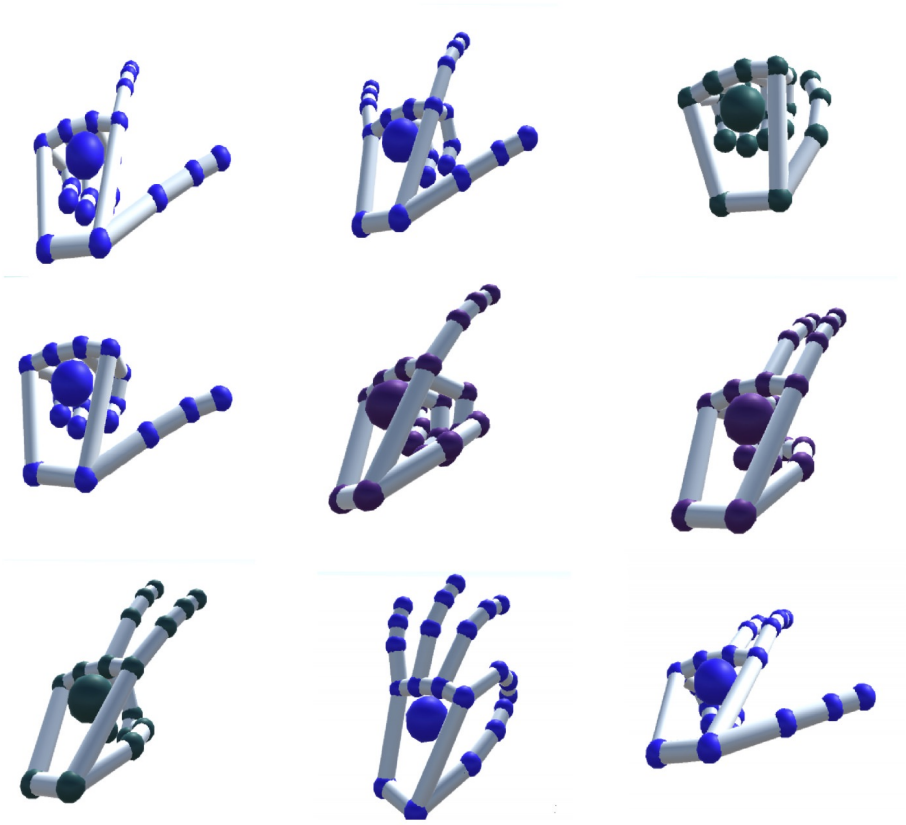


Fig. 3.11: Static Gestures

Dynamic Gestures

These gestures are easily distinguishable from static gestures as they involve motion. The total velocity magnitude between the fingers and palm is calculated, and if its greater than a specified threshold then the hand is deemed to be moving. A lot of factors are considered in the calculations from hand translation movement, hand rotation and finger movement.

3.4 Implementation

The Leapy application is web based and runs on any modern web browser. The design is minimal and designed to be easy to work with and intuitive. In this section

we will cover some of the important tasks a user performs with the Leapy application and preview what the application is capable of.

3.4.1 Launch

When you launch the application the screen as shown in figure 3.12, is presented to the user. The key areas have been highlighted with numbers in red color to help us distinguish the key feature areas of this application. As we mentioned earlier in chapter 3.2 when we talked about design, we want a mix of input methods, we therefore use keyboard, mouse and motion tracking to interact with the application.

