Universidade do Minho
Escola de Engenharia

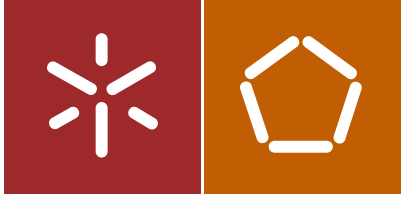
Sérgio Emanuel da Cruz Lopes

Touchscreen Biometrics for Continuous Authentication

December, 2014

Sérgio Emanuel da Cruz Lopes Touchscreen Biometrics for Continuous Authentication

UMinho | 2014



Universidade do Minho
Escola de Engenharia

Sérgio Emanuel da Cruz Lopes

Touchscreen Biometrics for Continuous Authentication

Master's Dissertation
Integrated Master's in Communications Engineering

Project done under orientation of
Professor Henrique Manuel Dinis dos Santos

December, 2014

DECLARAÇÃO

Nome: Sérgio Emanuel da Cruz Lopes

Endereço electrónico: a58656@alunos.uminho.pt

Telemóvel: 934492065

Número do Bilhete de Identidade: 13599566

Título da dissertação

Touchscreen Biometrics for Continuous Authentication

Orientador:

Professor Doutor Henrique Manuel Dinis dos Santos

Ano de conclusão: 2014

Designação do Mestrado:

Ciclo de Estudos Conducentes ao Grau de Mestre em Engenharia de Comunicações

Universidade do Minho, ___/___/_____

Assinatura: _____

Acknowledgements

This dissertation marks the end of a long journey, full of great memories and accomplishments. However, none of that would be possible without the precious help of some persons, to which I would like to thank.

First, I would like to thank my supervisor, Professor Henrique Santos, for all the help and advisors that he gave to me, for all the encouragement words and for all the joy that he transmits to all around him. Certainly, things would have been harder without him.

Secondly, I want to thank my friends and colleagues, for the good memories, for the fellowship and for the mutual aid that characterized these past five years.

I thank my family for all the support, to my parents for the courage they always gave me and for that "no" they never told me, and to my brothers for always being by my side.

Lastly, I want to thank my beloved Patrícia, for all the love, support and affection and for all those smiles. Thank you for sharing this journey with me.

Abstract

Nowadays, touchscreen devices are part of people's daily tasks and there is, progressively, less need to use traditional computers, whether to work or entertainment activities. Consequently, these devices contain an increasingly higher quantity of sensitive information that must be protected from unauthorized access. Generally, these devices have only a single layer of authentication and, once passed this phase, it is assumed that the device is used only by its owner or authorized user. However, no matter how strong is the authentication method, the device is still vulnerable to attacks from malicious users who gain access to it in an unlocked state. This way, is pertinent to adopt techniques that provide continuous authentication of users.

In this dissertation is proposed a system for continuous authentication in devices with touchscreen, aiming to serve as a proof of concept about the possibility of using the interactions of users with the touchscreen as a behavioral biometric characteristic. Was developed a mobile application to collect the touch data of some controlled gestures, performed by different users, and to extract a selected set of features, used to build their biometric template. The classification module was simulated through machine learning software. Lastly, are presented the system evaluation results.

Globally, the results show that the system, using the selected features and classifier, is not capable of perform a strong continuous authentication. In fact, was verified that the system behaved quite distinctly with different users. Besides the results, it is believed that, with the right set of features and classification techniques, the system has a promising future.

Resumo

Atualmente, dispositivos com *touchscreen* fazem parte das tarefas quotidianas das pessoas e há, progressivamente, menor necessidade de utilizar os computadores tradicionais, quer para trabalho quer para entretenimento. Consequentemente, estes dispositivos contém uma quantidade cada vez maior de informação sensível que deve ser protegida de acessos não autorizados. Geralmente, estes dispositivos têm apenas uma única camada de autenticação e, uma vez ultrapassada esta fase, assume-se que o dispositivo é usado pelo dono ou por um utilizador autorizado. No entanto, independentemente do quão forte seja a autenticação no início da sessão, o dispositivo continua vulnerável a ataques de utilizadores mal-intencionados que ganhem acesso ao mesmo num estado desbloqueado. Desta forma, é pertinente a adoção de técnicas que proporcionem uma autenticação contínua dos utilizadores

Nesta dissertação é proposto um sistema de autenticação contínua em dispositivos com *touchscreen*, com o objetivo de servir como prova de conceito sobre a possibilidade de usar as interações de utilizadores com *touchscreens* como uma característica biométrica comportamental. Foi desenvolvida uma aplicação móvel para recolher os dados de alguns gestos controlados, executados por diferentes utilizadores, e para extrair um conjunto de atributos selecionados, usados para construir o modelo biométrico dos utilizadores. O módulo de classificação foi simulado através de *software* de *machine learning*.

Globalmente, os resultados mostram que o sistema, usando os atributos e o classificador selecionados, não é capaz de desempenhar uma autenticação contínua robusta. De facto, verificou-se que o sistema se comportou de forma bem distinta com utilizadores diferentes. Apesar dos resultados acredita-se que, com os atributos e classificadores certos, o sistema tem um futuro promissor.

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

API	Application Programming Interface
CSV	Comma Separated Values
DET	Detection Error Trade-off
DTW	Dynamic Time Warp
EER	Equal Error Rate
FM	False Match
FMR	False Match Rate
FNM	False NonMatch
FNMR	False NonMatch Rate
kNN	k-Nearest Neighbor
Ms	Millisecond
PC	Personal computer
RBF	Radial Basis function
ROC	Receiver Operating-Characteristics
SVC	Support Vector Classification
SVM	Support Vector Machines
SVR	Support Vector Regression
PCA	Principal Component Analysis
PIN	Personal Identification Number
Px	Pixel
TM	True Match
TMR	True Match Rate

TNM	True Non-Match
TNMR	True Non-Match Rate
UI	User Interface
WEKA	Waikato Environment for Knowledge Analysis

1. Introduction

1.1 Context and Motivation

The world is currently in the era of mobility. The number of mobile devices, such as smartphones and tablets, has increased significantly in the past years and has surpassed, by far, the traditional computers in sales [1]. Due their evolution in terms of processing capabilities, storage capacity, applications and services offered, people are choosing mobile devices to perform their daily activities, rather than desktops or laptops. In US, the internet usage through mobile applications has outstripped computers in January of this year [2] and studies predict that, in 2017, smartphones and tablets will have 87% of the market share [3].

Information security always was, and always will be, a critical aspect whether at organizational or at personal level. With the usage growth of mobile devices, more and more sensitive information is accessed and stored in them, which leads to an upcoming interest of malicious users to explore vulnerabilities and gain access to that information, amplifying the necessity of enhanced procedures concerning the access control of these devices. In contrast with this necessity, most devices only provide user authentication based in secret challenges, such as PIN code or password insertion or drawing a defined pattern on the screen, which are vulnerable to *smudge attacks* and *shoulder-surfing attacks* [4]. Fortunately, there is a growing tendency to use biometrics to authenticate users, like fingerprint matching and face recognition [5], although they are still easily tricked. However, most of the utilization boils down to quick but frequent tasks, like checking the mail inbox and communication activities, and for those these authentication methods could be very inconvenient. People tend to give more importance to usability rather than security so, to not be bothered by a difficult authentication challenge every time they want to use the device, they often opt to insert easy secrets, increase the lock timeout, or even use no authentication at all [6]. Furthermore, this mechanisms just provide authentication at the session login phase, after that the device is vulnerable, for example, a user can leave his device unlocked while he takes a short break and a malicious individual can pick it and do whatever he wants with it. This is a very common reality, studies suggest that 60 to 70

percent of the attacks to information systems are made by known persons that had the opportunity to access a device after authentication phase [7].

With the implementation of continuous authentication systems it is possible to solve those flaws. After an enrollment phase, where they gather data about one or more behavioral characteristics of legitimate use, it is created a biometric profile. Then the system, continuously, monitors the activity in the device by matching the interactions of the one using it with the profile of the owner. Hence is guaranteed that the device is used always by legitimate user.

The fact that behavioral biometrics can be collected non-obtrusively [8] is what makes this possible. In these systems, users can not be aware that their behavior monitored and sampled, so that intruders don't change their actions before the matching completion. Behavioral biometrics don't need any additional hardware to be collected [8], which also makes them suitable to be used in mobile devices.

Is possible to find in the literature works regarding the use of behavioral biometrics with touchscreen devices, but for authentication only at the login. Likewise, several projects were conducted, and some commercial solutions are on the market [7] [9], using behavioral biometrics to continuous authentication, but not applied to touchscreen devices. There are only a few works that combine both and with encouraging results. [6], [10]–[12]

1.2 Goals

The main goal of the project described in this dissertation is to determine if it is possible to implement a system for continuous authentication using touchscreen behavioral biometrics. In order to do so there were various sub-goals to achieve:

- Propose the system architecture and functioning;
- Determine which gestures would be performed and which features would be extracted from their touch data;
- Develop an application, to run in a mobile device, in which users would perform the gestures.

- Through the application, collect the touch data from each gesture, extract the wanted features and generate the biometric template, saving it on the device's storage.
- Proceed to the classification of the stored samples, using machine learning classification software tuned for this project.
- Perform the evaluation of the system based on the classification results.

1.3 Research Methodology

The first task was to identify the problem and the objectives, and to search works in the literature to serve as a basis for this project. The search was made through keywords related to the project, such as "behavioral biometrics", "continuous authentication", "touchscreen biometrics" and "machine learning classifiers". Through the search was built a bibliographic pool.

Afterwards, was performed an extensive study on related works to find similar problems and the way that were solved, which were the steps to follow and the critical decisions taken. Was conducted a serious study also on the Android API documentation in order to understand how the application would be made.

In all the experimental processes, all steps were conducted carefully and more than once if needed.

1.4 Document Organization

In this first chapter is made a brief introduction to the project, as well as its objectives. It is also present the research methodology and the document organization.

In chapter 2, is given the state of the art regarding biometrics and biometrics systems. First are presented the theoretical concepts about biometrics, and about the architecture and operating mode of the biometric systems in the access control context. It is also presented some works related to this project.

Chapter 3 is related to the data collection and features extraction. There are justified every decision taken regarding that aspect, and is shown which were the gestures and features analyzed. In this chapter is also present a description about the experimental scenario and the mobile application development.

Chapter 4 presents all the procedures made in the classification process, such as the datasets data pre-processing, the classifier chosen, and lastly the training of the classifier and the classification itself.

In chapter 5 are shown the results of the project and its respective analysis, are also shown some additional tests that were made in order to understand the results and to see if was possible to obtain better results.

Lastly, in chapter 6, is presented the conclusion of all work. Are resumed the accomplishments of the project and which is the future work to be made.

2. Biometrics and Authentication

Human beings have unique biological traits, physical characteristics and natural behaviors, which distinguish them from the others. Within the technological and security scope, the use of those traits to recognition of individuals is referred as *biometrics*.

This chapter aims to provide an insight about the use of biometrics for continuous authentication purposes, in a progressive way: initially is given an overview about the advantages of biometrics when compared with traditional recognition means and the most popular biometrics in use, followed by a description of the components of a typical biometric system and how are the biometric characteristics used by them and, lastly, about the use of touchscreen biometrics in the context of continuous authentication.

2.1 Biometrics Overview

The evolution of technology and information systems increases the security concerns. The presence of biometrics in authentication systems is growing day by day because they introduce several benefits.

User authentication, in most cases, is still based on user knowledge, like passwords and PINs, or user tokens, such as access cards [13]. There are serious vulnerabilities in using passwords. If configured too long or too difficult, passwords and PINs can be easily forgotten. Because of this, users tend to choose basic passwords, such as names and birthday dates, which can be guessed or obtained by brute force dictionary attacks. If strong passwords are used, they are often written down in a paper in visible places [14]. To be secure it is recommended that passwords have numbers or special characters in it and should be renewed frequently, however, most of the times that doesn't happen. In a survey made in [15] is shown that almost 3 in 4 users rarely change the passwords and that only 17% combine letters, numbers and symbols. Is also shown that 42% of the survey population uses only one password for all their services and that almost half of them have at least one password shared with three or more persons. This last aspect is common to tokens, sometimes users share their access object with colleagues, and because of it a system cannot be sure if the individual using it is the

legitimate. Concerning tokens, these can also be easily stolen or lost. A major concern is also the fact that if one password or token is intercepted, all company security system could be compromised. By the contrary, biometrics are based on "what users are", they can't be forgotten or lost and are much more difficult to forge, copy and share. User don't need to memorize secrets or carry any object, just need to be present at the recognition moment. [14].

To be eligible as a biometric characteristic and to be used in a biometric system, the user traits must meet these requirements [16], [17]:

- Acceptability: People should accept provide that characteristic to the biometric system;
- Circumvention: It must be robust enough to not be easily fooled by fraudulent methods, like trait imitation;
- Collectability: Should be possible to measure quantitatively, without causing any inconvenient;
- Distinctiveness: The characteristic has to be sufficiently distinct among different user;
- Performance: The resources usage and the matching accuracy must comply the system requirements;
- Permanence: The trait should not change constantly in a certain period of time;
- Universality: Everyone should have the characteristic.

There are a considerable number of biometric characteristics, some are already in use by commercial solutions, while others are still investigation subjects. They are often classified as physiological, such as fingerprints, retina or palm scan, or behavioral, like keystroke dynamic, voice, gait or even odor. Beyond that, they can be classified as collaborative, if the user know about the process, or as furtive, if the characteristic extraction is done without user knowledge or consent [18].

