SigVerify
Real-Time Mobile Gesture-Based Authentication

Computer Science Tripos, Part II
St John’s College
May 11, 2014
Proforma

Name: Yunjae Lee
College: St John’s College
Project Title: SigVerify: Real-Time Mobile Gesture-Based Authentication
Examination: Computer Science Tripos, Part II, June 2014
Word Count: 11,742 words
Project Originator: Prof. Ross Anderson
Project Supervisor: Prof. Ross Anderson and Laurent Simon

Original Aims of the Project

The primary goal of the project is to design and implement a secure and usable gesture-based authentication mechanism. This entails carefully studying the biometric data in question and applying efficient classification algorithms to achieve reliable performance. The project’s original extent was to implement host-based authentication on the server and have individual devices communicate with it. As an optional extension, a standalone mobile authentication system was considered if development of a networked scheme proceeded well.

Work Completed

All major objectives have been met successfully. I implemented three authentication algorithms, one of which attained an Equal Error Rate of less than 2.5%. The performance of classifiers was thoroughly tested by crossvalidation with various combinations of features and other parameters. In addition to the Authentication-over-Network scheme, a fully-functional mobile authentication framework was designed on top. Robust evaluation of security and usability was carried out with special attention to the system’s usability as mobile software.

Special Difficulties

None.
Declaration

I, Yunjae Lee of St John’s College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed

Date May 11, 2014
Acknowledgements

Many people have contributed to this project in various ways, but I am particularly grateful to:

- **Prof. Ross Anderson**, for allowing me to take on this interesting project and for his general yet far-reaching advices.

- **Laurent Simon**, for his insightful ideas and very down-to-earth guidelines he provided throughout the entire process.

- People who kindly participated in data collection and user studies.
# Contents

1 Introduction

1.1 Motivation and Background .................................................. 1
1.2 Previous Work ........................................................................... 3
1.3 Overview of the Project ............................................................ 3
1.4 Challenges ................................................................................. 5

2 Preparation

2.1 Introduction to Pattern Classification ......................................... 7
  2.1.1 Binary Classification and Decision Theory ............................ 8
  2.1.2 Evaluation of a Binary Classifier ......................................... 9
  2.1.3 Training and Testing .......................................................... 11
2.2 Design Cycle of Pattern Recognition Systems ............................... 11
2.3 Threat Model and Mobile Security ............................................. 12
2.4 Requirements Analysis and Approach ....................................... 13
2.5 Data Collection and Formative Evaluation .................................. 14
  2.5.1 Formative Evaluation .......................................................... 14
  2.5.2 Data Collection ................................................................. 15
2.6 Choice of Tools .......................................................................... 16
  2.6.1 Programming Language .................................................... 16
  2.6.2 Additional Tools Used ....................................................... 17
  2.6.3 Development Environment ............................................... 17
2.7 Software Engineering Approach ................................................ 18
2.8 Summary ................................................................................... 18

3 Implementation

3.1 System Overview ........................................................................ 20
  3.1.1 High-level Workflow ......................................................... 20
  3.1.2 Data Structures ................................................................. 21
3.2 Preprocessing ............................................................................ 22
  3.2.1 Removing the Gravitational Acceleration ............................. 23
  3.2.2 Interpolation ...................................................................... 24
  3.2.3 Smoothing ........................................................................ 25
3.3 Segmentation ............................................................................. 26
3.4 Feature Extraction ................................................................. 27
## B Project Proposal

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Introduction and Description of the Work</td>
<td>82</td>
</tr>
<tr>
<td>B.2 Starting Point</td>
<td>83</td>
</tr>
<tr>
<td>B.3 Structure and Substance of the Project</td>
<td>83</td>
</tr>
<tr>
<td>B.4 Success Criteria</td>
<td>85</td>
</tr>
<tr>
<td>B.5 Resources Declaration</td>
<td>86</td>
</tr>
<tr>
<td>B.6 Work Timetable</td>
<td>86</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation and Background

Mobile devices are common in our everyday activities. We use them not merely for making phone calls, but also in various activities such as social networking, multimedia applications and financial transactions. Increasing performance and memory of mobile devices lend themselves to be used for interactive and engaging applications. As a result, mobile devices nowadays contain a considerable amount of personal and sensitive information. According to one research [19], the world mobile payments value amounted to $256 billion in 2012 and is expected to triple to $796 billion by 2014. Figure 1.1 shows the growth of the number of mobile payments over recent years.

![Figure 1.1: Number of global mobile payment transactions (billion), 2010-2014F. Source: World Payments Report 2013 [19].](image-url)
CHAPTER 1. INTRODUCTION

This phenomenon calls for a robust and convenient mechanism for authenticating mobile users. Unfortunately, current mobile identification schemes such as PINs, passwords and graphical patterns are limited at best, each having its own drawback. PINs and pattern locking mechanisms have small keyspaces, and therefore can be easily brute forced should a malware be planted in the device. Passwords are difficult to memorise and prone to typos, and also slow to enter on a touchscreen keyboard. Perhaps most importantly, all of these schemes are vulnerable to physical observation as they can easily be compromised by “shoulder surfing”. Hence, it seems that a more appropriate authentication mechanism is needed for mobile device users.

![Figure 1.2: Examples of current mobile authentication mechanisms.](image)

Biometric authentication identifies people based on their natural traits such as behaviour or physiology [2]. This is a very desirable feature as users do not have to memorise a long and complicated phrase made up of arbitrary characters. The biometric modality studied in this project is **signature**. Handwritten signature is an authentication method tried and tested throughout centuries. A key merit of signature is that it is commonly accepted as carrying a legal commitment of intent, making it particularly amenable to mobile financial applications.

For an authentication process to be nonintrusive, it is crucial that users need not carry extra hardware whenever they need to authenticate. Thus the natural choice is to use the sensors equipped in mobile devices. A typical smartphone contains a variety of sensors such as accelerometer, gyroscope, light and proximity sensor. Three-dimensional accelerometers and gyroscopes have commonly been exploited in previous work [13, 32] and are both studied in this project.
1.2 Previous Work

Significant work has been conducted on handwritten signature recognition primarily in two categories: online and offline. Offline signature verification takes place after the signature has been written. A scanned image of the signature is analysed. On the other hand, online signature verification involves collecting the signature data to be verified, often by using an electronic tablet. This captures not only the contours of the signature but also dynamic features such as velocity, acceleration, and pressure applied for each part of the signature.

In offline signature verification, Sigari et al. proposed a novel method which extracts features from signatures based on Gabor wavelet transform [37]. Another interesting study by Frias-Martinez et al. compared the system using Support Vector Machines (SVMs) with another using Multi-Layer Perceptrons [12]. Their results showed that SVMs perform better in both of the two feature extraction approaches studied.

Similarly, much work has been carried out in online signature recognition for the past twenty years. In fact, a number of public evaluations compared the performance of different classifiers: the Signature Verification Competition (SVC) [38] in 2004, the Signature Competition of the BioSecure Multimodal Evaluation Campaign (BMEC) in 2007 [24] and, most recently, the ICDAR Signature Verification Competition in 2009 [5]. Competing systems participated in numerous tasks with varying environments and constraints, and their performance was ranked in each cases. Two of the algorithms that consistently performed well were Hidden Markov Models (HMM) and Dynamic Time Warping (DTW).

A number of studies considered three-dimensional signature recognition using either special-purpose sensors or sensors in mobile phones. Irish studied the feasibility for a smartphone user to prove liveness by initialising a transaction with a 3-D signature using Naïve Bayes and Dynamic Time Warping [18]. An alternative approach by Pylvänäinen used Hidden Markov Models for accelerometer-based recognition, testing on a fixed set of 10 gestures [32].

1.3 Overview of the Project

This project attempts to show the feasibility of authenticating smartphone users using 3-D gestures captured with accelerometer and gyroscope. A smartphone user would register his/her signature to the system by waving the device in the air. An example is shown in Figure 1.3 Sensors in the device then record the acceleration and angular velocity at fixed time intervals. Feature extraction and classification are performed on this raw data.
During the initial stages, I collected and compiled a dataset of signatures from 20 people, aged from 18 to 53. The signature collection procedure was designed to avoid overfitting of data and to test the effectiveness of various forgery attacks against the system. Subsequently, raw signature data was pre-processed to remove the gravitational pull in acceleration data and improve classification performance.

Three authentication algorithms were chosen and implemented after a thorough testing on WEKA (a toolkit of machine learning algorithms) [14], namely Dynamic Time Warping, Hidden Markov Model and a Naïve Bayes Classifier (see Section 3.6). The Naïve Bayes Classifier performed consistently better than the rest, as expected from the initial test results in WEKA.

<table>
<thead>
<tr>
<th>Category</th>
<th>Project Requirements</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>The chosen classification algorithm achieves lower EER (Equal Error Rate) than others studied in this project.</td>
<td>✔</td>
</tr>
<tr>
<td>Usability</td>
<td>The time taken in both enrolment and authentication is reasonable to qualify as real-time.</td>
<td>✔</td>
</tr>
<tr>
<td>Usability</td>
<td>Users need not carry a physical item or memorise complex and arbitrary sentences.</td>
<td>✔</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional Extensions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Usability</td>
<td>The number of feature sets required for an acceptable authentication is small.</td>
<td>✔</td>
</tr>
<tr>
<td>Usability</td>
<td>The entire enrolment and authentication process can be performed in a mobile device alone.</td>
<td>✔</td>
</tr>
<tr>
<td>Usability</td>
<td>The mobile authentication process does not consume too much power or memory.</td>
<td>✔</td>
</tr>
</tbody>
</table>

*Table 1.1: Success criteria specified in the project proposal.*

As shown in Table 1.1, the project met all of its core success criteria, and fully independent mobile authentication system was developed as an extension to the host-based authentication scheme.
The end product is a fully functional mobile authentication system with Equal Error Rate of below 2.5% and high usability as a mobile system. To the best of my knowledge, it is the first system to support deviced-based signature recognition independently of any external computing power.

1.4 Challenges

Automatic signature recognition, although an active area of research due to the widespread use of signatures as an authentication method, is still considered a challenging problem. This is primarily due to the high intra-class variability and low inter-class variability of signature data. Indeed, the same person’s signatures are not identical when performed in different environments. Hence, it was necessary to identify the right level of preprocessing and to find ways of extracting salient features from the data before any classification was performed. This, in addition to the actual implementation of classification algorithms, added a major complication and ambiguity to the project.

The complexity of the optional extension to the project turned out to greatly exceed the initial expectations. Shifting the entire authentication process from the server to the individual devices introduced a whole new set of performance, energy and memory constraints. Even a perfect identification method would be useless if it consumed a significant proportion of the device’s battery life. This due consideration to usability as mobile software, on top of initial challenges of attaining security, significantly increased the complexity of the project.
Chapter 2

Preparation

This chapter starts off with a brief outline of pattern classification, followed by a description of pattern classification and the associated design cycle. Then it establishes a clear threat model that defines a set of possible attacks on 3-D signature authentication process. Based on these attack schemes, formal requirements of the end system are derived and analysed.

The later part of this chapter then moves on to describe my data collection process and formative evaluation by means of a survey. Finally, it describes software engineering techniques and tools employed throughout the development process.

2.1 Introduction to Pattern Classification

Pattern classification is a branch of machine learning pertaining to the assignment of a label or a class to a new observation by analysing patterns of known data called the training set. Given a training sequence, a learning algorithm constructs a model or a classifier based upon discriminative traits of the data referred to as features. Then, given new test data, the model computes how likely the test sample belongs to a particular class. (see Figure 2.1).

![Figure 2.1: Process of pattern classification.](image-url)
CHAPTER 2. PREPARATION

Pattern classification algorithms can be broadly divided into two categories depending on the method of learning the training data. *Supervised learning algorithms* assume that each sample (represented as a vector of features) in the training set has been labelled with its class membership. Examples of a supervised learning model are Naïve Bayes and Support Vector Machine (SVM). On the other hand, *unsupervised learning* constructs a classifier from an unlabeled set of training samples. Hidden Markov Models and clustering algorithms are examples of this.

2.1.1 Binary Classification and Decision Theory

Imagine a problem of classifying data into two classes: $C_1$ and $C_2$. A common approach is to train the model with data from a single class, e.g. $C_1$. Then, for each test sample, the classification algorithm outputs the value of some *similarity metric* between the test data and the representative feature set of $C_1$.

Suppose we have a test set $S_1$ containing only examples of class $C_1$, and another test set $S_2$ with only the data of class $C_2$. Then, we have a distribution of similarity scores for $S_1$ and $S_2$ (see Figure 2.2). The similarity score distribution of $S_1$ is called the *Authentics* distribution, as the test set is of the same class as the data used to train the model. For this reason we would generally expect the similarity scores for $S_1$ to be high. The similarity score distribution for $S_2$ is called the *Imposters* distribution. For any reasonable classifier, we would expect the Imposters distribution to have a lower mean value than the Authentics.

![Figure 2.2: Example decision environment.](image)

Given these two distributions, a discrimination threshold is placed such that all test samples with a smaller similarity will be classified as an imposter, and only the samples with a higher similarity will be accepted as authentic. When there is no overlap between two distributions, the
classification threshold is placed between them and the resulting decision will almost always be correct. However, the two distributions often overlap and the result is less clear-cut. The purpose of extracting salient features (with low intra-class variability and high inter-class variability) from raw data is to make the Authentics and Imposters distributions narrow and far apart from each other.

2.1.2 Evaluation of a Binary Classifier

Given a classifier and similarity threshold, there are four possible outcomes of a classification decision. If a positive instance is classified as positive it is called a *true positive*. If it is however labelled as negative, it is a *false negative*. If the instance is negative and is correctly classified as being negative, it is counted as a *true negative*. Otherwise, it is called a *false positive*. For a given classifier and test data, a two-by-two confusion matrix as illustrated in Table 2.1 can be constructed.

<table>
<thead>
<tr>
<th>Hypothesised Class</th>
<th>True Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Classification</td>
<td>Actually Positive</td>
</tr>
<tr>
<td>Negative Classification</td>
<td>Actually Negative</td>
</tr>
</tbody>
</table>

*Table 2.1: Confusion matrix of a binary classifier.*

Several metrics can be derived from the confusion matrix. The *true positive rate* (TPR) of a classifier is the ratio of positive instances that are correctly classified. The *false positive rate* (FPR) is estimated as the ratio of negative test samples incorrectly classified (classified as genuine).

\[
\text{TPR} = \frac{TP}{TP + FN},
\]

\[
\text{FPR} = \frac{FP}{FP + TN}.
\]
CHAPTER 2. PREPARATION

An ideal classifier would label all positives and negatives correctly and achieve TPR of 1 and FPR of 0. For most real classifiers, as soon as we move the similarity threshold in Figure 2.2 in one direction to improve one rate, the other rate immediately worsens.

In Signal Detection Theory, a Receiver Operating Characteristic (ROC) curve is used to visualise this tradeoff. An ROC curve is a 2-dimensional plot that graphically shows the classifier’s performance for all possible values of similarity threshold. Figure 2.3 shows an example of an ROC curve. The red dashed line on the graph represents the performance baseline resulting from random guessing. The blue dashed line represents performance of a classifier whose FPR is equal to $1 - TPR$, also known as False Negative Rate (FNR).

![Example ROC curve](image)

*Figure 2.3: Example ROC curve.*

Two additional metrics from the ROC curve are used in evaluating this project.

**Equal Error Rate (EER)** is defined as the rate at which false positive and false negative rates are equal. EER can be easily inferred from the ROC curve, namely the point (FPR, TPR) on the curve where $FPR = 1 - TPR$ holds (shown as the red dot in Figure 2.3). The EER of the classifier is then the length of the green dashed line. In general, the smaller the EER, the better the classifier.