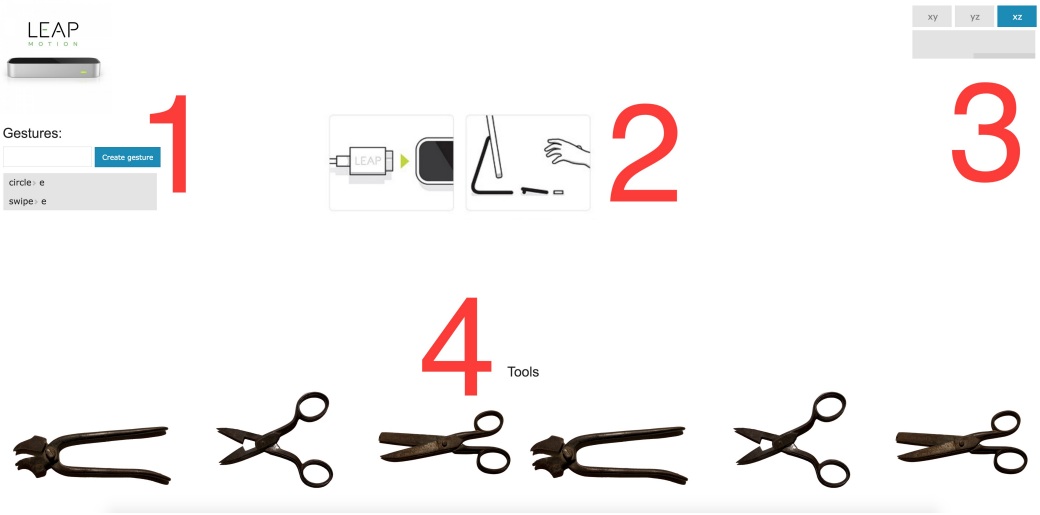


Fig. 3.12: Leapy Interface

- **Section 1**: This area is reserved for a user to create gestures, and they do so by typing a name of the gesture and then pressing the create button. It also shows already created gestures in a list, in the diagram we can see the circle and swipe gestures indicated.

- **Section 2**: This area serves as the centre of the application where the hands are drawn when a user puts their hands in the device's field of view. When the users hands aren't in view, the default image place holder is seen and this also serves as a flag to help the user know their hands aren't being tracked. The image placeholder is also an instruction tip, so the user knows to always plug in the leap motion controller before using the application.

- **Section 3**: This is the informational panel that represents which of the axes is being read by the algorithms and also provides messages to the user on whether there is recognition or matching happening in the application.

- **Section 4**: This area is populated by images of the tools i.e artifacts available in the system, depending on the gestures provided by the user, the results will be one of these tools.

3.4.2 Search

When a user puts his hands over the leap motion, the hands are drawn on the interface and when the user performs a gesture the application reads the gesture as shown on the top right corner of figure 3.13.

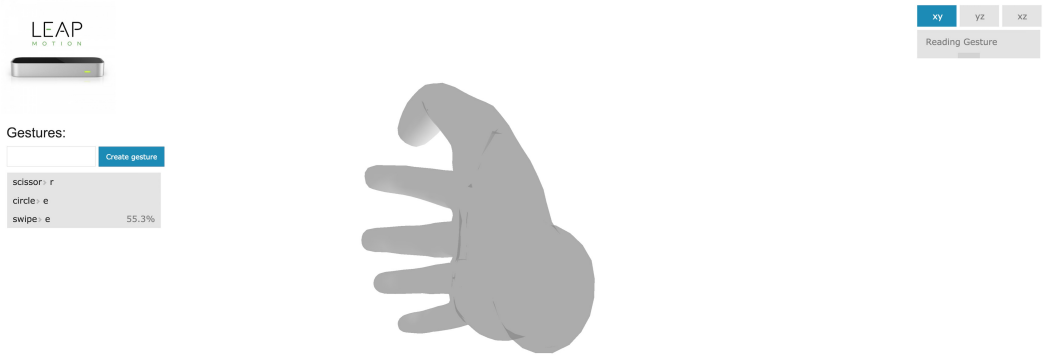


Fig. 3.13: Search Interface

Depending on the gesture being attempted you can also see the match score in percentage as shown in section 1 on the left of the interface. Its 55.3% for gesture swipe in the figure 3.13. When a gesture is recognized and matched, the corresponding gesture in the list turns green and the percentage match is indicated as shown in the figure 3.14.

Fig. 3.14: Successful search

This winning gesture that was being compared against internal known gestures after it hits the score threshold, the system gives back relevant results of the tool being searched for. The user is prompted with a result as shown in figure 3.15.

