Optimizing Gesture Recognition

A comparison of hidden Markov models and
linear gesture recognition

Håvard Kindem

Master of Science in Informatics

Submission date:	June 2013
Supervisor:	Alf Inge Wang, IDI
Co-supervisor:	Simon McCallum, Gjøvik University College

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

This thesis aims to compare a simple linear recognition algorithm to that of the well proven and reliable Hidden Markov Model. It implements a gesture recognition system able to recognize gestures using both algorithms and compares their performance before and after applying optimization techniques to improve their speed and accuracy.

The system used to retrieve the results is developed using Java and is aimed towards wearable devices as an alternate interaction technique for devices with limited processing power.

The final conclusion from this thesis is that even a very simple recognition algorithm can perform nearly as well as more complex ones if the data-sets are presented well to the system.

Keywords: Gesture Recognition, Hidden Markov Models, Augmented Reality, Alternate Interaction

Acknowledgements

This thesis is written as a result of the course *IT3905 - Informatics Postgraduate Thesis: Game Technology* under the Department of Computer and Information Science at the Norwegian University of Science and Technology.

I would like to thank my supervisors; professor Alf Inge Wang at the *Norwegian University of Science and Technology* and associate professor Simon J R McCallum at *Gjøvik University College* for feedback, ideas and support throughout this thesis.

Gjøvik, 6th of June, 2013.

Contents

List of Figures

List of Tables

Part I

Introduction and Research

Chapter 1

Project Introduction

The last years the number of smartphones has continued to increase as well as new types of mobile devices coming to the market called wearable devices. As some of these devices lacks typical interaction interfaces like touch screens or keyboards, this thesis is looking into alternative interaction techniques; specifically gesture recognition and how to improve it.

This chapter we will introduce the problem definition, the motivation of the project as well as it's context.

1.1 Motivation

As technology today is pushing towards more and more mobile solutions, one of the newest additions to this area of technology are wearable devices. These devices include watches, rings, glasses and clothing. This area of research is fairly unexplored as most of these the products has not yet been made available to the public. Many of the findings from research done towards mobile devices can be applied for wearable devices as well, though we are looking at challenges interacting with these devices.

As Google has announced their Project Glass[1] in 2012, this and simliar devices are going to be the origin for this research. One of the possible methods of interacting with mobile devices without classical input keys or touch screens is hand gestures. Using hand gestures is relevant for all devices able to get positional or accelerometer data from the hands of the user. This can either be through hand tracking using a camera or by attaching a device to

[1]`http://www.google.com/glass/start/`

the wrists of the user. One device that does the latter is the Pebble Watch[2] which includes an accelerometer that we can use to record hand movements.

Hidden Markov models is commonly used for temporal pattern recognition and has a central place in gesture recognition, I want to see how well it performs compared to a simple linear recognition algorithm.

1.2 Problem Definition

The main goal of this thesis is to conduct research towards alternate interaction techniques for mobile devices and Augmented Reality. The main focus will be gesture recognition, researching how it can be improved an comparing the results to existing temporal pattern recognizing algorithms; specifically Hidden Markov Models.

The results of this research aim to provide a good comparison between a very simple recognition algorithm and Hidden Markov Models(HMMs) used for temporal pattern recognition.

The research goals can be found in detail at Section 2.3.

1.3 Project Context

This project is a part of my master thesis at the Norwegian University of Science and Technology(NTNU)[3]. It is conducted under the Game Technology research program at the Department of Computer and Information Science(IDI). This research is also in collaboration with the Game Programming program at Gjøvik University College(HiG)[4].

1.3.1 Related Work

There is no research published that compares the performance of Hidden Markov Models(HMMs) with simplier alternatives that I managed to identify. There are however a lot of research on HMMs in gesture recognition.

[2]`http://getpebble.com/`
[3] `http://www.ntnu.no`
[4] `http://www.hig.no`

Wilson and Bobick[1] presents a parameterized version of the HMM, using an Expectationmaximization(EM) algorithm in addition to the HMM to improve the accuracy. This results in a gesture recognition system that as well as handling three dimensional gestures, can recognize two dimensional gestures based on the context from the EM algorithm.

Chen, Fu and Huang[2] uses HMMs to do gesture recognition with hand tracking, using both temporal and spatial data. The system is able to recognize new gestures as well. Their system results in a recognition rate above 90%.

Starner and Pentland[3] uses a four state HMM to decipher sign language with an accuracy of 99.2%.

There are much more research done into temporal pattern recognition using HMMs to achieve gesture recognition, voice recognition and handwriting recognition, though the mentioned ones stand out.

1.4 Reader's Guide

It is preferred that the reader of this thesis has a general knowledge about programming and a good understanding of math, though the content of this thesis is written simple enough for most people to understand. The research questions for this thesis are presented in Section 2.3.

For those that might not be familiar with gesture recognition, the prestudy is a good place to start. It explain what is needed to understand the work in this thesis:

Chapter 3: Augmented Reality and the Glass Technology

Chapter 4: Gesture Recognition

For those interested in the technical aspect of this thesis, the following parts should prove useful:

Chapter 6: Gesture Recognition Implementation

Chapter 8: Optimizing Gesture Recognition

Finally, for those interested in the results and findings of this thesis and the evaluation of the results the following parts should be interesting:

Chapter 7: Performance Comparison

Chapter 9: Results

Chapter 10: Evaluation and Future Work

Chapter 11: Conclusion

1.4.1 Nomenclature

This section lists abbrevations and expressions used throughout this thesis.

AR - Augmented reality

EM - Expectationmaximization

GPS - Global Positioning System

HMM - Hidden Markov Model

HUD - Heads Up Display

JVM - Java Virtual Machine

Nomenclature - This list

QR Code - A two dimensional barcode

UI - User Interface

Spatial - Related to space

Temporal - Related to time

VR - Virtual Reality

1.5 Stakeholders

Following is a list of stakeholders and their concerns related to this research.

1.5.1 Researcher

- Håvard Kindem

The researcher is primarily concerned with the quality of this research and it's future applications. As these topics are of interest of the researcher, its important that the findings can be applied in future applications and architectures.

1.5.2 Course Staff

- Alf Inge Wang - Supervisor
- Simon McCallum - Co-supervisor

The course staff are concerned with the quality of this report as well as it's findings. The quality of this report is vital as the readability should be good and the research should be of academic value. The findings in this research is also of value, as the course staff are looking to further improve their knowledge of using the best approaches for developing temporal pattern recognition solutions.

1.5.3 Developers

Developers are concerned with the readability of this thesis and the ability to quickly be able to retrieve and verify the results found in this research. The main concern is that the findings are correct and proven to be so to avoid non optimized solutions.

Chapter 2

Research

This chapter introduces the research goals as well as the methods applied in this research.

2.1 Goal Question metric

The Goal Question Metric(GQM) is an approach to software metrics by Basili, Caldiera and Rombach[4]. The GQM approach is used to better specify a measurement model. It clarifies the goal for questions and gives each question a well defined metric to compare the results found in various research. This is meant to aid the researcher keeping focus on the task at hand.

They suggest a measurement model with three levels:

Conceptual Level (GOAL): A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment.

Operational Level (QUESTION): A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.

Quantitative Level (METRIC): A set of data is associated with every question in order to answer it in a quantitative way. The data can be either objective or subjective.

The authors argue that the measurements must be defined in a top down fashion and to be effective, it must be:

1. Focused on specific goals.
2. Applied to all life-cycle products, processes and resources.
3. Interpreted based on characterization and understanding of the organizational context, environment and goals.

2.2 Research Methods

This section presents the research methods used in this thesis.

2.2.1 Literature Review

The *literature review* research method consists of reviewing existing literature on the subject. This is preferably done as early as possible in the research process to avoid the researcher reinventing the wheel. This ensures that the researcher has a general understanding of the subject, giving the researcher a good foundation for their own research.

2.2.2 Design Science

The definition by Hevner, March, Park and Ram[5] on design science says: *In the design-science paradigm, knowledge and understanding of a problem domain and its solution are achieved in the building and application of the designed artifact.* Put simply, this means that to find a solution to a problem using design science, the researcher develops a product to achieve the solution through the development process. The authors give seven guidelines for applying design science:

1. **Design as an Artifact**
   Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
2. **Problem Relevance**
   The objective of design-science research is to develop technology-based solutions to important and relevant business problems.

3. **Design Evaluation**
   The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.

4. **Research Contributions**
   Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.

5. **Research Rigor**
   Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.

6. **Design as a Search Process**
   The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.

7. **Communication of Research**
   Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

2.2.3 Use of Research Methods

This project starts with a research phase, using the literature review method. We will look into simliar research and research relevant to answering our research questions. This will give a good basic understanding of the topics covered in this project.

The *Goal Question Metric*(GQM) approach will be applied for the research questions. This will aid in setting up research questions that are relevant to the research goal as well as setting up metrics that are relevant to the research questions.

For the own contribution part, a mix of litterature review and design science will be applied. Litterature review will be used to identify and evaluate options found from the research questions. Design science will be used to evaluate the research questions covering the developed gesture recognizer, as I need to develop my own solution and be able to compare it to the results from other implementations.

2.3 Research Goals

This research will look into how to better can interact with wearable mobile devices, specifically aimed towards Augmented Reality Glasses. I will try to identify strengths and weaknesses of the gesture recognition implementation. The following research questions are the basis of the research:

Goal

Research the accuracy and performance of a gesture recognition system, using a simple gesture recognition algorithm and compare the results to the same system using hidden Markov models.

RQ 1

Question: How does a simple gesture recognition algorithm's accuracy compare to that of hidden Markov models?

Metric: Percentages showing the accuracies of both systems.

RQ 2

Question: How does the computational performance of a simple gesture recognition algorithm compare to that of hidden Markov models?

Metric: Timings for both algorithm's training time and classification time.

RQ 3

Question: What techniques can we apply to the gesture recognition system to optimize it's computational performance and accuracy?

Metric: Summary of- and discussion on techniques used.

RQ 4

Question: How does the techniques in RQ 1.3 affect the perfomances of the two algorithms?

Metric: Step-by-step comparison of the accuracy and computational performance.

RQ 5

Question: What benefits and tradeoffs can we identify in the developed gesture recognition system when using the simple gesture recognition algorithm?

Metric: Discussion of the benefits and tradeoffs based on the results in RQ 1.4.

RQ 6

Question: Based in the findings is RQ 1-5, are the vector components of a gesture coherent enough to be used as a single token?

Metric: Conclusion based on the results from the implemented systems and reasoning for it to be true or false.

Part II

Pre-study

Chapter 3

Augmented Reality and the Glass Technology

In this chapter the Augmented Reality(AR) technology, its current state and challenges will be presented.

3.1 What is Augmented Reality

In his paper *A survey of augmented reality*[6], Azuma defines AR: *Augmented Reality (AR) is a variation of Virtual Environments (VE), or Virtual Reality as it is more commonly called. VE technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. Therefore, AR supplements reality, rather than completely replacing it.*

AR requires the implementation to use a form of input, usually in the form of a camera, GPS and an orientation sensor(compass) as well as an output in the form of a screen. This has in the later years made mobile devices(phones and tablets) perfect for augmented reality. Today's mobile devices come with a ton of hardware sensors, ranging from Camera, GPS, Accelerometer, Compass, Light sensor and Proximity sensors. This gives us a lot of options when implementing an augmented reality application.

Due to the current performance of the mobile devices, AR applications can be run in real-time at ease.