**Area Under the Curve (AUC)** is the area under the ROC curve. Along with EER, it gives a concise representation of the classifier’s performance. The greater the AUC, the better the classifier in general.
2.1.3 Training and Testing

In assessing a classifier’s performance, available data is usually partitioned into separate non-overlapping groups such that data from one group is used for one purpose only. It is crucial that any sample used in training is never used for testing. In addition to training and test data, the validation data can be used to adjust the parameters of the classifier. Therefore, a typical evaluation of a classifier would consist of first training the model with the training data, optimising the parameters using the validation set and finally assessing its performance with the test data.

![Three datasets for supervised learning.](image)

In practice the amount of data available is often very limited. This can be remedied somewhat by using cross-validation. An $n$-fold cross validation splits the data into $n$ partitions of equal size and uses one for testing and the rest for training. In this project, 10-fold cross validation was adopted.

2.2 Design Cycle of Pattern Recognition Systems

The design process of pattern recognition systems encompasses of a repetition of five stages [10]. This approach, as shown in Figure 2.5, was adapted to fit in the timescale of this project. This section presents an overview for each stage and specific design choices made at each step of the project.

![Design cycle of pattern recognition systems.](image)
1. **Data Collection**: The method of data collection, size of the corpus and choosing the sample set can all have an impact on the overall quality of classification. See Section 2.5.2 for a discussion on data collection.

2. **Feature Selection**: A choice of a feature set is a crucial design choice in a pattern recognition system. Section 3.5 explains a general approach to selecting salient features. In the Evaluation chapter, the optimal choice of feature set is analysed for each classification algorithm implemented (Sections 4.2.2, 4.2.3 and 4.2.4).

3. **Model Selection**: Initial filtering was conducted by testing algorithms in WEKA on signature data collected. After studying this result and previous work, three algorithms were chosen to be implemented (Section 3.6).

4. **Classifier Training**: Framing the problem of signature authentication as a multi-class classification (one class per person) is not only inefficient but also computationally intractable. Hence it was treated as a binary classification (true user versus everyone else) and training was performed only on the samples of the true user.

5. **Classifier Evaluation**: 10-fold cross validation was used for evaluation. EER and AUC were extensively used to specify the classifiers’ performance.

### 2.3 Threat Model and Mobile Security

Before moving on to the threat model, it is important to address some security concerns of mobile authentication. Many smartphones’ open environment is vulnerable to malware that could get access to user passwords (in any form) used for authentication. For sensor-based authentication methods, unauthorised access to specific sensors used in authentication can potentially divulge the signal password. Hence, it should be stressed that a gesture-based authentication scheme in an insecure mobile environment is exposed to the risk of malware.

One suitable measure to enhance overall mobile security is the Trusted Execution Environment (TEE). The TEE is a separate execution environment that provides security services to the Rich Operating System (Rich OS) [29]. Most importantly, the TEE protects the resources and data belonging to authorised applications. Using the TEE will safeguard against attacks targeting sensors and memory storage to extract user’s information.

When designing a secure system, it is crucial to clearly define the capabilities and limitations of an attacker. It goes without question that the more capable the attacker is, the more
security threat he poses to the system. In this project, three different classes of attack scheme were defined according to the amount of information accessible. They are *Random Forgery*, *Limited Visual Forgery* and *Visual Forgery*:

- **Random Forgery**: The attacker has no information about the genuine user, therefore must resort to random guessing with arbitrary gestures. The effectiveness of this forgery was tested by using one person’s signature to authenticate as another.

- **Limited Visual Forgery**: The forger is allowed a limited number of physical observations. To see the robustness of the gesture-based authentication against shoulder surfing, each test subject in data collection was shown two brief video sequences of someone else’s signature gesture and was asked to mimic it.

- **Visual Forgery**: In data collection process most people used their real handwritten signatures rather than inventing a new one. To see the security threat in case the original handwritten signature signed elsewhere is exposed, each test subject was shown the image of the handwritten signature of someone else and was asked to imitate it.

### 2.4 Requirements Analysis and Approach

Identification and analysis of the system’s requirements was a formative step in the overall development process. The following questions were considered relevant and guided the project.

1. Does the biometric modality in question have enough entropy to be scalable?
2. What are the possible attacks against the system? How effective are they?
3. What are the factors potential users will look for in an authentication system? How usable is this system in such regards?

The first question deals with the effectiveness of signature as a discriminating identifier. Previous works have shown that signatures can be used to authenticate people, but signatures inherently have high intraclass variability and low interclass variability. Therefore, it is crucial to extract salient features from data. Question 2 addresses the security aspect of the system and the performance of gesture-based authentication compared to other knowledge-based mechanisms. The last question, perhaps the most important for the system to be actually deployed, regards the usability of the system.

The set of security and usability benefits suggested by Bonneau *et al.* [6] was adopted and tailored specifically towards mobile authentication.
2.5 Data Collection and Formative Evaluation

During initial stages of the project, 20 test subjects were recruited and their signatures collected. A detailed distribution of age, gender and preference for hand is presented in Table 2.3.

2.5.1 Formative Evaluation

Signature data was collected with an HTC Sensation mobile phone running on Android v4.0.3. A simple data logging application was implemented for this process. Along with signature data, some metadata of the subjects was also collected. The result is presented in Figure 2.6 and 2.7.

<table>
<thead>
<tr>
<th>Age</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;20</td>
<td>20 – 24</td>
</tr>
<tr>
<td>Male</td>
<td>3</td>
</tr>
<tr>
<td>Female</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2.3: Age, gender and handedness distribution of participants in data collection.
It was found that the majority of the survey group do not use any method to lock their mobile phones, mostly due to inconvenience and the limited user interface. The participants, however, expressed positive opinions about the idea of biometric authentication in general, and their willingness to use such a system was high (see Table 2.4).

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Familiarity with gesture interfaces (1 = &gt;10 times, 2 = 1–10 times, 3 = never)</td>
<td>3</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Willingness to use biometrics if offered by a mobile phone (1 = positive, 2 = neutral, 3 = negative)</td>
<td>11</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

*Table 2.4: Experience and impression about biometric authentication.*

### 2.5.2 Data Collection

The biometric modality studied in this project is a three-dimensional signature captured via the accelerometer and gyroscope in a mobile phone. 20 subjects who participated in the data collection were asked to choose their own unique gestures and wave it in the air while holding the phone. The majority chose their own handwritten signatures.

The subjects provided 30 signature samples and were explicitly asked beforehand to take a five-minute break for every five signatures provided. This measure was taken to prevent data
from overfitting. In addition, they were also asked to perform every fifth signature sitting down, and the rest of the signatures standing up. Their handwritten signatures were also collected.

With the participants’ consent, two brief video sequences of each person performing a gesture were taken. The camera setup for this video recording is illustrated in Figure 2.8. The blue dashed line shows the direction the subject was facing while performing the gestures.

![Figure 2.8](image)

*Figure 2.8: Camera setup during sample collection for forgery tests.*

In addition to drawing 30 original signatures, the subjects were then asked to forge two different people’s signatures after having seen the video sequences for each. They performed five forgery attempts for each target. These signatures were used in the Limited Visual Forgery testing.

Next, the handwritten signatures of two different people (different from the previous two whose video sequences were shown) were revealed to the forgers before they attempted to mimic the signatures five times each. These samples were used in the Visual Forgery testing.

### 2.6 Choice of Tools

#### 2.6.1 Programming Language

The mobile development was carried out on Android platform rather than iOS for the following reasons:

1. with a just single build settings file, Android applications are much simpler to configure and build;

2. Android supports an easy and seamless API for networking; and
2.6. CHOICE OF TOOLS

3. The development framework for iOS is difficult to use on a Windows machine.

Following from the choice of Android as a mobile platform, the natural choice of primary development language was Java. Additionally, the following factors contributed to this design choice:

1. The entire development process evolves around Android development from data collection to mobile authentication system.

2. There are many object-oriented concepts in the project that can be neatly represented in Java.

2.6.2 Additional Tools Used

This project did not require making use of many external libraries as I implemented almost all parts of the program myself. Three tools were employed, however:

- **Jama**: a linear algebra package for Java[^1] was used for efficient matrix operations in implementing multivariate normal distributions.

- **WEKA**: Machine Learning toolkit[^14] was used in the early stages to identify potential classification algorithms to be implemented.

- **Berkeley DB[^2]** was used in both server/client architecture and mobile authentication system to store user’s data and parameters of the model.

2.6.3 Development Environment

The development was primarily carried out on my personal machine running Windows 8.

**Integrated Development Environment:** The majority of the code was written in Eclipse IDE 4.3.2 (Kepler) along with the Android SDK and the ADT (Android Development Tools) Plugin for mobile development.

**Revision Control:** Two revision control systems were used. Subversion 1.8.8[^3] was used to backup the LaTeX code of my dissertation, allowing me to easily go back to any previous drafts. My project code base was committed nightly to my Git repository hosted on BitBucket[^4].

[^4]: [https://bitbucket.org/](https://bitbucket.org/)
CHAPTER 2. PREPARATION

**Backup Strategies:** Regular backups of my dissertation draft and references were committed to my Dropbox account.

### 2.7 Software Engineering Approach

This project has several components as discussed in Section 2.2, namely data collection, feature selection, model selection, classifier training and classifier evaluation. After considering the development stages and the timescale of the project, I decided to adopt the *Waterfall model* as the development approach.

For developing prototypes for both client-server authentication and mobile authentication schemes, the approach taken was to first lay down the fundamentals and iteratively incorporate each new increment to the previous design. This *incremental model* significantly helped streamline the design process.

### 2.8 Summary

This chapter discussed the work conducted before the actual implementation. It encompassed initial research into binary classification, the overall design cycle and the threat model of the system. Building upon the attack schemes, it then presented the data collection procedure and the formative evaluation conducted with the participants.

Chapter 3 discusses the implementation details of the classification process as well as two types of gesture-based authentication frameworks developed in this project.

[https://www.dropbox.com/](https://www.dropbox.com/)
Chapter 3

Implementation

This chapter first gives a high-level overview of SigVerify, then describes the implementation decisions made at each step from the data structures used to represent signature samples to the design of classification algorithms. It concludes with an outline of two frameworks for gesture-based authentication: a client-server architecture and mobile authentication.

SigVerify is a fully functional gesture-based authentication mechanism that supports both device-based and host-based authentication over a network to offload computation to a remote server.
3.1 System Overview

3.1.1 High-level Workflow

The typical workflow of SigVerify consists of three parts: processing, enrolment and authentication. The processing stage is responsible for polishing the raw sensor values into a more informative data representation and ultimately into feature vectors. At the enrolment stage, the details of the user are registered to the system and his model is constructed. Then, at authentication time, new test data is presented to the system and a classification decision is made based on its similarity level.

![Figure 3.4: Components of SigVerify.](image)

1. **Processing** is performed every time for each sample, both in training and authentication.

   (a) **Preprocessing**
   
   The raw sensor data is cleansed through a pipeline of preprocessing stages. First, the basis of the acceleration vector space is transformed so that the direction of gravitational acceleration is downwards. Then the effect of gravity is subtracted from sensor data. Afterwards, the data is interpolated to a fixed sampling frequency and smoothed.

   (b) **Segmentation**
   
   After preprocessing, the sensor data is split into segments of equal size. This is necessary because extracting one feature vector per entire signature sample results in a data representation of lower granularity than extracting features for each smaller segment. Distances between segments can be configured so that one might overlap with the next.

   (c) **Feature Extraction**
   
   Discriminative features are extracted from each segment. Selecting a feature set is
crucial in determining the accuracy and efficiency of the classifier. Both time-domain and frequency-domain features are studied in this project.

2. **Enrolment** is only performed once for each user.

   (a) **Model Construction**

   Based on the feature vector representation of training data, a model is constructed. The model is usually a set of parameters that allow the classification algorithm to compute the similarity level for new, unobserved training data.

   (b) **Similarity Threshold**

   Similarity levels for the validation set are computed and the threshold is determined. The exact value of threshold was chosen empirically.

3. **Authentication** is performed each time a user requests to be authenticated.

   (a) **Classification**

   Given new test data, the classifier makes a decision based on its similarity with the trained model.

### 3.1.2 Data Structures

#### 3.1.2.1 Raw Signature Data

A clear and efficient representation of signature data was required in many aspects of the project, including data collection, preprocessing, and feature extraction. Three different data structures were implemented, each with a different granularity of data.

The first one captures a single three-dimensional measurement from both accelerometer and gyroscope, sampled at the same point in time. This level of detail is needed for preprocessing tasks which modify raw data at each individual value.

The next data structure captures the entire raw dataset for a single signature. It is important to be able to handle each signature efficiently, for example in the case of crossvalidation, and this data structure provided a neat solution.

Lastly, a data structure for a collection of signatures was designed. This enabled a rapid manipulation of training, validation and test dataset. This three-level hierarchy allowed me to efficiently deal with signature data for needs of both high level and low level scale (see Figure 3.5).
CHAPTER 3. IMPLEMENTATION

![Diagram of data structures](image)

*Figure 3.5: Three-level hierarchy of data structures for raw signature data.*

### 3.1.2.2 Feature Vector

A set of data structures was implemented to represent a feature vector. The feature extraction module takes the signature data as an input and outputs a feature vector. Classification algorithms in turn take a set of feature vectors as an input and output classification decisions.

### 3.1.2.3 Data Structures for the ROC Curve

To efficiently build an ROC curve, a compact representation was implemented for a decision environment and a single point on the ROC curve. This allowed a streamlined construction of an ROC curve given the test results of a classifier.

### 3.2 Preprocessing

A series of preprocessing steps was performed before features were extracted from the data. (see Figure 3.6). The effect of each preprocessing step on signature data is visualised in Figure 3.7.

![Preprocessing pipeline](image)

*Figure 3.6: Preprocessing pipeline for signature data.*
3.2. PREPROCESSING

3.2.1 Removing the Gravitational Acceleration

Accelerometers in mobile devices have only one reference value to work with: force. For this reason, they cannot differentiate between gravitational pull and acceleration generated by the user’s hand gesture. This pollutes the signature data signals and the effect is aggravated by the fact that the device’s orientation continuously changes over the course of a gesture.

The Android Application Programming Interface (API) provides a module called the LinearAcceleration to mitigate this inaccuracy. It internally performs low-pass filtering to extract the gravitational pull and subtracts it from the raw data measurements. However, this simple method did not produce acceptable data quality. I hence took a more fundamental approach of shifting the basis of acceleration vector space so that the direction of gravity is always towards the negative $y$-axis, and simply subtracted the gravitational acceleration from that direction.\textsuperscript{1}

Let

$$A = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_N^T \end{pmatrix},$$

and $a_i$ be the vector of acceleration in $(x, y, z)$ dimensions sampled at time $i$. Also, let $g_N(A)$ be a linear function that returns the direction of gravity in $A$. Then, shifting the acceleration basis is equivalent to finding a transformation matrix $R$ such that $g_N(AR^T)^T = \alpha(0, -1, 0)$, where $\alpha$ is a scalar. As $g_N(A)$ is linear, this can be rewritten as $Rg_N(A) = \alpha(0, -1, 0)^T$.

In general, there exist an infinite number of solutions to this problem, therefore I impose a constraint that $R$ should be orthogonal, i.e. $R^TR = I$. When the tilting of the device is moderate, i.e. the angle between $g_N(A)$ and the negative $y$-axis is below 90 degrees, $R$ can be defined as:

$$R = \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix},$$

(3.1)

and $r_2^T = -\frac{g_N(A)}{|g_N(A)|}$. From the small tilting condition, it follows that $r_2^T\hat{x}$ and $r_2^T\hat{z}$ are both nonzero, where $\hat{x}$ and $\hat{z}$ are both unit vectors in the positive $x$ and $z$ axis respectively. Then I used Gram-Schmidt orthogonalisation to transform $R$ to a new basis. Let

$$r_1 = \frac{\hat{x} - \text{proj}_{r_2}(\hat{x})}{|\hat{x} - \text{proj}_{r_2}(\hat{x})|},$$

(3.2)

\textsuperscript{1}This method was adopted from the work by Pylvänäinen [32].
The choice for $g_N$ was the mean vector of acceleration dataset $A$, i.e. $g_N(A) = \frac{1}{N} A^T 1_N$, with $1_N$ being a vector of $N$ ones.