Fig. 3.15: Search result

3.4.3 Train Gestures

The application can also train and record user gestures. On the left area in section 1 a user is able to type in the name of the gesture they want to create, then they would put their hands in the device's field of view. If the device is picking up hand data the users hands would be drawn, at this point they press the create button and

a countdown animation appears over the users hands as shown in figure 3.16. The user would repeat the gesture over and over until the countdown runs out to get as many variations of the gesture as possible and then when they stop moving their hands the gesture is saved and a prompt appears confirming this. The new gesture is added to the list in section 1 of already saved gestures.

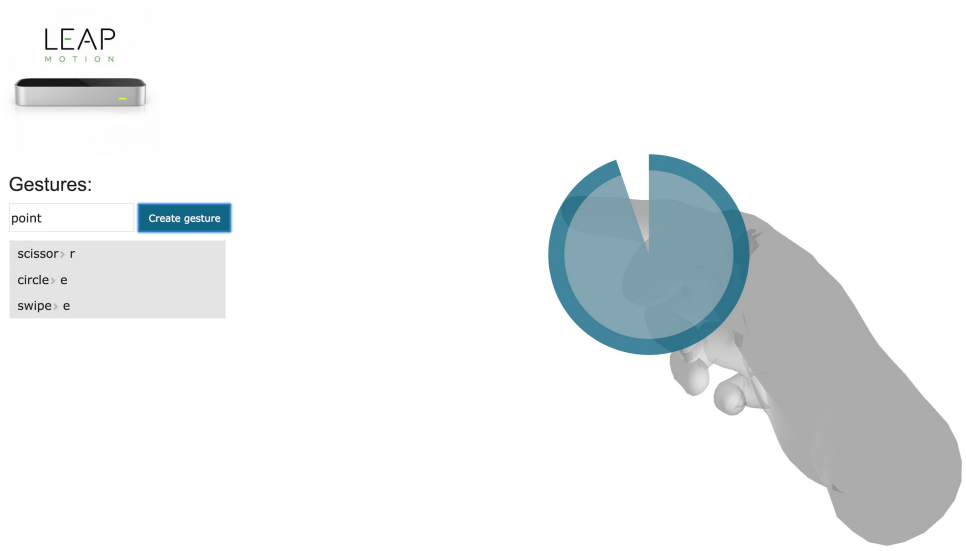


Fig. 3.16: Training a gesture

If a user wishes to delete a saved gesture all they have to do is click on it in the list and it will be deleted, and they can retrain a new gesture following the steps outlined earlier.

4 Evaluation

4.1 Device Performance

To properly evaluate the system, we have to look at the performance of the leap motion controller device in different scenarios which ultimately determines our results. Capturing and recognizing gestures in the system is a challenge as the device suffers frequent detection problems. This testing was done using the Leap motion built in visualizer that is used for troubleshooting purposes.

In the beginning we wanted to record gestures representing them in hand with the tools in operation. We quickly discovered this wasn't feasible as the device wasn't able to properly or reliably detect our hands. In the V2 software settings there is an option for tool tracking, but its only limited to simple tools that resemble pencils i.e a straight small object. So we needed to create custom gestures that can work with the device using just our hands, as examples can be seen in chapter 3.3.3.

The device recognizes hand motion using a best guess system, using what it can see with its cameras and an algorithm inferring to its internal hand models to determine what the hand and finger positions are. This makes it likely to not recognize or misread gestures. When a user's hands are pressed together, or their fingers are inter weaved together, or the hands are placed over each other the device won't detect this poses. When a hand is on top of another, the device is blind to any movement of the upper hand as its obstructed from view by the lower hand. So the software fails to recognize any gestures made. Sometimes even misinterprets this gestures in the case of the hands and fingers being close together.

Users will also encounter issues with when they perform a gesture how they place their hands over the device affects the recognition, sometimes rotating the hands will give the device better view and aid recognition.