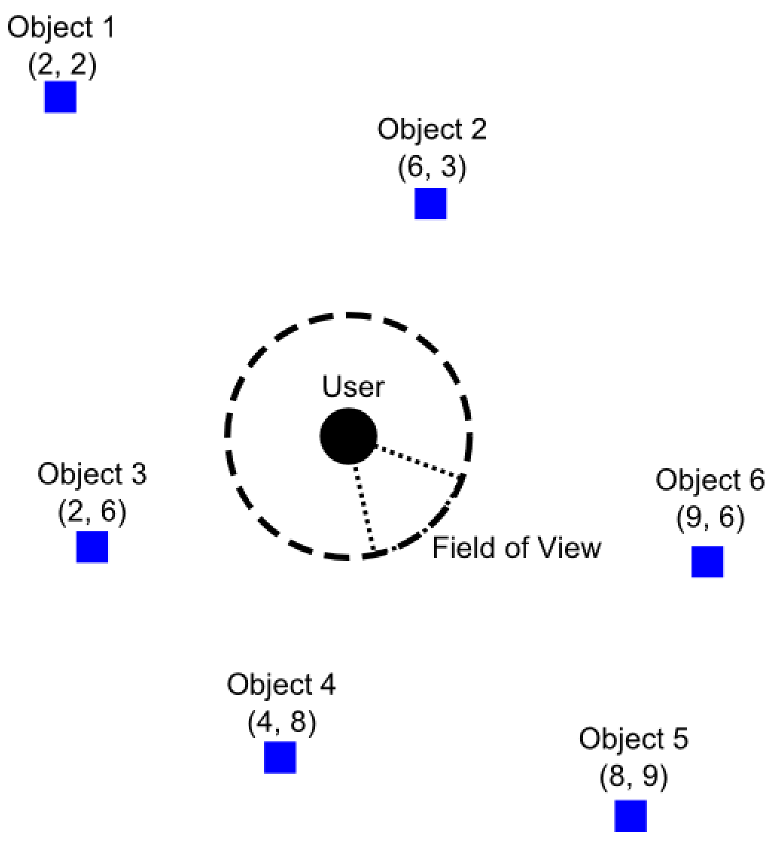

Figure 3.1: Illustrating an AR client using only positional and orientation data

3.2 Registering real world objects to the AR device

An augmented reality application needs to have some way of transferring a real-world position into screen space, this can be done is a series of ways. The simplest and maybe most common way at the moment is using orientation and position, then overlaying objects on the screen based on the user's and object's real-world position, taking account for the user's orientation to create a field of view as shown in Figure 3.1.

One other way of getting the real-world information to the device is image recognition. This uses from basic algorithms, recognizing QR codes to complex image recognition algorithms and are able to trace images or shapes defined in the program. Usually this is done using so called AR-markers then using this position to augment the world around us, though it can also be used to recognize hand movement etc.

3.3 Applications of Augmented Reality

AR is applicable to multiple fields. Azuma[6] mentions six different applications: Medical, Manufacturing and repair, annotation and visualization, robot path planning, entertainment and military aircraft. As these examples are fairly specific, this list will generalizes them further.

Training: AR can be used for training for certain tasks, a doctor could train for surgery on a virtual copy of a patient even before the real surgery begins. This could be supplemented by real-world objects as well to further improve the authenticity of the training.

Information: AR is very suitable to provide information about the real world and it is widely used on mobile devices today. This can provide information about the scenery, signs, even commercials. Augmented reality applications can also be used as a substitution for other types of Heads Up Displays(HUDs), like the ones used by various aircraft.

Entertainment: One other application of AR is in games and entertainment. This could be anything from treasure hunting games, adding actors or characters to the real-world in the way of a movie or moving the game from a computer to outside.

3.4 Real-world Aumented Reality applications

There is an ever-growing set of AR applications who use various techniques to achieve their goals. This section presents some of the most noteable of them.

3.4.1 Wikitude

Wikitude is an application that allows the user to search or browse for places in you vicinity and get information on them. This has been expanded to contain everything from tweets, wikipedia articles, restaurants, hotels etc. The application also allows for sharing the user's favourite places and even play games. Figure 3.2 shows the UI with the search results on screen.

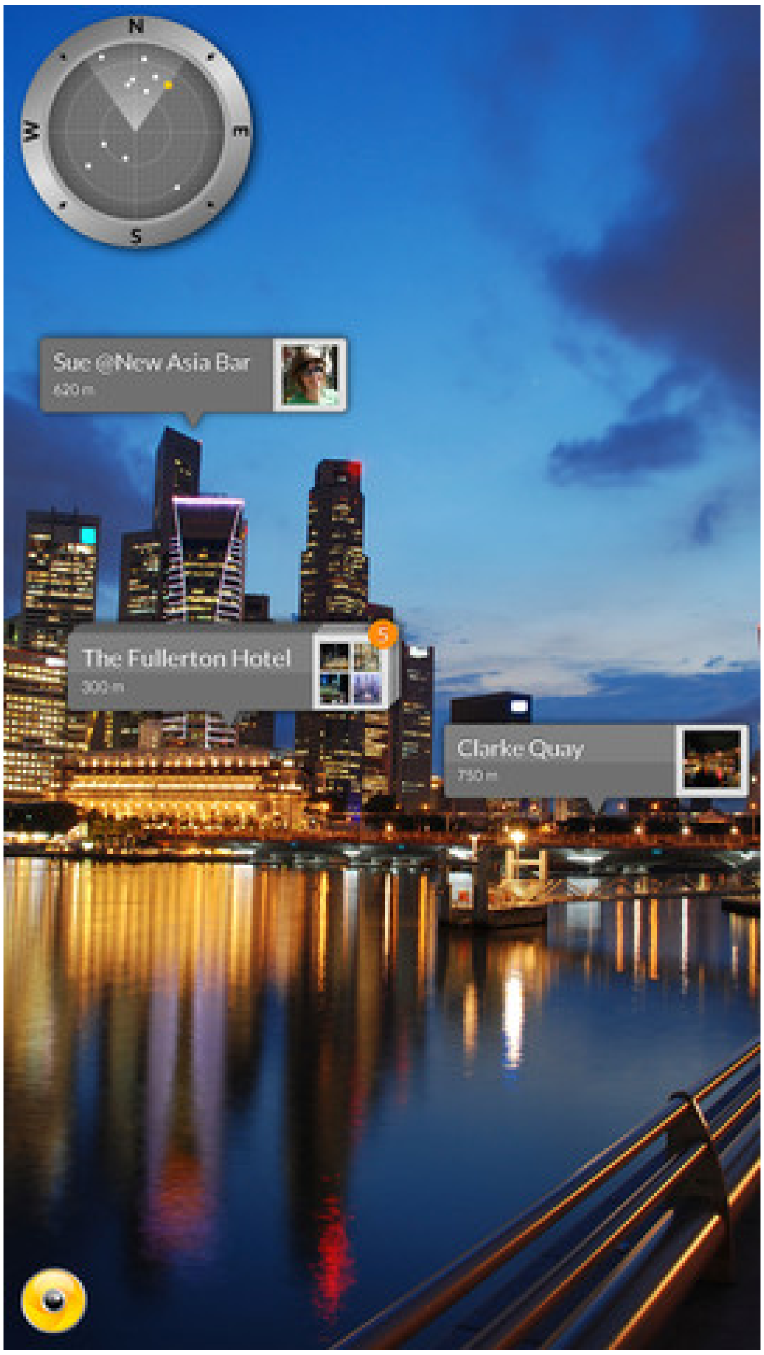

Figure 3.2: Wikitude showing the search results on top of the camera view.

It uses a location based approach to AR, meaning that it places the objects on screen based on the user's lcoation and orientation. Wikitude has been voted to be the *Best Augmented Reality Browser* four years in a row.

3.4.2 Google Goggles

Google Goggles uses image recognition to get information about the image content. It can be used to get information about landmarks that the user takes pictures of and it can use the *Google Image Search* to recognize logos and products and get their respective information. The application can also recognize text to translate it to a language preferred by a user or add contact data from a business card. It can scan QR codes to get information from it simliar to other applications that work as *barcode scanners* as well.

This kind of image recognition has also allowed for the application to be able to solve sudoku puzzles as shown in Fig. 3.3.

3.4.3 iOnRoad Augmented Driving

The iOnRoad application aims to improve on car safety. It uses positional data(GPS) as well as image recognition to analyze the road ahead to ward the user about possible dangers. The application also tracks the car's position on the road, notifying the user if the car should drift off when the driver is sleepy.

Figure 3.4 shows the application in action.

3.5 Wearable devices

Wearable devices or *mobile wear* are small devices that is worn as or with clothing. This includes E-textiles, fabrics that allow digital devices to be embedded in them and what is commonly referred to as accessories, like wrist watches, glasses or other jewelry. The last couple years there has been a significant increase in wearable devices, though few has been made publicly available yet.

Google has announced their Project Glass[1], which are Android powered glasses. Pebble has just begun shipping their Pebble Watch[2], a wrist watch

[1]`http://www.google.com/glass/start/`
[2]`http://getpebble.com/`

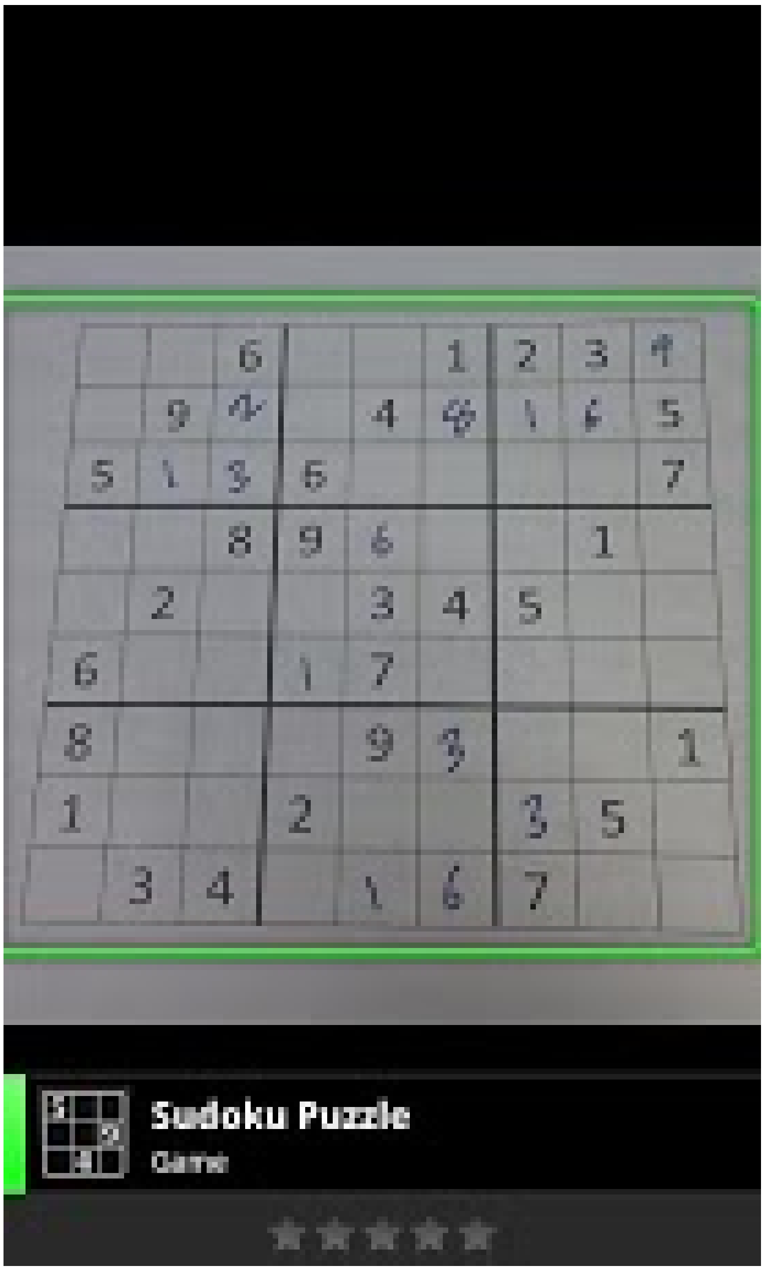

Figure 3.3: Google Goggles solving a sudoku puzzle.

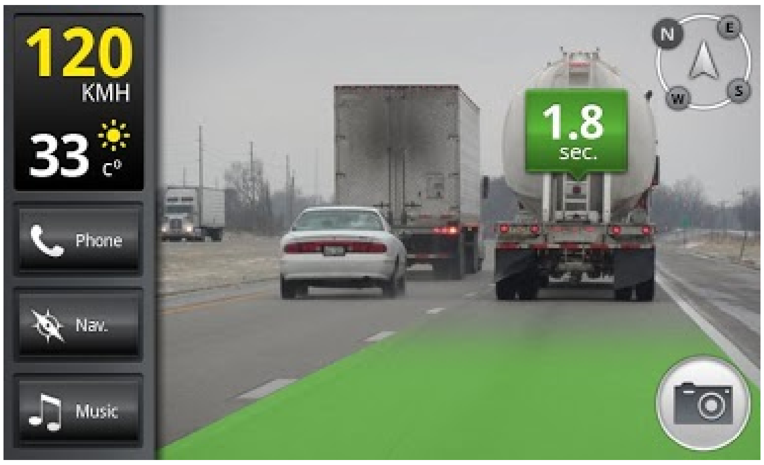

Figure 3.4: The iOnRoad application tracking the lanes on a road.

that is capable of using Bluetooth to connect to other mobile devices, displaying various information and even interact with the device both ways(sending and receiving information).