2.1.1 Physiological Biometrics

Physiological biometrics are based in characteristics directly retrieved from physical parts of the user's body. Next is presented a briefly description of some of the most common and well-stablished biometrics of this type.

Fingerprint

A fingerprint is a pattern of interleaved ridges and valleys on the surface of a fingertip. These patterns are unique for each person's finger, even twins have different fingerprints. The fingerprint matching is a technique being used from long time to personal identification, whether to civil registration or criminal investigation, being, undoubtedly, the most used biometric characteristic, with a higher acceptance level by people. The accuracy provided in recognition is appropriate for verification, although may consume too much computational resources in identification processes. As the implementations grow in number and diversity of applications, the price of the scanners is becoming more affordable. [10] [11]

However, fingerprints scanners can't differentiate well an actual finger of a "dead finger", and there are already tutorials to synthesize fingerprints, undermining their reliability. [19] Besides, there are some situations when fingerprint matching is not possible due modifications on persons fingers, due aging or labor reasons.

Figure 2.1 - Different fingerprint patterns
taken from http://www.sfis.ca.gov/how_afis.html

Face

Face recognition is an activity that humans always used, naturally, to recognition of individuals. This process starts with an image capture of the user's face, then that image is compared with models stored in a database. One of the most typical implementations is based on the comparison of positions of facial attributes (eyes, chin, nose, lips and eyebrows) and their special relations [14].

This technique has problems regarding the image capture. Different head orientations, illumination changes, face accessories or even aging can damage the recognition. Therefore is necessary to adapt the system to all these conditions before capturing faces.

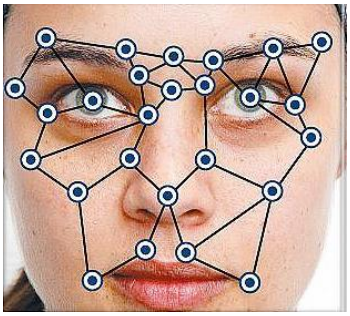


Figure 2.2 - Face recognition
taken from http://www.dailytelegraph.com.au/news/national/nsw-government-recording-features-for-facial-recognition/story-e6freuzr-1225874819392?nk=3395e9e1099503b1a0ce770b267cb510

Iris

Similarly to face recognition, iris scan is also a non-intrusive technique. Iris is the annular region of the eye. Iris, contrasting with other characteristics last for the lifetime [13], and its texture carries very complex and distinct information. The process also consists in record an image to be compared later. It has a very low false accepted rate, which is good to identification. [13]. However, not all people is willing to put an eye in a sensor, so this characteristic has a very low acceptance level. [14]

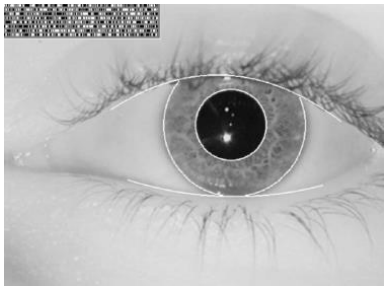


Figure 2.3 - Iris pattern, taken from [20]

Retina

Retinal scan techniques compare the vasculature of the back part of the eye. The pattern composed by the vasculature is unique and it has great accuracy levels. Retinal scans are not easily tricked because the vasculature is very difficult to replicate. It involves user cooperation and a considerable effort and, like iris scan, the user has to look directly and very closer to the sensor, which can lead to an uncomfortable situation, decreasing the user acceptability. [11] [16]

Figure 2.4 - Retina vasculature,
taken from http://www.eastoneyecare.net/services/New-Services-Page,440166

Hand Geometry

The recognition of the geometry of a hand is made by analyzing the hand measurements, such as shape, size of palm and length and width of fingers. It can be made with these measures only, or by fixing points in the captured image and calculate the distances between the points and the features. [19]. These techniques are very simple and effortless, and are not biased by bad skin conditions of user's hand like on fingerprint case.

On the other hand, hand measures are not a very discerning factor, they can work in small population but cannot be implemented in large scale because are big risks of not being able to distinguish users. More, it can't be used on children because they are in constant growth. Other disadvantage lies on the size of the scanner, it's too big to be integrated in relatively small devices as happens with fingerprint scanners or the built-in webcam in laptops. [14]

Figure 2.5 - Hand geometry scanning,
taken from http://fingerprinting.umwblogs.org/adoption/

Face and Hand Thermogram

The body temperature is not the same in all members of the human body. Using infrared cameras is possible to obtain a heat print of the user, which is unique, being more used in hands and face. In these systems is necessary to pay attention to the capturing scenario because there can't be present heat generator objects.

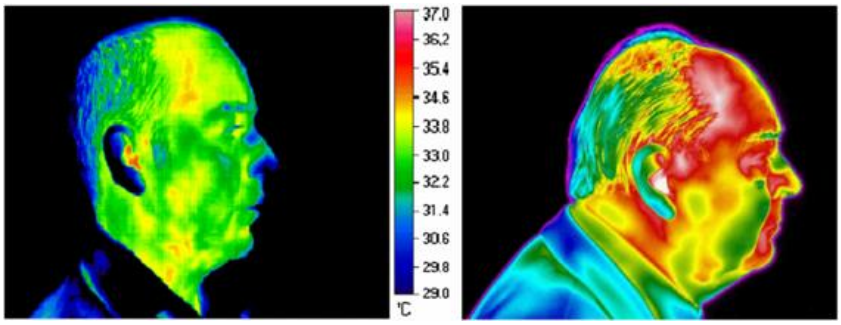


Figure 2.6 - Two different thermograms,
taken from http://hplusmagazine.com/2013/02/19/extending-the-human-sensorium-part-i-touching-the-invisible/

2.1.2 Behavioral Biometrics

Behavioral biometrics are usually less popular than physical ones, however they have some advantages. In contrary to the physical biometrics, these don't need any special hardware to be collected, which makes them cheaper, and that can be done furtively or non-obtrusively. [8]

Voice

Despite being behavioral, voice has a physic origin. It's due to different physiological characteristics that user have different voices. The process is made by converting the voice signal in an amplitude spectrum, analyzing the location and size of the spectral peaks. There are three types of voice recognition systems, depending on user's speech freedom: fixed text systems, where the speaker says a defined word, to enrollment procedures; text dependent systems, where the user has to say a phrase or a set of phrases and, finally, text independent systems, where the user says whatever he wants. This biometric characteristic is integrated with ease, and in many application, as the material that it is needed is just a microphone. [8]

Gait

Gait is a muscle control-based and complex spatial-temporal biometric, and translates the peculiar way that a user walks. It is not much distinct between users, being just enough to allow verification but in low security levels. The acquisition is made by video-sequence footage and has in account the arm swing, walking rhythm, bounce, steps length, vertical distance between head and foot, distance between head and pelvis and maximum distance between both feet [8]. This characteristic can be severely affected by injuries that compromise the natural walking behavior.

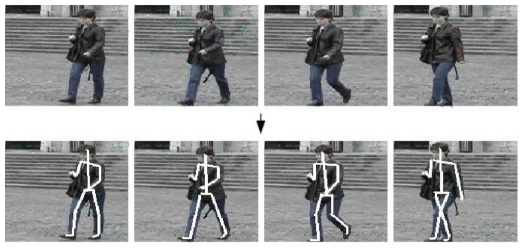


Figure 2.7 - Gait analysis, taken from http://homepages.inf.ed.ac.uk/rbf/CVDICT/cvg.htm

Signature

Signature verification are a widely accepted mean of identifying people. There are two types of verification that can be used in these systems, depending on the data collection procedure. If the signature is scanned after the signing moment it is called the offline or static verification. If the signature is collected in real time is the online or dynamic verification and, in this case, special equipment is needed so the dynamics of the signature can be collected. In the static way are collected the trajectories of the signature, while in online verification is also collected the pen pressure, acceleration and pen tilts. Consequently, dynamic verification presents better results that static verification [8].

The downside is that this is a characteristic easily replicable by attackers, fooling the system. Besides, people's signatures change over time and even the legitimate user doesn't perform his own signature exactly the same way more than once. [14]

Figure 2.8 - Online signature verification, taken from
http://www.azcentral.com/business/articles/2008/05/28/20080528biz-SkySong0529-ON-CP.html

Keystroke dynamics

Each person has its own typing pace, and its own way of pressing the keys. Keystroke dynamics is a behavioral biometrics that explore those differences. It can be used as for verification as for identification, requiring different amounts of samples. The keystroke features mostly about times: time duration between keystrokes, inter-key strokes and dwell time (time pressed down); but also measure the typing speed, sequence of errors, use of numpad, the order in which user presses *shift key* to and pressure. These systems can be used in login

authentication or collect data furtively to perform continuous user authentication, as will be seen ahead. [8] [11]

Figure 2.9 - Keystroke analysis,
taken from http://www.deakin.edu.au/research/cisr/research-areas/pma-lab.php

Mouse Dynamics

The same way that users tend to type on keys in a unique rhythm, the way that users use the mouse is also unique. By monitoring the mouse action is possible to create a user profile. Movement, drag and drop, point and click and stillness are among the possible actions to monitor [8]. These systems can also be used to perform continuous authentication.

2.2 Biometric Systems

"A biometric system is essentially a pattern-recognition system that operates by acquiring biometric data from an individual, extracting a feature set from the acquired data, and comparing this feature set against the template set in the database" – Jain, 2004

Through the previous definition, it is clear that a biometric system comprises four main modules: biometric sensor, feature extraction module, classification module and database. [11] [18]

2.2.1 Architecture

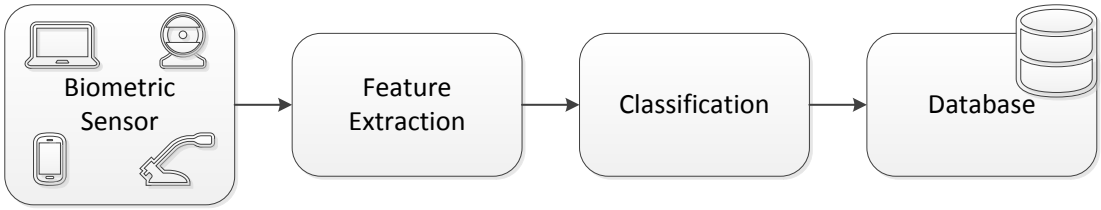


Figure 2.10 - Components of a Biometric System

1. Sensor Module

It's through the sensor that users interact with the system. It collects the biometric data of the individuals. Sensors can be special hardware, purpose made to that task, like fingerprint or hand scanners, or built in modules in devices like webcams, microphones and keyboard in laptops. In the case of this dissertation project, the sensor will be the touchscreen of the mobile device.

2. Feature Extraction Module

The feature extraction module is one of the most important in the entire system. In this module the data collected with the sensor is processed and a set of biometric features is extracted. Those features should be chosen in order to discriminate one user from the others, in other words, they must be the features with more variance among their samples.

Generally, between this module and the previous one is made a quality check before extracting the features. The quality check serves to prevent compromised samples of being part of the user biometric profile, taking the risk of jeopardize the classification.

3. Classification Module

The classification module is where the extracted features are compared with the templates stored in the database to make a decision. The matching is made through classification algorithms, usually called classifiers. In order to correctly match the samples, the classifiers have to be trained, generating the user templates.

There are many classifiers in use nowadays. Despite having the same main goal, there are classifiers more appropriate than others in a specific domain, some are better in text classification (i.e. email spam), others in image classification. Hence, was made an investigation work about which classifier would best serve the project. After analyzing the literature, there were four classifiers who stood out. Namely, K-Nearest Neighbors (kNN), Naïve Bayes, Neural Networks and Support Vector Machines (SVM).

When comparing the performances of those four algorithms, some studies and experiments stated that SVM presented better results [22] [23] [24] . In [6], SVM was compared with kNN, in a situation very similar to this dissertation project, and SVM had always the least number of wrong classifications.

Despite having multi-class implementations arising, SVM is used generally as a binary classifier, working through the separation of two sets of multidimensional data. For example, considering samples from two different classes in the same dataset, SVM finds a hyperplane (in the cases of two-dimensional data it finds a line) that separates the two classes of data. However, there are more than one possible hyperplane that can separate the data and other techniques that does the same. SVM stands out by using the concept of "margin" to optimize the separation, being the margin given by the distance between the hyperplane and the nearest samples of each class, the support vectors. This way, maximizing the margin is obtained the best separation of classes. Figure 2.11 is an example of the data separating, with a bi-dimensional dataset.

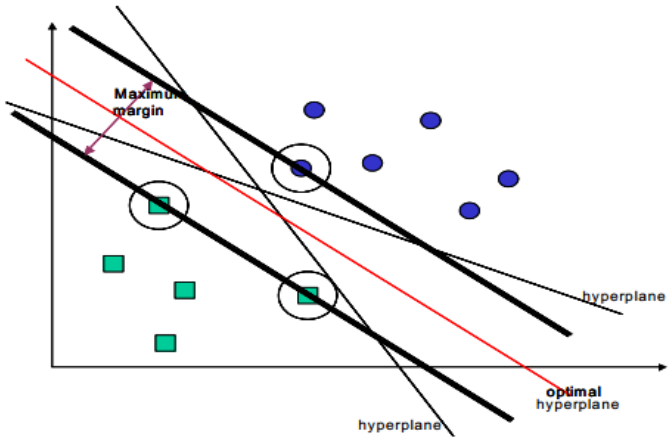


Figure 2.11 - SVM separating hyperplane and margin maximization, taken from [25]

Unfortunately, two classes aren't always easily separated, most of the times are outlier samples in the dataset placed in the wrong side of the hyperplane. Therefore, to include those samples it is added an extension of the margin, the so called soft margin, controlled by a parameter C, wherein the lower the value of C the bigger will be the soft margin, by the contrary as C tends to infinity the margin tends to be thinner and may even narrow the original margin. However, this has an associated risk, because by extending the margin there's a bigger chance of inclusion, in one class, samples from the other. Therefore, C must be set with caution because it controls the trade-off between the maximization of the margin and wrong accepted samples. The Figure 2.12 is an adaptation of the last example, where it is used the soft margin to include the green outlier sample, however it is included also the blue samples in between.