Given the rotation matrix $R$, the acceleration vector $A$ is transformed and gravitational acceleration is subtracted from the result:

$$A' = AR^T - \begin{pmatrix} 0_N & g_N & 0_N \end{pmatrix}$$

where $0_N$, $g_N$ are both $N \times 1$ vectors of $N$ ones and 9.8s (the gravitational acceleration), respectively. This method has additional advantage over applying a low-pass filter, namely that it cancels out the effect of the same user holding the mobile device differently each time, as the dataset is rotated to a new basis.

### 3.2.2 Interpolation

The data logging application was configured to sample data at 50Hz. However, the actual sampling frequency was often irregular. This is due to both limitations in the Android API and the variable load on the mobile processor. For devices running on Android platform, sensor values are only accessible via the `onSensorChanged` module that gets called whenever sensor values have changed, which is not always in sync with the sampling frequency specified by the programmer.

Therefore, the data was linearly interpolated to a fixed sampling rate 50Hz. Before I move on, let us denote the signature data as $s_d(t_i)$, where $s \in \{a, g\}$ (a for accelerometer and $g$ for gyroscope), $d \in \{x, y, z\}$ (denoting each dimension) and $t_i$ denotes the $i$th time step sampled. The same notation is used throughout this report.

Given two data points $s_d(t_i)$ and $s_d(t_{i+1})$ sampled at $t_i$ and $t_{i+1}$, a linearly interpolated value for time point $t \in (t_i, t_{i+1})$ is:

$$s_d(t) = s_d(t_i) + \left( s_d(t_{i+1}) - s_d(t_i) \right) \frac{t - t_i}{t_{i+1} - t_i}.$$
3.2.3 Smoothing

In addition to the effect of gravity and irregular sampling frequency, the raw sensor measurements from both accelerometer and gyroscope were corrupted with high frequency noise. One approach that has commonly been used in previous work is smoothing \[31\]. Hence, the signature data was smoothed using a \textit{Simple Moving Average} (SMA) with a span of 10. This method recomputes each value in a sequence as the arithmetic mean of the last 10 values.

\[
s_d(t_i) = \frac{1}{10} \sum_{j=i-9}^{i} s_d(t_j).
\]

The effect of each processing step in order is shown in Figure \[3.7\]. The figures are projections of acceleration values into the $x - z$ plane, with $z$ axis facing the device’s screen and $x$ axis lying sideways to the device.

\begin{figure}[h]
\centering
\begin{subfigure}{0.4\textwidth}
\centering
\includegraphics[width=\textwidth]{raw_acceleration_data}
\caption{Raw acceleration data.}
\end{subfigure}\hfill
\begin{subfigure}{0.4\textwidth}
\centering
\includegraphics[width=\textwidth]{acceleration_trace_after_removing_gravity}
\caption{Acceleration trace after removing gravity.}
\end{subfigure}
\begin{subfigure}{0.4\textwidth}
\centering
\includegraphics[width=\textwidth]{acceleration_trace_after_interpolation}
\caption{Acceleration trace after interpolation.}
\end{subfigure}\hfill
\begin{subfigure}{0.4\textwidth}
\centering
\includegraphics[width=\textwidth]{acceleration_trace_after_smoothing}
\caption{Acceleration trace after smoothing.}
\end{subfigure}
\caption{Two-dimensional trace of acceleration data after each preprocessing step.}
\end{figure}
3.3 Segmentation

Most gesture classification methods perform segmentation to split the raw data into smaller time segments. Classification algorithms are then applied separately to each segment. Various types of segmentation policy exist depending on the nature of the underlying data. The basic windowing scheme is shown in Figure 3.8. The raw data is partitioned into segments with no overlap. An alternative policy allows adjacent segments to share some fraction of data (see Figure 3.9). Segment size in this project was chosen to be 0.8s, with an intersegment overlap of 40%.

![Figure 3.8: Windowing scheme along gyroscope signal without intersegment overlap.](image1)

![Figure 3.9: Windowing scheme along gyroscope signal with intersegment overlap.](image2)
3.4 Feature Extraction

Previous work on gesture recognition has employed a wide range of features to discriminate instances. Applied to a segment of preprocessed data, feature extraction methods output a vector of features that is then used as an input to the classification algorithm. This section outlines the features used in this project, in both time- and frequency-domain.

Features were extracted from each of $x$, $y$ and $z$ axis as well as the magnitude vector $s_m(t) = \sqrt{s_x(t)^2 + s_y(t)^2 + s_z(t)^2}$. Of course, it is possible to not perform feature extraction and directly use the preprocessed data for training and testing. This is comparatively studied in the Evaluation chapter.

For a given segment of values from each dimension and each sensor, one feature vector was extracted, containing one or more of the features presented below.

**Mean**  
Arithmetic mean of the segment.

**Med**  
Median of the segment.

**Min**  
Minimum value of the segment.

**Max**  
Maximum value of the segment.

**Std**  
Standard deviation of the segment.

**Span**  
Difference between the maximum and the minimum of the segment.

**RMS**  
The root mean square of the segment, i.e. the square root of the mean of the squares of the original values.

**Bin**  
Relative frequency histogram distribution across equally spaced bins from the minimum to the maximum of the segment. The bin sizes used in this project are five and ten.

**MFCC**  
Mel-frequency cepstral coefficients. Explained below.

**Mel-frequency Cepstral Coefficients**

MFCCs are the amplitudes of the cosine transform of the real logarithm of the short-term energy spectrum expressed on a mel-frequency scale [39, 26]. A compact representation for acoustic signals, they are predominantly used in speech recognition systems. Figure 3.10 shows the steps in extracting MFCCs.
CHAPTER 3. IMPLEMENTATION

Figure 3.10: Process of extracting MFCCs from signature data [27].

**Pre-emphasis** A high-frequency filter was applied to the signal. This process emphasises the energy of signal at high frequencies:

\[ s'_d(t) = s_d(t) - 0.95 * s_d(t - 1), \quad d \in \{x, y, z, m\}. \]

**Windowing** A Hamming window [28] was used to divide segments further into overlapping windows.

**DFT** A Discrete Fourier Transform was applied to each window.

**Mel Filter Bank Processing** As the frequency range in the Fourier spectrum is non-linear, the power spectrum was mapped onto mel scale using triangular windows:

\[ M(f) = 2595 \times \log_{10}(1 + \frac{f}{700}). \]

**DCT** Discrete Cosine Transform was applied to the log powers of the mel frequencies to decorrelate the spectral features, after which the amplitudes of first 13 spectra are obtained as MFCCs.

### 3.5 Feature Selection

Once the single features were extracted from each segment, they were combined into a feature vector. In this project, I adopted two methods for feature selection. For each classification algorithm, the single best performing features were identified and combined into an optimal feature vector. Each classifier was found to have its own best feature set; this is studied in Sections 4.2.2, 4.2.3 and 4.2.4.

In addition, a more general approach to feature selection known as Discriminative Potential Score (DPS) was used for the initial classifier evaluation on WEKA. In the original paper by Mrácek et al. [25], DPS was used to select salient features for recognising anatomical features from three-dimensional face scans.
The underlying idea of DPS is to identify features with low intra-class variability and high inter-class variability. To compute the DPS, the value of each feature was normalised to the range of [0, 1]. For each person \( i \) and feature \( j \), the range of values of feature \( j \) over all instances belonging to person \( i \) was calculated. Let us denote that range of values as \( v_{ij} \). Then, following statistical values were computed:

**Mean Range**  \( \mu_j \), mean difference between the maximum and minimum in \( v_{ij} \).

**Standard Deviation**  \( s_j \), mean standard deviation of values in \( v_{ij} \).

**Maximum Range**  \( m_j \), maximal difference between the maximum and minimum in \( v_{ij} \) over all \( i \).

These values indicate the intra-class variability of each feature. For the inter-class variability, it was assumed that feature values of all subjects should be uniformly distributed to have high inter-class variability. Hence the correlation of feature values to uniform probability density function \( c_j \) was computed. From these values, the DPS of feature \( j \) was defined to be:

\[
dps_j = \left( 1 - (\mu_j + s_j + m_j) \right) + c_j.
\]

The DPS for each feature was computed and is presented in Table 3.1. The dataset used in computation was a combination of each subject’s training data. In the original calculation, features with multiple elements, e.g. Bin and MFCC, were treated as multiple separate features. Also, each feature was extracted from eight different dimensions (accelerometer/gyroscope and \( x/y/z/m \)), hence feature Bin5 for example has 40 different feature-axis combinations. The result shown is averaged over all such combinations.

<table>
<thead>
<tr>
<th>Feature</th>
<th>( \mu )</th>
<th>( s )</th>
<th>( m )</th>
<th>( c )</th>
<th>( dps )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin5</td>
<td>0.341</td>
<td>0.082</td>
<td>0.900</td>
<td>-0.0017</td>
<td>-0.323</td>
</tr>
<tr>
<td>Bin10</td>
<td>0.350</td>
<td>0.084</td>
<td>0.910</td>
<td>-0.0061</td>
<td>-0.350</td>
</tr>
<tr>
<td>Max</td>
<td>0.281</td>
<td>0.069</td>
<td>0.679</td>
<td>-0.0259</td>
<td>-0.054</td>
</tr>
<tr>
<td>Med</td>
<td>0.263</td>
<td>0.060</td>
<td>0.568</td>
<td>0.1144</td>
<td>0.223</td>
</tr>
<tr>
<td>Mean</td>
<td>0.221</td>
<td>0.052</td>
<td>0.522</td>
<td>-0.1100</td>
<td>0.095</td>
</tr>
<tr>
<td>MFCC</td>
<td>0.345</td>
<td>0.088</td>
<td>0.794</td>
<td>0.0247</td>
<td>-0.202</td>
</tr>
<tr>
<td>Min</td>
<td>0.320</td>
<td>0.079</td>
<td>0.748</td>
<td>0.0779</td>
<td>-0.069</td>
</tr>
<tr>
<td>RMS</td>
<td>0.217</td>
<td>0.052</td>
<td>0.458</td>
<td>0.0740</td>
<td>0.346</td>
</tr>
<tr>
<td>Span</td>
<td>0.273</td>
<td>0.067</td>
<td>0.634</td>
<td>-0.0094</td>
<td>0.016</td>
</tr>
<tr>
<td>Std</td>
<td>0.240</td>
<td>0.058</td>
<td>0.518</td>
<td>-0.2450</td>
<td>-0.061</td>
</tr>
</tbody>
</table>

*Table 3.1: DPS and its subscores for each feature.*
3.6 Choice of Classifiers

Given the timescale of eight months, it was necessary to set aside a model selection phase. I decided to use WEKA to identify suitable classification algorithms for signature recognition.

As an input, I extracted two features (Med and RMS) from preprocessed data for each dimension \( d \in \{ x, y, z, m \} \) and both accelerometer and gyroscope. Overall, each signature sample was transformed into a feature set of 16 features.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TPR</th>
<th>FPR</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>MCC</th>
<th>ROC</th>
<th>PRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost</td>
<td>0.952</td>
<td>0.097</td>
<td>0.951</td>
<td>0.952</td>
<td>0.951</td>
<td>0.870</td>
<td>0.949</td>
<td>0.954</td>
</tr>
<tr>
<td>Bagging</td>
<td>0.930</td>
<td>0.162</td>
<td>0.931</td>
<td>0.930</td>
<td>0.926</td>
<td>0.808</td>
<td>0.954</td>
<td>0.959</td>
</tr>
<tr>
<td>Decision Table</td>
<td>0.948</td>
<td>0.103</td>
<td>0.947</td>
<td>0.948</td>
<td>0.947</td>
<td>0.859</td>
<td>0.915</td>
<td>0.933</td>
</tr>
<tr>
<td>J48</td>
<td>0.928</td>
<td>0.116</td>
<td>0.930</td>
<td>0.928</td>
<td>0.927</td>
<td>0.812</td>
<td>0.905</td>
<td>0.911</td>
</tr>
<tr>
<td>Logistic</td>
<td>0.951</td>
<td>0.095</td>
<td>0.950</td>
<td>0.951</td>
<td>0.949</td>
<td>0.866</td>
<td>0.962</td>
<td>0.963</td>
</tr>
<tr>
<td>MLP</td>
<td>0.979</td>
<td>0.049</td>
<td>0.980</td>
<td>0.979</td>
<td>0.979</td>
<td>0.941</td>
<td>0.979</td>
<td>0.980</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.978</td>
<td>0.054</td>
<td>0.979</td>
<td>0.978</td>
<td>0.976</td>
<td>0.940</td>
<td>0.976</td>
<td>0.978</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.961</td>
<td>0.102</td>
<td>0.962</td>
<td>0.961</td>
<td>0.958</td>
<td>0.891</td>
<td>0.978</td>
<td>0.979</td>
</tr>
<tr>
<td>SGD</td>
<td>0.977</td>
<td>0.061</td>
<td>0.979</td>
<td>0.977</td>
<td>0.975</td>
<td>0.938</td>
<td>0.958</td>
<td>0.964</td>
</tr>
<tr>
<td>SVM</td>
<td>0.968</td>
<td>0.084</td>
<td>0.972</td>
<td>0.968</td>
<td>0.966</td>
<td>0.915</td>
<td>0.943</td>
<td>0.951</td>
</tr>
</tbody>
</table>

Table 3.2: Averaged performance of classifiers in WEKA on signature data.

Table 3.2 shows the classifiers’ performance averaged over all test subjects. The columns represent metrics; some of them were previously discussed in Section 2.1.2 (Area Under the ROC Curve). Precision, recall, F1 and MCC measure the accuracy and sensitivity of the classifier, with higher values being desirable. Based on precision and recall, a Precision-Recall Curve (PRC) can be constructed in a similar way to an ROC curve. The rightmost column corresponds to the area under the PRC. A brief explanation is given for each classifier algorithm tested above [15].

**AdaBoost:** An ensemble learning algorithm that constructs a set of base learners and combines them [11].

**Bagging:** Another method to aggregate classifiers [7].

**Decision Table:** A decision table with a default rule mapping to the majority class [21].

**J48:** A decision tree where each node chooses a feature to split its samples most effectively [33].

**Logistic Regression:** A multinomial logistic regression model [22].
3.6. CHOICE OF CLASSIFIERS

**Multilayer Perceptron (MLP):** Based on a backpropagation neural network [35].

**Naïve Bayes:** Standard probabilistic Naïve Bayes classifier [20].

**Random Forest:** Ensemble learning with multiple decision trees [8].

**Stochastic Gradient Descent (SGD):** An efficient variant of Gradient Descent to sample a subset of all the element functions to be summed.

**Support Vector Machine (SVM):** A supervised learning method for binary classification [30].

![Figure 3.11: Average latency of Naïve Bayes, Multilayer Perceptron and Random Forest (ms).](image)

The top three classifiers in terms of AUC were Multilayer Perceptron, Naïve Bayes and Random Forest. Among these, Naïve Bayes was a clear winner in speed while offering classification accuracy on a parallel level with others. Figure 3.11 shows the average latency of the three classifiers (30 authentics and 20 imposters samples were used for each subject). For these reasons, I decided to implement a Naïve Bayes classifier.

Additionally, previous work in gesture recognition [17] found Hidden Markov Models and Dynamic Time Warping to be consistently effective. Although not supported by WEKA, they were chosen to be implemented as well.
CHAPTER 3. IMPLEMENTATION

3.7 Classification Algorithms

This section introduces three classification algorithms I implemented in this project, namely Dynamic Time Warping, Hidden Markov Models and Naïve Bayes Classifiers. It illustrates the basic idea underpinning each algorithm and also describes different variants and particular issues in the implementation process.