3.6 Glass Technology

Though Google were the first ones that announced that they wanted to make the glass technology commercially available, there are a few alternatives being developed so we might be looking at a new trend in technology.

The concept is to have a pair of glasses with a screen, an integrated camera, sound, GPS, orientation sensors all packed as a fully working mobile device. This opens up a lot of possibilities in terms of interaction. Using these kind of devices, we can free up our hands and allow for an even better human-computer interaction, making it more painless to use as well as more accessible in terms of availability when using the device(no more looking for you mobile phone).

3.7 Interaction Challenges with the Glass Technology

As the point of the glass technology is to be as mobile as possible, not having any wires or large interaction panels there are some interaction challenges of using these kind of devices. Though at least the Google Glass is coming with hardware buttons on the device frame, it is limited how many buttons the device can have before being cluttered. This forces us to look into other interaction methods. The Google Glass is shipping with voice recognition, allowing the user to use voice commands to control the device. Though this is simple to use, it has it's faults: In the real-world there are a lot of noise, which will in terms lower the accuracy of speech recognition.

Hirsch[9] found that the recognition accuracy was especially diminished when the training data was considered *clean*. In various noisy environments, the accuracy would then be as low as 66% compared to the multi-condition training data at 83%.

As an alternative, we can use hand gesture recognition to interact with the devices, though this suffers from some challenges, similar to the ones with voice recognition.

3.8 Performance Challenges with Mobile Devices

One of the ever evolving challenges of mobile devices, as well as traditional computational devices is keeping up in performance. This comes especially true in mobile devices as they are normally restricted in physical size. To increase the overall performance, we can do one of two things:

Increase computational power: Assuming the same processor architecture, increasing the clock speed gives us a linear increase in performance. Ie. doubling the clock speed (1Ghz to 2Ghz) halves the runtime of a program. This is historically what has been evolving the complexity of developed applications.

Code optimization: This is normally done when there is a limitation on the hardware or the developer wants to make something available to a wider audience. Code optimization is also used to develop faster routines for computational operations, decreasing the need for computational power. For example: solving the P versus NP problem(also known as the traveling salesman problem) would result in drastic improvements in multiple fields, like artificial intelligence, cryptology and math.

Chapter 4

Gesture Recognition

This chapter introduces gesture recognition. It presents typical implementations and discuss their applications.

4.1 What is Gesture Recognition

Mitra and Acharya's[10] definition of gesture recognition goes as follows: *Gesture recognition pertains to recognizing meaningful expressions of motion by a human, involving the hands, arms, face, head, and/or body.*

Gesture recognition uses either spatial or temporal data to decipher movement into complete gestures that are then used as an input source for a system.

Spatial data is gesture data that are based on real world positioning. This is a collection of points consisting of x, y and z.

Temporal data is often used when referring to non-positional data. This data is based on changes in movement over time, rather than actual positioning. This consists of three dimentional vectors.

Both kind of data can be used to recognize gestures in a very simliar way. The only change in handling the data is their actual origin. By this, I mean that the spatial data must be based on a delta from an origin point, while the origin point of the temporal data is the gravity vector. After the initial handling of the data, it is handled by algorithms suitable for recognizing patterns and the result is sent to the application.

4.2 Applications of Gesture Recognition

While there are an ever expanding set of possible applications of gesture recognition, some of the ones that are used today are:

Sign language recognition is one area that is massively researched and can majorly benefit mute or hearing-impaired people. When done in real-time, it can efficiently translate sign language to text or voice, allowing for effortless communication between people using sign language and people that does not know it.

Recognition of emotions is a known difficulty for people suffering from autism or Asperger syndrome. Gesture recognition can also be used to decode facial expressions, aiding in decoding what emotions the subject is feeling. This could significally improve the quality of life for people suffering from these kind of impairments.

Alternate computer interaction can help people unable to use a traditional computer and mouse as well as improving on existing interaction. It can be used as interaction in games, making them more immersive. Even more realistic simulators can be made, allowing the player to interact with the game world like he or she would in the real world.

4.3 Difficulties in gesture recognition

There are several difficulties in gesture recognition, amongst the main ones are: accuracy, usefulness and processing cost.

Accuracy: First of all, there will be a fair amount of noise in the data when doing gesture recognition. This needs to be handled before the recognition algorithms can be run at all. After the initial cleaning, there must be enough accurate, clean training data for the gesture recognizer to compare the input to. The system must also be calibrated correctly to avoid false positives, as well as being able to correctly identify the gesture. If we fail to obtain a good accuracy, the usability of the system will be diminished and the user is likely to look for another input method.

Usefulness: Gesture recognition should not just be a neat new way of interacting with a system. There are other well known, working input devices that are very easy to use (touch screens, keyboards, mice, buttons).

Gesture recognition should aim to augment these methods or provide interaction techniques that are not possible on the other input devices. It should be as easy or easier to use then other input devices for it's application.

Processing cost: As the noise reduction or filtering, as well as the common recognition algorithms are very computational consuming, it is a great challenge to be able to run a gesture recognition system in real-time without sacrificing significant processing power, better used on other areas of an application.

4.4 Contexted versus context-free gestures

Context-free gestures generally mean that the gestures have no additional information. All possible gestures are considered equally likely to occur. We can think of this as a globally available variables in an application. It can be accessed from anywhere and everyone has access to it. If we have a finite set of gestures $A = \{G_1, G_2, .., G_n\}$ wi the size n, when using context-free gestures we are trying to match the input data with all gestures in the set A.

With contexted gestures, we are actively limiting our matchable set. To match our explanation of context-free gestures, this time we are using a private variable so that not everyone has access to it. This means that there are fewer points in the apoplication where the variable can be accessed. Let's say that we have the same set of gestures A, with contexted grammar we are now deciding what gestures are available to be recognized at this time, reducing our set to a subset of A, called B. An example of this kind of subset can be $B \subset A = \{G_2, G_4, G_7\}$.

As you can tell, we now have a much smaller set of matchable gestures. In a gesture recognition system we might want both contexted and context-free gestures, this gives us a faster matching time and still leaves us the possibility of globally recognizable gestures for when the context does not matter.

Chapter 5

Summary

The sections covered in the pre-study shows that there are a lot of interesting applications of Augemented Reality. As there are coming new mobile devices capable of doing AR, the challenge of interacting with these devices needs to be addressed. Gesture recognition is only one of many options.

Different AR applications has been presented along with the difficulties and applications of gesture recognition to hopefully allow the reader to get an idea about what is possible to obtain with AR in these new devices.

As this thesis presents an alternate way of implementing gesture recognition and how to improve it, understanding these challenges of gesture recognition and interacting with these devices are vital.

Part III

Own Contribution

Chapter 6

Gesture Recognition Implementation

In this chapter, the implementation of the gesture recognition system is presented.

6.1 Requirements

The goal is to implement a gesture recognition system as simple and performance effective as possible. The purpose is to look into what accuracy we can obtain when using simple algorithms and trying to prove that because the vector components of a gesture is linearly connected, the system can be simplified to an extent that drastically improves the performance. Following is the list of requirements for the implementation.

Feature Requirements:

1. FR1: Comparing a gesture to another, giving it a score based on it's simliarity.
2. FR2: Storing gestures in a way that makes comparison quick and easy.
3. FR3: Isolating gestures in a buffer of vector data.
4. FR4: Reducing noise from the incoming accelerometer data.
5. FR5: Pruning irrelevant data from a vector sequence.
6. FR6: Returning a list of gestures with a simliarity score to a given gesture.

7. FR7: Calibrating the system to recognize true positives and ignore false positives.

Non-functional Requirements:

1. NFR1: Implementation developed in Java.

2. NFR2: Recognition accuracy ≥ 90.

Feature requirements 1, 3, 4 and 6 are the most important ones. These are essensial to the recognition system and needs to be prioritized in the implementation. FR7 is essensial to obtain the accuracy defined in NFR2, this will be part of the calibration process.

6.2 Design

This section is about the hardware used, platforms, languages, libraries etc.

6.2.1 Platform and Language

The implementation will be using Java[1] as our language as it is platform independant and it has an easy way of handling networking and dynamically sized collections. Using Java also allows us to easily use the application as a library for later Android development. For any data manipulation needed, Python[2] will be used. Figure 6.1 shows the basic class diagram for the gesture recognition system.

6.2.2 Input device

It was originally planned that the Pebble Watch[3] would be used for getting input data. As I have been unable to get a hold on a Pebble Watch due to it's popular demand, it was decided to use an Android device instead. It serves the same purpose and can replicate the data received from a Pebble Watch. Instead of using Bluetooth, we will be using WiFi for simplicity. This will have no effect on the results.

[1]http://www.java.com/
[2]http://www.python.org/
[3]http://getpebble.com/

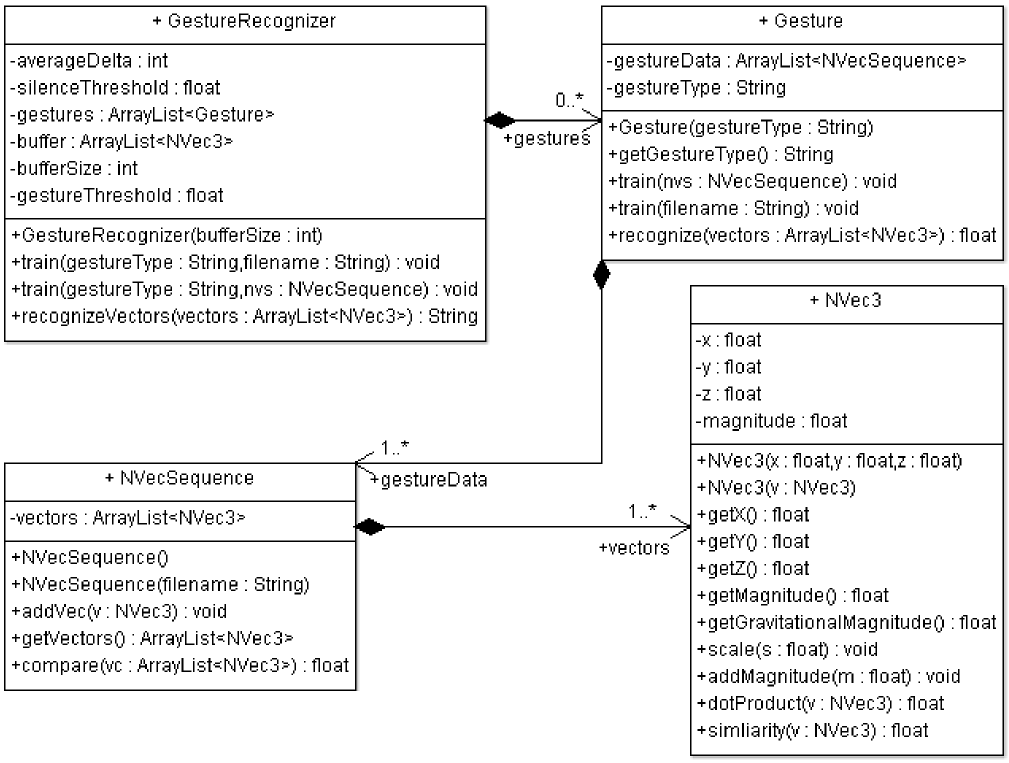
+ GestureRecognizer
-averageDelta : int
-silenceThreshold : float
-gestures : ArrayList<Gesture>
-buffer : ArrayList<NVec3>
-bufferSize : int
-gestureThreshold : float
+GestureRecognizer(bufferSize : int)
+train(gestureType : String,filename : String) : void
+train(gestureType : String,nvs : NVecSequence) : void
+recognizeVectors(vectors : ArrayList<NVec3>) : String
0..*
+gestures
+ Gesture
-gestureData : ArrayList<NVecSequence>
-gestureType : String
+Gesture(gestureType : String)
+getGestureType() : String
+train(nvs : NVecSequence) : void
+train(filename : String) : void
+recognize(vectors : ArrayList<NVec3>) : float
+ NVec3
-x : float
-y : float
-z : float
-magnitude : float
+NVec3(x : float,y : float,z : float)
+NVec3(v : NVec3)
+getX() : float
+getY() : float
+getZ() : float
+getMagnitude() : float
+getGravitationalMagnitude() : float
+scale(s : float) : void
+addMagnitude(m : float) : void
+dotProduct(v : NVec3) : float
+simliarity(v : NVec3) : float
1..*
+gestureData
+ NVecSequence
-vectors : ArrayList<NVec3>
+NVecSequence()
+NVecSequence(filename : String)
+addVec(v : NVec3) : void
+getVectors() : ArrayList<NVec3>
+compare(vc : ArrayList<NVec3>) : float
1..*
+vectors

Figure 6.1: The class diagram for the gesture recognition system.