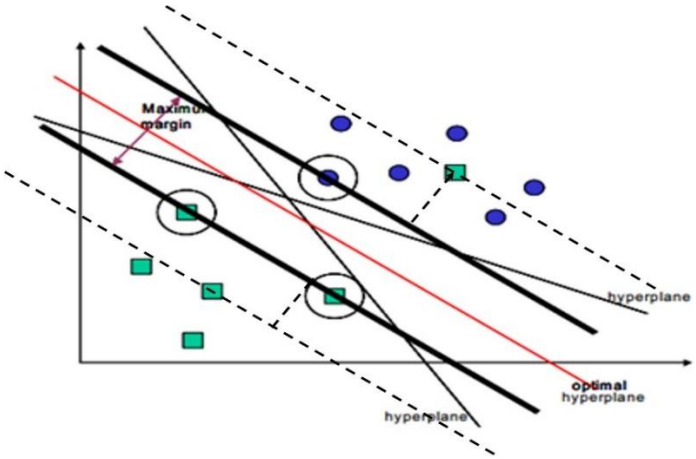


Figure 2.12 - SVM separating hyperplane and margin maximization with soft margin, adapted from [25]

Despite the use of soft margin may improve the separation, with a well-tuned *C* parameter, in most cases that is just not enough. In real cases the data rarely is linearly separable, so it needs to be applied the "kernel trick" to the SVM. It consists in using a function that converts the dataset data in a higher-dimensional space, and apply then the hyperplane to separate the data. The original space is called *input space* and the new high-dimensional space is called *transformed feature space*.

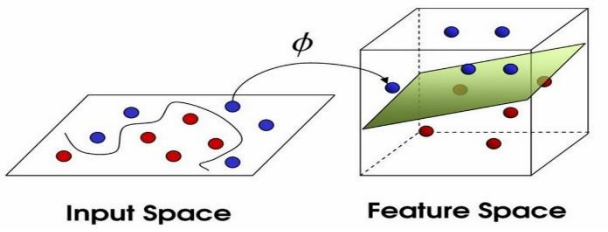


Figure 2.13 - Transformation to a higher-dimensional space, taken from http://www.imtech.res.in/raghava/rbpred/svm.jpg

To train the SVM algorithm one must build the training dataset, containing data from the two classes, with an additional feature indicating the class. After that the kernel is chosen, the parameters are tuned and the SVM is run. Once calculated the hyperplane, the train is done and a training model is generated. To proceed to classification the classifier is fed with a data sample, then that sample is compared with the training model. The classifier determines to which side of the hyperplane belongs the testing sample, classifying it accordingly.

4. Database

This is the last module of the biometric system. Database is where the user's biometric profiles are stored. The information must be always available to be matched and must be secured because if the stored templates are compromised, so are also the future classifications.

2.2.2 Operating Modes

A biometric system has three operating modes: enrollment, verification or identification.

Enrollment Mode

In this phase, the system receives the biometric data from the user to create his profile and store it in the database, next to information concerning the user [17].

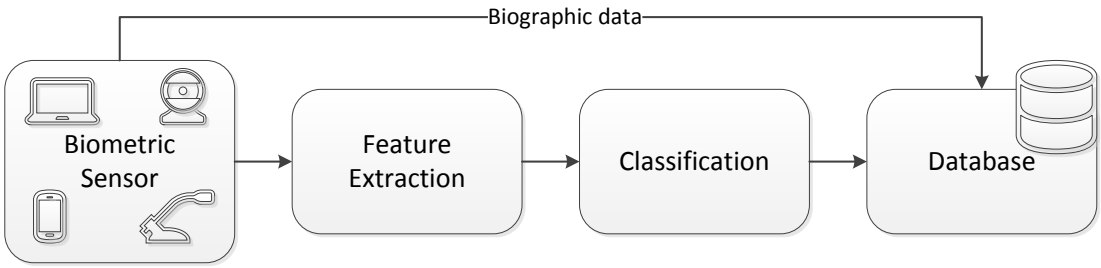


Figure 2.14 - Enrollment Mode

The biometric characteristic is collected and sampled by the sensor, before the feature extraction starts, is made the quality test, to check if is valid. Once extracted the variables, in the classification mode, the classifier is trained with them creating the user biometric templates, which is stored in the database.

Verification Mode

This is where the authentication procedures are made. In this case, the classifier will match an authentication attempt with the corresponding profile.

User informs the system about who he claims to be, then the system sends to the classifier the template of that user. After the typical procedure (collection, quality assurance and feature extraction), the actual input sample of the challenger is matched to the profile already in the classification module. The classification generates a score match that is passed to the decision rule. The decision rule has configured a threshold value, if the score is higher, then the attempt

is considered valid, if is below the threshold the user is considered an intruder. This is a scenario of positive recognition.

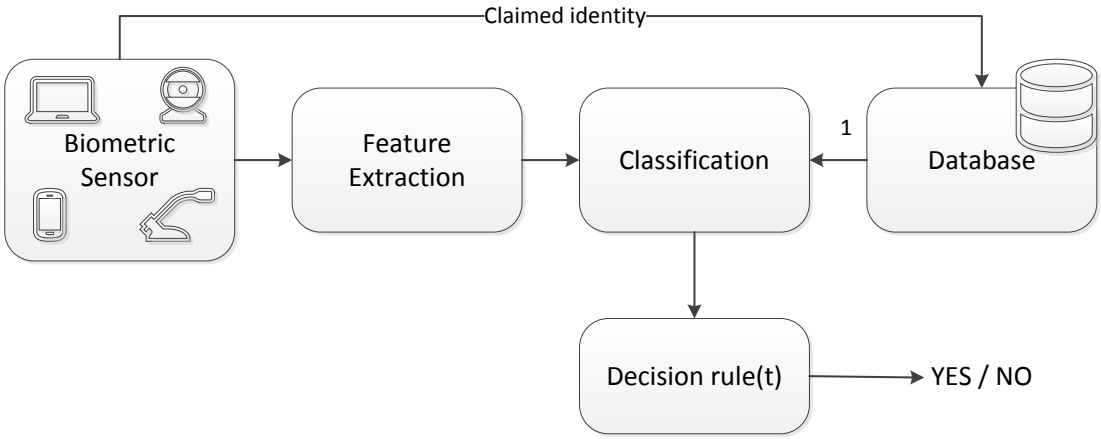


Figure 2.15 - Verification Mode

Identification Mode

The identification mode is the most challenging in terms of resources consumption. In this case the user don't have to tell who he is, that's a scenario of negative recognition.

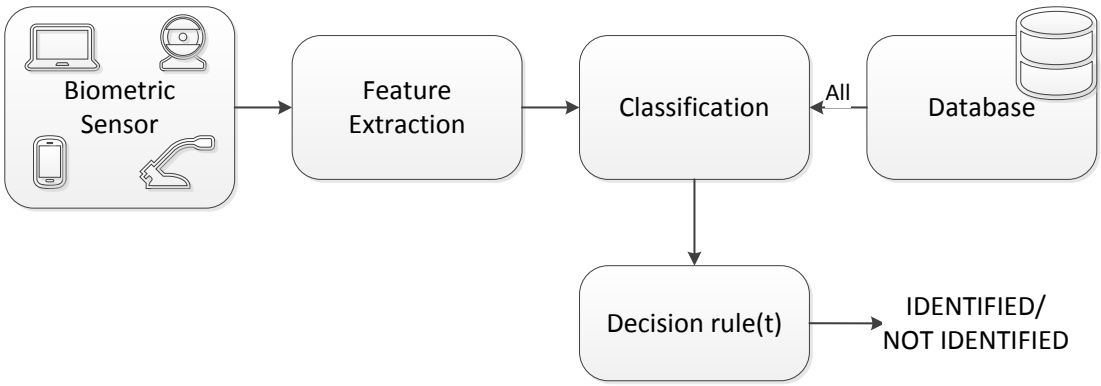


Figure 2.16 - Identification Mode

The classifier now receives all templates stored in the database and matches one by one with the user attempt data. In the same way, the final decision is made comparing the score value of each classification with the threshold.

In the end of all iterations, if the user still not identified it's because he is not enrolled in the system. Negative recognition has the special intention of not let the same individual use multiple identities. This kind of recognition is exclusive of biometrics while positive recognition can be implemented with the traditional authentication means.

2.3 Continuous Authentication using Touchscreen Biometrics

Behavioral biometrics, despite being less popular than physical biometrics, have a grown up body in literature in authentication. However, most of the works focus on session login authentication rather than continuous authentication. Despite that, characteristics like keystroke dynamics have some work done, and that were base to commercial solutions [7] [9].

The use of touchscreen devices was only thought for first time a few years ago. In [26] was proposed behavioral manners of using a touchpad like a touchscreen. Was used a combination of finger pressure and keystroke dynamics with 10 participants. The features extracted were hold-time, inter-keys and finger pressure. It was found that finger pressure alone was more discriminative than keystroke dynamics using k-NN classifier, and had an accuracy of 99%.

More recently the capabilities of mobile devices started to be explored. In [4], was explorer a two-factor authentication in smart devices, using the lock pattern and with biometric information. Was achieved an EER of 10.39% from the experiment with 32 participants and using Random Forest as classifier. This suggested that users could be identified by the pattern they draw.

In [27], it was created a prototype for mobile user identification that provides continuous authentication, using voice, location, multitouch and locomotion. The results suggested that those modalities were suitable as data sources for implicit mobile identification.

Then started to appear, a couple of years ago approaches that tried to establish a continuous authentication method based on interactions on touchscreen. In [12] was presented an approach with the purpose of exploit touch screen data of common smartphones to identify users based on the way they performed an action. Were evaluated unlock gestures and password

patterns of Android platform. Was obtained an accuracy rate of 77%, FAR of 21% and FRR of 19%, using dynamic time warp (DTW).

In [28] was presented a touch-screen based user authentication approach for mobile devices. The system used touch gestures on the touchscreen as input, and also used a digital sensor glove to add extra gesture information. Were used as classifiers Decision Trees, Random Forest and Bayes Net Classifier. At his best the model achieved a FAR of 4.66% and a FRR of 0.13%, which proves that it can be used.

In [6], was proposed a model with 30 features extracted from raw touchscreen logs. It were used just scroll moves, in three different sessions. Were used kNN and SVM as classifiers and SVM had always lower error rates. It was achieved EER values from 0% to 4%, depending on the session. Although, the authors disqualify this method as a standalone solution.

It was combined, in [10], the interaction in the touchscreen with the walking patterns of the user. To do that were used the motion sensors of the smartphone, namely the accelerometer and the gyroscope. It was used SVM as the classifier. The results were encouraging as they got a FAR and FRR below 1% after collection 10 actions.

These last works are the ones with more in common with this project. However all these works were focused only at the device level, because in most of them users could perform free gesture.

3. Data Collection and Feature Extraction

Data collection and feature extraction are the first steps that any biometric system has to perform. Regardless of the context, whether for enrollment or recognition, the system has to gather the subject's biometric information and select which features will be extracted.

In this chapter is described the practical work regarding the data collection and subsequent biometric feature extraction done in this dissertation project. For start, are given two introductory sections, the first presents the proposed biometric system that served as a guide for the project and mostly for the decisions taken along the way, the second reflects about the choice of the operating system in which the collection application would be implemented.

Later, the chapter includes sections concerning aspects such as: which gestures would be performed by the users and which features would be extracted. It also contains, in the end, the data collection scenario and the description of the application development.

3.1 Biometric System Proposed

The desired biometric system proposed is a system that provides a continuous authentication of users using only behavioral biometrics, unlike other systems that use the traditional authentication methods or in some cases a combination of both.

The system is not to be applied to the entire device because that would require changes in the operative system layers, and it would only work in rooted devices, which is not reachable for everyone. Instead, this system is to be implemented in applications, no matter its nature, the user or the device, as long as has a touchscreen. This is so because different applications requires distinct actions from users and, this way, is possible to adjust the system to each app.

In an earlier phase, the legitimate user of the device is asked to perform an enrollment process, a sequence of defined gestures, so the system can create a biometric profile of him.

That profile is created using the features extracted from the gestures and training the classifiers with them. Once created, this step should no longer be available, this way there is no risk of an attacker perform this process in order to pass by the authentication system.

The system should now be able to compare the input gestures with the legitimate user's profile, through the classifiers, which calculates a matching score between the two datasets. If the matching score is above a given threshold the gesture is classified as legitimate, if the score is below the threshold then the gesture is classified as not legitimate or as an attack. As a system for continuous authentication the decision about the user's identity is not taken immediately, it must be defined the number of not legitimate attempts needed to consider that the user is an attacker. Once reached that number, the system should act in order to block the device and possibly warn the legitimate user.

Users tend to change their behavior over time, either by being more used to the device or even to physical limitations. Therefore, for the system to remain accurate, the user's biometric profile must be updated, as long as the user keep using the device. Obviously, not all input data can be used to update the profile, new biometric information should only be kept after the user be classified as legitimate.