3.7.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a pattern matching algorithm based on dynamic programming with non-linear time-normalisation. It measures similarity between two sequences in time (of possibly different lengths) and is widely applied in various fields including speech recognition, bioinformatics and also gesture recognition [13].

3.7.1.1 Distance Function

I used weighted \( p \)-norm as a distance function between two feature vectors to account for difference in scaling between various features. Given two feature vectors \( a \) and \( b \) of size \( n \), the \( p \)-norm difference between \( a \) and \( b \) is defined as:

\[
||a - b||_p = \left( \sum_{i=1}^{n} \left( w(i) \ast |a_i - b_i| \right)^p \right)^{1/p},
\]

where the weight function \( w(i) \) normalises values of various features to an even standard, as acceleration and rotational speed have different value ranges. I chose \( w(i) \) as the reciprocal of the mean magnitude of the \( i \)th element in all feature vectors in the dataset.

\( p \) values of 1 and 2 were used in this project (known as Manhattan and Euclidean Norm respectively). Throughout this chapter, \( ||a - b||_p \) is abbreviated to \( d(a, b) \) for distance metric.

3.7.1.2 Description of the Algorithm

DTW computes the similarity of two temporal sequences by finding the cheapest warp or alignment. Both sequences are warped in the sense that some parts are of them are extended in time and others reduced to find the best alignment. To do this, the algorithm computes an alignment matrix \( D \) that stores the local warping decisions. Algorithm[1] shows an outline of the algorithm.

The cheapest warp path is found by backtracking from the last element of the array \( D(I - 1, J - 1) \) and remembering each local decision when computing the minimum of three possible predecessors.
Algorithm 1: Outline of the basic Dynamic Time Warping algorithm.

1. **Input**: sequences of feature vectors \( A \) and \( B \) of length \( I \) and \( J \).

\[
A = (a_1, a_2, \ldots, a_I) \\
B = (b_1, b_2, \ldots, b_J)
\]

2. **Initialisation**: Create an \( I \times J \) two-dimensional array \( D \).
   
   (a) Set \( D[i, 0] = \infty \), \( \forall i \).
   
   (b) Set \( D[0, j] = \infty \), \( \forall j \).

3. **Iteration**: For each \( i \) from 1 to \( I \) and \( j \) from 1 to \( J \), do
   
   (a) Set cost = \( d(a_i, b_j) \).
   
   (b) Set
   
   \[
   D[i, j] = \text{cost} + \min \left\{ D[i - 1, j], D[i - 1, j - 1], D[i, j - 1] \right\}.
   \]

4. **Termination**: \( D[I - 1, J - 1] \) contains the warp score of \( a \) and \( b \).

The time complexity of the DTW algorithm is \( O(IJ) \) as it entails populating the two-dimensional array \( D \). However it is possible to speed up the process by constraining the possible paths, as suggested in the original paper by Sakoe et al. [36].

### 3.7.1.3 Restrictions on the Warping Function

**Adjustment window ratio** limits the maximal timing difference between two signatures. A large timing difference implies bad warping, hence warping is only allowed within a fixed linear band as visualised in Figure 3.12a. Of course, the exact value of the allowed timing difference depends on the length of the two signatures. Hence, the window size was specified as a fraction of the shorter signature’s length. Formally, this can be expressed as:

\[
\text{Compute } D[i, j], \text{ if } |i - j| \leq \min(I, J) * K.
\]

\( K \) value of \( \frac{1}{2}, \frac{1}{3} \) and \( \frac{1}{4} \) were studied in this project.

**Slope constraint** restricts the gradient of the warp path. Clearly, either too steep or too gentle a gradient is undesirable. This condition was realised by restricting the set of possible adjacent moves in the path. More specifically, for every \( m \) consecutives moves in the direction of either \( x \) or \( y \) axis, no more move in the same direction was allowed before
moving $n$ steps in the diagonal direction. The rigidity of this condition was specified by the following value:

$$P = \frac{n}{m}$$

<table>
<thead>
<tr>
<th>$P=0$</th>
<th>$P=1$</th>
<th>$P=2$</th>
</tr>
</thead>
</table>

![Table 3.3: Three different slope constraints for DTW.](image)

Figure 3.12 shows an example warp path within permitted regions for two different restrictions. The left figure demonstrates an adjustment window of six time units, and the right figure visualises a slope constraint of $P = 1$.

The exact methods of implementing the adjustment window and slope constraint are presented in Algorithm 2.
### Algorithm 2

Outline of the algorithm used to restrict the warping path in DTW.

For each $i$ from 1 to $I$ and $j$ from 1 to $J$, do

1. **Adjustment window:** if $|i - j| > \min(I, J) \times R$, $D[I, J] = \infty$

2. **Slope constraint:**

   - $P = 0$:
     
     $D[i, j] = \min \begin{cases} 
     D[i - 1, j] + d(i, j) + H \\
     D[i - 1, j - 1] + 2 \times d(i, j) \\
     D[i, j - 1] + \text{cost} + H 
     \end{cases}$

   - $P = 1$:
     
     $D[i, j] = \min \begin{cases} 
     D[i - 2, j - 1] + 2 \times d(i - 1, j) + d(i, j) + H \\
     D[i - 1, j - 1] + 2 \times d(i, j) \\
     D[i - 1, j - 2] + 2 \times d(i, j - 1) + d(i, j) + H 
     \end{cases}$

   - $P = 2$:
     
     $D[i, j] = \min \begin{cases} 
     D[i - 3, j - 2] + 2 \times d(i - 2, j - 1) + 2 \times d(i - 1, j) + d(i, j) + H \\
     D[i - 1, j - 1] + 2 \times d(i, j) \\
     D[i - 2, j - 3] + 2 \times d(i - 1, j - 2) + 2 \times d(i, j - 1) + d(i, j) + H 
     \end{cases}$

### 3.7.1.4 Similarity Metric for DTW

Both the adjustment window and the slope constraint favor diagonal steps over horizontal or vertical steps. In most cases, the cost of one diagonal step is smaller than the sum of one horizontal and one vertical cost. However, the diagonal step must only be chosen when its cost is clearly smaller than that of the others. Therefore, the costs of diagonal steps were doubled to cancel out this effect. Yet at the same time, non-diagonal moves should not be made excessively, as a large number of them result in a poor warp quality. For this reason, horizontal and vertical steps were additionally penalised with a fixed cost $H$. This approach was adopted from the work by Guse [13].

The final DTW score in $D[I - 1, J - 1]$ is heavily dependent on the lengths of the two sequences compared, and hence prefers shorter sequences to long ones. To compensate for this unfairness, the final score was normalised with the summed length:

$$D'[I - 1, J - 1] = \frac{D[I - 1, J - 1]}{I + J}.$$

The value $D'[I - 1, J - 1]$ was used as the similarity metric between two feature vectors. This value is referred to as the DTW score between $a$ and $b$, or DTW$(a, b)$ throughout this report.
3.7.1.5 Training Procedure

DTW is inherently a binary comparison mechanism. As such, the trained model has to represent every enrolment instance in a single signature sample. This is perhaps a weakness of DTW, but there are a few workarounds. Three methods introduced by Guse [13] and Abdulla et al. [1] are explained below. In this section, the set of enrolment samples is denoted as \( M = (m_1, m_2, \ldots, m_K) \).

Choosing the most representative sample. This is the most straightforward approach to training a model. For each enrolment sample \( m_i \), the DTW score is computed with every other enrolment sample \( m_j \), \( (j \neq i) \). Then, the sample with the minimum sum of costs is chosen as the most representative.

\[
m_{\text{mod}} = \arg \min_{m_i} \frac{1}{K} \sum_{j=1 \atop j \neq i}^{K} \text{DTW}(m_i, m_j).
\]

Although simple to implement, one immediate shortcoming of this method is that information about the rest of the training set cannot be encoded into the model.

Computing the average by averaging and duplication. A different approach suggested by Abdulla et al. is to take an average of all training samples. It selects a reference template or model by first taking the average of the length of every training sample. Then, a sample with length nearest to the average length is chosen as the initial template. Subsequently, all other training samples are aligned to the initial template by the DTW process. More specifically, longer samples are compressed and shorter ones expanded. The model is computed as the average of the aligned training samples. The full method is presented below:

1. Find the initial template \( m_{\text{tem}} \).

2. Compute the DTW score for each training sample \( m_i \) with \( m_{\text{tem}} \), while memoising the choices made in each step.

3. Trace back from the last point in the alignment matrix to the first frame. Expand or compress \( m_i \) along the way.

When comparing \( m_{\text{tem}}(x) \) with \( m_i(y) \), three cases arise depending on the step taken at \( D[x, y] \).

**Horizontal:** \( m_i(y) \) is expanded, and a new frame \( m_i(y - 1) \) is inserted with the same value of \( m_i(y) \).
3.7. CLASSIFICATION ALGORITHMS

**Diagonal:** $m_i(y)$ is kept. Nothing changes.

**Vertical:** $m_i(y)$ and $m_i(y - 1)$ are compressed into a single frame. The average value is taken.

4. Take the average of the aligned samples across each frame.

**Computing the average by summation and distribution.** A caveat of the previous approach is that while aligning a sample to the initial reference, its path as a signature might change significantly as a result of expanding and compressing frames. For acceleration and rotational speed it is important to preserve the sum of the magnitudes for the trajectory of the signature to remain close to the original.

Consequently, I adopted a method suggested by Guse [13] which instead of replicating and averaging templates splits them into equal values or adds them to preserve the local sum.

### 3.7.1.6 Authentication Procedure

Given a DTW model, new test data are classified as either authentic or imposters depending on the DTW score of aligning the model and the test sample. If the DTW score is less than some threshold, the authentication succeeds. Otherwise, the new test sample is classified as false.

The discrimination threshold was computed using both genuine and false validation samples. True validation set contains genuine user’s samples, whereas false validation set consists of different users’ samples. Assuming both validation sets are of size $n$, let the vectors of similarity scores $V_T = (v_1^T, v_2^T, \ldots, v_n^T)$ and $V_F = (v_1^F, v_2^F, \ldots, v_n^F)$, where $v_i$ is the DTW score between the $i^{th}$ validation sample (either true or false) and the model. Then, the threshold $\delta$ was defined as

$$\delta = Q_T \cdot \left( \frac{\mu(V_T)}{\sigma(V_T)} \right) + Q_F \cdot \left( \frac{\mu(V_F)}{\sigma(V_F)} \right),$$

where $Q_T$ and $Q_F$ are column vectors of size 2. The exact values of the vectors $Q_T$ and $Q_F$ were chosen empirically.

### 3.7.1.7 Parameter Setting for DTW

The configuration of parameters as shown in Table 3.4 was used in evaluating DTW. The choices of values that outperformed others are in bold.
### Table 3.4: Various parameter configurations of DTW.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjustment window ratio</td>
<td>$K = 1/2, 1/3, 1/4$</td>
</tr>
<tr>
<td>Slope constraint</td>
<td>$P = 0, 1, 2$</td>
</tr>
<tr>
<td>P-norm</td>
<td>$p = 1, 2, 3$</td>
</tr>
<tr>
<td>Non-diagonal penalty</td>
<td>$H = 5, 10, 15, 20$</td>
</tr>
<tr>
<td>Training Procedure</td>
<td>Choosing the most representative sample</td>
</tr>
<tr>
<td></td>
<td>Averaging by expanding and compressing</td>
</tr>
<tr>
<td></td>
<td>Averaging by summing and dividing</td>
</tr>
</tbody>
</table>

3.7.2 **Hidden Markov Model**

A Hidden Markov Model (HMM) is a statistical model which assumes that the process being modelled is a Markov process. A Markov process is a stochastic process that satisfies the Markov property, i.e. the probability distribution of next state depends only on what the current state is. In a HMM, the actual states of the process are unknown, and each state emits a certain observation based on a probability distribution. HMMs are used in a wide range of applications because they have a rich mathematical structure and work very well in practice, including for gesture recognition tasks [32].

#### 3.7.2.1 Basic Elements of an HMM

In this section, I use the following notation to describe the HMM. I denote instances of time in the process as $t = 1, 2, \cdots$, and the actual state at time $t$ as $q_t$. Also, we have a fixed set of possible states and observations of finite size, $S = (S_1, S_2, \cdots, S_N)$ and $E = (E_1, E_2, \cdots, E_M)$.

An HMM can be fully characterised by following parameters:

1. The number of states $N$.
2. The number of possible observations $M$.
3. The state transition probability distribution $A = \{a_{ij}\}$, where
   $$a_{ij} = P(q_{t+1} = S_j | q_t = S_i), \quad 1 \leq i, j \leq N \text{ and } \sum_{j=1}^{N} a_{ij} = 1.$$
4. The emission probability distribution $B = \{b_j(k)\}$, where
   $$b_j(k) = P(E_k \text{ observed at time } t | q_t = S_j), \quad 1 \leq j \leq N, \quad 1 \leq k \leq M.$$
3.7. CLASSIFICATION ALGORITHMS

5. The initial state distribution \( \pi = \{ \pi_i \} \), where
\[
\pi_i = P(q_1 = S_i), \ 1 \leq i \leq N.
\]

An HMM is often abbreviated to \( \lambda = (A, B, \pi) \) to indicate the complete parameters of the model. Figure 3.13 shows a sequence of states in blue and corresponding emissions in white. Similarly, blue arrows denote state transitions and white arrows denote probabilistic emission.

![Figure 3.13: First order Markov process with emissions](image)

3.7.2.2 Training and Authentication Procedures

Given an HMM \( \lambda = (A, B, \pi) \) and an observation sequence \( O = (O_1, O_2, \cdots, O_T) \), there are three basic problems as explained by Rabiner [34].

**Evaluation**: what is the probability that the HMM \( \lambda \) generated the observation sequence \( O \), i.e. \( P(O|\lambda) \)?

**Decoding**: what is the most likely sequence of states \( Q = (q_1, q_2, \cdots, q_T) \) given the observation sequence \( O \), i.e. \( \arg \max_Q P(O|Q, \lambda) \)?

**Learning**: how do we adjust the parameters \( \lambda = (A, B, \pi) \) to maximise \( P(O|\lambda) \)?

Of the three problems, two are relevant for this project. The first problem addresses the likelihood that a given model generated an observation, thus it is immediately applicable as a similarity metric. Known as the *forward probability*, it is used in both training and authentication process.

The third problem, on the other hand, deals with training the model. Given a training set of observations, it attempts to find the optimal set of model parameters. To do this, a variant of the Expectation-Maximisation algorithm known as the Baum-Welch algorithm [3] was implemented.
An overview of the classification mechanism for HMM is shown in Algorithm 3. As discussed in Section 2.2, only the samples of the genuine user were used in training the model, and the forward probability was used as a similarity metric.

**Algorithm 3: Outline of training and authentication procedure for HMM.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Training:</strong></td>
<td>Given a training set, compute the optimal HMM parameters $\lambda = (A, B, \pi)$ for the training observation sequence using the Baum-Welch algorithm.</td>
</tr>
<tr>
<td>2. <strong>Computing threshold:</strong></td>
<td>For each sample in the validation set, compute its forward probability given the model computed above. Determine the similarity threshold according to the forward probabilities of all validation samples.</td>
</tr>
<tr>
<td>3. <strong>Authentication:</strong></td>
<td>Given an HMM and a test sample, compute the forward probability that the model emitted the test sample. Accept the sample if the probability exceeds the threshold. Reject otherwise.</td>
</tr>
</tbody>
</table>

### 3.7.2.3 Application to Gesture Recognition

HMMs are very amenable to representing signatures due to their structure with hidden states and observation. In the context of signature recognition, the hidden process modelled by an HMM is the dynamics of the hand movement made by the person performing the signature. A state can be understood as a particular shape drawn with a device. Transition probabilities between states then carry a meaning of the likelihood of one shape being followed by another in someone’s signature. An initial probability distribution indicates how likely someone’s signature begins with a particular shape.