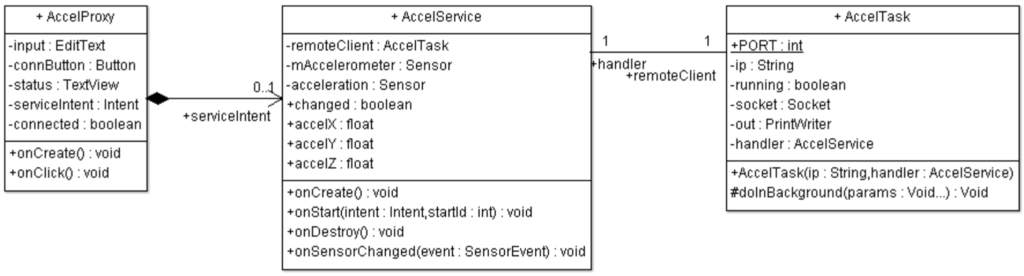

Figure 6.2: The class diagram for the application used to collect input data.

6.2.3 Gathering test data

As it was chosen to transfer the gesture data over WiFi, training data can easily be collected by using the NetCat[4] application and outputting the transferred data to a text file. This is exactly the same data transferred by the device, so we will keep a concurrent dataset in our application. To output the test data to a file, the command used is *nc -vv -l 8889 >> input_data.txt*. This sets NetCat to listen on port 8889 which is the recording application's connection port, outputting the data into *input_data.txt*.

The class diagram for the application developed for gathering test data is shown in Figure 6.2.

To make the data usable to jahmm[5], we need to convert the data to the form:

```
[ -0.37861265 9.498847 1.8283978 ] ;  [ -0.34029235 9.74154233333
1.48351503333 ] ;  [ -0.110370486 9.69044833333 1.4451947 ] ;
[ -0.020956422 7.37845643333 1.61124936667 ] ;
```

Multiple vector sequences are split into separate lines and it's components are encapsulated by square brackets and separated by semicolons. We will write a custom loader for our implementation and using the built in file reader to use the sequences in jahmm. The file can be loaded in jahmm with the following code:

```
List<List<ObservationVector>> sequences;
try {
    Reader reader = new FileReader(dataSetFile);
    sequences = ObservationSequencesReader.readSequences(
        new ObservationVectorReader(), reader);
    reader.close();

} catch (IOException e) {
    e.printStackTrace();
} catch (FileFormatException e) {
    e.printStackTrace();
}
```

During the development, the custom loading functions and the ones integrated with jahmm got interchanged and the results started to come out with

[4]http://netcat.sourceforge.net/
[5]https://code.google.com/p/jahmm/

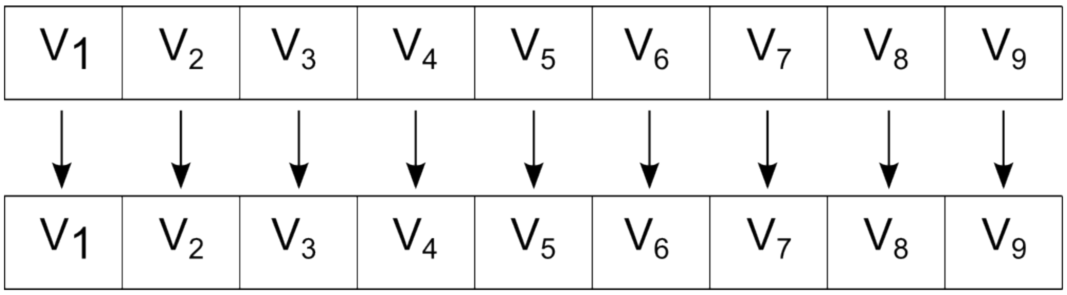

Figure 6.3: Linear comparison

much less precision. To countneract this, the custom loader was modified to handle the *ObservationVector* sequence from jahmm as well.

6.2.4 Recognition algorithm

As mentioned earlier, the purpose is to make the recognition process as fast and simple as possible. To achieve this, a scaled linear recognition algorithm as seen in Fig. 6.4 will be used. This is a simple way to remove the time dimension of the results, compared to a normal linear comparison (see Fig. 6.3). This recognition algorithm should have a performance comparable to a lazy learning algorithm, though using statistical analysis instead of actual learning.

This way of recognizing gestures will most likely have a loss in accuracy at least with a data-set that contains noise. As the the development progresses and we get to reduce the noise and classify the vectors nicely, we should see an improvement in accuracy. We will scale the regognition algorithm so that it can handle different length comparisons. This algorithm is dependant on a good implementation of isolating gestures to avoid having to brute-force the input buffer.

The implementation of the scaled linear comparison is very straightforward. It produces a step size for both sequences based on the length of the smallest sequence and iterates over the minimum length. This then calls the simliarity function which compares two vectors and quantifies the result (see Section 6.2.5) for more on the simliarity classification.

```
public static double compareVec3Sequences(List<Vec3> a, List<Vec3> b) {
    int minCount = (a.size() < b.size()) ? a.size() : b.size();
    float aStep = (float)a.size() / (float)minCount;
    float bStep = (float)b.size() / (float)minCount;
```

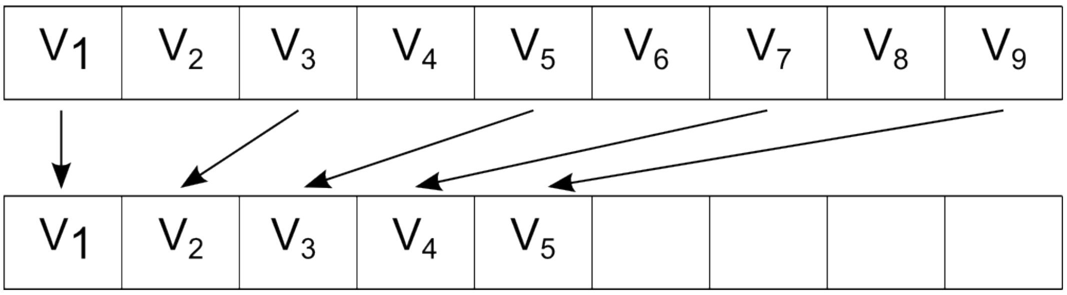

Figure 6.4: Scaled linear comparison

```
    double result = 1.0f;

    for(int i = 0; i < minCount; i++) {
        int ai = (int) Math.round((aStep * (float)i));
        int bi = (int) Math.round((bStep * (float)i));

        result *= a.get(ai).simliarity(b.get(bi));
    }

    return Math.min(result, 1.0);
}
```

6.2.5 Vector simliarity quantification

As the application will be comparing vectors it needs to be a quantifiable measure so that the sequence of vectors can be summarized to give a total simliarity, the dot product is perfect for this.

$$X \cdot Y = |X||Y| \cos \theta \tag{6.1}$$

The equation for the dot product (See Eq. 6.1) measures the angle between two vectors. As cosine is a periodic function, $\cos \theta$ ranges from -1 to 1. -1 meaning that it is exactly opposite of the compared vector and 1 means that it is parallel to the compared vector.

$$\cos \theta = \frac{X \cdot Y}{|X||Y|} \tag{6.2}$$

As this fits as a good measure of simliarity we will be using the value of $\cos \theta$ (See Eq. 6.2) as our quantification in addition to a percentage length comparison. Having negative numbers will not aid in the accuracy of the

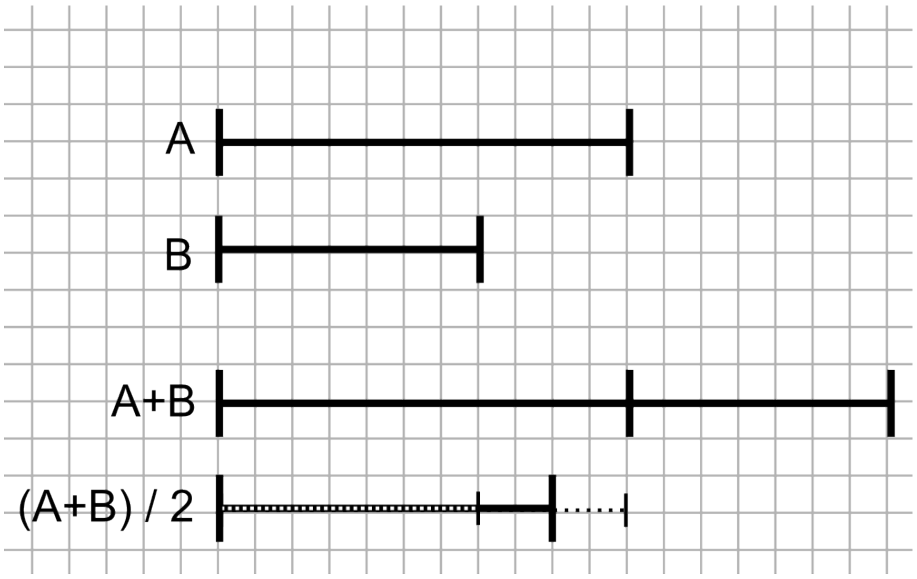

Figure 6.5: Comparison of vector magnitudes

comparison, so it will be normalized to be in the range $[0 \dots 1]$. The equation for calculating the simliarity can be seen in Eq. 6.3.

$$Q_d = \frac{\cos\theta + 1}{2} \tag{6.3}$$

The system also needs to be able to differ twi vectors based on their respective lengths. Even if two vectors are parallel, they might be very different in terms of length. To make an example of this: A bicycle and a car is traveling on the same road, analyzing their movement only would result in them might being classified as the same thing. This is why the system needs to look at the magnitudes of the vectors as well.

To compare the the lengths of two vectors, we want a quantifiable measure that says how simliar the distances are. We want this measure to be in the range of $\langle 0 \dots 1]$ so that it can easily be applied to the direction comparison measure. The equation for the length comparison can be found in Equation 6.4, Figure 6.5 illustrates this and shows that we are basing the measurement on the average of the two lengths and compare it to the vector with the greatest magnitude. If both these vectors were of simliar length, Q_m would be 1.0.

$$Q_m = \frac{1}{\max(L_1, L_2)/\frac{L_1+L_2}{2}} \tag{6.4}$$

Finally, both Q_m and Q_d are multiplied to generate the total simliarity of the two vectors and implemented in java it looks like it does in the *simliarity(Vec3 function)*.

```
public double simliarity(Vec3 v) {
    double a = (dotProduct(v) + 1.0) / 2.0;
    double b = 1.0 / (Math.max(length(), v.length()) /
         ((length() + v.length()) / 2.0));
    return a * b;
}
```

6.3 Testing

The linear recognition algorithm will be compared to the same system, using hidden Markov models(HMM) instead. The tests will be done step by step to see the performance increase gained by the different steps. For each step, the following attributes will be recorded:

- Learning time
- Classification time
- Classification accuracy
  - True positive percentage
  - True negative percentage

The recordings will be based on timings from the JVM's function: $System.nanoTime()$ to get a time-measure as precise as possible. The routine for measuring this will be:

```
long startTime = 0;
long endTime = 0;

startTime = System.nanoTime();
functionToBeTimed();
endTime = System.nanoTime();

System.out.println("Recorded time: " + endTime-StartTime);
```

Chapter 7

Performance Comparison

7.1 Comparison Subject

The algorithm we are comparing the linear recognition algorithm's performance against is called *hidden Markov model*(HMM), a java implementation of this called *jahmm*[1] will be used. This is a HMM library implemented by Jean-Marc Francois. This is an open source project and gives us the ability to see the working code. The version used is *jahmm-0.6.1.* Jahmm also contains well-known algorithms that are related to HMMs and due to it's open source status, it is well suited for research.

7.1.1 Hidden Markov Model

Hidden Markov Model (see Fig. 7.2) is classified as a statistical Markov model. This means that it assumes that the system is a Markov process with hidden states. A Markov process implies that one can predict the future results based on it's present and previous states. An example of this could be estimating the amount of money in a bank account, based on the owner's actions. In this case, the amount of money in the account(states) is hidden to the observer while the different actions(output states) of the owner would be observable.