For the purpose of implement this biometric system, it is necessary to study which touchscreen features are measurable, which must or must not be used in the analysis, and if the resulting biometrics are enough to provide an accurate authentication. And that is what this project is about.

3.2 Choice of Operating System

Prior to develop the application to collect touch data, it was necessary to identify which operating system better suited the project. There were some obvious options due their presence on portable devices: Android, iOS and Windows (Windows 8 and Windows Phone). These operating systems were analyzed in terms of their market share, if their API methods allow to monitor and extract touch data, requirements to develop, ease of programming and support documentation.

In terms of market share, Android has a great advantage over its rivals. In the second quarter of 2014, Android had 84.7% of market share, with 255.3 Millions of devices, growing 33.3% compared to the same period of 2013. It is also the mobile operating system most present in low-end and mid-range devices, contrasting with iOS which has no presence in low-end devices. Windows Phone has a nearly insignificant market share comparing with Android and iOS, and Windows 8, until now, runs only in PCs and in a limited range of tablets. [29]

Each platform has a main programming language and particular requirements to develop applications.

Android requires knowledge of, at least, Java and XML. The code can be written using various IDEs alongside with Android SDK Tools. However, it is recommended using Eclipse ADT or Android Studio, because these IDEs already have everything set up. It is multiplatform, can be installed in Windows, Mac OS or Linux. Android applications are free to develop and to publish in the store, but it has a one-time registration fee of 25$.

The iOS app development main programming language is Objective-C. To build apps with iOS it is needed Xcode IDE and iOS SDK. Unlike Android, it can only run on a Mac operating system, which means that the developer has to own a Mac computer or use a virtual machine to run Mac OS in other platforms. It charges the developers an annual fee of 99$ for the developer license, needed to publish apps in the store.

To develop Windows apps, whether to Windows 8 or Windows Phone, the main programming language is C#. Like iOS it requires an exclusive IDE and operating system, Visual Studio and Windows, respectively, which forces developers to own a windows PC or to use virtual machines. It costs a one-time fee of 19$ to individual developers or 99$ to companies.

In terms of support documentation all of them have platform has a well-established developer center, with tutorials, complete information of every API method, examples of applications, design guides, among other utilities.

The monitoring of touchscreen inputs and gesture recognition are possible with the API of any of these platforms, so this was no key factor in the decision.

Ease of programming also was not an exclusion factor. In fact, all these programming languages are suited for the job. However, the author has already a Java background, so it is

less time-consuming to develop in a known language instead of learning a new one, because no matter how good the learning curve might be, it always consumes a considerable amount of time.

Having all of the previous information into account, the chosen platform to develop the application was Google's Android.

3.3 Analyzed Gestures

After choosing Android as the application development platform, the next step was to determine which gestures could be monitored by Android's API [30].

In Android, each touch on the screen, with one or more pointers[1], generates an event. Those events, isolated or in sequence, can be interpreted as gestures, if they follow a particular pattern.

It was used the `GestureDetector.SimpleOnGestureListener` class to detect the gestures. This is a nested class from `GestureDetector` class[2], which receives the events, triggered by the touches on screen, collect all the information related to them and determines if they have the same pattern of any supported gesture. Each gesture has a respective method to assign any action to it.

This class supports the following gestures: [31]

Tap down: When the screen is touched, this is the first event of any gesture.

Single tap up: Occurs every time that a pointer is lifted.

Single tap confirmed: Occurs when a pointer is lifted, but is not followed by another tap down (e.g. double tap).

Long press: When a pointer presses the screen, without moving, for a long time.

[1] Pointer is the object that makes contact with the screen, it could be a finger, stylus pen, among others.

[2] It was used the GestureDetectorCompat class instead of GestureDetector to grant compatibility with older versions of Android.

Show press: When a tap down occurs and still down for a long time. It occurs alongside with long presses or scrolls.

Scroll: When there is a displacement between the coordinates of a single tap down and the respective tap up, with one or more pointers. It is one of the most common gestures.

Fling: When a scroll is performed and the pointer is lifted so quickly that gesture remains accelerated, making the scroll continues. The gesture continues until is completely decelerated and stop.

Double tap: When a double-tap is performed

Double tap event: When occurs a double tap with displacement between the first tap down and the final tap up (e.g. double tap with dragging).

To implement a biometric authentication system, the gestures to analyze must meet some criteria, such as:

- Simplicity
- High frequency
- Universality
- Distinctiveness

The gestures must be simple and common, so that any person can perform them, regardless the experience with the device. They also need to occur with high frequency, so they should, preferentially, be part of common actions that everyone performs naturally (e.g. reading a document and navigate between menus). Finally, these gestures must be performed differently, as much as possible, by different users.

Under these circumstances, it was necessary to review all the supported gestures to decide if they fulfill the previous requirements.

Gestures like **Show press** and **Long press** were discarded. The first one because it occurs always alongside scrolls and long presses, so it would only add redundancy to the dataset. Among the actions that a user could perform, apart from selecting or highlighting, that is no use for a static long press, even those two actions are not very usual. Every common action involving long presses are made combining a movement, like when a user drags an item to other place. For this reason, long presses were also not taken into account.

Isolated events, such as, **tap down** and **tap up** were also discarded. Despite being simple, universal and be executed several times, these single events do not have enough distinctiveness to discern users. A single tap down event is always followed by a tap up, and a tap from one user does not differ much from a tap of other user, there is no displacement and no velocity, besides, they are quite random so they cannot be associated to a particular action when performed isolated.

Related to taps, there are some works made by studying the keystrokes. These works analyze the behavior of an individual when he writes on a keyboard, the time that he takes to step from one key to another, the pressure made by the stoke, the time spent to tap a sequence of keys, among other features. This could be studied on this dissertation project, however, this idea falls apart because of the unlimited amount of different keyboards that a user could install on his device. That way, a key mapping is impossible to make, unless the experiment was conducted on a particular keyboard, made for the purpose or selected between the many keyboards on the market (tough this would raise permission problems). Nevertheless, this would focus on only a part of the device utilization, so it was left aside as well.

On the other hand, there are gestures that users inevitably do in order to complete certain action. Navigation through menus, web browsing, reading and gaming are some of the most common uses of touchscreen devices. When someone navigate between menus, inside an application or in the app drawer, that action is made using swipes, which are nothing more than scrolls with fling. Whether to scroll a web page or an application view, or even in reading, there are always the same base actions: scrolls to apply movement and double taps or multitouch gestures for scaling. In fact, these gestures are present in almost every usage.

Given these points, the chosen gestures to get touch data from, were **single or multitouch scrolls**, **flings** and **double taps**.

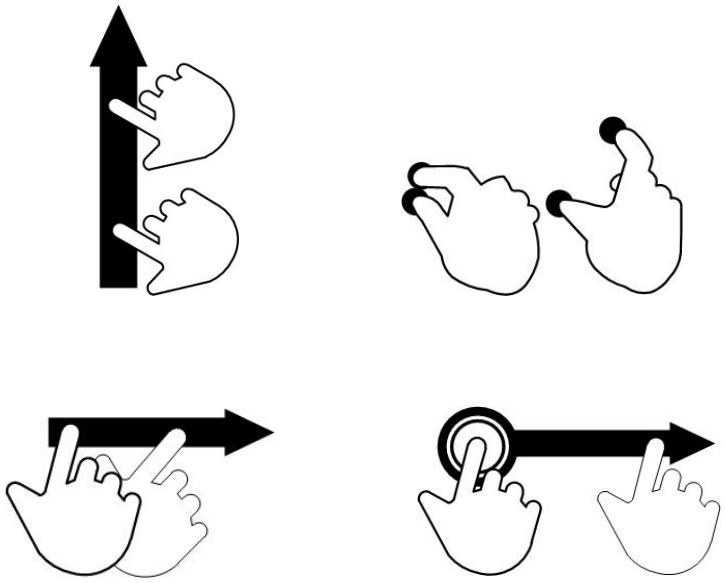


Figure 3.1 - Analyzed gestures[3]

3.4 Extracted Features

As has been mentioned before, the class that detects gestures receives events and gather all the data that they hold. Those events are `MotionEvents`, objects that contain data from a whole gesture or just a part. This is so because, while a pointer is in contact with the screen, events are triggered from time to time, generally 16 or 17 ms. Because of that, a tap down has just one event but is typical to find a scroll with several events.

With the methods of this class it was possible to retrieve, from each event, the following data:

Action: Returns the kind of action that was performed, such as, down, up, move and cancel. This information is useful to interpret to which part of the gesture it belongs. If action is down represents the beginning of the gesture, if it is up represents the end and if it is move represents a movement event between the beginning and the end of the gesture.

[3] Figures made using *Touch Gesture Reference Guide visio stencils* by Luke Wroblewski, http://www.lukew.com/ff/entry.asp?1071

Down time: Returns the time of the gesture's first action down, in other words, the first touch on the screen, in milliseconds.

Event time: Returns the time in which the event occurred. If it is related to the first event of the gesture, the event time and down time are the same, in milliseconds.

Number of Pointers: Returns the quantity of pointers in contact with the touchscreen at the same time. In Android, the gestures can be performed, at the most, by 10 fingers simultaneously.

Orientation: Returns the orientation of the device: portrait, landscape.

Pressure: Returns the pressure made by every pointer in contact with the screen.

Size: Returns the size of the pointers, in other words, returns the area of the screen pressed by the pointers.

X, Y: Returns the X and Y coordinates of every pointer.

The data from each event was organized in vectors. Since the only gestures that have just one event are the tap down and tap up, and those were discarded from analysis, each gesture has more than one vector. Therefore, it was necessary to compile, for each gesture, all vectors in one, representing the gesture's biometric template. This final vector has the following format:

1. Initial time;
2. Fling time;
3. Total time;
4. Initial X;
5. Final X;
6. Initial Y;
7. Final Y;
8. Minimum Size;
9. Average Size;
10. Maximum Size;
11. Minimum Pressure
12. Average Pressure
13. Maximum Pressure
14. Distance in X-axis
15. Distance in Y-axis
16. Velocity in X-axis
17. Velocity in Y-axis

For example, the Figure 3.2 shows the data gathered from a scroll with double tap. Is possible to count 12 rows, which are the 12 events that compose the gesture. Then, those vectors were compiled and saved in the format showed in Figure 3.3. As one can see, the final

vector has more variables than the event's vectors. Those features were included because they could be useful to differentiate users.

1	2	3	4	5	6	7
id	gesture	action	downTime	eventTime	time_total	time
0	DoubleTap	ACTION_DOWN	290211728	290211728	0	0
0	DoubleTap	ACTION_MOVE	290211728	290211862	134	134
0	DoubleTap	ACTION_MOVE	290211728	290211879	151	17
0	DoubleTap	ACTION_MOVE	290211728	290211893	165	14
0	DoubleTap	ACTION_MOVE	290211728	290211912	184	19
0	DoubleTap	ACTION_MOVE	290211728	290211928	200	16
0	DoubleTap	ACTION_MOVE	290211728	290211945	217	17
0	DoubleTap	ACTION_MOVE	290211728	290212142	414	197
0	DoubleTap	ACTION_MOVE	290211728	290212159	431	17
0	DoubleTap	ACTION_MOVE	290211728	290212175	447	16
0	DoubleTap	ACTION_MOVE	290211728	290212175	447	0
0	DoubleTap	ACTION_UP	290211728	290212188	460	13

8	9	10	11	12	13
pointerCount	pointerId	x	y	size	pressure
1	0	644.6428	524.5714	0.41935483	0.16800001
1	0	641.9643	552.6482	0.32258064	0.23520002
1	0	643.13464	587.6065	0.3548387	0.26880002
1	0	640.1786	624.8571	0.3548387	0.26880002
1	0	633.9243	657.45	0.3548387	0.26880002
1	0	616.3694	758.53766	0.3548387	0.2976
1	0	603.3908	835.9025	0.3548387	0.31680003
1	0	592.6937	1357.2437	0.3548387	0.24480002
1	0	597.9573	1372.4028	0.3548387	0.2544
1	0	618.17993	1407.5039	0.3548387	0.22080001
1	0	618.75	1408.2858	0.32258064	0.16800001
1	0	618.75	1408.2858	0.32258064	0.16800001

Figure 3.2 - Event vectors of a scroll with double tap gesture

1	2	3	4	5	6	7	8	9
time_init	time_fling	time_total	x_init	x_final	y_init	y_final	size_min	size_ave
134	0	460	644.6428	618.75	524.5714	1408.2858	0.32258064	0.35215053

10	11	12	13	14	15	16	17
size_max	pressure_min	pressure_ave	pressure_max	distanceX	distanceY	velocityX	velocityY
0.41935483	0.16800001	0.24000001	0.31680003	80.34607	883.71436	0.17466536	1.9211181

Figure 3.3 - Final vector with the compiled data

There are three different times saved on this vector. The **initial time** is the difference between the downtime and the event time of the first move, in other words it's time between the first tap down and the first move event; the **total time** shows the total time of the gesture, since the first tap down until the tap up or, in case of fling, until the animation stops; finally the **fling time** represents the time between the tap up of a gesture and the instant that the view stops scrolling, occurring mostly on gestures performed quickly, which can be a good way to distinguish users that perform gestures quicker than other.

The starting and ending positions in X-axis and Y-axis are depicted in **Initial X**, **Initial Y**, **Final X** and **Final Y**. With these positions was calculated the distance covered by the gesture, more specifically the **distance in X** and the **distance in Y**. Using these distances and the total time was possible to calculate the **velocity in X** and the **velocity in Y**, given in px/ms.