One major complication in applying HMMs to modelling signatures is that the observations from accelerometer and gyroscope are continuous real numbers, whereas our definition of HMM assumed the observation space to be discrete and finite. One solution is to discretise the space of measurements using a clustering algorithm. However, this results in an inaccurate model due to a loss of observation data. For this reason, a set of continuous observation probability distributions was implemented in this project, namely a mixture of multivariate Gaussian distributions.
3.7.2.4 Computing the Similarity Metric

The forward probability of an observation sequence given an HMM was computed by a dynamic programming approach [34]. The example code of the initialisation and the recursive procedure is presented in Listing 3.1.

The final forward probability was obtained by summing the last row of the forwardProb matrix. This was used as the similarity metric for the HMM.

```
// Computes the initial column of the forward probability matrix forwardProb.
public void forwardInit(HMM hmm, Emission e, int i){
    double pi_i = hmm.pi(i);
    double b_i = hmm.emissionProb(i).decode(e);
    forwardProb[0][i] = pi_i * b_i;
}

// Recursively populates the matrix forwardProb, one column after another.
public void forwardRecur(HMM hmm, Emission e, int t, int i){
    double acc = 0;
    for (int j = 0; j < hmm.statesNumber; j++)
        acc += forwardProb[t-1][j] * hmm.transProb(j, i);
    forwardProb[t][i] = acc * hmm.emissionProb(i).decode(e);
}
```

Listing 3.1: Initialisation and recursive step of computing the forward probability.

3.7.2.5 Training the Model

Given a set of training sequences, the HMM was trained via the Baum-Welch algorithm. A full derivation of the Baum-Welch algorithm is beyond the scope of this report, but derived by Bishop, [4]. In this section, I present a high-level overview of parameter reestimation mechanism.

Algorithm 4 shows the process of one iteration of the Baum-Welch algorithm. The parameters are reestimated repeatedly in this way until the change in the probability $P(O|\lambda)$ between successive iterations is negligible.
CHAPTER 3. IMPLEMENTATION

Algorithm 4: One iteration of the Baum-Welch algorithm.

1. To reestimate the parameters of an HMM $\lambda$, we compute the following variables.
   - $\beta_t(i)$ (backward probability): the probability of observing the emission sequence from $t + 1$ to $T$.
   - $\gamma_t(i)$: the probability of being in state $S_i$ at time $t$ given the observation sequence and the model.
   - $\xi_t(i,j)$: the probability of being in state $S_i$ at time $t$ and state $S_j$ at time $t + 1$.

2. We can reestimate the parameters $\lambda = (A, B, \pi)$ as:
   \[
   \bar{\pi}_i = \sum_{t=1}^{T-1} \xi_t(i,j)
   \]
   \[
   \bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \quad b_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.
   \]

3. Update $\lambda = (A, B, \pi)$ to $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$.

3.7.2.6 Implementing Continuous Emission Probabilities

To represent continuous emission densities, I assumed that each value in the feature vector is normally distributed. Then it was possible to represent the probability density function (pdf) of the observation as a multivariate Gaussian density. To allow for more flexibility in signature data, the observation probability distribution was modelled as a mixture of multivariate Gaussian distributions:

\[
b_j(O) = \sum_{m=1}^{M} c_{jm} \mathcal{N}[O, \mu_{jm}, U_{jm}],
\]

where $O$ is the observation vector and $c_{jm}$ is the mixture coefficient. $\mathcal{N}$ is a normal distribution with a mean vector $\mu$ and covariance matrix $U$.

The most challenging aspect in implementing a mixture of multivariate normal distributions was choosing the initial values of the parameters. The Baum-Welch algorithm is prone to finding local maxima of the likelihood function [34]. Hence, the initial parameters were manually segmented into separate states and then averaged over a large amount of training data.
3.7.2.7 Scaling

The number of parameters in the Forward-Backward algorithm grows proportionally to the size of the observation sequence (or to its square in the case of Baum-Welch algorithm). As both algorithms involve iteratively multiplying probabilities to a small value, the values of parameters tend to zero very quickly and yield a floating point underflow. This was particularly aggravated when continuous emission densities were used. After a few iterations of parameter reestimation, most elements of the covariance matrix approached to zero. To resolve these issues with floating point computation, a scaling method described by Rabiner was used.

In the forward algorithm, the exact values of $\alpha$ on each column are inconsequential, as long as the ratios between them are preserved so the ordering is the same. Thus, after each step, the forward probabilities were uniformly scaled by a factor $c_t$ defined as:

$$c_t = \frac{1}{N \sum_{i=1}^{N} \alpha_t(i)}.$$  

As for the Baum-Welch algorithm, the estimation procedure of $\xi$ was modified accordingly:

$$P(O|\lambda) = \frac{1}{T} \prod_{t=1}^{T} c_t,$$

and

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \alpha'_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j).$$

3.7.2.8 Authentication Procedure

The authentication process for HMM is similar to that for DTW (see Section 3.7.1.6). Given a trained HMM, the forward probability of a new test sample yields its similarity with the model. If it is greater than threshold calculated from the validation samples, the authentication succeeds. The probability threshold is computed using both true and false validation sets. Namely, it is defined as:

$$\delta = Q_T \cdot \left( \mu(V_T) \ \sigma(V_T) \right) + Q_F \cdot \left( \mu(V_F) \ \sigma(V_F) \right),$$
CHAPTER 3. IMPLEMENTATION

3.7.2.9 Parameter Settings for the HMM

The implementation of HMM was evaluated in this project by varying two parameters: the number of states in the model and the number of Gaussian mixtures to represent the observation distribution.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of states</td>
<td>S = 2, 4, 8</td>
</tr>
<tr>
<td>Number of mixtures</td>
<td>M = 2, 4, 8</td>
</tr>
</tbody>
</table>

Table 3.5: Parameter configurations of HMM.

3.7.3 Naïve Bayes Classifier

A Naïve Bayes Classifier (NBC) is a simple yet powerful classification method based on Bayes’ Theorem with simplistic independence assumptions between different features.

3.7.3.1 Algorithm Overview

This section explains the implementation of Naïve Bayes in this project. The standard formula for the Naïve Bayes is the following:

\[
P(c|f_1, f_2, \ldots, f_k) \approx \prod_{i=1}^{k} P(f_i|c) \cdot P(c). \tag{3.6}
\]

Hence, the posterior class probability is the product of the prior class probability and the conditional probability of each feature \( f_i \) given the class. But, as discussed in Section 2.2, every sample used in training is a genuine sample of a single user. As there is only one class, the second term on the right hand side is uniform. Therefore, we have

\[
P(c|f_1, f_2, \ldots, f_k) \approx \prod_{i=1}^{k} P(f_i) \cdot P(c) = \prod_{i=1}^{k} P(f_i), \tag{3.7}
\]

where \( P(c) = 1 \) as there is only one class, namely the genuine user, hence is dropped. This probability measure was used as the similarity metric for the model trained for a particular user. Given the similarity distribution of validation samples, a suitable threshold was chosen with which to discriminate test data. Listing 3.2 shows a code excerpt for calculating Equation 3.7.

To avoid floating-point underflow, the logarithm of each Gaussian density was summed.
3.8. CLIENT-SERVER AUTHENTICATION FRAMEWORK

```java
// Computes the conditional probability of the test sample given the class.
public double getProb(double[] means, double[] vars, double[] features){
    double p = 0;
    for(int i=0; i<features.length; i++){
        p += logNormDistProb(means[i], vars[i], features[i]);
    }
    return p;
}
```

*Listing 3.2: Calculating the similarity metric of a test sample using Naïve Bayes.*

### 3.7.3.2 Training

For Naïve Bayes, I needed to calculate the probability of each feature given the class of the authentic user. Therefore a feature set was extracted per signature, whereas for DTW and HMM, features were extracted from each segment of a signature.

For discrete features, this can be estimated by applying relative frequencies from the training set to either a multinomial or a Bernoulli distribution. As every feature in this project is continuous, it was assumed that all features are normally distributed.

The training procedure involved calculating the mean and variance of each feature over all samples in the training set. For a new test sample, the conditional probability $P(f_i|c)$ was computed as a Gaussian density with mean and variance computed with training data.

### 3.7.3.3 Authentication Procedure

Equation [3.7] was used as the similarity metric for a test sample. The discrimination threshold was then calculated from the similarity scores of validation samples. In the following, $V_T$ is a vector of similarity scores for the true validation samples and $V_F$ is a vector for false validation samples:

$$\delta = Q_T \cdot \left( \mu(V_T) \sigma(V_T) \right) + Q_F \cdot \left( \mu(V_F) \sigma(V_F) \right).$$

### 3.8 Client-server Authentication Framework

As originally planned, a server-based authentication framework was designed. For this purpose I implemented a Java server application running Berkeley DB.
Most tasks in the authentication workflow is carried on the server side, including training, authentication and managing users’ data. Each client device only collects signature data from the user and then offloads the rest of the work to the server. This has the advantage of imposing less compute demand to the mobile processing unit, making the authentication system accessible to a wider range of mobile devices with different computational power.

Figure 3.14: Client-server framework for signature authentication.

This authentication framework involves uploading the signature data of about 35 KB to the server, but this upload delay is somewhat mitigated by the server’s fast computing capabilities. An additional advantage of this scheme is that more signature data are available for testing as signature data from all clients are stored centrally at the server.

This centralised scheme, however, introduces some security and reliability weaknesses. Specifically, it is critical in the authentication process that the server is functional, hence it is the single point of failure of the system. Moreover, as all of the users’ authentication details are stored in the server’s databases, the security consequences are severe should the server be compromised.

Listing 3.3 shows the code snippet of the example query of user data to the database.

```java
Database systemDatabase = env.openDatabase(null, "System DB", dbconf);
SequenceConfig seqconf = new SequenceConfig();
DatabaseEntry user = new DatabaseEntry();
Sequence dbseq = systemDatabase.openSequence(null, user, seqconf);
return dbseq.get(null, 0);
```

Listing 3.3: Retrieving the user data with Berkeley DB.

### 3.9 Mobile Authentication System

Following the implementation of the client-server architecture, the mobile authentication framework was developed as an optional extension. The major components of the mobile system are explained below.
There were a number of challenges involved in developing an independent device-based authentication. The limited computing power and memory of mobile processors posed a challenge, but yet a more fundamental issue was the lack of negatively-labelled signature samples to use as validation data. Both true and false validation samples are needed in computing the similarity threshold, but only true signature samples are available in a mobile environment. Next parts of this section describe how these complications were dealt with.

3.9.1 Components of the Mobile System

The high-level design of SigVerify is explained below.

![Diagram of components and interactions of the mobile authentication framework.](image)

*Figure 3.15: Components and their interactions of the mobile authentication framework.*

The **User Interface** allows a user to interact with the SigVerify authentication framework. In the system preferences, users can choose the number of samples required for training, the method of authentication and even whether between the client-server and mobile scheme. Screenshots of the system are shown in Figure 3.17–Figure 3.22.

The **Authentication Master** is the central module that arbitrates and manages the overall authentication process. Most importantly, it makes the authentication decision based on the input from the Authentication Module and the user’s data stored in the Storage Unit. User’s system preferences are communicated to the Authentication Master, which then sends them to other components.

The **Data Collection Unit** provides access to accelerometer, gyroscope and other sensing devices (if any). It collects the user’s biometric data and forwards it to the Processing Unit.

The **Data Processing Unit** performs preprocessing tasks and feature extraction on the
signature data. It is generally one of the most time- and energy-consuming steps in the authentication process, thus should be constrained not to drain system resources excessively.

The Authentication Module is a complete implementation of a classification algorithm including both training and authentication. It trains the model of the user and computes the discrimination threshold. For authentication, the module calculates similarity level of a test sample and transfers it to the Authentication Master. At present, SigVerify supports three authentication modules, but additional methods can be easily integrated to the system.

The Data Storage Unit stores information including the user’s metadata, the parameters of a trained model, the history of user’s authentication and the discrimination threshold.

3.9.2 Optimisations for a Mobile Environment

There exists a huge performance difference in processors and memory systems employed in general-purpose desktops, commercial servers and mobile devices. Perhaps not surprisingly, the latency of the system as developed on a desktop machine was unacceptable when run on a mobile device. To account for the gap in hardware performance, several optimisations were designed and applied to the system.

Need-based Feature Extraction was motivated by the observation that computing and storing every feature in advance per segment basis and returning the ones that are needed is too costly while only providing minimal benefits. In the client-server architecture, it makes sense as the server might have to service multiple requests for the same person’s signature data for different sets of features. However its benefits for an individual mobile system are limited. As a result, only the necessary features were extracted from raw signature data.

Fine-grained Data Manipulation is necessary since, given the size of signature data corpus, handling an entire dataset for each person was often beyond the size and bandwidth of the mobile device’s memory system. Therefore, manipulating the signature data, e.g. passing it as an input to the preprocessing module, is performed on an individual signature basis. Also, only the relevant signatures are queried from the database to avoid unnecessary latency from memory operations.
3.10. SUMMARY

3.9.3 Generating False Validation Samples

Unlike in the client-server architecture, there are no readily available false validation data in the device-based authentication. The only available data are samples of the genuine user.

The idea of using completely arbitrary signals as false data was quickly rejected, as the entropy of actual signatures is rather limited: a typical signature lasts 5–6s on average, the maximum range of a signature is limited by the arm’s length and signatures usually consist of letters rather than arbitrary shapes. Furthermore, a forged signature would be far more similar to the genuine one than a random sequence of values.

For these reasons, I decided to store different subjects’ genuine data in the mobile database and use them as false samples. Additionally, I perturbed authentic user’s signatures with a random variation in each dimension and timescale to create negatively labelled samples. Figure 3.16 shows the effect of random perturbation on an acceleration trace.

![Original acceleration trace.](image1) ![Acceleration trace perturbed with random noise.](image2)

Figure 3.16: Two-dimensional trace of acceleration data before and after random perturbation.

3.10 Summary

This chapter discussed the main implementation work of the SigVerify project. Starting with a brief outline of the system, it considered the mechanisms involved in preprocessing the raw data and feature extraction. Then a set of features studied in this project was explained, followed by feature selection methods used in the Evaluation chapter as well as initial testing using WEKA.
Next described was the process of choosing the classifiers and the implementation details of each algorithm. The chapter finishes with an outline of two modes of authentication supported by SigVerify, client-server authentication over a network and standalone mobile authentication.
### 3.10. SUMMARY

- **Figure 3.17:** Main page.
- **Figure 3.18:** Home menu.
- **Figure 3.19:** User enrolment.
- **Figure 3.20:** Metadata collection.
- **Figure 3.21:** System settings.
- **Figure 3.22:** Authentication modules.
Chapter 4

Evaluation

The primary goal of this project was to design and implement a secure and usable 3D gesture-based authentication mechanism. This chapter discusses both quantitative and qualitative results, showing how major objectives were met. The original success criteria are restated, followed by a visual comparison of the performance of different classification methods.

In addition, this chapter describes the performance of SigVerify from various usability perspectives with several tests and a user study. Finally, the unit testing framework for the project is described briefly.

4.1 Overall Results

The original success criteria specified in the project proposal are restated below.

Requirement 1: *the chosen classification algorithm achieves better performance than others implemented.*

Detailed evaluation of classifiers’ performance is presented in Section 4.2. In performance tests in various settings, Naïve Bayes outperforms other algorithms both in terms of EER and latency by a significant margin. As it is also a strong performer in tests with WEKA, this indeed aligns with the results from a standard implementation.