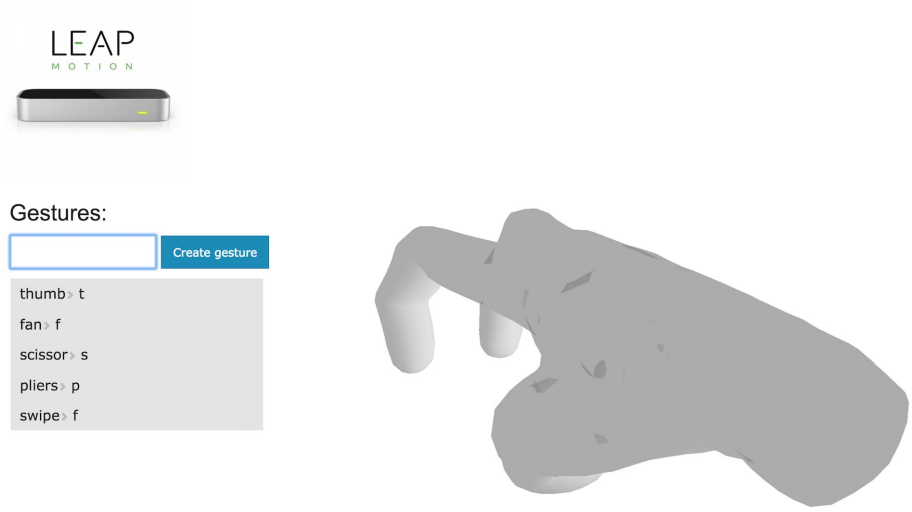
In the application some gestures have close similarities as can be seen in figures 3.9 and 3.10. A new user who isn't familiar with the system is very unlikely to perform correct gestures and get a correct match unless they exaggerate the gesture. This limits our possibilities, as to properly differentiate the gestures we have to

create gestures that are quite different as similarity is a bottleneck and will result in mismatches.

Lighting conditions also affect the performance of the device, the software does detect external infrared light and compensates for this but more often than not this can reduce the tracking capabilities of the device. Best tips for the environment is to avoid direct sunlight, or any other direct light source as this causes interference because the device sees the over exposed parts of the hands. The device can see in the dark just fine, and a lot of background environmental factors have been accounted for in the leap motion software like Light, darkness, reflective walls, monitors, desktops, coffee cups, posters e.t.c.

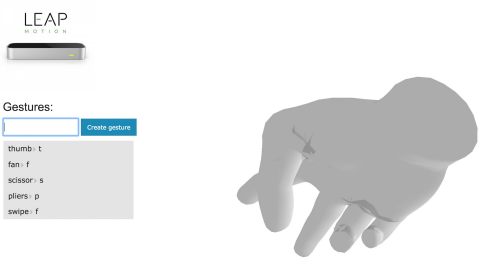
4.2 Recognition Performance

A core feature in this application is the recognition of gestures, and its important that the system can distinguish between them as accurately as possible. Testing was done with five gestures as shown in figure 4.1:

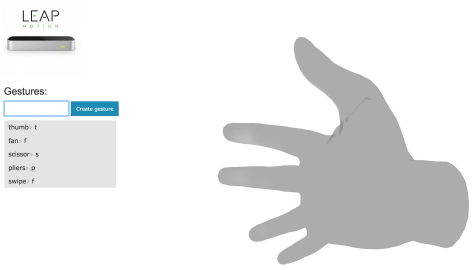


(a) Pliers gesture

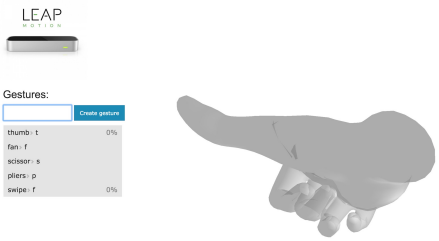
(b) Grip gesture



(c) Scissor gesture



(d) Swipe gesture



(e) Thumb

Fig. 4.1: Test gestures

The Leapy system was cleared of all data and the gestures were trained into the system. The user attempted all the gestures in consecutive order, making attempts at each gesture and recording the matching scores of recognition given by the system. The data was as shown in table 4.1:

Gesture	Expt: 1	Expt: 2	Expt: 3	Expt: 4	Expt: 5
Grip	26.5%	0	0	0	0
Thumb	0	73.7%	74.1%	56.5%	52.7%
Pliers	0	0	0	0	0
Scissor	0	0	0	69.1%	0
Swipe	0	42.9%	38.9%	25.2%	85.1%

Tab. 4.1: Match scores

In experiment 1 the grip gesture was attempted and the system gave back a score of 26.5% for the grip gesture, although this is a low score its also not a negative result as no other gestures were triggered, as all of them gained a score of 0, and the system recognizes this gesture as the closest match to the known gestures in the system.