Analyzing the values of size and pressure of all event vectors, it were calculated their **minimum, average and maximum** values.

The orientation was left out because it was decided that the device had to be with the same orientation for every collection.

Those new features were measured through basic calculations because, as previously stated, the extracted features should be nearly raw data, ensuring that device's processing requirements are kept to a minimum.

Again, these features were collected through the android application, and the data vectors were saved in csv files, because it's an easily readable format and can be imported by many software.

3.5 Data Collection Scenario

The data collection scenario was planned considering the goals of the project, therefore this was a very oriented scenario, and one of the reasons why this project is different from other projects in this area.

Users had to do exactly the same gestures, those selected in 3.3, in the same order, in the same conditions, meaning the same device and with the same posture. This is so because the

study focuses on the applications and not in free use of the operating system, and most of the time applications require the same gestures, with limited performance, to execute an action. So, what matters is the variance of the features. For instance, it is wished to know if the set of features of the same gesture (i.e. swipe from bottom left from top right), performed by two different persons, is different enough so that those users can be distinguished by the system.

That being so, it was prepared a sequence of gestures for users to perform on the touchscreen, each one with a set of instructions, such as the part of the screen where the gesture should start and end of the gesture, how many times they would have to do it and a description of the gesture.

The device used was the *Asus Google Nexus 7 2013,* running Android KitKat 4.4.4. This device has a capacitive touchscreen with a resolution of 1200x1920 px.

3.5.1 Procedure Description

The process started with an explanation of the project to the participants, such as the purpose of project, what they would have to do and what would be done with their touch data.

Right before start performing gestures, it was asked the participants to complete a small form where they indicated their name (just for data cataloging purposes), age, gender, handedness and experience level in using touchscreen devices. The data was used to build a characterization of the participants, and to determine if those parameters could have any influence in the results.

After the completion of the form, the participants performed the gestures. The users were seated with the device placed on a flat table, in front of them. The orientation of the device was set to portrait and the user wasn't allowed to change its position so that the conditions were the same for all. It was not said nothing about how the user was supposed to make the gesture, in terms of speed or pressure or the exact spot to place the fingers, because that could affect their natural performance

The sequence of gestures was:

1. Swipe from bottom left to top right;

2. Swipe from bottom right to top left;
3. Swipe from top left to bottom right;
4. Swipe from top right to bottom left;
5. Swipe horizontally from left to right;
6. Swipe horizontally from right to left;
7. Swipe vertically from bottom to top;
8. Swipe vertically from top to bottom;
9. Static double-tap;
10. Double tap and swipe from bottom left to top right;
11. Double tap and swipe from bottom right to top left;
12. Double tap and swipe from top left to bottom right;
13. Double tap and swipe from top right to bottom left;
14. Double tap and swipe horizontally from left to right;
15. Double tap and swipe horizontally from right to left;
16. Double tap and swipe vertically from bottom to top;
17. Double tap and swipe vertically from top to bottom;
18. Zoom in from center to bottom left and top right
19. Zoom in from center to bottom right and top left
20. Zoom out from bottom left and top right to the center
21. Zoom out from bottom right and top left to the center

In each twenty-one gesture sequence, the participants were asked to perform every gesture five times. It was provided a counter in the application so that the participant knew how many gestures were missing, and if any gesture was done wrong the user had the opportunity to start over. Once the zoom gestures were made using two fingers, each finger was analyzed individually, meaning that for every zoom performed was originated two vectors of data.

For classification purposes, was required to collect data to train the classifiers and data to test the classifiers. By that reason, each user had to complete the sequence two times, the data collected from the first sequence was used to build the training dataset and the data from the second sequence was used to build the testing dataset. Consequently, in total, were collected 250 vectors of touch data per participant.

To conclude the procedure, it was verified if everything gone as planned, by checking the csv files in the device's storage, searching flaws that could contaminate the data.

3.5.2 Characterization of the Participants

The touch data was collected from ten participants, with ages between 10 and 51 years, equally divided among men and women.

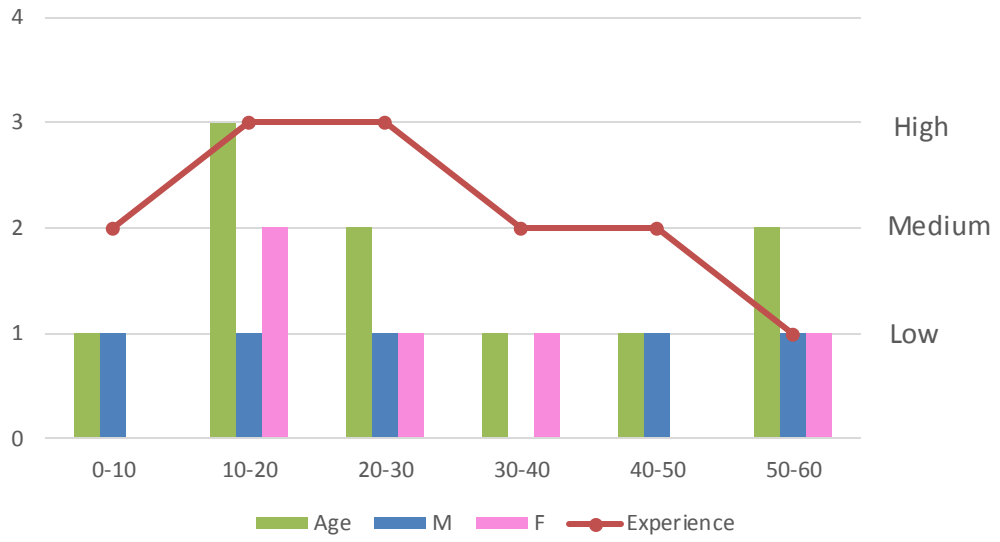


Figure 3.4 - Characterization of the test population

The experience level was classified in three levels, from low to high. It is possible to understand that the younger participants have more experience in using touchscreen devices. By the other hand, one cannot say that the gender of the participant has influence on experience. The handedness is not part of the analysis because all users were right-handed.

3.6 Android Application

The application, developed for devices running Android 2.3+, is responsible for handle users' information, through a form, collect the touch data from the gestures, process that data and store the resulting data templates. Those tasks are accomplished through a set of chained activities. An activity is an application component that provides a screen containing a view with the user interface, and where the developer implements some callback methods to be called when the it transits from one state of its lifecycle to another (when its created, stopped, resumed or destroyed), or when occurs any user interaction with the activity.

Android activities can start other activities and be paused to be ready when needed again, however, that strategy wasn't followed. Each step of the workflow has its own activity and resources, it was created an activity for the initial form and a new one per gesture, stepping forward as the user performs the five samples. When the application transits to other activity, the previous is destroyed, this way there is no consumption of unnecessary resources.

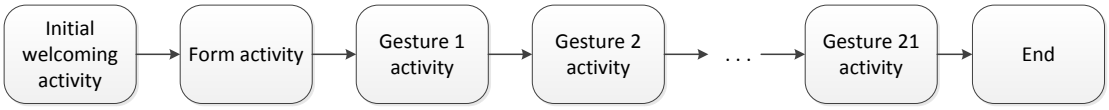


Figure 3.5 - Application workflow

In order to collect the information of the participants and to build the characterization of the previous section (Figure 3.4), was built an activity with the form on the Figure 3.6. The checkbox at the bottom is to indicate if the samples are for training or testing. Once filled the form is created a directory in the device's storage with the user's name and the information is saved in a text file in that location. It is also created in that directory a folder to both training and testing samples in which are stored the csv files with the touch data, i.e. "device_storage/app_name/ user_name/train/". The form was built with security checks in every field, so that users were reminded if was some information to fill. This is considered an extra activity because it is not part of a biometric system, therefore it won't be part of further description.

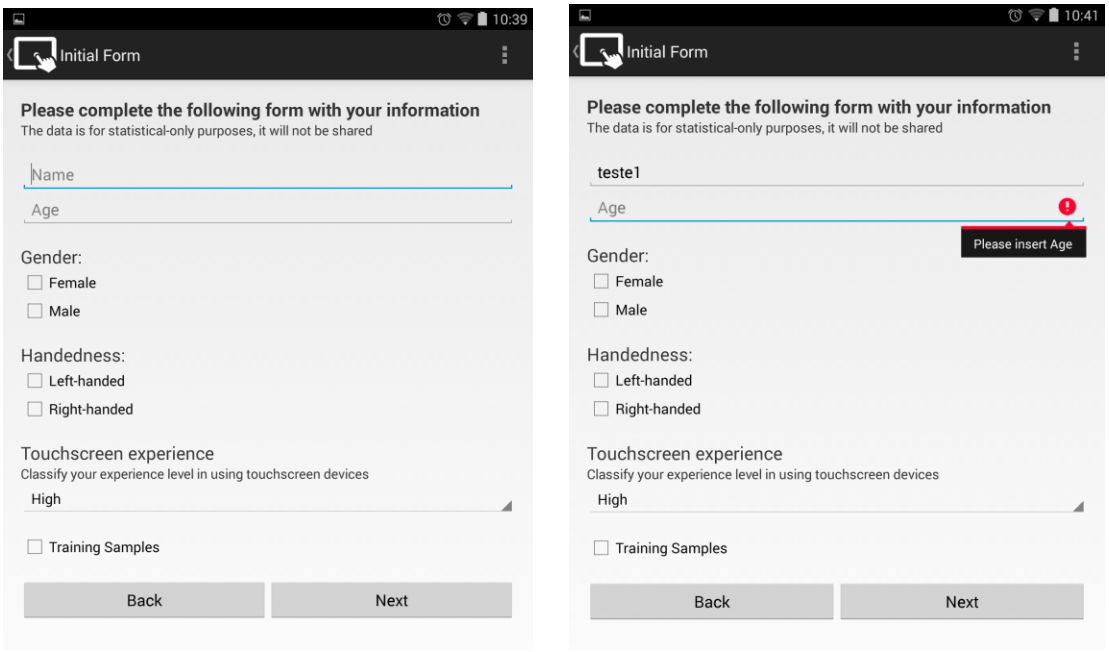


Figure 3.6 - Initial form with verification

3.6.1 Architecture

The application is divided in two main components, the Activity and the TouchListener, in Figure 3.7 is shown the system architecture.

The activities, as explained before, provide the user interface, through which participants interact with the system, and have an implementation of the `GestureDetector` class. For every interaction with the touchscreen the android system generates `MotionEvents`, which triggers the Gesture Detector. Once "awaked", the Gesture Detector receives the event, the activity changes the value of the counter presented in the UI, and sends the event to the TouchListener module to be processed. Once again, activities are independent, so each one has its own Gesture Detector and its own TouchListener module.

The TouchListener is the most important component of the application, it is there were all the features are extracted and organized. Once called, this module processes the event information in order to see which gesture it refers. Those methods select the required data and hold it in temporary vectors. When the gesture ends, and all its events are handled, the remaining

features are calculated, all features are organized in a final vector and then saved to a csv file, in the corresponding storage directory.

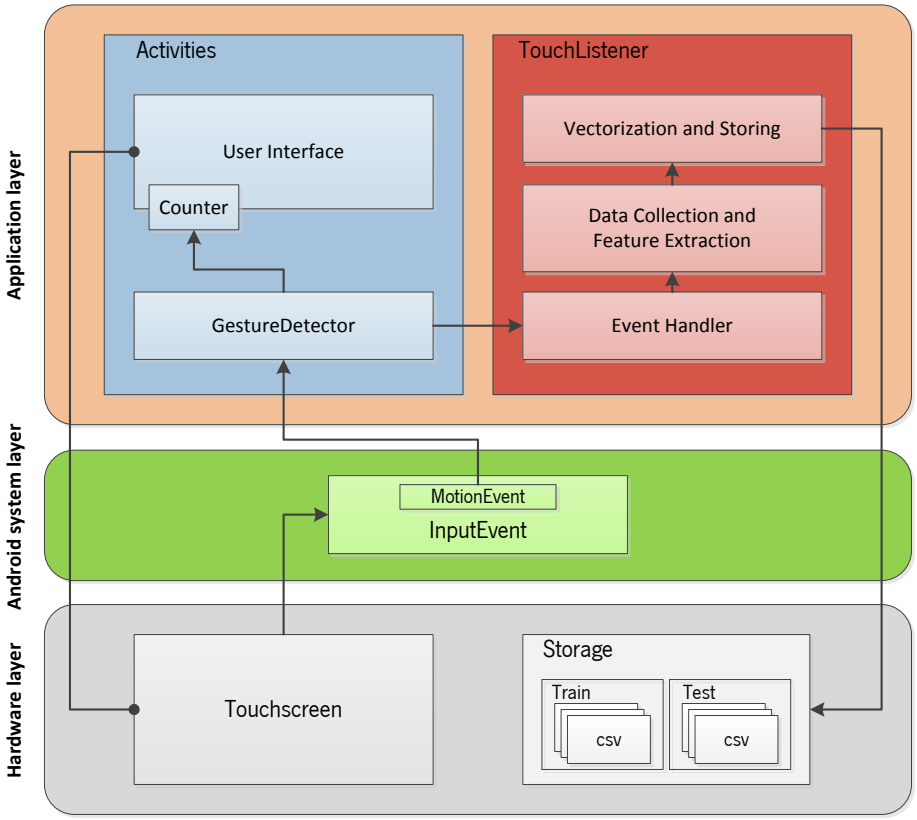


Figure 3.7 - Application Architecture

3.6.2 Implementation

In this section it is explained how the components are implemented. As the application is not the main goal of the project, the description won't be exhaustive, instead, will be provided general explanations using a few diagrams to better understanding.