Requirement 2: *the authentication process takes reasonable amount of time to qualify as real time.*

Section 4.3 contains evaluation of SigVerify as a usable mobile system with performance criteria including time required to authenticate, battery life and CPU consumption. Overall, SigVery achieved very high usability in all aspects considered.
Requirement 3: the authentication process does not require users to have a large memory, additional hardware or technical knowledge.

A qualitative evaluation of SigVerify is presented in Section 4.4. In short, the gesture-based authentication mechanism does not require anything from the users’ point of view apart from the physical activity involved in performing signatures. Compared to existing knowledge-based mechanisms, it is a significant improvement in this regard.

In addition to the requirements above, all optional extensions were also successfully implemented. These are:

- Implementing the authentication process entirely on the mobile phone (Section 3.9).
- Authentication with a minimal number of features (Section 4.2).
- Evaluation of the usability of the authentication scheme as mobile software (Section 4.3).

### 4.2 Classification Performance

The goal of this evaluation is to estimate how often an enrolled user is successfully authenticated or incorrectly rejected. The metrics used are Equal Error Rate (EER) and Area Under the ROC Curve (AUC). The evaluation framework is executed on my Windows 8-based personal computer with Intel Core i5-2410M CPU, 2.30 GHz clock speed and 4GB RAM. The mobile test environment is my personal HTC Sensation with 1.2 GHz dual-core Scorpion processor and 768MB RAM running on Android v4.0.3 (Ice Cream Sandwich).

#### 4.2.1 Training and Testing

This section briefly introduces the exact number of samples used in training, validation and testing.

First, the optimal size of the training set was determined empirically. Given 30 authentic samples for each subject, three values (10, 15 and 20) for the training data size were tested for every classification algorithm. Figure 4.1 shows the values of EER obtained for each algorithm and training set size. A set size of 10 was found to generate the best outcome on average, and hence was chosen for the evaluation.
4.2. CLASSIFICATION PERFORMANCE

Figure 4.1: EERs obtained with varying amounts of training data (lower = better).

Figure 4.2 shows a complete set of the data used in evaluating one trained model (using signature data of one subject). Each of the six datasets contains 10 signature samples gathered from different parts of the signature corpus.

First, 30 genuine samples of the user were partitioned into three disjoint sets of equal size, shown as the blue datasets on the left.

Green datasets on the right of the figure represent the false test data in each of three Forgery schemes introduced in Section 2.5.2.

For each test subject there were multiple ways of dividing genuine samples into training, validation and test set. I arbitrarily selected 10 such different partitionings and pooled the classification results for each. Also, the results for all 20 test subjects were combined to construct a representative ROC curve that is not biased towards any person.

4.2.2 Dynamic Time Warping

Table 4.1 shows the parameter settings that were used for DTW. Among these, the adjustment window size was found to have little or almost no effect on classification performance. It was hence fixed at $1/2$. The optimal choice for non-diagonal penalty $H$ varied depending on the number of features extracted. For three other parameters, the values in bold yielded the lowest EER.

To determine the optimal feature set for DTW, the performance of each individual feature was evaluated separately. Table 4.2 shows the performance of each feature in terms of EER. The features discussed in Section 3.4 were extracted from each segment. The result shown is
averaged over all possible combinations of parameters. The number of features is one except for Bin5, Bin10 and MFCC which have 5, 10 and 13 features respectively. Clearly, the error rate subsides as the data from different axes and sensors are combined.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjustment window ratio</td>
<td>K = 1/2, 1/3, 1/4</td>
</tr>
<tr>
<td>Slope constraint</td>
<td>P = 0, 1, 2</td>
</tr>
<tr>
<td>P-norm</td>
<td>p = 1, 2, 3</td>
</tr>
<tr>
<td>Non-diagonal penalty</td>
<td>H = 5, 10, 15, 20</td>
</tr>
<tr>
<td>Training Procedure</td>
<td>Choosing the most</td>
</tr>
<tr>
<td></td>
<td>representative sample</td>
</tr>
<tr>
<td></td>
<td>Averaging by expanding</td>
</tr>
<tr>
<td></td>
<td>and compressing</td>
</tr>
<tr>
<td></td>
<td>Averaging by summing and</td>
</tr>
<tr>
<td></td>
<td>dividing</td>
</tr>
</tbody>
</table>

Table 4.1: Parameter configurations of DTW.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Acceleration</th>
<th>Gyroscope</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>Bin5</td>
<td>20.3%</td>
<td>21.4%</td>
<td>18.1%</td>
</tr>
<tr>
<td>Bin10</td>
<td>19.6%</td>
<td>18.8%</td>
<td>17.8%</td>
</tr>
<tr>
<td>Max</td>
<td>20.1%</td>
<td>22.9%</td>
<td>23.3%</td>
</tr>
<tr>
<td>Med</td>
<td>24.4%</td>
<td>18.3%</td>
<td>19.0%</td>
</tr>
<tr>
<td>Mean</td>
<td>21.7%</td>
<td>19.5%</td>
<td>18.9%</td>
</tr>
<tr>
<td>MFCC</td>
<td>20.8%</td>
<td>18.4%</td>
<td>17.8%</td>
</tr>
<tr>
<td>Min</td>
<td>21.7%</td>
<td>21.3%</td>
<td>20.6%</td>
</tr>
<tr>
<td>Rms</td>
<td>20.2%</td>
<td>20.0%</td>
<td>19.5%</td>
</tr>
<tr>
<td>Span</td>
<td>20.4%</td>
<td>21.1%</td>
<td>21.1%</td>
</tr>
<tr>
<td>Std</td>
<td>20.5%</td>
<td>21.2%</td>
<td>21.6%</td>
</tr>
</tbody>
</table>

Table 4.2: Averaged single feature performance of DTW in EER (M for Magnitude and O for Overall).

Subsequently, high performance features were combined into a feature vector (see Table 4.3). The features were extracted from all 8 axes (x, y, z, m for both accelerometer and gyroscope).

The performance of the best feature set was evaluated comparatively to the case when no feature extraction was done. Furthermore, the effectiveness of each attack scheme described in Section 2.3 was studied as a separate ROC curve.
4.2. CLASSIFICATION PERFORMANCE

<table>
<thead>
<tr>
<th>Set</th>
<th>Feature Set</th>
<th>Size</th>
<th>EER</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bin5, Mean, RMS</td>
<td>56</td>
<td>8.89%</td>
<td>0.964</td>
</tr>
<tr>
<td>2</td>
<td>Bin5, Mean, Std</td>
<td>56</td>
<td>8.43%</td>
<td>0.960</td>
</tr>
<tr>
<td>3</td>
<td>Bin5, Med</td>
<td>48</td>
<td>8.31%</td>
<td>0.958</td>
</tr>
<tr>
<td>4</td>
<td>Bin5, Med, Mean</td>
<td>56</td>
<td>7.22%</td>
<td>0.966</td>
</tr>
<tr>
<td>5</td>
<td>Bin5, Med, Mean, Std</td>
<td>64</td>
<td>6.08%</td>
<td>0.973</td>
</tr>
<tr>
<td>6</td>
<td>Bin5, Med, Mean, Span</td>
<td>64</td>
<td>8.94%</td>
<td>0.963</td>
</tr>
<tr>
<td>7</td>
<td>Bin5, Med, Mean, RMS</td>
<td>64</td>
<td>8.82%</td>
<td>0.959</td>
</tr>
<tr>
<td>8</td>
<td>Bin5, MFCC</td>
<td>144</td>
<td>11.63%</td>
<td>0.877</td>
</tr>
<tr>
<td>9</td>
<td>Bin10, MFCC</td>
<td>184</td>
<td>13.17%</td>
<td>0.850</td>
</tr>
</tbody>
</table>

Table 4.3: Performance of each feature set for DTW.

Figure 4.3: ROC Curves of DTW using varying feature extraction and forgery schemes (Red = feature set 5 in Table 4.3. Blue = raw signature data with no feature extraction).

Figure 4.3 shows the ROC curves of DTW evaluated under six different conditions. The red curves show the result when feature set 5 was used, while the blue curves indicate the result when no feature was extracted and raw values were supplied to the DTW algorithm. The effect of feature extraction for DTW is highly pronounced, as shown by the distance between blue and red curves.

For each feature extraction scheme, three different forgery classes were compared. The ROC curve for Visual Forgery is much further away from the upper left corner than that of Random Forgery. This is in accordance with the expectation that as more information is
available to the forger, the better he manages to mimic the original signature.

From the ROC curve, it can be inferred that if a false positive rate of, e.g. \( \leq 2\% \) is desired, the true positive rate will be 80\% for visual forgeries. Alternatively, if a low false negative rate (insult rate) is required, as is the case in banking and financial systems, roughly an insult rate of 5\% is achieved when the false positive rate is 95\% for limited visual forgeries.

### 4.2.3 Hidden Markov Models

For HMM, the parameters used were the number of states in the model and the number of Gaussian mixtures. No significant change in performance was observed with varying number of states. On the contrary, 8 mixtures yielded the lowest EER value.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of states</td>
<td>( S = 2,4,8 )</td>
</tr>
<tr>
<td>Number of mixtures</td>
<td>( M = 2,4,8 )</td>
</tr>
</tbody>
</table>

*Table 4.4: Parameter settings of HMM.*

The single feature performance for HMM was estimated by again averaging over all combinations of parameter settings.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Acceleration</th>
<th>Gyroscope</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>Bin5</td>
<td>24.1%</td>
<td>19.0%</td>
<td>25.5%</td>
</tr>
<tr>
<td>Bin10</td>
<td>26.6%</td>
<td>21.3%</td>
<td>19.4%</td>
</tr>
<tr>
<td>Max</td>
<td>21.1%</td>
<td>30.9%</td>
<td>22.8%</td>
</tr>
<tr>
<td>Mean</td>
<td>24.2%</td>
<td>23.9%</td>
<td>24.4%</td>
</tr>
<tr>
<td>Med</td>
<td>31.9%</td>
<td>22.4%</td>
<td>21.1%</td>
</tr>
<tr>
<td>MFCC</td>
<td>21.3%</td>
<td>25.3%</td>
<td>26.6%</td>
</tr>
<tr>
<td>Min</td>
<td>30.0%</td>
<td>22.1%</td>
<td>34.0%</td>
</tr>
<tr>
<td>Rms</td>
<td>25.8%</td>
<td>26.5%</td>
<td>26.1%</td>
</tr>
<tr>
<td>Span</td>
<td>20.7%</td>
<td>16.2%</td>
<td>16.1%</td>
</tr>
<tr>
<td>Std</td>
<td>35.8%</td>
<td>30.0%</td>
<td>31.3%</td>
</tr>
</tbody>
</table>

*Table 4.5: Averaged single feature performance of HMM in EER (M for Magnitude and O for Overall).*

Overall, HMM performed better than DTW. An interesting result was that on a single axis basis, each individual feature performed worse than DTW. If several axes were combined, the
4.2. CLASSIFICATION PERFORMANCE

Performance was significantly better than DTW. As HMM is probabilistic by nature, different results were obtained with consecutive runs even when exactly same samples were used in training, validation and testing. Hence the classification performance for HMM was averaged over 10 runs.

High performance features were combined into a feature set. The optimal feature vector for HMM was Bin5, RMS and Std, totalling a set of 56 features. The performance of this feature vector was compared with raw data.

<table>
<thead>
<tr>
<th>Set</th>
<th>Feature Set</th>
<th>Size</th>
<th>EER</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RMS, Std</td>
<td>16</td>
<td>9.56%</td>
<td>0.936</td>
</tr>
<tr>
<td>2</td>
<td>RMS, Std, Min</td>
<td>24</td>
<td>8.56%</td>
<td>0.950</td>
</tr>
<tr>
<td>3</td>
<td>RMS, Std, Min, Span</td>
<td>32</td>
<td>8.33%</td>
<td>0.953</td>
</tr>
<tr>
<td>4</td>
<td>Bin5, RMS</td>
<td>48</td>
<td>7.40%</td>
<td>0.964</td>
</tr>
<tr>
<td>5</td>
<td>Bin5, RMS, Std</td>
<td>56</td>
<td>5.21%</td>
<td>0.971</td>
</tr>
<tr>
<td>6</td>
<td>Bin5, RMS, Std, Span</td>
<td>64</td>
<td>6.87%</td>
<td>0.965</td>
</tr>
<tr>
<td>7</td>
<td>Bin5, MFCC</td>
<td>144</td>
<td>10.31%</td>
<td>0.927</td>
</tr>
<tr>
<td>8</td>
<td>Bin10, MFCC</td>
<td>184</td>
<td>12.08%</td>
<td>0.892</td>
</tr>
</tbody>
</table>

Table 4.6: Performance of HMM for each feature set.

![Figure 4.4: ROC Curves of HMM using varying feature extraction and forgery schemes (Red = feature set 5 in Table 4.6. Blue = raw signature data with no feature extraction).](image-url)
Figure 4.4 shows six different ROCs for HMM. One noticeable difference from Figure 4.3 is that the curves of three forgery classes are much densely clustered. This implies that HMM is less vulnerable to physical observation than DTW.

HMM provides 93% True Positive Rate while maintaining False Positive Rate for visual forgeries under 3%. Although this is significantly better than DTW, it cannot achieve both False Negative Rate and False Positive Rate under 5%.

4.2.4 Naïve Bayes Classifier

Unlike its competitors, Naïve Bayes does not have configurable parameters. Despite its incredible simplicity it outperformed others by far, as shown in Tables 4.7 and 4.8.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Bin5</th>
<th>Bin10</th>
<th>Max</th>
<th>Mean</th>
<th>Med</th>
<th>Mfcc</th>
<th>Min</th>
<th>Rms</th>
<th>Span</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>M</td>
<td>O</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>17.0%</td>
<td>17.7%</td>
<td>15.2%</td>
<td>14.2%</td>
<td>11.3%</td>
<td>27.2%</td>
<td>30.4%</td>
<td>24.3%</td>
<td>23.3%</td>
</tr>
<tr>
<td></td>
<td>14.4%</td>
<td>15.5%</td>
<td>18.4%</td>
<td>16.3%</td>
<td>10.0%</td>
<td>28.5%</td>
<td>32.3%</td>
<td>24.7%</td>
<td>26.7%</td>
</tr>
<tr>
<td></td>
<td>31.1%</td>
<td>26.8%</td>
<td>31.6%</td>
<td>31.1%</td>
<td>30.4%</td>
<td>20.6%</td>
<td>22.2%</td>
<td>25.6%</td>
<td>18.6%</td>
</tr>
<tr>
<td></td>
<td>16.6%</td>
<td>18.8%</td>
<td>17.7%</td>
<td>25.2%</td>
<td>10.3%</td>
<td>27.2%</td>
<td>23.9%</td>
<td>22.2%</td>
<td>18.4%</td>
</tr>
<tr>
<td></td>
<td>19.4%</td>
<td>23.7%</td>
<td>22.7%</td>
<td>25.5%</td>
<td>8.3%</td>
<td>33.3%</td>
<td>34.4%</td>
<td>26.8%</td>
<td>19.4%</td>
</tr>
<tr>
<td></td>
<td>18.2%</td>
<td>16.6%</td>
<td>21.2%</td>
<td>20.9%</td>
<td>15.4%</td>
<td>19.8%</td>
<td>22.6%</td>
<td>26.2%</td>
<td>17.3%</td>
</tr>
<tr>
<td></td>
<td>26.1%</td>
<td>32.2%</td>
<td>34.0%</td>
<td>30.9%</td>
<td>23.8%</td>
<td>23.1%</td>
<td>25.2%</td>
<td>21.7%</td>
<td>33.9%</td>
</tr>
<tr>
<td></td>
<td>18.0%</td>
<td>22.7%</td>
<td>22.7%</td>
<td>26.8%</td>
<td>10.0%</td>
<td>15.6%</td>
<td>21.1%</td>
<td>21.1%</td>
<td>17.8%</td>
</tr>
<tr>
<td></td>
<td>28.8%</td>
<td>31.4%</td>
<td>36.6%</td>
<td>34.5%</td>
<td>21.6%</td>
<td>19.9%</td>
<td>22.2%</td>
<td>20.2%</td>
<td>18.3%</td>
</tr>
<tr>
<td></td>
<td>22.6%</td>
<td>25.7%</td>
<td>20.0%</td>
<td>30.4%</td>
<td>10.5%</td>
<td>15.0%</td>
<td>21.1%</td>
<td>22.6%</td>
<td>16.4%</td>
</tr>
</tbody>
</table>

Table 4.7: Averaged single feature performance of Naïve Bayes in EER (M for Magnitude and O for Overall).