Hidden Markov Models are known for their application in temporal pattern recognition like speech, face, handwriting and gesture recognition. It was first described by Leonard E. Baum in his paper *Statistical inference for probabilistic functions of finite state markov chains*[11] in 1966. It was

[1]https://code.google.com/p/jahmm/

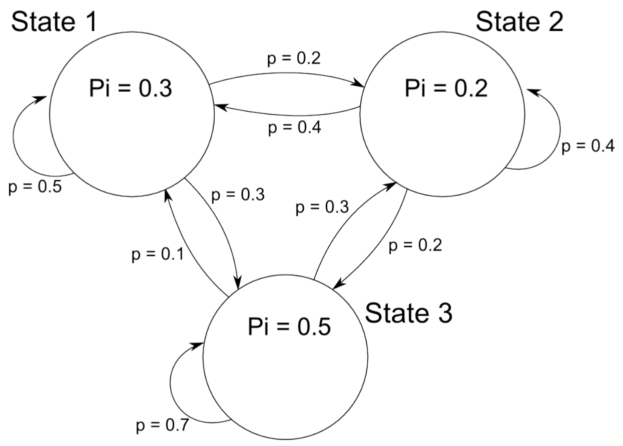

Figure 7.1: Markov chain with three states

later described in depth by Lawrence R. Rabiner in his paper *A Tutorial On Hidden Markov Models And Selected Applications in Speech Recognition*[12] in 1989.

Markov chains(see Fig. 7.1) are simpler HMMs, the difference between these two is that in Markov chains, the states are directly visible to the observer. This system also transisions between states, though the next state only depends on the current state, not previous states.

There are three problems a HMM can solve, given the parameters of the model:

- Learning from a data-set to make predictions on future outcomes.
- Classifying the most probable set of states that caused a particular output sequence.
- Calculating the probability of a sequence originating from the learned sequence.

7.2 Comparison Points

The linear recognition algorithm will be compared to the HMM implementation on the following points. This will give us a good measurement on how they compare as well as an indication on what improvements we gain from the different steps.

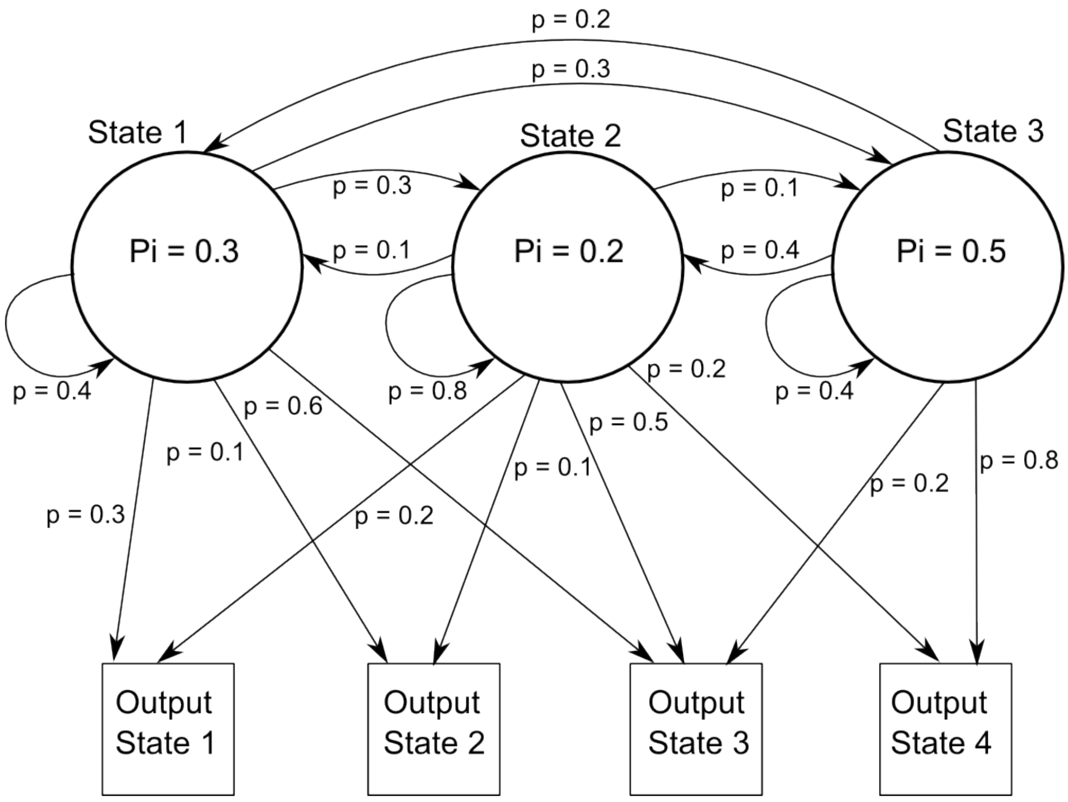

Figure 7.2: Visualization of a Hidden Markov Model

7.2.1 Learning execution time

The execution time for the learning process gives us an indication on how heavy the learning process is. This will be the delta time from the function begins executing until it has finished. This will include function overheads for both implementations, so our results will not be skewed but the direct comparison of the execution times will be affected to a minor degree when the execution times differ.

If the result times are 1.0 and 0.5 seconds, both with a 0.1 second overhead a direct comparison would be affected: $S = (1.0 + 0.1)/(0.5 + 0.1)$. In our case, the overhead will be so small that it will not be significent enough to eliminate.

I am interested in the learning time, as this will affect the initialization time of the recognition system in addition to the normal program initialization. When we have a lot of gestures, this can affect the user experience of an application using a gesture recognition system.

7.2.2 Classification execution time

The time it takes to correctly classify a gesture is the most important thing in a gesture recognition system. This is the direct input of the user and we must strive to keep the time between a user's gesture and the effect of the gesture to a minimum. Recognizing an incorrect gesture happens, this occurres for all systems that use user input that require interpretation. This is generally accepted by the users and does not affect the user experience of a program if not excessively doing mistakes.

When the input starts lagging behind on the other hand, the users tend to describe the systems as sluggish or slow. This goes for all kind of inputs; buttons, touch interfaces or non-standard input devices. As we are checking our scaled linear recognition algorithm(Fig. 6.4) against HMMs(Fig. 7.2), this is one of the major points of interest.

The classification time will be the delta time from the recognition function call to it's completion. This checks all gestures stored in the gesture recognizer and returns the probablitity of a vector sequence being one of the previously learned gestures.

7.2.3 Classification accuracy

When it comes to quality of a temporal pattern recognizer (voice-, gesture-, handwriting recognizer), the accuracy is key. We want to be able to have an accuracy as high as possible to retain the usefulness of the system. It is expected that some of the recognition accuracy will be traded for classification time in our linear recognition algorithm, what we are interested in is *how much.*

Classification scenarios

There are a few scenarios we can encounter when we are measuring accuracy, these are as follows:

True positives are considered a correct classification of a gesture. This is when the system correctly recognizes the buffer data as possible gesture and returns the correct gesture esimate, ie. a left gesture is recognized as a left gesture.

True negatives is a measurement on how well we are handling noise in our buffer. If the user is doing random movement, we do not want to recognize this as a gesture. For example: We have random movement data in our buffer and this might or might not be recognized as a possible gesture, though it is not confirmed to be a gesture by the recognition system. This can be because it has been filtered out as noise and does not get sent to the recognition system or the data is not simliar enough to the learned gestures in the system.

False positives are gestures or noise that are classified falsely. The system classifies the data in the buffer as a gesture existing in the recognition system, although it is not the correct gesture or not a gesture at all, ie. a left gesture or noise is recognized as a downward gesture.

False negatives occurs when the system fails to recognize vector data as a gesture. This is when a user tries to do a left gesture, but the system fails to recognize it and discards it as noise.

Accuracy measurements

There are three measurements we are looking for in term of accuracy: Total accuracy, true positive accuracy, true negative accuracy. This is measured in a percentage, representing the various accuracies. We are interested in the performance on multiple levels, therefore we are calculating the accuracy three ways.

Firstly, we are trying to find the true positive accuracy. This represents how good we can correctly classify gestures passed to the recognition system. It ignores noise data and only focuses on how well we can classify a gesture in the system. See Eq. 7.1.

$$A_{tp} = \frac{N_{tp}}{N_{tp} + N_{fp}} \tag{7.1}$$

We also want to classify how well the system handles false or noisy data. This is to see how well it handles when the user is gesticulating while not trying to interact with the system as well as ignoring gestures that does not exist in our system. The true negative accuracy is calculated as shown in Eq. 7.2.

$$A_{tn} = \frac{N_{tn}}{N - N_{tn} - N_{fn}} \tag{7.2}$$

Finally, the total accuracy descibes the overall accuracy performance of the system. This gives us an overview compared to the A_{tn} and A_{tn} accuracy. Equation 7.3 shows how it is calculated.

$$A_{total} = \frac{N_{tp} + N_{tn}}{N} \tag{7.3}$$

Chapter 8

Optimizing Gesture Recognition

In this section, different methods of improving the performance in a gesture recognizing system will be presented. The purpose of these techniques are either to improve the recognizer's accuracy or calculation performance.

8.1 Noise Reduction

The first step I will be doing with to the data is to reduce the noise. Figure 8.1 illustrates the magnitudes of the component vectors, where vector number 11 shows the noise that is not coherent with the rest of the set and should be removed. There are a few ways to remove these kind of noise spikes, some programs use adaptive noise filtering or algorithms for dynamic noise filtering. As the whole vector sequence of a gesture should be more important than the content(or noise) of one vector, smoothing will be applied instead.

To smooth out the result vector sequence, averaging will be used. The implemented function is able to either do a straint up averaging of the elements

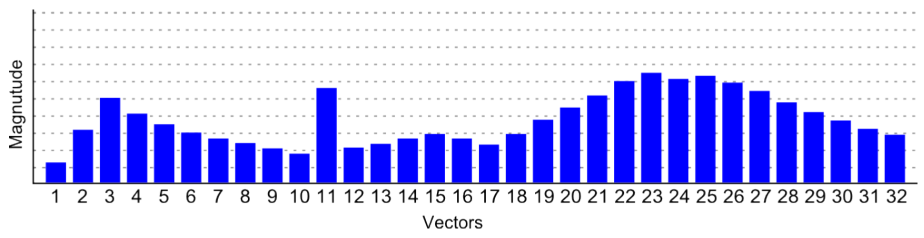

Figure 8.1: Illustration of vector magnitudes.

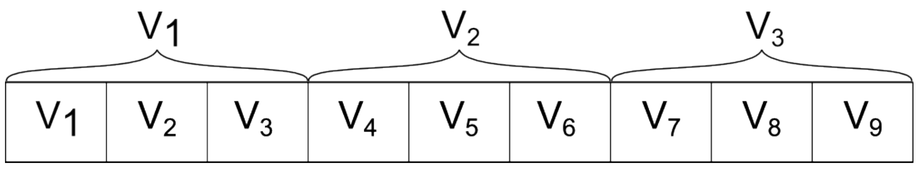

Figure 8.2: Averaging sections of 3 elements with no overlap.

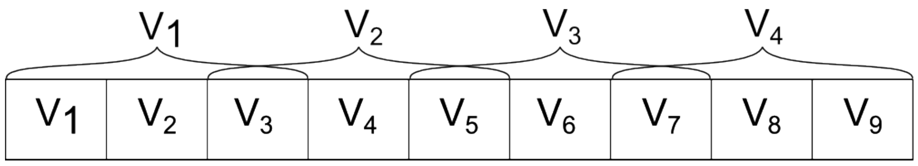

Figure 8.3: Averaging sections of 3 elements with 1 element overlap.

or ose overlaps to retain even more of the information. During the implementation, these numbers are tweaked to best fit the data-set. The tweaking of these values (overlap and chunk size) are dependant on the sample rate of the vectors and should be set depending on the delta time of the recording device.

Figure 8.2 shows the averaging process with no overlap, this reduces the sample size to one third of the original size. As we are canculating each vector, we are not directly removing data, we are generalizing it to remove unwanted spikes that might not be coherent with the rest of the sample set.

Fig. 8.3 however uses a slight overlap to retain even more information of the data set while still smoothing the content of the vector sequence.

8.2 Isolating Gestures

Isolating gestures helps greatly in dealing with noise. Even after the vector sequences has been smoothed, the application still has to handle non-important data. As gestures consist of a set of movement, a quick and simple way of doing this is to look at the magnitudes of the vectors to decide where the useful information is located. This is based on the concept that movement is required to do a gesture. Figure 8.4 illustrates this.