The interactions between the application components and the application lifecycle can be resumed in four states, displayed in Figure 3.8.

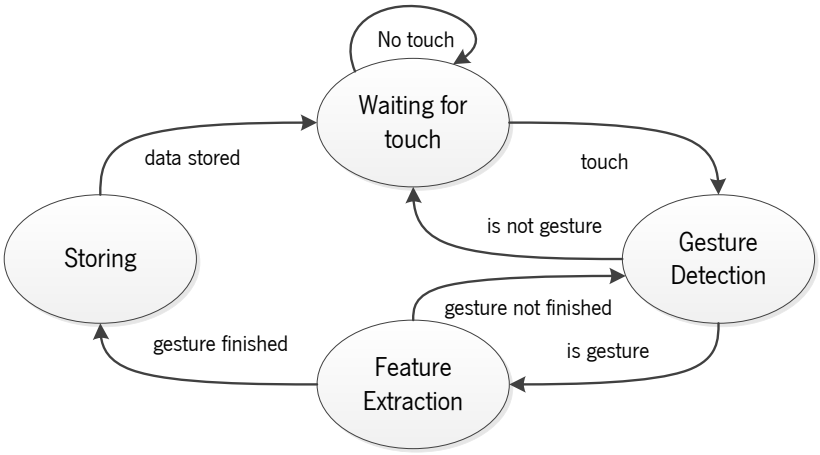


Figure 3.8 - Application statechart diagram

User Interface

The user interface main purpose is to serve as a guide to users, showing how they should perform the gestures.

In each activity is given a description of the gesture, the starting and ending positions, a gesture counter and a reset button. Although the gestures have a conditioned performance, the starting and ending marks are big enough so users can start the gesture pressing a slightly different position. In Figure 3.9 are shown user interfaces of four different gesture-performing activities, namely, swipe from bottom left to top right, double tap and swipe vertically from bottom to top, zoom out from bottom right and top left to the center and swipe horizontally from right to left. As one can see, in the fourth activity, the countdown is replaced by an error message, this message appears when the participant does not perform the gesture that the activity wants to receive. Those alerts were created so that the user could not spoil the experiment by perform wrong gestures.

If the participant is aware that may have made a performing mistake he can reset the procedure, clicking in the reset button, or go back to the previous gesture by clicking on back button. Once finished the performance of the five gestures the counter shows an “OK” message and no longer accepts touch inputs, so user must click on next to proceed. The other UIs can be consulted in Appendix A.

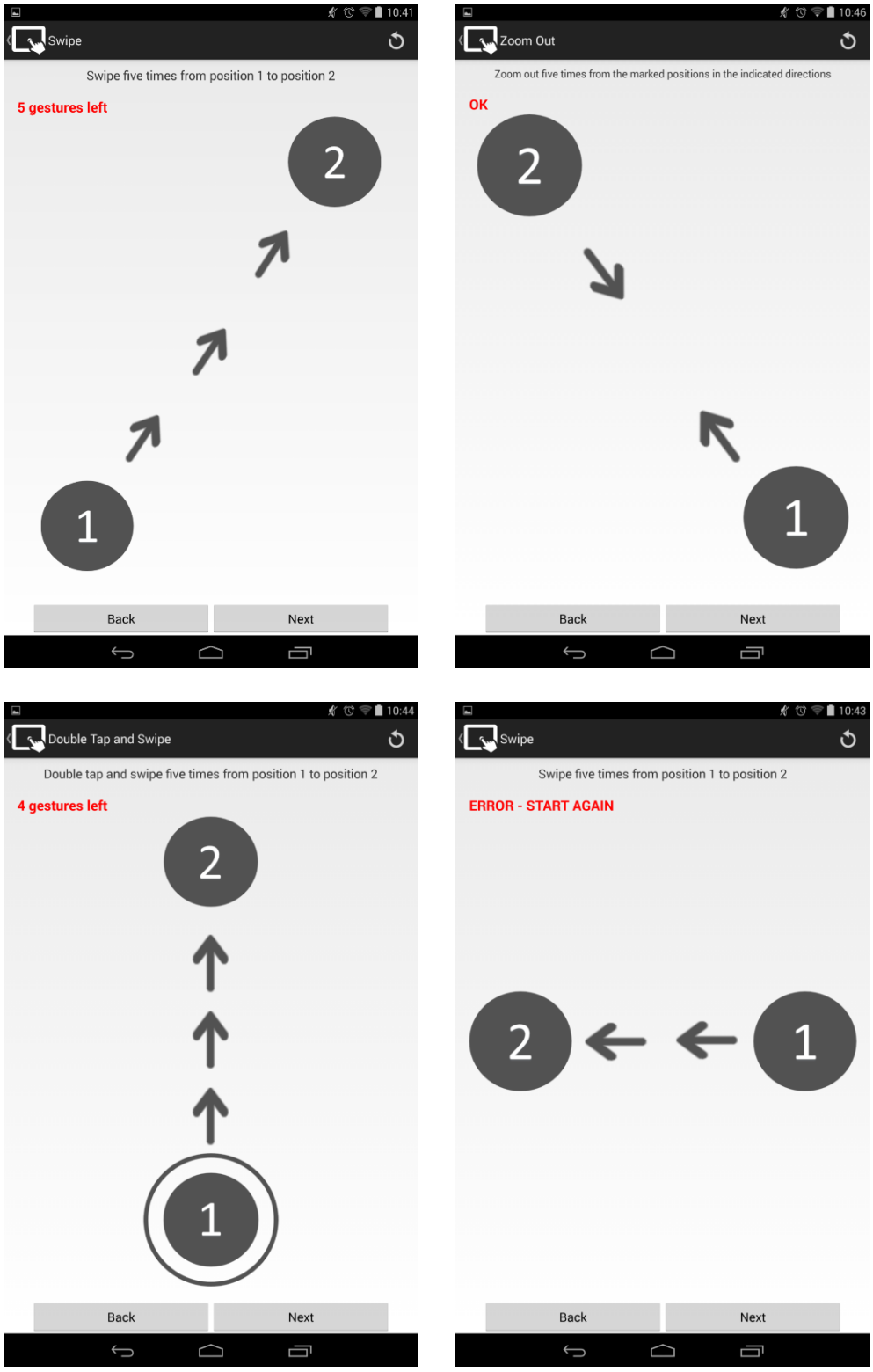


Figure 3.9 - User interface of four different activities

Activities

There are three types of activities: activities that are expecting swipe gestures, activities that are expecting double tap gestures and activities that are expecting zoom gestures. With the flowchart diagram presented on Figure 3.10, it's possible to understand how activities work.

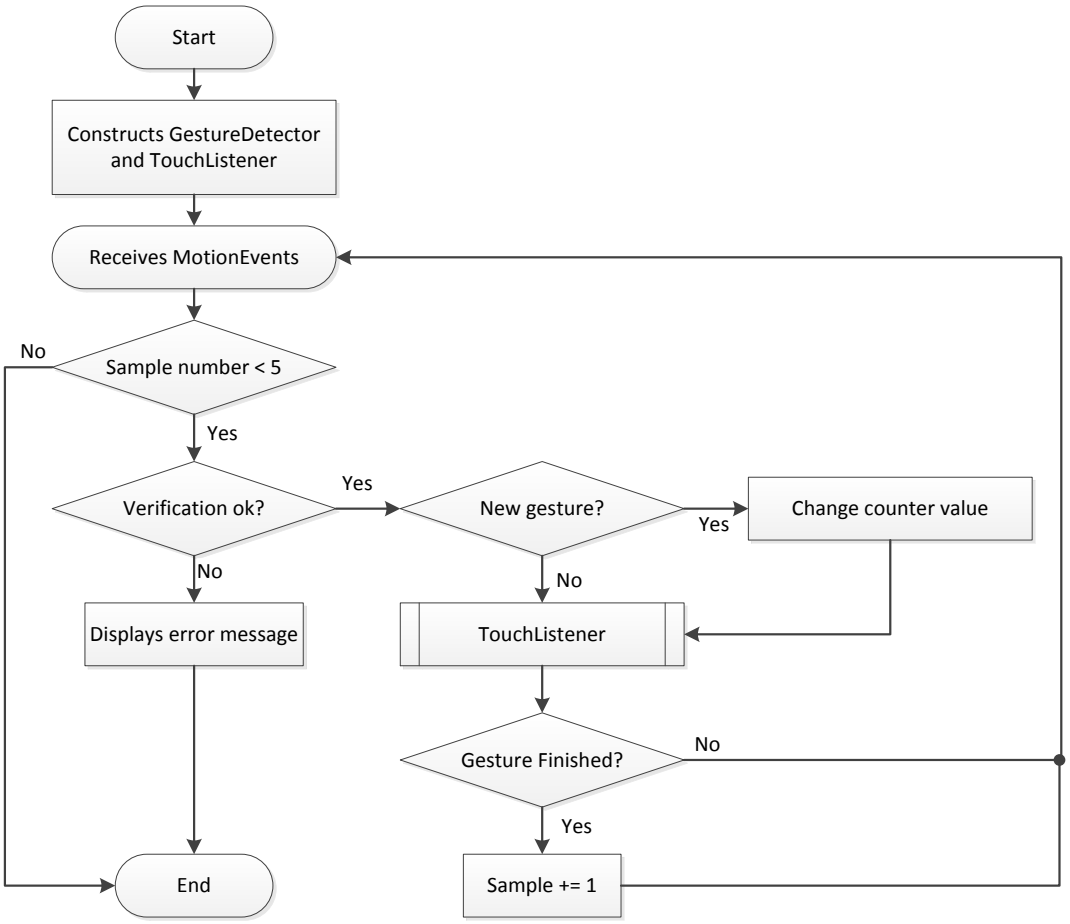


Figure 3.10 - Activities flowchart diagram

Once are only needed five samples of each gesture at time, activities control the number of samples performed, if the sample number is bigger than 5 it doesn't process further received events.

Besides the UI, activities differ in the verification made to the events. That verification serves to guarantee that the events are from the gestures that are supposed to be performed. Activities that are expecting swipe gestures discard any multitouch event and any event that has a double tap. Activities that are expecting double taps also reject both the multitouch events and any event that does not start with a double tap. Finally, activities that are expecting zoom events reject the double tap events, discarding also any event that is not performed with two pointers. If those constraints are not met the error message is shown and the user restarts the gesture.

Once the event is accepted in the verification, the TouchListener module is called and the event is passed as a parameter. But before, if the event is the first of a new gesture, the counter value is changed. Since events from same gestures have the same downtime, that can be verified by comparing the actual event downtime with the previous.

The final part of the activity workflow is to check if the processed event is the last of the gesture, it is true if the event is an action up, which is always the final action of any gesture. If so the sample value is incremented. In the end the activity starts waiting for new events.

TouchListener

The TouchListener module is where the data processing and storing is made. It is configured to accept only events of the type double tap, scroll or fling, any other type is not considered. The flowchart diagram of Figure 3.11 an overview about how it works.

The first step is to determine if the event received is the first of its gesture or if it belongs to a gesture already in processing. As already said, that is made by comparing the downtime. If so, the gesture vectors are initialized and the variables are cleaned.

When it comes to extract the data from the events, described in 3.4, are used the public methods of the `MotionEvent` class shown in Table 3.1. Since gestures can be performed with more than one pointer, in this case the zoom gestures, it is necessary to process each pointer individually because some data is different, such as the X,Y coordinates, size and pressure. This way, the data extraction process is made as many times as the number of pointers. The data extracted in each turn of the loop is saved in temporary vectors, with the number of the corresponding event.

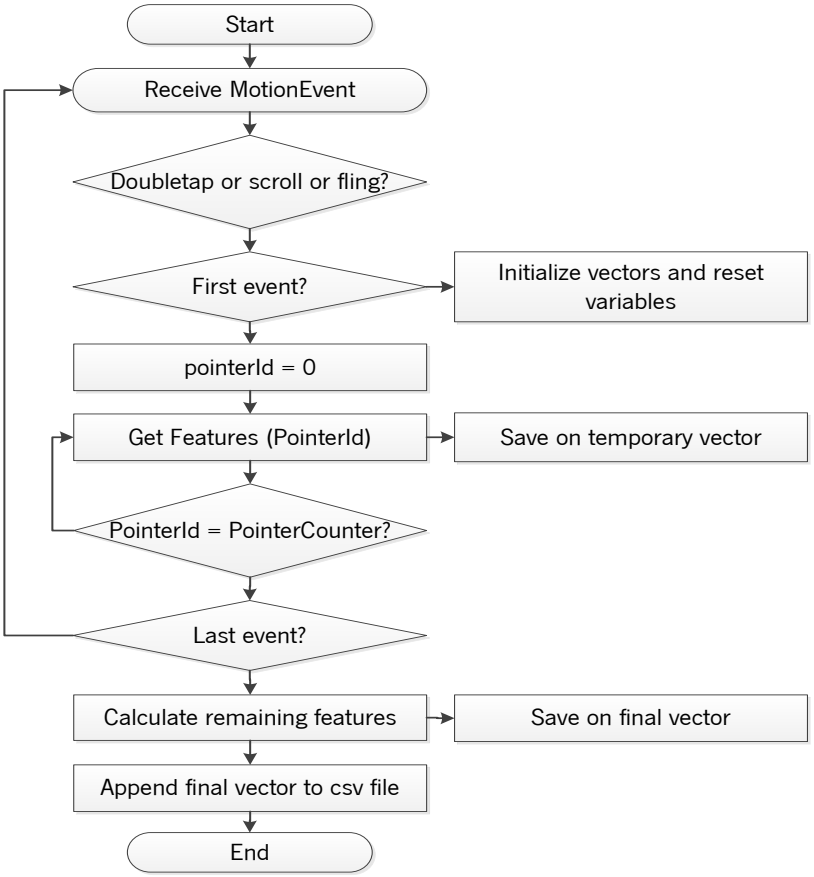


Figure 3.11 - TouchListener flowchart diagram

Data to extract	MotionEvent method
Type of action	getActionMasked()
Down time	getDownTime()
Event time	getEventTime()
Number of pointers	getPointerCount()
Pressure	getPressure(PointerId)
Size	getSize(PointerId)
X coordinate	getX(PointerId)
Y coordinate	getY(PointerId)

Table 3.1 - MotionEvent public methods used

The process is repeated until the last event is processed. When that happens, through a sequence of loop operations on the temporary vectors, the remaining features are calculated,

such as global times, distances, velocities, minimum, average and maximum values. At last, all features are compiled to a single vector, which is appended to the csv file that stores that type of gesture, located in the device's storage.