In experiment 2 the thumb gesture was attempted and in this case the system recognized this gesture to be either the thumb or swipe gesture. But it gave a higher score to the thumb gesture or 73.7% which was the intended gesture by the user. So the winning gesture will be the thumb gesture and this shows that the system is working as expected.

In experiment 3 the Pliers gesture was attempted, but the system didn't recognize this gesture as it gave a result score of 0. Instead we see the thumb and swipe gestures get scores of 74.1% and 38.9% respectively. This is unexpected behavior from the system, as the intended gesture was not recognized. We investigated this scenario by running the same gesture on the device's built in visualizer which is provided for troubleshooting purposes, and we got interesting results as shown in figure 4.2.

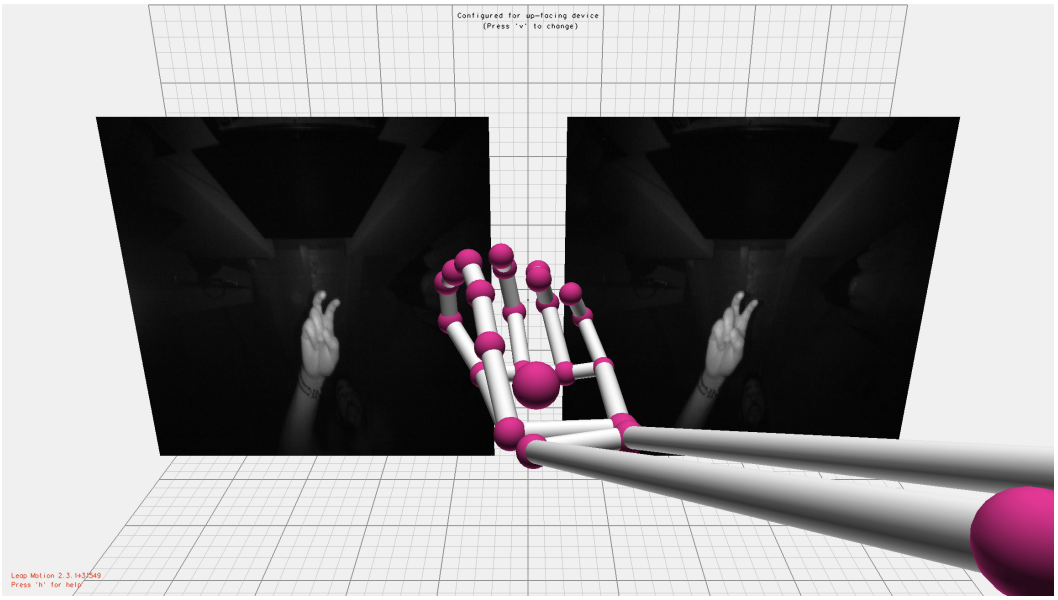


Fig. 4.2: Pliers gesture

From the diagram you can see two views, the 3D hand in pink is what the leap motion has drawn from what it perceives to be seeing. The two rectangle frames with hands in the back is what the actual two cameras as shown in figure 2.2 on top of the device are seeing in real time. You can see clearly that the cameras show the user performing the pliers gesture but the device draws them not as they are but differently showing the hand inverted with palm up and doing some kind of pinch pose. To be fair the device did represent the hands the way they were intended a few times, but also got it wrong other times as shown in figure 4.2. We are left with questions was it the device that caused failures in experiment 3, was it the algorithm or was it a mix of the two that led to these results. A possible solution would be to retrain the gesture, or find a more suitable gesture that is distinguishable. In this experiment we would need to investigate further to establish the causes, and in chapter 5.1 dealing with future work we talk about making further iterations to the system and refining these gestures.

In experiment 4 the scissor gesture was attempted and the system recognized three possible matches that could potentially be the intended gesture i.e thumb, scissor and swipe gestures. Each was given a score and the winning gesture was the scissor with a match score of 69.1%. The system worked optimally in this situation, giving the correct result that the user intended.