<table>
<thead>
<tr>
<th>Set</th>
<th>Feature Set</th>
<th>Size</th>
<th>EER</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bin5, Std</td>
<td>48</td>
<td>4.44%</td>
<td>0.968</td>
</tr>
<tr>
<td>2</td>
<td>Bin5, Std, Mean</td>
<td>56</td>
<td>3.09%</td>
<td>0.980</td>
</tr>
<tr>
<td>3</td>
<td>Bin5, Std, Mean, Span</td>
<td>64</td>
<td>2.78%</td>
<td>0.982</td>
</tr>
<tr>
<td>4</td>
<td>Bin5, Std, Mean, Span, RMS</td>
<td>72</td>
<td>2.38%</td>
<td>0.985</td>
</tr>
<tr>
<td>5</td>
<td>Bin5, Std, Mean, Span, RMS, Mod</td>
<td>80</td>
<td>3.61%</td>
<td>0.979</td>
</tr>
<tr>
<td>6</td>
<td>Bin10, Std</td>
<td>88</td>
<td>5.67%</td>
<td>0.97</td>
</tr>
<tr>
<td>7</td>
<td>Bin10, RMS</td>
<td>88</td>
<td>6.11%</td>
<td>0.965</td>
</tr>
<tr>
<td>8</td>
<td>Bin10, Mean</td>
<td>88</td>
<td>6.11%</td>
<td>0.971</td>
</tr>
<tr>
<td>9</td>
<td>MFCC, Std</td>
<td>112</td>
<td>7.23%</td>
<td>0.928</td>
</tr>
</tbody>
</table>

Table 4.8: Performance of Naïve Bayes for each feature set.
Interestingly, the performance of the single best feature for NBC (Std) was better than the best feature set’s performance for DTW. The combination of nine features per axis (Bin5, Std, Mean, Span and RMS) was the optimal feature vector for NBC with an average EER of below 2.5%.

For Naïve Bayes, it is impossible to use raw signature values as an input to the algorithm. Therefore, the performance of the feature set 4 was evaluated with and without preprocessing the training and testing samples. The performance improvement after preprocessing is highly distinguished in the figure as a wide gap between blue and red ROC curves.

As shown by the ROC curves, even an attacker who has physical access to the original handwritten signature can impersonate the true user with success probability of 2.5%. The true user, on the other hand, can be recognised by Naïve Bayes 95% of the times.

### 4.2.5 Overall Comparison

Overall, the Naïve Bayes Classifier clearly performed best. Figure 4.6 shows the averaged performance of three classification algorithms using the optimal feature set for each. It was possible to keep the false positive and false negative error rate of NBC both under 5%, whereas for DTW and HMM, to bring down one error rate, the other must be significantly higher.

---

*Figure 4.5: ROC Curves of Naïve Bayes using varying preprocessing and forgery schemes (Red = pre-processed data. Blue = raw signature data with no preprocessing).*
4.2.6 Constructing a Confidence Interval

For the classification results to be statistically valid, it was necessary to estimate the level of confidence with which the empirical ROC curve of each classifier estimates its true, unknown ROC curve.

For each ROC curve in Figure 4.6 an upper and lower confidence band known as Simultaneous Joint Confidence Region (SJR), introduced by Campbell [9], was constructed independently for TPR and FPR. For each point on the curve, a rectangular confidence interval centred at that point was obtained using the Kolmogorov-Smirnov one-sample test statistic.

The Kolmogorov-Smirnov statistic was used to test whether two sampled distributions are drawn from the same distribution (the null hypothesis) by considering the maximal vertical and horizontal distance in their cumulative distribution functions allowed from the given ROC curve without rejecting the null hypothesis. These distances along with corresponding confidence level are well tabulated [16]. For large set sizes ($n > 35$), some of the distances and confidence levels are as follows.

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0.20</th>
<th>0.15</th>
<th>0.10</th>
<th>0.05</th>
<th>0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Size</td>
<td>$\sqrt{n}$</td>
<td>$\sqrt{n}$</td>
<td>$\sqrt{n}$</td>
<td>$\sqrt{n}$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>&gt; 35</td>
<td>1.07</td>
<td>1.14</td>
<td>1.22</td>
<td>1.36</td>
<td>1.63</td>
</tr>
</tbody>
</table>

*Table 4.9: Kolmogorov-Smirnov critical values for rejecting the null hypothesis.*
Given the desired confidence level \( 1 - \delta \), let the corresponding Kolmogorov-Smirnov critical distances be \( d \) and \( e \) for FPR and TPR respectively. Then, for each point on the ROC curve (fpr, tpr), the simultaneous join confidence region (SJR) is given as \( (fpr \pm d, tpr \pm e) \), i.e. a rectangular region of size \( 4d \times e \). Combining the SJRs of all data points yields a global interval of confidence level \( (1 - \delta)^2 \), as TPR and FPR are assumed to be independent.

As discussed in Section 4.2.1, 10 genuine samples and 10 false samples were used in evaluating each subject’s model. Considering the ROC curve of each single test, the sample set size is 10 per axis (TPR and FPR). As the result in Figure 4.6 is pooled from 20 test subjects and 10 runs for each, the number of samples per axis amounts to 2,000. The 99% confidence distance for both TPR and FPR for \( n = 2000 \) is 0.0364. Therefore we have a global band of confidence level \((0.99)^2 = 0.98\) for each pooled ROC curve.

In Figures 4.7a, 4.7b and 4.7c the regions between the black bands on either side represent the theoretical values of all (fpr, tpr) points with confidence coefficient 0.98. In fact, Provost et al. [23] showed that while the theoretical confidence level is \((1 - \delta)^2\), the empirical bound is actually tighter: \(1 - \delta\), corresponding to 99% in the case of this evaluation. For Naïve Bayes, the upper band was out of the range of ROC curve and is not shown.

![Figure 4.7](image)

*Figure 4.7: Confidence bands around the ROC curves for three authentication algorithms.*

### 4.3 Other Performance Measures

**System latency.** The foremost usability factor for an authentication mechanism is latency, and its significance is higher in a mobile environment where users are more active and less patient. Latency was measured for both the client-server architecture and the mobile authentication scheme.
In addition to the result above, data collected on the device must be uploaded to the server for the client-server architecture. An average signature is approximately 50KB in size. Ten signatures (size of the training set) took roughly 5s to upload using GPRS and 1s with EDGE.

As expected, the latency on a mobile device was an order of magnitude greater than on a laptop. For both cases, Naïve Bayes performed significantly better than other classifiers, and the difference was even more pronounced for mobile authentication. When executed entirely on the test mobile device, Naïve Bayes required on average 5.2 seconds for both training and authentication. As the enrolment happens only once, the actual time cost of each authentication is almost negligible, thereby sufficiently qualifying as a real-time authentication module.

**Battery Usage.** As the authentication process takes a relatively short amount of time compared to the overall battery life of the device, the battery-life impact of a single authentication is negligible. Hence, only the power consumption level was for enrolment was compared. The battery consumption was traced with an application called PowerTutor\(^1\).

<table>
<thead>
<tr>
<th>Energy Usage (J)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading emails with WiFi</td>
<td>7.6</td>
</tr>
<tr>
<td>Using camera</td>
<td>2.4</td>
</tr>
<tr>
<td>Playing an mp3 file</td>
<td>3.4</td>
</tr>
<tr>
<td>DTW enrolment</td>
<td>50.7</td>
</tr>
<tr>
<td>HMM enrolment</td>
<td>329.6</td>
</tr>
<tr>
<td>NBC enrolment</td>
<td>7.6</td>
</tr>
</tbody>
</table>

\(^1\)http://ziyang.eecs.umich.edu/projects/powertutor/index.html

---

**Table 4.10:** Computation latency of each classification algorithm (ms).

**Table 4.11:** Energy usage of various tasks over their running times.
4.4. COMPARATIVE EVALUATION

The values for the first three rows are the total energy usage of the first minute of using each function. It is clear from the table that HMM is the most power hungry method, while the power consumption of NBC is comparable to that of a routine activity like reading emails.

**RAM Usage**  Along with power consumption, RAM usage of the enrolment process was monitored. Memory usage was moderate for all three authentication methods and no significant difference between the algorithms was observed. RAM usage was monitored using an application called Memory Usage.

<table>
<thead>
<tr>
<th>RAM Usage (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading emails with WiFi</td>
</tr>
<tr>
<td>Using camera</td>
</tr>
<tr>
<td>Playing an mp3 file</td>
</tr>
<tr>
<td>DTW enrolment</td>
</tr>
<tr>
<td>HMM enrolment</td>
</tr>
<tr>
<td>NBC enrolment</td>
</tr>
</tbody>
</table>

*Table 4.12: RAM usage of various tasks.*

4.4 Comparative Evaluation

In this section, the performance of SigVerify is compared with existing mobile authentication mechanisms such as passwords and PINs using qualitative criteria presented by Bonneau et al. [6]. In their paper, various Web authentication alternatives to passwords are suggested and evaluated in term of three major categories: usability, deployability and security. As SigVerify is a mobile authentication mechanism using sensors built into devices, deployability criteria for Web authentication schemes were regarded irrelevant and hence are not considered.

Table 4.13 summarises the key security benefits of SigVerify over existing knowledge-based schemes. Performance evaluation of classifiers confirms that SigVerify attains the first three security benefits. Concretely, Naïve Bayes achieved EER of less than 3% for Limited Visual Forgery (when forgers were allowed a limited number of physical observations before mimicking the signature). The effectiveness of targeted impersonation against SigVerify was also found to be in a safe range (about 4% EER for Naïve Bayes).

Table 4.14 shows the usability benefits offered by SigVerify and existing schemes. The most noticeable difference between the two is that the signature recognition mechanism does not

[https://play.google.com/store/apps/details?id=mem.usage]
require users to memorise anything. In fact SigVerify attains every usability benefit considered except one; it involves a physical action of drawing a signature.

In short, SigVerify provides significantly more security and usability benefits than passwords or PINs. The next section confirms this with a summative user study.

<table>
<thead>
<tr>
<th>Security Benefits</th>
<th>SigVerify</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resilient-to-Physical-Observation</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Resilient-to-Targeted-Impersonation</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Resilient-to-Throttled-Guessing</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Resilient-to-Unthrottled-Guessing</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Resilient-to-Internal-Observation</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Resilient-to-Leaks-from-Other-Verifiers</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Resilient-to-Phishing</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Resilient-to-Theft</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>No-Trusted-Third-Party</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Requiring-Explicit-Consent</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Unlinkable</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

Table 4.13: Comparative security evaluation of SigVerify against current mobile authentication schemes such as PINs or passwords. ( ✔ = offered; ✗ = not offered.)

<table>
<thead>
<tr>
<th>Usability Benefits</th>
<th>SigVerify</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memorywise-Effortless</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Scalable-for-Users</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Nothing-to-Carry</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Physically-Effortless</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Easy-to-Learn</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Efficient-to-Use</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Infrequent-Errors</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Easy-Recovery-from-Loss</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Table 4.14: Comparative usability evaluation of SigVerify against current mobile authentication schemes such as PINs or passwords. ( ✔ = offered; ✗ = not offered.)
4.5 User Study

As the final part of the evaluation, 24 people were recruited for a user study to assess the overall usefulness of SigVerify as an authentication system. Figure 4.15 shows the distribution of age and gender of participants in this user study.

<table>
<thead>
<tr>
<th></th>
<th>&lt; 20</th>
<th>20 - 24</th>
<th>25 - 30</th>
<th>&gt;30</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>Female</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Total</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>4</td>
<td>24</td>
</tr>
</tbody>
</table>

*Table 4.15: Age and gender distribution of participants in the user study.*

Every participant was asked to try out the mobile application SigVerify; both training and authentication. The device used in this study is HTC Sensation Z710e and the test environment was made consistent for each person to rule out any environmental bias in the result. Furthermore, each person was given the same information about the system and what they were asked to do.

From the result shown in Table 4.16, it is clear that they had very little difficulty with the overall authentication process. Similarly, the physical activity required to perform a gesture did not turn out to be a major drawback based on the users’ answers. One pleasing result is a clear preference for SigVerify over existing authentication schemes.

<table>
<thead>
<tr>
<th>Question</th>
<th>Mean</th>
<th>Mode</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of the authentication process (1=very easy – 5=very hard)</td>
<td>1.75</td>
<td>1</td>
<td>0.897</td>
</tr>
<tr>
<td>General acceptability of the system (1=very good – 5=very bad)</td>
<td>1.79</td>
<td>1</td>
<td>0.833</td>
</tr>
<tr>
<td>Willingness to use the system for sensitive applications (e.g. mobile banking) (1=very positive – 5=very negative)</td>
<td>1.92</td>
<td>1</td>
<td>0.974</td>
</tr>
<tr>
<td>Preference of the system over passwords/PINs (1=very positive – 5=very negative)</td>
<td>1.83</td>
<td>2</td>
<td>0.761</td>
</tr>
</tbody>
</table>

*Table 4.16: Results of four multiple-choice questions regarding the usability of SigVerify.*
CHAPTER 4. EVALUATION

4.6 Testing

SigVerify consists of several interdependent parts. A failure or an error in one component propagates to the entire system. Therefore, each program component was first tested individually alongside the implementation process and later in unison to ensure the validity of the system in general.

The code base was divided into six separate components and each component was verified with a set of modular unit tests. For each major module, a test case was defined with an expected output given a certain input. As Figure 4.8 shows, all major test cases successfully completed without errors or failures.

![JUnit Test Results](image.png)

Figure 4.8: The result of the unit testing framework for each component of the system.

4.7 Summary

In this chapter, I have presented a thorough security and usability evaluation of SigVerify. First, the initial requirements were recapitulated to assess the success of the entire project. I then presented a series of detailed performance evaluations of three classifiers implemented. Various combinations of features and parameter settings were taken into account by pooling the results from many runs of the algorithm.

Extending the project to enable independent device-based authentication introduced a whole
new set of complexities to the project. In terms of system latency, power consumption and RAM usage, the Naïve Bayes Classifier was a clear winner in all aspects.

Towards the end of this chapter, a comparative evaluation of SigVerify and existing mobile authentication methods was conducted, highlighting SigVerify’s advantages that would appeal to potential users. Finally, an outline of unit testing framework and the final user evaluation were presented.
Chapter 5

Conclusion

I am very glad that I embarked on this ambitious project. Its subject matter combines two fields in computer science of my primary interest: machine learning and security.

The course of this project took an abrupt turn as I started implementing the device-based authentication. Designing a practical mobile system entailed a whole new set of tasks on top of host-based authentication system.

This section reflects on the design and implementation decisions made in undertaking the project. It will complete this report with a concise recapitulation of the successful result, what was learned throughout the process and what practical and academic implications could be deduced from it.

5.1 Achievements

All major objectives have been successfully accomplished. In fact, the project went further than its initial scope and satisfied all optional security and usability requirements specified at its inception.

With SigVerify, smartphone users now have an additional authentication mechanism that has major advantages over knowledge-based schemes. The primary strengths of SigVerify are summarised below:

1. The authentication process is natural and largely effortless, as users do not have to memorise any complex knowledge.

2. The system is robust against shoulder-surfing attacks and more advanced forgeries with access to original handwritten signatures.
3. The authentication mechanism is perfectly feasible on a mobile device alone regarding aspects of system latency and memory/power consumption.