In the end of the experiment can be found forty-two csv files per participant, half for training purposes and the other half for testing.

3.7 Summary

This chapter covered the data collection and feature extraction process, which was made through an application, in a device running Android.

Before any implementation, were considered which gestures could be analyzed, justifying also the choice of the extracted features. It was decided to analyze just gestures with motion, such as scrolls and flings, static double taps and double taps with movement and multitouch gestures such as zoom.

Was asked the user to perform two sequences of 21 oriented gestures in the touchscreen, five times each. The first sequence for classifier training and the second for testing. The application recorded the data from those gestures, selected the required features and saved them in a file, using a specific format. Once zoom gestures are multitouch in the datasets was saved a touch data vector by each finger used.

4. Classification

After the data collection and feature extraction procedures, the next phase of a biometric system is the classification. This is the phase where the decisions are taken, meaning, if the biometric information captured by the system belongs to the legitimate user or if it belongs to an impostor.

In this section is explained how the classification was performed. First is made a briefly reference to the classifier that was used and, afterwards is detailed all steps of the classification process.

In order to make decisions related to the authenticity of biometric data, it is necessary to include a classification algorithm or technique, known as classifier, in the system. As explained in 2.2.1, a classifier compares a stored biometric profile composed of processed information, behavioral biometric information in this case, with a new incoming dataset of data, previously processed by the feature extraction procedures, producing a result.

The classifier chosen was Support Vector Machines (SVM). This classifier has a good reputation in the machine learning scope, being widely used for pattern recognition. SVM was preferred over the others because it is one of the most robust and accurate methods of classification [32] and, as referred in 2.2.1, obtained better results when compared with others.

There are a few popular kernels that could have been used, in this project was used the "radial basis function kernel" (RBF), because is it one the first choices in pattern recognition processes[33].

This project, as a proof of concept, does not required an integration of SVM in the application. As said before, the samples were collected, processed and saved in csv files and the implementation of SVM and further classification was made using LIBSVM library. In fact, LIBSVM is more than a library, is an integrated software for support vector classification, being one of the most used SVM software by now. It is a very convenient solution due the fact that provides implementations for a wide range of SVM applications on several platforms, and a tools package

with some utilities, in python. As the development was made in Windows platform, were used the already compiled LIBSVM executable files for Windows.

The classification process was divided in six steps, according to the LISVM guide [34], in the following order:

1. Build the training and testing datasets;
2. Convert from csv to LIBSVM format;
3. Scale the samples values;
4. Find the better values for parameters to tune the SVM;
5. Train the classifier with training dataset;
6. Test the classifier with testing dataset;

All these steps were executed through some purpose made batch scripts, because, given the quantity of data to process (250 gestures per user), it would be impractical to proceed each step, one gesture at a time.

4.1 Building the datasets

Training datasets

In order to train a classifier, to authentication of users, is has to be created a dataset with samples of the legitimate user and samples of other users, in this case those samples are the vectors with data from the gestures that are already stored, being used only data from the first sequence each user performed in the collection procedure.

To each vector was included an attribute called class, which could be 1 or -1, 1 for indicate that belongs to the legitimate user and -1 to indicate the opposite. That way, SVM maps all data and the hyperplan separates the "biometric print" of the legitimate user from the others. The first samples to be placed in the training dataset were those from the legitimate user. Here, legitimate user is the one to whom the dataset is built. The files were opened, the samples inside copied to the dataset, going file by file until reached the 21th gesture and a total of 125 samples. The order in which the files are opened is the same of the collection procedure, see 3.5.1.

After the legitimate samples, were placed in the dataset the samples of other users. The dataset is balanced, containing the same quantity of samples of both classes. To achieve that without biasing the dataset it were taken random samples from files of random users. The order of gestures was the same, in other words, were copied five random samples of a gesture, followed by another five random samples of the next gesture, in case of zoom gestures instead of five samples were copied ten because it is a two-touch gesture. In Table 4.1 is an example of this random picking.

...	...
3rd sample of swipe_down_up.csv from user 2	Swipe vertically from bottom to top
1st sample of swipe_down_up.csv from user 9	
4th sample of swipe_down_up.csv from user 5	
3rd sample of swipe_down_up.csv from user 1	
2nd sample of swipe_down_up.csv from user 7	
...	...

Table 4.1 - Example of random sample picking to training dataset

Once each zoom gesture has two samples, the process is slightly different. The samples of a zoom are contiguous, so once selected a random sample, from a random user, it was copied also the next sample if the sample number was odd or the previous sample if the number was even. In Table 4.2 is an example for zoom gestures.

...	...
3rd sample of zoomin_bl_tr.csv from user 2	Zoom in from bottom left to top right
4th sample of zoomin_bl_tr.csv from user 2	
7th sample of zoomin_bl_tr.csv from user 5	
8th sample of zoomin_bl_tr.csv from user 5	
...	...

Table 4.2 - Example of random sample picking with zoom gesture

Were built thus the datasets for training the classifier, with 250 vectors of touch data, half from legitimate user and half from random users. In the Figure 4.1 is possible to see an overview about the building process and in Figure 4.2 is shown, more specifically how it was made.

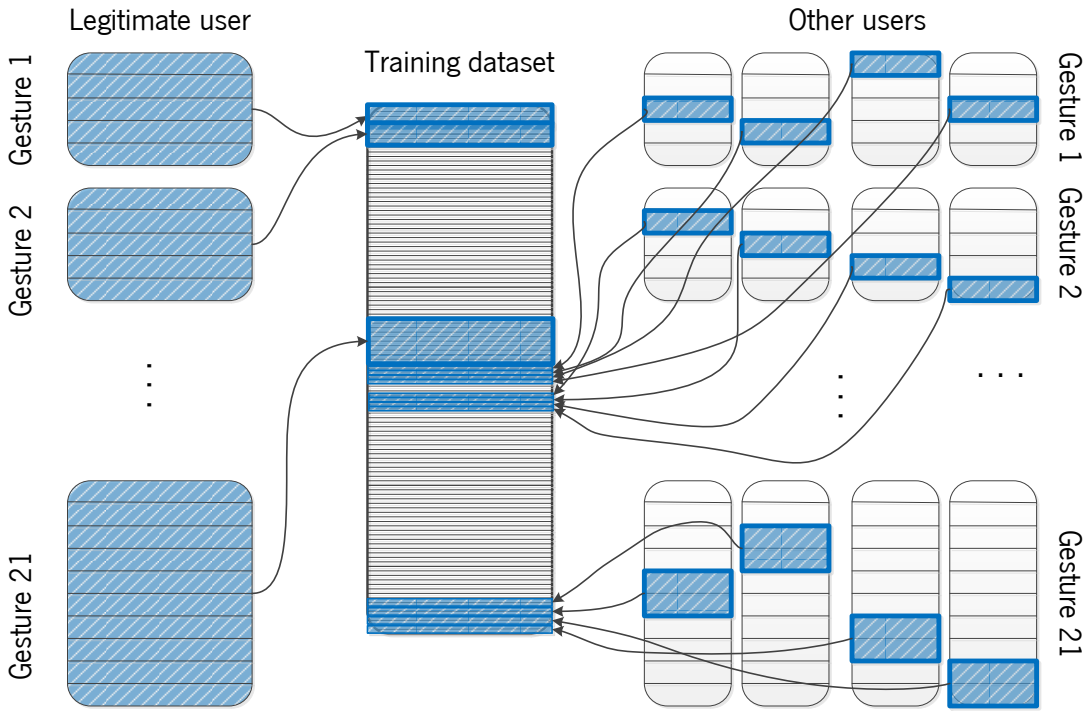


Figure 4.1 - Global overview of training dataset building

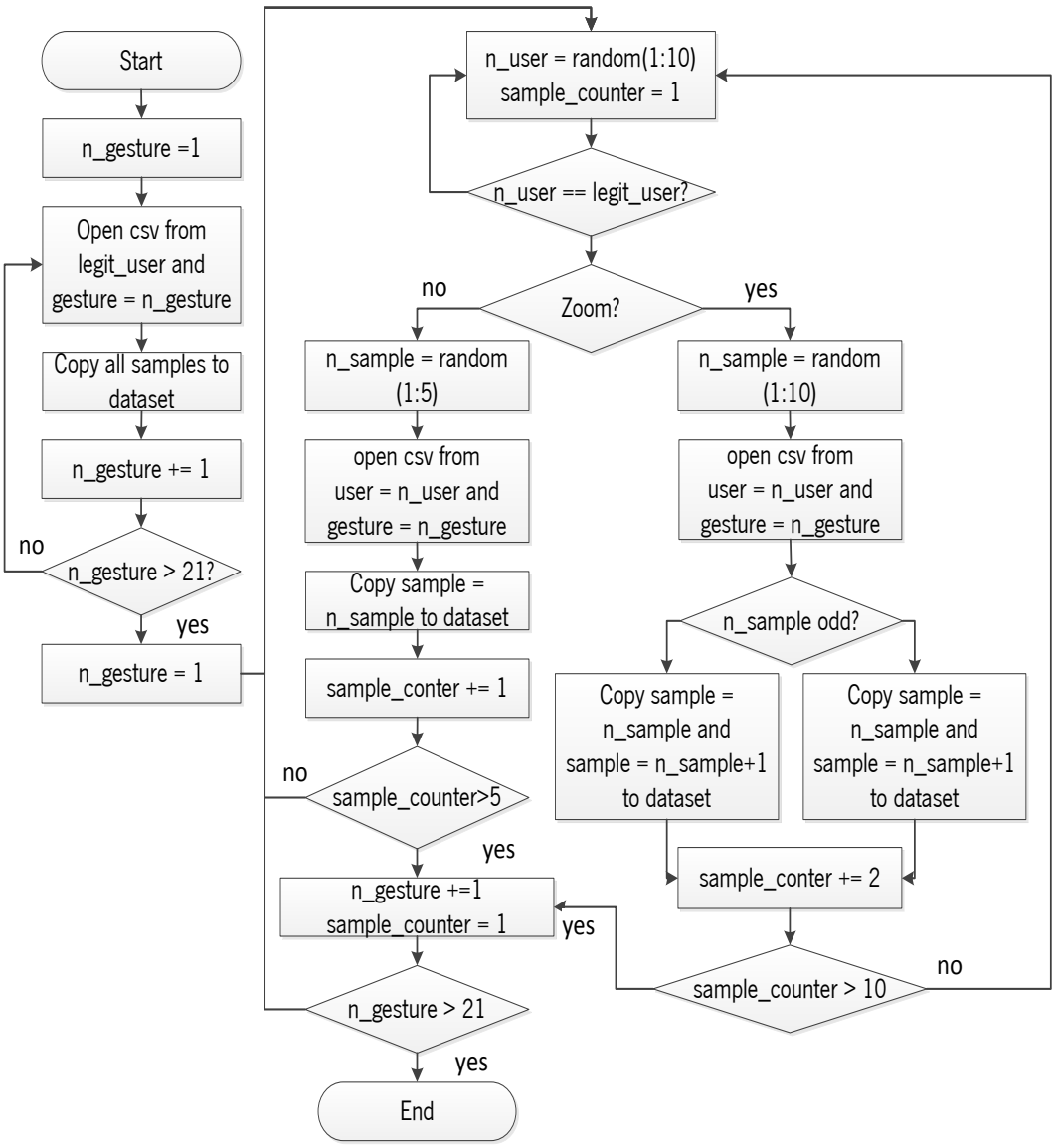


Figure 4.2 - Flowchart diagram of training dataset building

Testing datasets

The building process of testing datasets was very similar to the training datasets. For each user were created two testing datasets, one with only legitimate test samples, to test the classifier for legitimate authentication attempts, and the other with just samples from other users, to simulate attacks to the system. In fact, the only difference in the building process relies in splitting the script in two parts, to generate separated files. The legitimate samples are extracted to the testing dataset the same way that were extracted for the training dataset, and the random sample

picking applied in the first process is also applied here. Unlike the previous process, to build both testing datasets was used only the data from the second sequence of gestures made by users.

4.2 Converting to LIBSVM input format

LISBVM has its own input format, so in order to feed the classifier it was necessary to convert the csv datasets, created in the previous step, into LIBSVM format files.