In experiment 5 the swipe gesture was attempted and the system recognized two possible matches namely the thumb and swipe gestures. The swipe gesture won with a score of 85.1%, the highest score recorded in these experiments. This also

shows that the more distinguishable and different a gesture is, the better it is for the system to recognize it.

Time wasn't considered as the application returns results between 0.20ms and 0.50ms and is consistent and well below 5ms as specified in chapter 3.1.1 where we discussed requirements of the system. Training of gestures also is constant, it takes the same amount of time to record any gesture. Using this sample we can conclude that the system has 4 of 5 success rate, around 80% and this is a positive outcome for the system.

5 Conclusion

When we began this project we laid out specific goals in chapter 1.2 which guided how the Leapy application was designed and developed and looking back at what we set out to achieve we can determine the success of this project. The goals were:

- Analyzing a collection of hand operated tools to identify their attributes not just physical but also their cultural and historical significance.
- Investigate models to represent these tools based on how they are operated and represent them using hand gestures.
- Train and recognize these gestures using a leap motion controller.
- Querying a database and matching these gestures with digitized representations of these tools.

In the first goal we were successful in analyzing the collection of hand operated tools, identifying their attributes as can be seen in chapter 2.3.1. We adapted this method by Xiangyang [16] of interpreting cultural assets by their factors which include physical, immediate reasoning and cultural influential factors. It helps us classify artifacts, discovering their function and as a consequence determine which gestures are appropriate for these tools.

In the second goal when it came to investigating models to represent the tools, it was challenging as it was not enough to just represent them based on how they were operated in certain situations. As discussed in chapter 3.3.3, for some tools its appropriate to represent them as they are used as the gesture is unique and distinguishable and the system can recognize it easily as seen in the results of experiment 1 in chapter 4.2. In some scenarios the tools operations are similar e.g scissors for cutting paper and shears for cutting wool and so we need to add another dimension to differentiate them and as proposed in chapter 3.3.3 using a mix of both static and dynamic gestures. We can also see from the experiments in chapter 4.2 tools with similar gestures have a hard time being recognized or even misread in some cases.

The third goal was to train and recognize these gestures using a leap motion controller. The training of gestures was successful, as the system allowed a user to train their own gestures and name them, the time taken was well within allowed limits. The application performed as expected as long as the user followed the criteria of training a gesture i.e making sure the device was plugged in, keeping their hands in the field of view of the device, moving their hands and repeating the desired gesture and pressing the save button when done. Recognizing of gestures is the other part of this goal and its success had mixed results that were influenced by other factors as discussed in chapter 4.1 and chapter 3.3.3. If a gesture was accommodating to the limitations of the device and also very distinguishable the system would recognize it easily. Users have to be weary of the degree of similarity of their gestures and can implement methods laid out in chapter 3.3.3 to improve the reliability of the systems recognition.

The fourth goal was to query a database and match these gestures with digitized representations of these tools. Assuming the recognition of gestures was successful and all factors outlined in the third goal were optimal, the system would give back a result of a hand tool that the gesture is best matched against. This goal was partly achieved as the database is yet to be implemented for this system. The tools are stored as image assets and retrieved by simple conditional statements. Further work for the future is discussed in chapter 5.1.

5.1 Future Work

There are many areas to improve the application and some of the ideas will be discussed in this section. There will be split into two parts covering the hardware and software parts of the project.

Hardware

This application uses the leap motion controller device as the primary input source and as accurate as it is in tracking hand motion it has its challenges as discussed in chapter 4.1. One big gap it exhibits is its data inference capability, its not able to read hand movement in conditions where hands are stacked over each other, or fingers are interlaced and sometimes fails to calculate the right positions of hand rotations and finger positions. A possible solution for this problems would be to use multiple leap motion devices as illustrated in figure 5.1.