One of the problems with this way of isolating gestures is that it is not adaptive. If there should be a lot of noise and we want to analyze very noiseful data, a lot of information would be sent to the recognition system and the movement would then need to be discarded there instead. This is however very fast, so the time needed by the recognition system to discard

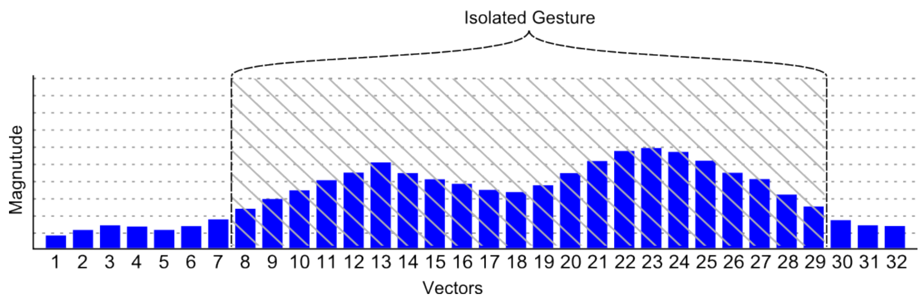

Figure 8.4: Isolating the relevant data of a vector sequence.

the information can easily be gained here.

By isolating gestures like this, the application accepts much less irrelevant data, it trims all data sent to the recognition system (both training data and testing data) so that we are looking at the more relevant parts of the sequence, not the padding on both sides of the movement. It is important that this is implemented in a good way though, as an application looking for gestures would have a buffer that is constantly changing, it should not remove half-complete gestures.

8.3 Optimizing Vector Comparisons

Trigonometric functions are legendary slow to perform and there has been various tricks of speeding them up through time. Earlier, when some languages did not have hardware support for trig. functions, lookup tables were used. This was generated ahead of time to speed up the lookup time of the various angles. When comparing vectors using the dot product (see Eq. 6.1) we are normally using the *inverse cosine* to get the angle between two vectors as we ll as two square roots to calculate the length of each vector as seen in Equation 8.1.

$$l = \sqrt{x^2 + y^2 + z^2} \tag{8.1}$$

As the implementation of the vector comparisons are made for speed, the angle is disregarded and $\cos\theta$ is used instead, removing the need for the *cosine* call. It is also partially possible to remove the need for the square roots as well, this although is more of a semantic problem. As the calculation of the lengths are in need of the square root function this needs to be removed.

One way of removing the need for the length calculation is to extend the dimensions of the vector by one. Normally a three-dimensional vector has three components: *x, y* and *z*. To optimize this, the implementation uses normalized vectors instead. The vectors are now represented with *x, y,z* and *m*, where *m* is the magnitude.

Now instead of needing to calculate the length for both the dot product and the length comparisons, it the magnutude can easily be looked up. This requires that any modification of the optimized vectors gets normalized again and each operation (scaling, multiplication, addition, etc.) are programmed with this in mind.

Chapter 9

Results

In this chapter the results from the implementation and its HMM comparison will be presented. Each step comments on the results found and talks about how the results reflect the changes in the implementation. Furthermore, the last section, 9.5 gives an overlook on how the results evolved step by step.

There are four steps done in the implementation, firstly the system is run using raw data. This is expected to have a low accuracy measure due to noise, no isolation and a simple recognition algorithm.

The second part attempts to smooth out the raw data while reducing noise. Most applications using statistical analysis uses this in some way, especially in audio analysis. The third step is to isolate the gestures, as the implementation is expected to be more precise when analyzing only relevant data.

Finally, the fourth part is done only for the linear recognition implementation as the *jahmm* library has it's own built in comparators for vectors. This step is included to present the possible speedup gained from optimizing vector comparisons.

The tests are run with four gestures in the system: *left, right, up* and *down*. These are tested against ten occurences of six sequences: *left, right, up, down, noise* and *random*, where the *noise* sequences are random movement and the *random* sequences are ther gestures not learned by the system.

All of these steps uses the same configuration and has the same level of strictness to provide a results that are as precise as possible. For details on the numbers generated, see Section 7.

9.1 Step 1: Raw data

The first step uses only raw data, and can be compared to looking for a needle in a haystack. It has no knowledge about what is relevant or not, it is a straight up data comparison.

Accuracy Results	
True Positives	6
True Negatives	20
False Positives	0
False Negatives	34
Atp	1.0
Atn	0.37037036
Atot	0.43333334
Time Results	
Training Time 1	0.0024753ms
Training Time 2	0.0025657ms
Training Time 3	0.0027771ms
Training Time 4	0.0025959ms
Training Time 5	0.0025658ms
Classification Time 1	0.31329786ms
Classification Time 2	0.26877992ms
Classification Time 3	0.30384366ms
Classification Time 4	0.29277154ms
Classification Time 5	0.3536261ms
Average Training Time	0.00259596ms
Average Classification Time	0.306463816ms

Table 9.1: Linear Recognition with raw data

The linear recognition algorithm performed very poorly in this step as expected. The total accuracy is $\sim 43\%$ which mens that less than half of the gestures were treated correctly. The true positives accuracy is very good, while the true negative accuracy is very bad. This shows that the linear algorithm in this step acts very strict and treats many of the actual gestures as noise and its unable to recognize them, most likely due to the noise. The training time however is very good, from the numbers this can be classified as a lazy learner as expected.

Accuracy Results	
True Positives	33
True Negatives	19
False Positives	1
False Negatives	7
Atp	0.9705882
Atn	0.7307692
Atot	0.8666667
Time Results	
Training Time 1	37.0542999ms
Training Time 2	37.614819ms
Training Time 3	37.9955217ms
Training Time 4	37.1280739ms
Training Time 5	36.6714843ms
Classification Time 1	0.1623027ms
Classification Time 2	0.1573281ms
Classification Time 3	0.15215426ms
Classification Time 4	0.15322888ms
Classification Time 5	0.15839668ms
Average Training Time	37.29283976ms
Average Classification Time	0.156682124ms

Table 9.2: HMM Recognition with raw data

The recognition system using hidden Markov models on the raw data performed suprisingly well. HMMs are known for performing well when used to recognize patterns and this can clearly be seen from the results even with very noisy data tha has not been isolated in any way, the HMMs are able to find patterns matching the gestures.

In regards to training time, the HMMs performed much worse than the linear recognition algorithm. The linear recognition algorithm trains $\sim$ 144% faster, which is can be explained by what the algorithms actually does when training. The linear recognition algorithm at this point is simply moving the training data to it's buffers, not handling it at all while the HMM is being generated by the training data, resulting in the actual model.

This is also reflected in the classification time: The HMM system classifies the data twice as fast, as it does not need to check the input data against any other sequences like the linear learning algorithm. The sequence is run through the HMMs of the different gestures which is a lot faster than checking the input sequence against all known sequences.

The system is still unable to handle the most noiseful sequences, treating them as pure noise instead.

9.2 Step 2: Noise Reduction

The second step is to remove the noise from the vector sequencese through smoothing. This results in a smaller sample size, but the information is retained in the remaining averaged vectors.

Accuracy Results	
True Positives	26
True Negatives	13
False Positives	7
False Negatives	14
Atp	0.7878788
Atn	0.4814815
Atot	0.65
Time Results	
Training Time 1	0.17227ms
Training Time 2	0.1663234ms
Training Time 3	0.1569357ms
Training Time 4	0.1629427ms
Training Time 5	0.1650557ms
Classification Time 1	0.1977951ms
Classification Time 2	0.1712558ms
Classification Time 3	0.18677732ms
Classification Time 4	0.16640798ms
Classification Time 5	0.2037236ms
Average Training Time	0.16470556ms
Average Classification Time	0.18519196ms

Table 9.3: Linear Recognition with noise reduction

As seen in Table 9.3, the linear gesture recognition algorithm now recognizes a lot more gestures. The true positive accuracy has lowered, but there are now 20 more gestures correctly recognized. The number of false positives has also increased, this might be because the data is now more generalized. Overall, total accuracy has increased to $\sim 65\%$, performing much better.

The training time is much longer now, as the data is actually being handled in the training process, not just bein copied. The one feature that might be suprising is that the classification time has been reduced, the explanation for this is that the data set is now a lot smaller, resulting in fewer required comparisons.

Accuracy Results	
True Positives	40
True Negatives	11
False Positives	9
False Negatives	0
Atp	0.81632656
Atn	1.0
Atot	0.85
Time Results	
Training Time 1	24.1008119ms
Training Time 2	23.5388439ms
Training Time 3	22.8029154ms
Training Time 4	22.0887207ms
Training Time 5	22.8647359ms
Classification Time 1	0.23431384ms
Classification Time 2	0.14797656ms
Classification Time 3	0.14964886ms
Classification Time 4	0.12955722ms
Classification Time 5	0.16585256ms
Average Training Time	23.07920556ms
Average Classification Time	0.156682124ms

Table 9.4: HMM Recognition with noise reduction

The HMM recognition's total accuracy has now decreased a light amout. This might seem as it is performing worse but looking at the other accuracies, it now has no false negatives meaning that it handles more gestures as actual gestures, not discarding them as noise. The true positives accuracy is still quite good ($\sim$ 82%), still after the data has been generalized to this extent.

While the classification time has not changed to a notable extent(due to the classification being done in the HMMs, no direct comparisons), the classification time has been cut in half. This is becuase of the same reason as the classification time of the linear recognition algorithm improving: there is less content in the vector sequences.

9.3 Step 3: Gesture isolation

In the third step of the tests, the gestures(both training- and test gestures) are being isolated (see Section 8.2 for details). This further reduces the size of the vector sequences, trimming them by checking the magnitude and removing the outlaying vectors that does not pass over the treshold.

It is expected that this step greatly improves the performance of both the HMM and the linear recognition system, especially the latter as it up until now has been checking the outlaying irrelevant vectors of both the training- and test sequences.

Accuracy Results	
True Positives	39
True Negatives	18
False Positives	2
False Negatives	1
Atp	0.9512195
Atn	0.94736844
Atot	0.95
Time Results	
Training Time 1	0.2285967ms
Training Time 2	0.2139565ms
Training Time 3	0.2179712ms
Training Time 4	0.2181222ms
Training Time 5	0.2231632ms
Classification Time 1	0.0993594ms
Classification Time 2	0.10054268ms
Classification Time 3	0.13603508ms
Classification Time 4	0.09548354ms
Classification Time 5	0.13708553ms
Average Training Time	0.22036196ms
Average Classification Time	0.113701246ms

Table 9.5: Linear Recognition with gesture isolation

The linear recognition system has now reached a total accuracy of $\sim 95\%$, higher than ever expected. During the first tests, this was believed to be incorrect so the test data was scrapped and more was added, this ended up performing even better, resulting in these results. The linear recognition algorithm is now able to correctly classify $\sim 95\%$ of the gestures and discarding the same amount of incorrect/noise gestures.

The training time has increased, due to the additional handling of the data. Again, the numbers show that the classification time has been reduced: again due to the fact that there are fewer checks to be made.

Accuracy Results	
True Positives	40
True Negatives	18
False Positives	2
False Negatives	0
Atp	0.95238096
Atn	1.0
Atot	0.96666664
Time Results	
Training Time 1	15.6299071ms
Training Time 2	16.3723254ms
Training Time 3	15.3566657ms
Training Time 4	15.4031215ms
Training Time 5	15.6687863ms
Classification Time 1	0.08068046ms
Classification Time 2	0.06163928ms
Classification Time 3	0.0657506ms
Classification Time 4	0.06750136ms
Classification Time 5	0.0574797ms
Average Training Time	15.6861612ms
Average Classification Time	0.06661028ms

Table 9.6: HMM Recognition with gesture isolation

The HMM recognition system in step 3 recognized all gestures sent to the system, while still allowing only two other gestures to pass as actual gestures. The additional accuracy comes from better classifying the sequences, there are now much less clutter and irrelevant data that ends up being both trained and attempted recognized by the HMMs.

In terms of training- and classification time, both have decreased. The training time reduction because of the now notably smaller data set and the recognition time because the clutter has been removed so the HMMs are cleaner.

9.4 Step 4: Optimizing vector comparison

The fourth and last step is done to improve the computational performance of the linear recognition system. This step has no effect on the HMM recognition system, as the *jahmm* library does not directly compare vectors. The accuracy of the results should remain unchanged.

Accuracy Results	
True Positives	39
True Negatives	18
False Positives	2
False Negatives	1
Atp	0.9512195
Atn	0.94736844
Atot	0.95
Time Results	
Training Time 1	0.2970579ms
Training Time 2	0.3161353ms
Training Time 3	0.3096756ms
Training Time 4	0.2945223ms
Training Time 5	0.3047553ms
Classification Time 1	0.073556619ms
Classification Time 2	0.072083559ms
Classification Time 3	0.070532ms
Classification Time 4	0.07391884ms
Classification Time 5	0.0718964ms
Average Training Time	0.30442928ms
Average Classification Time	0.0723974836ms

Table 9.7: Linear Recognition with gesture isolation

From the results in Table 9.7 it can be seen that the training time has increased while the classification time has decreased. The system now uses constantly normalized vectors and as these are normalized when creating new vectors, it is expected that creating them are slower. The comparisons are now avoiding the *cosine* and *square root* functions, resulting in a quicker comparison.