5.2 Lessons Learned

At the beginning of the project, one of my overseers advised me that preprocessing the data and extracting salient features might be even more important than choosing exact classifiers. To be honest, I did not at the time grasp the extent to which this is true. From the implementation stages towards to end of the project, I kept finding better ways of preprocessing and feature extraction, and each time I had to go back and redo the work based on my previous implementation. I will aim to conduct a proper level of research at each stage before moving on to the next in the future.

Another lesson I learned from this project is that data collection should be conducted such that the data represents reality as much as possible. If I were to go back to the initial stage, I would ask participants to record their signatures every few hours or even days, not every few minutes. I realised that data overfitting can only be prevented when each part of the design cycle—starting from data collection—is designed accordingly with care.

5.3 Future Work

Despite successfully meeting the requirements and optional criteria, a number of interesting opportunities remain for future work.

**Preprocessing:** although major irregularities found in raw signature data were eliminated by methods used in this project, I have not used any dimensionality reduction techniques such as Principal Component Analysis (PCA). A possible extension would be to perform PCA before applying machine learning algorithms to reduce the size of the dataset and improve performance.

**Classification algorithms:** support more models such as Random Forests, Multilayer Perceptrons and Support Vector Machines. It would also be interesting to apply ensemble classification methods to gesture-based authentication.

**Sample data collection:** A sample dataset of larger size, collected in a more organised way would increase the accuracy of the results and improve practical effectiveness of the system.
Bibliography


A.1 INFORMATION SHEET FOR PARTICIPANTS

Genuine Sample Collection

The following is the guideline in the data collection procedure. It is crucial that you follow these instructions carefully when performing gestures using the phone. Please make sure you understand the rules, or feel free to ask if anything is unclear.

First You are going to provide 30 samples of a unique gesture of your own choosing. The gesture need not be the same as your handwritten signature, but it should be something you can easily reproduce without effort.

Second For every five gestures you provide, please take a five minute break.

Third Please perform every fifth signature while sitting down. The rest should be done standing up.

Forgery Sample Collection

Forgery samples are collected to assess the robustness of the authentication system. Please carefully adhere to the following instructions.

Limited Visual Forgery You will mimic two different people’s gestures five times each. For each victim you will be allowed to observe two video sequences of the genuine person performing his gesture in the air, shot from different angles.

Visual Forgery You will mimic another two different people’s gestures five times each. For this forgery case you will have access to the original handwritten signature of each victim. Having just seen this, you will attempt to mimic each victim five times. Video sequences for these victims will not be revealed.
## A.2 AGREEMENT FORM

**SigVerify Data Collection Agreement**

<table>
<thead>
<tr>
<th>Name:</th>
</tr>
</thead>
</table>

1. I agree that I have read and understood the information sheet for the SigVerify project and have had the opportunity to ask questions.

2. I support this project conducted by Yunjae Lee, Part II candidate for Computer Science Tripos, by participating in this data collection.

3. I confirm that the signature samples are indeed my genuine samples and none others.

<table>
<thead>
<tr>
<th>Date:</th>
<th>Sign:</th>
</tr>
</thead>
</table>

A3. Formative Evaluation Form

SigVerify User Study Questionnaire

Thank you very much for participating in this user study. You are encouraged to ask any questions regarding any item in the questionnaire.

About you

1. Your name: ________________________________

2. How old are you? I am _______ years old.

3. Are you left or right handed?  ○ Left-handed  ○ Right-handed  ○ Ambidextrous

4. Do you usually lock your mobile phone?
   ○ Yes, I do lock my phone regularly.
   ○ No, I don’t use any authentication scheme to access my phone.

5. If you answered no to the previous question, why not use any mobile authentications?
   ○ They are slow.
   ○ I get frequent errors.
   ○ Interface is not user-friendly.
   ○ It is hard to remember passwords.
   ○ Other: ____________________________

Please pick one of the boxes below

<table>
<thead>
<tr>
<th>6a. I have experiences with gesture interfaces, e.g. Nintendo Wii or Microsoft Kinect</th>
<th>Over 10 times ○—○—○—○—○ None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>6b. If given a choice over following biometric modalities, I would choose...</th>
<th>○ Face  ○ Fingerprint  ○ Signature  ○ Iris</th>
</tr>
</thead>
</table>

| 6c. I am willing to use biometrics authentication if offered by my mobile phone | Absolutely ○—○—○—○—○ Never                  |
A.4 Summative Evaluation Form

SigVerify User Study Questionnaire

Thank you very much for participating in this user study. You are encouraged to ask any questions regarding any item in the questionnaire.

About you

1. Your name: ____________________________________________
2. How old are you? I am _______ years old.

Please fill in the answers to the following questions

3a. I found the overall authentication process with SigVerify straightforward and easy to follow.
   □ Strongly Agree
   □ Agree
   □ Neutral
   □ Disagree
   □ Strongly Disagree

3b. Other comments.
   ____________________________________________

4a. I think SigVerify is generally usable in most social situations or contexts.
   □ Strongly Agree
   □ Agree
   □ Neutral
   □ Disagree
   □ Strongly Disagree

4b. Other comments.
   ____________________________________________

5. If I can download SigVerify on my smartphone, I will use it.
   Definitely □—□—□—□—□ Absolutely not

6. I prefer SigVerify to passwords or PINs.
   Definitely □—□—□—□—□ Absolutely not

7. Other general comments about SigVerify
   ____________________________________________
   ____________________________________________
Computer Science Tripos Part II Individual Project Proposal

3D Gesture-Based Mobile Authentication

October 24, 2013

Project Originator: Prof Ross Anderson

Resources Required: See attached Project Resource Form

Project Supervisor: Prof Ross Anderson and Laurent Simon

Signature:

Director of Studies: Malte Schwarzkopf

Signature:

Overseers: Dr Markus Kuhn and Dr Neal Lathia

Signatures:
APPENDIX B. PROJECT PROPOSAL

B.1 INTRODUCTION AND DESCRIPTION OF THE WORK

Mobile phones are an essential part of our daily lives. We use them for a variety of purposes and those devices often contain personal and sensitive information. Also, smartphones are widely being used for secure applications such as online shopping and bank transfers. Yet, PINs, passwords or other user identification mechanisms on mobile devices are vulnerable to a number of security threats. For example, user interfaces on those mechanisms are often limited and they are very easy to peek and forge. Also, such knowledge-based mechanisms are lacking in usability in that they require users to memorise arbitrary and complex information. Hence, there is a clear need for a better means to authenticate smartphone users.

Biometrics attempts to identify people with some parts of their behaviour or physiology. It is desirable in the sense that one can be authenticated by his natural characteristics without serious effort or difficulty. Nowadays, mobile devices contain several sensors including 3D accelerometers and gyroscopes, and there has been some research effort to develop a new biometric authentication mechanism using those sensors. For example, it has been shown to be possible for a smartphone user to prove liveness by initialising a transaction with a 3D phone gesture [18]. However, out of three machine learning algorithms introduced in the paper, only two were actually tested and their results compared. This project will attempt to explore other classification mechanisms and compare the results with each other.

The aim of this project is to develop a secure and usable 3D gesture-based authentication mechanism. We use a 3D gyroscope and an accelerometer to collect a sample of user’s signature in midair, which can be used to authenticate its owner via machine learning algorithms. Compared to a 2D handwritten signature on paper, it is much harder to replicate a 3D signature just by looking someone drawing it in the air.

Data collection will be implemented at the client side and the actual authentication mechanism will be implemented on a server, which the phone will communicate with via a separate channel. As an unidentified user provides his signature sample, the data collected by the gyroscope and accelerometer will be sent over to the server side for authentication. Optionally, if time budgets permit, it would be interesting to investigate a phone-based authentication, unlike in [18]. I plan to look at this as an optional extension to the project.
B.2 STARTING POINT

I have a good working knowledge of C and Java. Last year, I have developed an Android application as a group project in Part IB. Also, I have studied Markov Models, Bayesian Probability Theory and Artificial Intelligence in Part IA and IB. This experience should be useful in learning some of the machine learning algorithms.

In the initial part of the project, WEKA will be used to choose the best machine learning algorithm to classify 3D signatures, but ultimately those algorithms will be implemented in this project.

B.3 STRUCTURE AND SUBSTANCE OF THE PROJECT

The authentication process is composed of two parts: enrollment and authentication. In the enrollment process, a user provides a set of signature samples to the system. Machine learning algorithms are then applied to create a model out of the enrollment samples. When an unknown user provides his signature, it is only accepted if its similarity level with the model is above a certain threshold.

Several machine learning algorithms will be used in creating a model. While the exact algorithms to be implemented will be chosen in the course of this project, the following three are good candidates: Dynamic Time Warping, Hidden Markov Models and Naïve Bayesian Classifiers. Dynamic Time Warping is a method to minimize the difference of two sequences by varying the time axis of the datasets. Given two series of vectors, DTW finds the cheapest temporal alignment between them. The model is trained by choosing an enrollment sample. Then the cost of the cheapest path is used as metric for similarity and the threshold is chosen using the enrollment samples.

A Hidden Markov Model statistically models the gesture as a Markov process. To classify a gesture using a HMM, the enrollment samples are used to generate a HMM. Subsequently, we calculate the probability that the model generated the observed sequence. The similarity metric for a HMM is the the logarithm of the probability penalized by the length of the observed sequence. If the metric is within the predefined threshold, the observed sequence is accepted.

To classify a sample with Naïve Bayesian Classifiers, we first convert the sample to a
feature vector. For each feature vector, the class with the maximum probability is assigned using the Bayesian theorem.

In the early phases of the project, I will recruit and select 10 ~ 15 participants for data collection. My personal HTC Android phone will be used to collect gyroscope and accelerometer data. The participants will be asked to provide six iterations of their signature, each consisting of five samples. There will be a gap between recording each iteration to prevent the data from being overtuned, and this is also to prevent causing fatigue to the participants. The 3rd iteration will be used as a training set as it is the most representative of all samples. Validation samples will be 2nd, 4th and 5th iterations while the last iteration will be used in testing. The first iteration will contain some variance with regards to the rest due to the lack of training, hence will be dropped. In addition, the participants will be asked to provide five attempts at each other’s signature after viewing it twice. The forged samples, as well as some arbitrarily generated sample data, will be used as negatively-labelled parts of my dataset.

After the data collection, the signature samples will be pre-processed before machine learning algorithms are applied for classification. This would involve taking out the effect of gravity in accelerometer readings and extracting the most relevant features to simplify the dataset.

In evaluating the project, the following metrics in biometric mechanisms will be used. Failure-to-Enroll (FTE) and Failure-to-Authenticate (FTA) measure the usability of the mechanism in terms of the rate of users who cannot use the mechanism and the rate of users who cannot authenticate due to possible error. Also, False-Negative-Rate (FNR) measures the times false users are accepted as authentic. On the other hand, False-Positive-Rate (FPR) measure the rate of genuine users rejected as false. True-Positive-Rate (TPR) is the rate of false users classified correctly, and is one minus FNR. The FPR and the TPR are highly correlated, and the Receiver Operating Characteristic (ROC) curve is used to visualise the trade-off between the two. Generally, the classifier system is considered to be efficient if the area under the curve is close to one. The last part of the evaluation will be a user study about the usability of the system in general.

This project can be broken down into several main stages, each consisting of several sub-tasks that need to be completed.

1. Collecting and compiling signature accelerometer/gyroscope example data from human participants.
2. Preprocessing the collected signature data, i.e. reducing its dimension.
3. Applying various machine learning algorithms in WEKA and choosing the best one in
B.4 SUCCESS CRITERIA

The primary success criteria for this project are illustrated in the following. They must be completed for this project to be considered a success.

- The chosen machine learning algorithm is more secure and generalises better than other algorithms in terms of metrics such as TPR and FPR.
- The mechanism takes reasonable amount of time to authenticate a user.
- Users can be authenticated with convenience, i.e. there is no need for technical knowledge, special hardware or great memory for identification.

Also, following items are possible extensions of the project, which would increase the usability of the mechanism.

- Minimizing the set of features needed for an accurate authentication of a user signature.
- Implementing authentication process on the phone and developing a stand-alone application that collects data and authenticates a user without the need of a server.
- Optimising the mobile side authentication so that it would consume minimum battery life to authenticate a user.
APPENDIX B. PROJECT PROPOSAL

B.5  RESOURCES DECLARATION

Project development will largely be done on my Windows 8-based personal computer with Intel Core i5-2410M CPU, 2.30 GHz clock speed and 4G RAM. To safeguard against machine failure, I will backup each iteration of my project regularly in my personal DropBox account. Also, I will save my entire work in the Computer Laboratory MCS so I could use that to continue development in case of hardware failure. As for version control, I will use BitBucket and Git.

For data collection, I will use my personal HTC Sensation, running on Android v4.0.3 with Dual-core 1.2 GHz CPU. Should it fail in the duration of this project, I will purchase the same one from Amazon.

I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.

B.6  WORK TIMETABLE

• 11th Oct ~ 25th Oct
  – Plan out the project in detail with supervisors and DoS.
  – Set up LaTeX, WEKA and Android SDK.
  – Plan out collecting the data samples from the user.
  – Submit Project Proposal.
  – Acquire an Android phone. Set up and familiarise with Gyroscope/Accelerometer API.

• 26th Oct ~ 8th Nov
  – Set up the communication channel between the phone and the laptop computer.
  – Study machine learning algorithms via textbooks and online courses.
  – Collect and organise accelerometer/gyroscope data from human participants.

• 9th Nov ~ 22nd Nov
  – Finish collecting data from test participants and compile the sample data, separate them into three parts: training, validation and test sample.
B.6. WORK TIMETABLE

- Perform any pre-processing necessary to extract the most relevant data from the samples collected.
- Contingency time for over-run of collecting data from participants.

• 23rd Nov ∼ 6th Dec
  - Try out different Machine Learning(ML) algorithms on WEKA using the training set, refine each algorithm with the validation set.
  - Choose 2~3 ML algorithms that are most accurate classifiers.
  - Build the first prototype using the implementation of the chosen algorithms on WEKA.
  - Evaluate the performance of the prototype, e.g. security and usability.

• 7th Dec ∼ 20th Dec
  - Implement the first iteration of the first machine learning algorithm of my choice. Implement the training algorithm for enrollment and the classification algorithm for authentication.
  - Work out the model acceptance function and similarity threshold for the first ML algorithm.
  - Find and use the right mathematics toolkit needed for the algorithm.

• 21st Dec ∼ 3rd Jan
  - Test and debug the first algorithm. Evaluate its performance and identify room for optimisation.
  - Start implementing the second ML algorithm for classification, again work on classification and training.

• 4th Jan ∼ 17th Jan
  - Add final implementation to the second ML algorithm code, try various input signature users samples to evaluate its performance accurately.
  - Produce a working prototype with two identification mechanism.
  - Catchup time for any incomplete tasks so far.

• 18th Jan ∼ 31st Jan
  - Implement the classification and training algorithm for the final classifier mechanism.
  - Debug and test the last algorithm, evaluate its performance.
APPENDIX B. PROJECT PROPOSAL

- Generate the last prototype.
- Write up and submit the progress report.

• 1st Feb ∼ 21st Feb
  - Start on implementing optional extensions for the project. Implement authentication process on the mobile device and put it altogether into a standalone application.
  - Start on the first draft of the dissertation with a general outline.

• 22nd Feb ∼ 14th Mar
  - Complete the dissertation with figures, tables, results from user study and evaluation.
  - Typeset the dissertation into a format ready to be submitted.

• 15th Mar ∼ 11th Apr
  - Finish the dissertation.
  - Seek feedback from supervisors and director of study.
  - Fix any typos and typesetting issues.

• 12th Apr ∼ 17th May
  - Make final changes to the dissertation.
  - Submit dissertation.