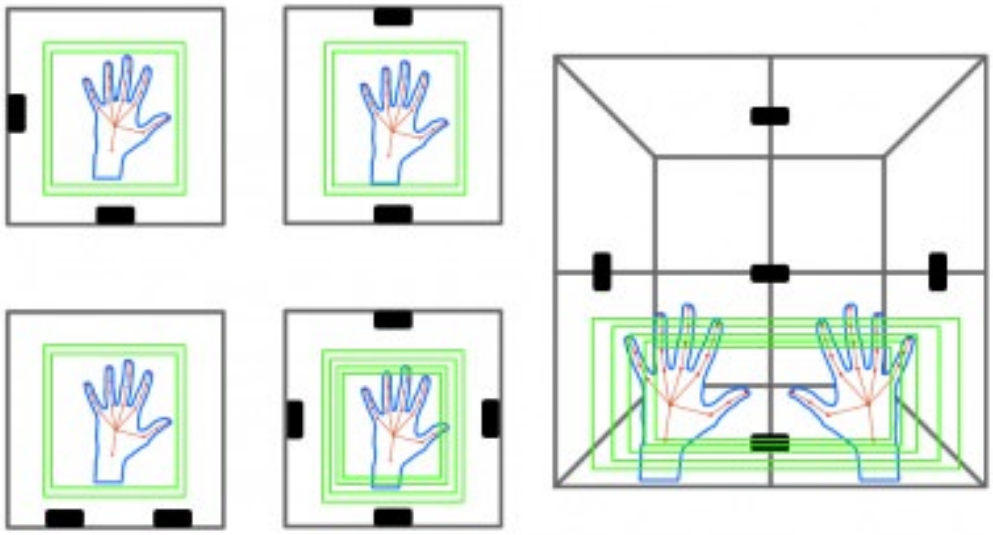


Fig. 5.1: Multiple devices

There would be this interaction box area that would allow the leap motion devices to see a users hands from different angles and positions, making it very unlikely that the users hand movements are obscured or missed. However, this also increases complexity of the application as we have to synchronize the data of multiple devices and represent it as one image frame.

Updates to the device software would be beneficial in increasing tracking capabilities, improvements to the API and SDK to allow developers better access and flexibility with the data.

Software

As mentioned in chapter 5, the database and a database manager for this application would be developed further allowing for a robust system where the logic and data are separated, and can be worked on independently. Artifacts could also be represented using 3D objects and not just as images. The system could adopt a similarity measure for the tools, so that when one is queried, tools are ranked and there are close recommendations that are given back as results.

The recognition of gestures could also be improved by modifying the $P algorithm further and also looking at other algorithms and machine learning techniques which are available. Improvements to speed and performance would also greatly increase

the reliability of this application. The system can also be tested with users to put the gestures into practise and iterate over numerous versions and improvements discovering new gestures that come more naturally to a normal user and that work well with the system and hardware available.

5.2 Final thoughts

Overall I was satisfied with the project, it met most of the project goals and the implementation stands out as more of a proof of concept in this new domain of gesture based querying of cultural assets. The project developed further my technical and management skills, helping to also put in practice all the experience and knowledge acquired during my university training.

List of Figures

1.1 Egyptians traveled far to mine copper for the tools to cut the stones for the pyramids. . . . 1

2.1 Leap motion device . . . 5
2.2 Top PCB of a leap motion device . . . 6
2.3 Leap motion field of view . . . 6
2.4 Hand features. . . . 7
2.5 Bones in a hand . . . 8
2.6 Predefined Gestures . . . 9
2.7 Drawing a Square . . . 11
2.8 Interpreting Cultural Artifacts. . . . 12

3.1 Use case diagram . . . 18
3.2 Wire frame diagram . . . 22
3.3 Search result diagram . . . 23
3.4 System Architecture . . . 24
3.5 Display a hand . . . 24
3.6 Recording a Gesture . . . 25
3.7 Recognition/Matching of a Gesture . . . 25
3.8 Examples cultural artifacts used in this project . . . 27
3.9 Scissor gesture . . . 28
3.10 Pliers gesture . . . 29
3.11 Static Gestures . . . 30
3.12 Leapy Interface . . . 31
3.13 Search Interface . . . 32
3.14 Successful search . . . 33
3.15 Search result . . . 33
3.16 Training a gesture . . . 34

4.1 Test gestures . . . 37
4.2 Pliers gesture . . . 39

5.1 Multiple devices . . . 43

Bibliography

[1] Lisa Anthony and Jacob O Wobbrock. „A lightweight multistroke recognizer for user interface prototypes“. In: *Proceedings of Graphics Interface 2010*. Canadian Information Processing Society. 2010, pp. 245–252 (cit. on p. 11).