9.5 Step-by-step comparison

This section shows the change in the accuracy and computational performance through steps for easy comparison.

Step	Training time	Classification time	Atp	Atn	Atot
Step 1					
Linear	0.0026ms	0.3064ms	1.0	0.3704	0.4333
HMM	37.2928ms	0.1566ms	0.9706	0.7308	0.8667
Step 2					
Linear	0.1647ms	0.1851ms	0.7879	0.4815	0.65
HMM	23.0792ms	0.1566ms	0.81632656	1.0	0.85
Step 3					
Linear	0.2203ms	0.1137ms	0.9512	0.9474	0.95
HMM	15.68ms	0.0666ms	0.9524	1.0	0.9667
Step 4					
Linear	0.3044ms	0.0723ms	0.9512	0.9474	0.95

Table 9.8: Step-by-step comparison of the test results.

Chapter 10

Evaluation and Future Work

This chapter evaluates the results summarized in chapter 9, adresses the research questions and discusses future work.

10.1 Accuracy results

First of all, the accuracy obtained by the linear recognition algorithm was suprisingly good. It reached 95% during in the final step. The results show that the linear recognition algorithm was very dependant on having clean isolated gestures. This makes sense as the algorithm treats the sequences as entire gestures so it needs to be assisted in telling what is a gesture and what is not. The noise reduction by doing smoothing turned out to be good as the shape of the gesture remains the without loosing important information.

In the earlier stages especially when testing with raw data, the linear recognition algorithm didn't do much better than guessing. From the following results I deduce that this is because of the importance placed on each vector, when it then hits a vector affected by noise, the results get diminished. By smoothing, the vectors get more connected. They now follow the macro movement of the user instead of the micro movements.

The HMM performed suprisingly well in all conditions, it managed to detect patterns in all three stages. This means that using a HMM for temporal pattern recognition requires very little work as it can run reasonably well even with raw data. It seems (after step 2) that the HMMs prefers as much data as possible, which makes sense as it is a statistical model.

It is expected that as the number or gestures in the system increases, the

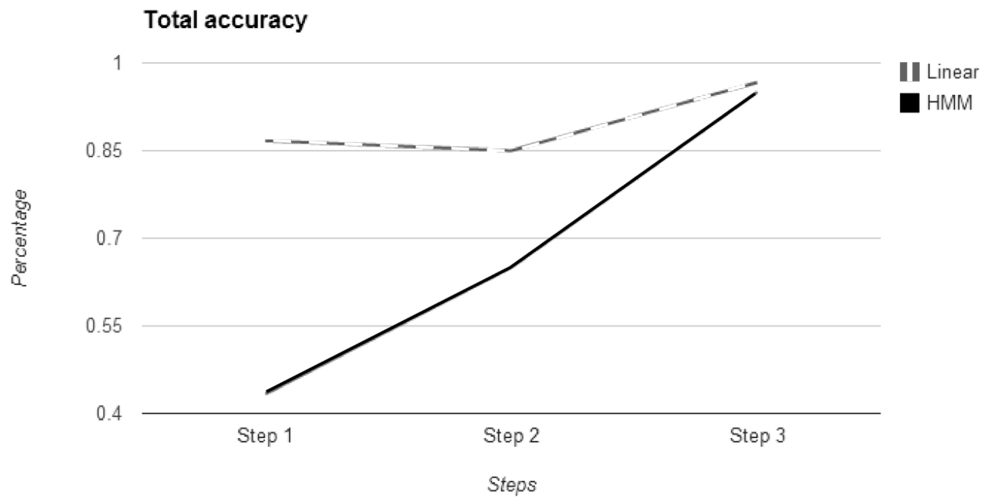

Figure 10.1: The total accuracy of the linear recognition algorithm and HMMs.

accuracy will decrease. Figure 10.1 shows the changes in the total accuracy between each step.

10.2 Computational Performance Results

The training times speak for themselves, they were so different for the two algorithms that the only way of combining them in one graph would have been to use a logarithmic scale. The HMM training times ranged from 37ms to 15 ms, while the linear recognition algorithm ranged from 0.003ms to 0.22ms(0.3ms in step four).

While the linear algorithm beat the HMMs in training time, the HMMs performed better than the linear algorithm in all steps. The only step that came close was the optimization step(step 4), even then the HMM performed a little better. This would apply ever more to larger data-sets, as the data only need to be sent through the HMM once for each gesture, while the linear one needs to check against all instances of the known gestures.

If there are a lot of gestures in a system the training times for the HMMs would increase even more, this is a one time load though and the HMMs require very little memory as only the resulting model is stored for later use.

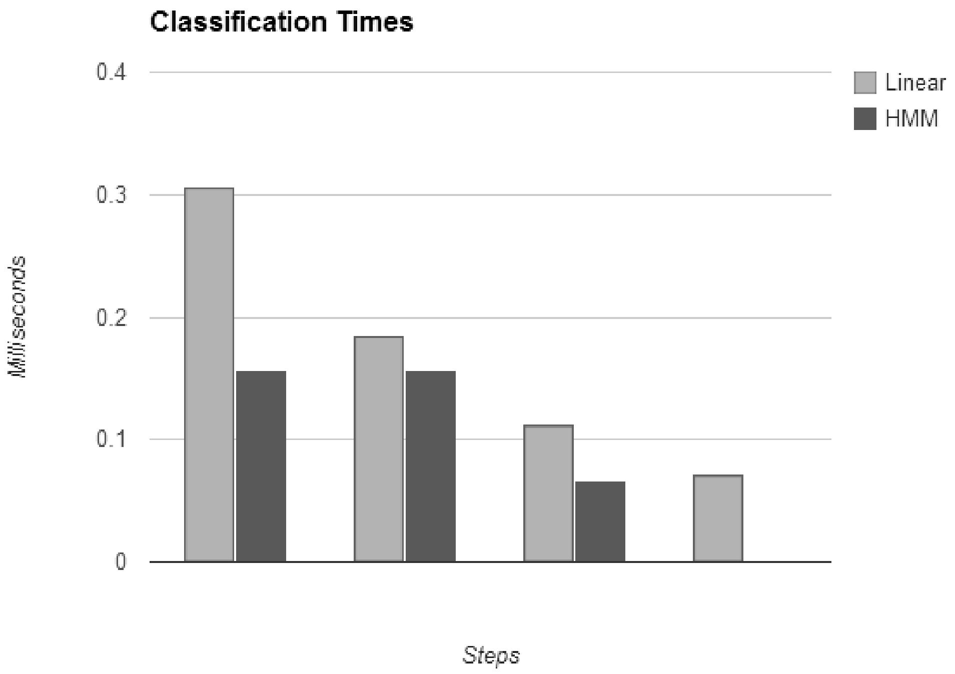

Figure 10.2: The classification times through each step.

The results also showed that optimizing the vector comparison for the linear system improved the classification time by 36%, clearly showing the large computational cost of trigonometric functions.

10.3 Research Goal Answers

This section answers and discusses the research questions.

10.3.1 RQ 1

How does a simple gesture recognition algorithm's accuracy compare to that of hidden Markov models?

The results in Chapter 9 shows that when the data is cleaned and the gestures are isolated in a good way, the perfomance of a linear recognition algorithm can be compared to that of HMMs.

I suspect though that when the trained gesture set becomes larger, HMMs would perform better if the data set contains data from accelerometers. This is because the vectors appears very simliar due to the gravitational force, if using spatial data however where the positions have their origin in *0, 0, 0* rather than *0, -9.81, 0* the performance could still compete with that of HMMs as there would be a greater difference between the elements. For more specific data, see Section 9.5.

The end results were:
Simple algorithm: 95%
Hidden Markov Model: 96.67%

10.3.2 RQ 2

How does the computational performance of a simple gesture recognition algorithm compare to that of hidden Markov models?

The results found shows that HMMs perform a little better than the linear recognition algorihm when it comes to classification times. The training times however were much lower for the linear recognition algorihm.

It comes down to how a developer wants to apply the algorithms. If the system should be able to load on demand, using HMMs would be a bad idea as they require a lot of time for their training process. When the training process is over however, the HMMs would take up less memory and would generally be faster when classifying gestures. For more specific data, see section 9.5.

The end times for training:
Simple algorithm: 0.3ms
Hidden Markov Model: 15.68ms

The end times for classification:
Simple algorithm: 0.07ms
Hidden Markov Model: 0.06ms

10.3.3 RQ 3

What techniques can we apply to the gesture recognition system to optimize it's computational performance and accuracy?

In the implementation there are mainly three techniques used: *Noise Reduction*, *Gesture Isolation* and *Vector Comparison Optimization*. There are of course many other techniques that can be applied to solve the same problems as well as further improving the data-sets.

The steps are explained in detail in Chapter 8.

10.3.4 RQ 4

How does the techniques in RQ 1.3 affect the perfomances of the two algorithms?

Table 10.1 shows the accuracies obtained from the raw data and after applying *Noise Reduction* and *Gesture Isolation*.

Step	Linear	HMM
1	43.33%	86.67%
2	65%	85%
3	95%	96.67%

Table 10.1: Total accuracy for the linear recognition algorithm and HMM.

The *Vector Comparison Optimization* increased the classification time by 36%.

Refer to Section 9.5 for more details.

10.3.5 RQ 5

What benefits and tradeoffs can we identify in the developed gesture recognition system when using the simple gesture recognition algorithm?

After implementing the linear gesture recognizer and comparing it to the same system using HMMs, I found that the linear version had some benefits over the HMM version. These are mostly when it comes to computational performance. The linear version relied very on handling the data correctly while the version using HMMs was much more forgiving towards the data-sets. The linear system can however handle small data-sets compared to the HMM system which requires a larger data-set to be able to train the model.

Benefits

- Easy to implement.
- Much faster learning time.
- Handles small data-sets.
- Able to look at a gesture as a whole.

Tradeoffs

- Vulnerable to noisy data.
- Requires isolated gestures.
- Classification time just slightly better than that of HMMs.

- Slows down significantly when testing large data-sets.
- Not as accurate as HMMs.

The linear solution is valid for some implementations where the developer wants something simple and fast. If the developer wants a solution that is very reliable and robust, HMMs would perform better.

10.3.6 RQ 6

Based in the findings is RQ 1-5, are the vector components of a gesture coherent enough to be used as a single token?

From the performance of the linear recognition system, it seems that a gesture can be used as a single token. The individual vector data is connected enough that the movement can be considered coherent. The reasoning behind this is that there would be no gaps in a gesture, the movement is connected throughout the gesture.

This means that a gesture from left to right could be tokenized as {*begin movement right, stop movement right*}. What happens between these state changes when using temporal data could possibly be disregarded in some systems as they have no useful information except from orientation information.

Translating this to spatial data, this would result in a vector describing the total movement along a direction. Chaining these should also be possible, having one vector right and one vector down for the *right and down* gesture.

An implementation could take advantage of this by using individual accelerometer vector samples to record any significant changes instead, treating every directional or magnitude change as an event then disregarding the vector samples after the event has been registered.

To make a conclusive statement if this is true or false more research in the area is needed.

10.4 Project evaluation

Developing the project in Java worked well, as expected and the implementation can easily be made into a library for use in other applications. The end result of the implementation differs a bit from it's class digram (see Fig.

6.1) as a result of making the application more modular. I will definitely be using this implementation to develop an application for Android at a later time, as the results were very good.

By doing this implementaion in CC++ instead, I expect that it would be even faster as there are some sub-optimized sections of the code where a lot of new memory is assigned for the vector sequences. This could be avoided in CC++ by managing the memory myself. The gains from using Java is of course that the development time is much shorter.

Personally,I found the results to be suprisingly good, and this sparks an interest in continuing to improve on the system in the future.

Chapter 11

Conclusion

The results found from this thesis shows that a linear recognition algorithm can compete with that of Hidden Markov Models. The linear solution was found to be very dependant on the input data being clean and isolated to be able to perform well. It did however outperform HMMs in terms of training time and the classification times was very close.

Though a linear recognition algorithm performs reasonably well, Hidden Markov Models are still more reliable and more forgiving of the data-sets it analyzes. As the number of gestures in the system increases, it is expected that the accuracy of the HMMs will outperform the linear recognition algorithm even more.