[2] Roger E Axtell and Mike Fornwald. „Gestures: The do's and taboos of body language around the world“. In: (1998) (cit. on p. 16).

[3] Mingyu Chen. „Universal motion-based control and motion recognition“. PhD thesis. Georgia Institute of Technology, 2013 (cit. on p. 10).

[4] Alex Colgan. „How does the leap motion controller work?“ In: *Leap Motion Blog* 9 (2014) (cit. on pp. 6, 8).

[5] Jesús García González, Andrea Bellucci, and Ignacio Aedo Cuevas. „Definición de gestos 3D por usuario final: una experiencia práctica con Leap Motion“. In: () (cit. on p. 15).

[6] Sven Helmer, Gerhard Glüher, and Christian Upmeier. „Show, Don't Tell: Retrieving Cultural Assets Via Gestures“. In: 4th Interdisciplinary Workshop The Shape of Things (SHAPES 4.0), 2017 (cit. on pp. v, 9).

[7] Fazlur Rahman Khan, Huey Fang Ong, and Nurhidayah Bahar. „A sign language to text converter using leap motion“. In: *International Journal on Advanced Science, Engineering and Information Technology* 6.6 (2016), pp. 1089–1095 (cit. on p. 15).

[8] David Koller, Bernard Frischer, and Greg Humphreys. „Research challenges for digital archives of 3D cultural heritage models“. In: *Journal on Computing and Cultural Heritage (JOCCH)* 2.3 (2009), p. 7 (cit. on p. 1).

[9] I Scott MacKenzie. „Input devices and interaction techniques for advanced computing“. In: *Virtual environments and advanced interface design* (1995), pp. 437–470 (cit. on pp. v, 2).

[10] Mohamed Mohandes, S Aliyu, and M Deriche. „Arabic sign language recognition using the leap motion controller“. In: *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*. IEEE. 2014, pp. 960–965 (cit. on p. 10).

[11] Robert O'Leary. *Leap Trainer*. 2013 (cit. on p. 15).

[12] Lin Shao. „Hand movement and gesture recognition using Leap Motion Controller". In: *Virtual Reality, Course Report* (2016) (cit. on p. 29).

[13] Babak Toghiani-Rizi, Christofer Lind, Maria Svensson, and Marcus Windmark. „Static Gesture Recognition using Leap Motion". In: *arXiv preprint arXiv:1705.05884* (2017) (cit. on p. 16).

[14] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O Wobbrock. „Gestures as point clouds: a $ P recognizer for user interface prototypes". In: *Proceedings of the 14th ACM international conference on Multimodal interaction*. ACM. 2012, pp. 273–280 (cit. on pp. 10, 11).

[15] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. „Analysis of the accuracy and robustness of the leap motion controller". In: *Sensors* 13.5 (2013), pp. 6380–6393 (cit. on pp. 2, 5).

[16] Xin Xiangyang, Cagan Jonathan, et al. „Interpreting Cultural Artifacts". In: *Guidelines for a Decision Support Method Adapted to NPD Processes* (2007), pp. 171–172 (cit. on pp. 12, 27, 41).

List of Tables

2.1 Interpreting a chisel tool 14

3.1 Scenario for the "Search" use case 19
3.2 Scenario for the "Train Gesture" use case 19
3.3 Scenario for the "Match Gesture" use case 20

4.1 Match scores 38

Declaration

I certify that the material contained in this dissertation is my own work and does not contain un-referenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation. Regarding the electronically submitted version of this submitted work, I consent to this being stored electronically and copied for assessment purposes, including the Department's use of plagiarism detection systems in order to check the integrity of assessed work. I agree to my dissertation being placed in the public domain, with my name explicitly included as the author of the work.

Bolzano, April 5, 2018

Edgar Obare