The final conclusion from this thesis is that even a very simple recognition algorithm can perform nearly as well as more complex ones if the data-sets are presented well to the system.

Chapter 12

Future Work

This chapter presents some suggested future work as well as remaining work to be able to use a linear gesture recognition system instead of Hidden Markov Models in an enterprise solution.

Further research to conclude if gestures are coherent enough to be used as a single token

As this research was unable to conclude if RQ 6 is true or false, further research should be done to determine this. If it is true, this means that recognition systems can be based on this fact, resulting in a great decrease in computational cost while still retaining the accuracy a gesture recognition system needs.

It is required that this is tested for multiple systems to ensure that the conclusion is correct.

Applying a linear recognition algorithm for spatial data

Using the linear recognition algorithm for temporal data resulted in good results. If the data is required to be temporal could be an interesting subject to look into. This would require a different input device able to record spatial information. Some possibilities are: *Kinect*, *Playstation Move* or *image recognition*.

Further optimizations to improve performance

As this thesis only covers three techniques for improving the computational time and accuracy, more techniques could be researched to find an optimal mix of techniques to achieve a resulting system that performs as good as possible.

Simliar solution for sound analysis

Voice recognition uses simliar techniques for determining the words spoken by a user. Would a linear recognition system work for this kind of temporal pattern recognition?

Bibliography

[1] Wilson, A. D., & Bobick, A. F. (1999). Parametric hidden markov models for gesture recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 21(9), 884-900.

[2] Chen, F. S., Fu, C. M., & Huang, C. L. (2003). Hand gesture recognition using a real-time tracking method and hidden Markov models. Image and Vision Computing, 21(8), 745-758.

[3] Starner, T., & Pentland, A. (1997). Real-time american sign language recognition from video using hidden markov models. In Motion-Based Recognition (pp. 227-243). Springer Netherlands.

[4] Caldiera, V. R. B. G., & Rombach, H. D. (1994). *The goal question metric approach.* Encyclopedia of software engineering, 2, 528-532.

[5] Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). *Design science in information systems research.* MIS quarterly, 28(1), 75-105.

[6] Azuma, Ronald T. "A survey of augmented reality." Presence-Teleoperators and Virtual Environments 6.4 (1997): 355-385.

[7] Layar. *What is Layar?* Layar. Retreived date: 8. May 2013. From `http://www.layar.com/what-is-layar/`

[8] Wikitude GmbH. *Wikitude- The App.* Wikitude. Retreived date: 9. May 2013. From `http://www.wikitude.com/app/`

[9] Hirsch, H. G., & Pearce, D. (2000). *The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions.* In ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW).

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

[11] Baum, L. E., Petrie, T. (1966). *Statistical inference for probabilistic functions of finite state Markov chains*. The annals of mathematical statistics, 37(1), 1554-1563.

[12] Rabiner, L. R. (1989). *A tutorial on hidden Markov models and selected applications in speech recognition*. Proceedings of the IEEE, 77(2), 257-286.

Appendix A

Conversation program

Conversion program from raw data to jahmm readable data in Python.

```
import sys
import math

if len(sys.argv) < 3 :
    print("Not enough arguments")
    print("Usage: <output name> <input file 1> <input file 2> ...")
    exit()

out = open(sys.argv[1], 'w')

for i in range(2, len(sys.argv)) :
    print("Handling file: " + sys.argv[i])
    f = open(sys.argv[i])
    for line in f :
        a = line.strip('\n')
        b = a.partition(":")
        xyz = b[2].rsplit(",")
        x = xyz[0]
        y = xyz[1]
        z = xyz[2]
        out.write("[ " + x + " " + y + " " + z + " ] ;  ")
    f.close()
    out.write('\n')
out.close()
```

Appendix B

Raw Data File

```
1370451316888:-0.6724017,9.102871,-0.8795709
1370451316906:-0.51912045,9.601034,-1.3010944
1370451316917:-0.51912045,9.524394,-1.1478131
1370451316945:-0.6724017,9.524394,-1.2244537
1370451316957:-0.8640033,9.562715,-1.377735
1370451316984:-0.8640033,9.4860735,-1.377735
1370451317002:-0.710722,9.4860735,-1.3394147
1370451317016:-0.5957611,9.4860735,-1.262774
1370451317036:-0.6724017,9.4860735,-1.1861334
1370451317055:-0.7873627,9.447753,-1.3010944
1370451317075:-0.7873627,9.409433,-1.4543756
1370451317096:-0.6724017,9.371113,-1.5310162
1370451317116:-0.6340814,9.371113,-1.4926959
1370451317136:-0.710722,9.447753,-1.4926959
1370451317155:-0.710722,9.409433,-1.5310162
1370451317175:-0.74904233,9.371113,-1.5693365
1370451317196:-0.74904233,9.409433,-1.5693365
1370451317216:-0.710722,9.409433,-1.4160553
1370451317235:-0.5574408,9.217832,-1.3010944
1370451317255:-0.36583924,8.872949,-1.2244537
1370451317275:-0.17423767,8.259824,-0.99453187
1370451317296:0.09400452,7.186855,-0.5730084
1370451317316:0.43888733,5.5390816,0.0017962647
1370451317336:0.20896545,3.546425,0.15507752
1370451317355:-0.6340814,1.936972,-0.036524046
1370451317377:-1.1322455,0.825683,0.0017962647
1370451317396:-0.825683,-0.13232483,0.6915619
1370451317416:-0.13591737,-1.5501764,1.8411713
1370451317436:0.20896545,-3.4278717,2.7608588
1370451317455:-0.059276734,-5.573809,3.1440618
```

```
1370451317476:-0.6724017,-8.10295,3.4889448
1370451317496:-1.2088861,-9.597442,3.7955072
1370451317515:-0.6724017,-8.217911,3.5655854
1370451317535:1.5118561,-3.0446687,3.4123042
1370451317555:4.46252,5.883964,4.1787105
1370451317576:5.535489,14.889237,6.0564055
1370451317596:3.1213093,18.491346,9.236992
1370451317616:0.13232483,18.75959,14.027031
1370451317635:-1.2088861,18.75959,17.437538
1370451317655:-1.6304096,18.75959,16.249609
1370451317675:-1.7453705,18.75959,12.034374
1370451317696:-1.1705658,18.75959,7.81914
1370451317716:-0.44247985,18.184784,5.36664
1370451317735:0.20896545,16.153809,4.9451165
1370451317755:1.0903326,13.586347,5.711523
1370451317775:1.5501764,11.708652,6.1713667
1370451317796:1.0903326,9.869277,5.749843
1370451317816:0.55384827,8.0299015,5.021757
1370451317835:0.55384827,6.956933,4.3703117
1370451317855:0.78377014,6.841972,4.0254292
1370451317876:0.9753717,7.2251754,3.8338275
1370451317896:1.2436138,7.570058,3.9871087
1370451317916:1.4735358,7.8766203,4.14039
1370451317935:1.7417779,8.259824,4.2170305
1370451317955:1.9333795,8.719667,4.523593
1370451317975:1.7034576,8.987909,4.830156
1370451317997:1.2819343,8.911269,4.9451165
1370451318016:0.9753717,8.719667,4.830156
1370451318035:1.013692,8.681347,4.4469523
1370451318055:1.2819343,8.834628,4.0637493
1370451318075:1.4352155,8.949589,3.8721478
1370451318096:1.3968952,8.872949,3.8338275
1370451318116:1.3968952,8.719667,3.7955072
1370451318135:1.3585749,8.719667,3.8338275
1370451318155:1.3968952,8.757988,3.8721478
1370451318175:1.3968952,8.796308,3.8338275
1370451318196:1.2436138,8.796308,3.7955072
1370451318216:1.1286529,8.834628,3.9871087
1370451318235:1.0520123,8.872949,4.14039
1370451318255:1.013692,8.796308,4.293671
1370451318275:1.0903326,8.681347,4.2170305
1370451318296:1.0903326,8.604707,3.9871087
1370451318316:1.0520123,8.489745,3.8721478
```

Appendix C

Converted Data File

```
[ -0.6724017 9.102871 -0.8795709 ] ;  [ -0.51912045 9.601034 -1.3010944 ] ;
[ -0.51912045 9.524394 -1.1478131 ] ;  [ -0.6724017 9.524394 -1.2244537 ] ;
[ -0.8640033 9.562715 -1.377735 ] ;  [ -0.8640033 9.4860735 -1.377735 ] ;
[ -0.710722 9.4860735 -1.3394147 ] ;  [ -0.5957611 9.4860735 -1.262774 ] ;
[ -0.6724017 9.4860735 -1.1861334 ] ;  [ -0.7873627 9.447753 -1.3010944 ] ;
[ -0.7873627 9.409433 -1.4543756 ] ;  [ -0.6724017 9.371113 -1.5310162 ] ;
[ -0.6340814 9.371113 -1.4926959 ] ;  [ -0.710722 9.447753 -1.4926959 ] ;
[ -0.710722 9.409433 -1.5310162 ] ;  [ -0.74904233 9.371113 -1.5693365 ] ;
[ -0.74904233 9.409433 -1.5693365 ] ;  [ -0.710722 9.409433 -1.4160553 ] ;
[ -0.5574408 9.217832 -1.3010944 ] ;  [ -0.36583924 8.872949 -1.2244537 ] ;
[ -0.17423767 8.259824 -0.99453187 ] ;  [ 0.09400452 7.186855 -0.5730084 ] ;
[ 0.43888733 5.5390816 0.0017962647 ] ;  [ 0.20896545 3.546425 0.15507752 ] ;
[ -0.6340814 1.936972 -0.036524046 ] ;  [ -1.1322455 0.825683 0.0017962647 ] ;
[ -0.825683 -0.13232483 0.6915619 ] ;  [ -0.13591737 -1.5501764 1.8411713 ] ;
[ 0.20896545 -3.4278717 2.7608588 ] ;  [ -0.059276734 -5.573809 3.1440618 ] ;
[ -0.6724017 -8.10295 3.4889448 ] ;  [ -1.2088861 -9.597442 3.7955072 ] ;
[ -0.6724017 -8.217911 3.5655854 ] ;  [ 1.5118561 -3.0446687 3.4123042 ] ;
[ 4.46252 5.883964 4.1787105 ] ;  [ 5.535489 14.889237 6.0564055 ] ;
[ 3.1213093 18.491346 9.236992 ] ;  [ 0.13232483 18.75959 14.027031 ] ;
[ -1.2088861 18.75959 17.437538 ] ;  [ -1.6304096 18.75959 16.249609 ] ;
[ -1.7453705 18.75959 12.034374 ] ;  [ -1.1705658 18.75959 7.81914 ] ;
[ -0.44247985 18.184784 5.36664 ] ;  [ 0.20896545 16.153809 4.9451165 ] ;
[ 1.0903326 13.586347 5.711523 ] ;  [ 1.5501764 11.708652 6.1713667 ] ;
[ 1.0903326 9.869277 5.749843 ] ;  [ 0.55384827 8.0299015 5.021757 ] ;
[ 0.55384827 6.956933 4.3703117 ] ;  [ 0.78377014 6.841972 4.0254292 ] ;
[ 0.9753717 7.2251754 3.8338275 ] ;  [ 1.2436138 7.570058 3.9871087 ] ;
[ 1.4735358 7.8766203 4.14039 ] ;  [ 1.7417779 8.259824 4.2170305 ] ;
[ 1.9333795 8.719667 4.523593 ] ;  [ 1.7034576 8.987909 4.830156 ] ;
[ 1.2819343 8.911269 4.9451165 ] ;  [ 0.9753717 8.719667 4.830156 ] ;
[ 1.013692 8.681347 4.4469523 ] ;  [ 1.2819343 8.834628 4.0637493 ] ;
[ 1.4352155 8.949589 3.8721478 ] ;  [ 1.3968952 8.872949 3.8338275 ] ;
[ 1.3968952 8.719667 3.7955072 ] ;  [ 1.3585749 8.719667 3.8338275 ] ;
[ 1.3968952 8.757988 3.8721478 ] ;  [ 1.3968952 8.796308 3.8338275 ] ;
[ 1.2436138 8.796308 3.7955072 ] ;  [ 1.1286529 8.834628 3.9871087 ] ;
[ 1.0520123 8.872949 4.14039 ] ;  [ 1.013692 8.796308 4.293671 ] ;
[ 1.0903326 8.681347 4.2170305 ] ;  [ 1.0903326 8.604707 3.9871087 ] ;
[ 1.0520123 8.489745 3.8721478 ] ;  [ 1.0903326 8.489745 3.8338275 ] ;
```