Csv stand for comma separated values so, as name suggests, in each array of data the values are separated by commas. Typically the first line of a csv is where the labels are. By the contrary, in LIBSVM format values are not separated by commas but by spaces, and there are no labels. In the top of Figure 4.3 is shown a chunk of a csv file from a dataset, and in the bottom is the same data but in LIBSVM format.

```
class, time_init , time_fling , time_total , x_init , x_final , y_init , y_final
+1 , 139 , 12 , 451 , 221.42857 , 1068.75 , 1541.1428 , 321.4286
+1 , 103 , 13 , 449 , 208.92857 , 1091.0714 , 1544.5714 , 296.57144


1  1:139.000000 2:12.000000 3:451.000000 4:221.428574 5:1068.750000 6:1541.142822 7:321.428589
1  1:103.000000 2:13.000000 3:449.000000 4:208.928574 5:1091.071411 6:1544.571411 7:296.571442
```

Figure 4.3 - CSV and LIBSVM formats

As on can see, LIBSVM treats differently the first attribute of the line and then enumerates the other attributes, in the same order, separating the index of the value by a colon. The first value is interpreted as the class label. Again, the label for legitimate samples is “1” and the label for the other samples is “-1”.

Although being an easy process, there were too many samples to convert, so the process was automated according to the flowchart diagram of Figure 4.4.

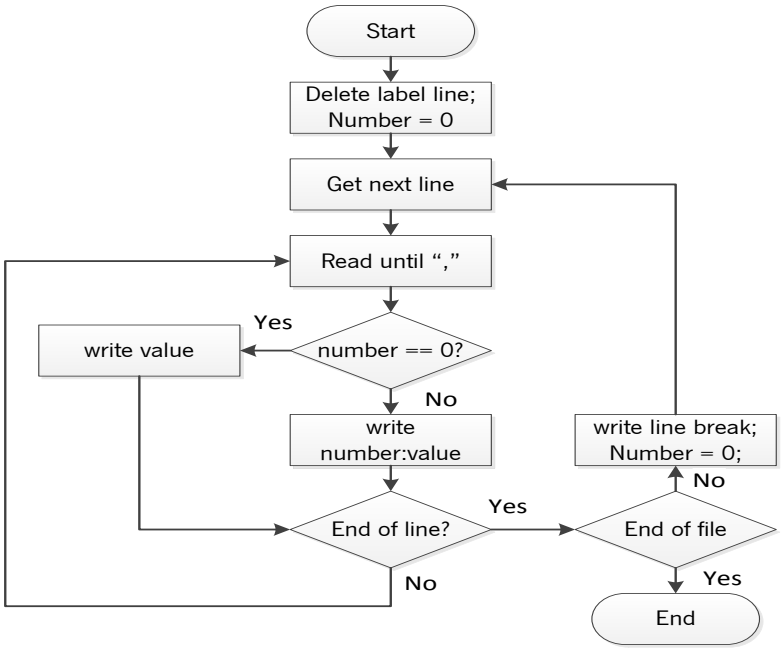


Figure 4.4 – Flowchart diagram of CSV to LIBSVM conversion

4.3 Scaling the values

The datasets have values from very different ranges, reaching the thousands barrier when related to the distance traveled by the finger, but in the case of velocities, since it is given in px/ms, the values are less than a dozen. If the classifier training was made with this discrepancy in the magnitude of values, the highest values would have had more influence than the lower values. On the contrary, as the values are from independent features, they must have the same influence on the final result. Therefore is essential to normalize the datasets.

The data was scaled using a tool provided by LIBSVM, `svm-scale`. By default, values are normalized in the [-1, 1] scale, however it is possible to indicate which are the lower and upper values to scale in a personalized scale. More importantly, `svm-scale` provides an option to save the scaling parameters in a file, so that those parameters could be used to scale other dataset. This is extremely important in to the project, once the classifier is trained with a scaled dataset and, consequently, generated its training model, all the data used in classifications with that model must be scaled with the same parameters. This process prevents that values from

same attributes are transformed with different scales, resulting in wrong classifications and accuracy loss.

For this reason, for each user, the training dataset was scale to the default range, [-1, 1], and the scaling parameters were saved in a file and, posteriorly, both testing dataset (legitimate samples and attack samples) were scaled using the parameters of that file.

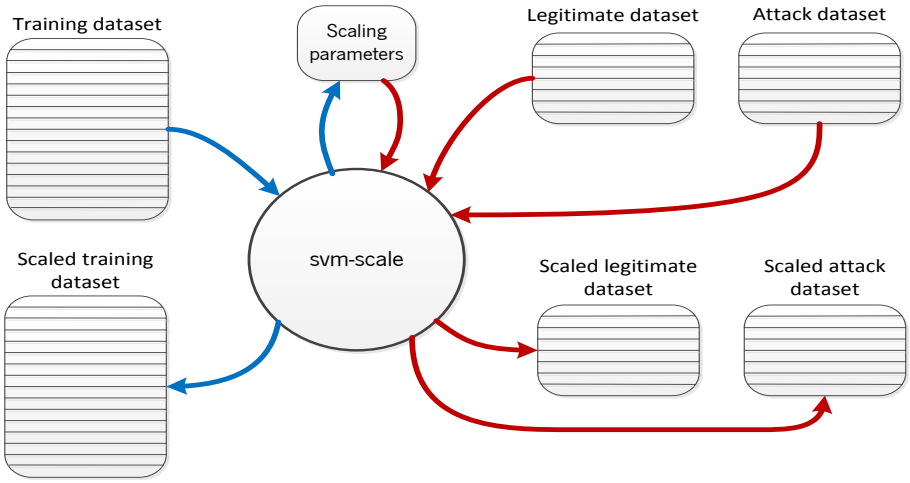


Figure 4.5 - Scaling procedure

The `svm-scale` commands syntax are:

Training dataset:

svm-scale –s range_file –l lower_limit –u upper_limit training_dataset_file > output_file

Testing dataset:

svm-scale –r range_file testing_dataset_file > output_file

In the first command the –s argument is to indicate the file in which the scaling parameters are to be saved, and the –l and –u argument are to indicate the lower and upper values of the scale range. In the second command, for testing datasets, there's no need of lower or upper values because the range is already defined in the file with the scaling parameters. That information is imported through the –r argument.

In the next figure are two pieces of data, before scaling and after scaling with the [-1, 1] range.

```
1  1:139.000000 2:12.000000 3:451.000000 4:221.428574 5:1068.750000 6:1541.142822 7:321.428589
1  1:103.000000 2:13.000000 3:449.000000 4:208.928574 5:1091.071411 6:1544.571411 7:296.571442

1 1:0.269406 2:0.142857 3:0.191888 4:-0.874674 5:0.838489 6:0.780564 7:-0.921801
1 1:-0.0593607 2:0.238095 3:0.185647 4:-0.899043 5:0.878666 6:0.78558 7:-0.956161
```

Figure 4.6 - Data before and after scaling

4.4 Finding the best values for parameters

As said before, the kernel applied to the SVM was the RBF kernel, so there were two parameters to tune, C and γ. SVM is very sensitive to values of the parameters and a little variation can result in a considerable difference in classification accuracy. So this is a critical procedure to execute.

By default, C = 1 and γ = 1/number of features but, just as there is no equal datasets, so the parameters to use in SVM won't be the same from one to another. The problem is that one doesn't know which values would serve better to train a particular dataset, then is necessary to perform a parameter selection process. As recommended by the creators of LIBVSM, a grid-search with cross validation was made for each parameter. Cross validation is a very common practice among the training methods, consists in split the dataset in a given quantity of subsets, called folds, with equal number of samples. One subset is used to test the classifier, previously trained by the remaining subsets using the current values of C and γ. When the classification is done, the process is repeated with other subset, going on until all samples of the dataset have been used to test and to train. The accuracy result is given by the correct classified sample rate. (Correct classified samples / all samples).

Grid-search is based on the trial-error method, it tests various (C, γ) pairs with the training dataset and keeps the one with better accuracy. It was made with a python script provided by LIBSVM, in the "tools" package. In Figure 4.7 is shown the output of an example of grid-search made with this tool.

In a demand for the better accuracy value, was conducted a study with different number of fold in the cross-validation. Grid-search, by default uses a 5-fold cross validation, but in this experiment were used also 10-fold, 15-fold and 20-fold cross validation. The reason of this test lies on the discussion about the number of folds that should be used in cross validation. Rule of thumb, it is used 5-fold or 10-fold, but there is no solid support to justify that option. The procedure was realized with the training datasets of all users. The Table 4.3 presents the results of the study.

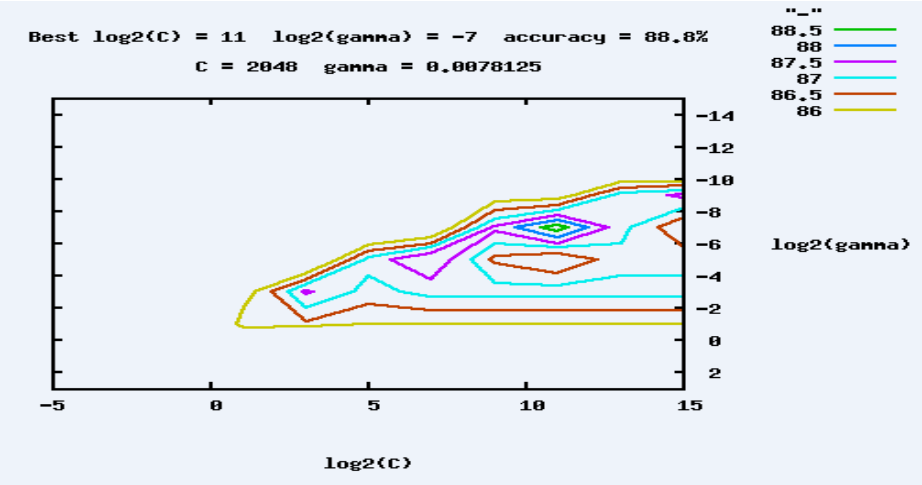


Figure 4.7 - Example of grid search

	Accuracy (%)			
Users	5-fold CV	10-fold CV	15-fold CV	20-fold CV
1	80.4	81.2	81.6	79.6
2	79.2	77.2	77.6	79.6
3	78	78	77.6	76.8
4	88.8	88.4	90.4	89.6
5	89.6	90	90.4	90.8
6	83.2	86	86.4	86.4
7	91.6	92	92	91.6
8	80	81.6	82.4	82.8
9	86.4	87.6	89.2	87.2
10	96	96.4	96.8	96.4

Table 4.3 - Accuracy values from cross validation study

The green cells have the highest accuracy value by user. As one can see the 15-fold cross validation was who got more highest values in general, so this was the adopted number of folds.

The grid-search script was embedded in a batch script, so that, once found the values of C and γ that provide better accuracy, those values could be saved in a text file to be used posteriorly in the real training.

4.5 Training

At this point of the process, the datasets are scaled and the best values for parameters are chosen so the classifier is ready to be trained.

To train the classifier it is used `svm-train`, from LIBSVM library. It can be trained for classification (SVC), to regression (SVR) or to one-class SVM. The kernel to use can be chosen from among these four: the linear kernel, polynomial kernel, rbf kernel and sigmoid kernel.

The command syntax is:

svm-train –s svm_type –t kernel_type –c c_value –g γ_value –b prob_en dataset_file

Once the objective was classification, it was used the C-SVC as SVM type. It was possible to choose also the nu-SVC, though. The kernel type used, as known by now, was the radial basis function, however, LIBSVM supports also linear, polynomial and sigmoid kernels. C-SVC and rbf are the default options. For C and γ were used the values previously determined and saved by the grid-search process.

The argument –b provides an additional functionality, the probability estimates. With this option disabled, the SVM classification output just informs the class to which each sample belongs and the accuracy value of the whole classification. However, if the probabilities estimates option is enable, instead of only tell the class it also inform about the probability of belonging to that class. In the training this option was enabled to supply more information about the classification, allowing further analysis.

Finished the training, is generated the train model file, which contains the hyperplan information, its coordinates and training parameters. It was with it that the predictions were made.

4.6 Testing

The final phase to the classification process is the testing, the classification itself. As was discussed in 4.1, there were two different testing datasets per user. The dataset with only test samples from the legitimate user simulated the true authentication attempts, while the dataset with a compilation of test samples from all the other users simulated the false authentication attempts in the system, or attacks.

The testing was made using the `svm-predict`. The command has the following syntax:

svm-predict –b prob_en test_dataset train_mode_file output_file

There was just one argument to configure this time, because of the fact that the additional information needed is already on the training model. Since the probability estimates was enable when training the classifier, now it also has to be enable.

The test samples, similarly to the training samples, also have a first attribute representing the class. As a matter of fact, that attribute is always set as “1”. That is because, whether being a legitimate attempt or an attack, in an authentication scenario the user always tries to convince the system that he belongs to the rightful class.

Once started the script, each vector of touch data, depicted in the dataset rows, was analyzed by the classifier. All its data, except the class attribute, were mapped and the probabilities of belonging to both classes was calculated. After that, the classifier compared if the predicted class in the result matched the class attribute of the vector. In affirmative case it was counted as a positive classification, otherwise was counted as a negative classification, being

the accuracy given by the percentage of positive classifications. The accuracy value was saved in a text file.

```
Accuracy = 97.6% (122/125) (classification)
```

Figure 4.8 - Example of classification accuracy given by svm-predict

The Figure 4.8 shows the resulting accuracy of a classification made by svm-predict. The values between brackets are the number of correct matches and the total number of samples, respectively.

In Figure 4.9 is shown a piece of the output file generated by the classification process.

```
labels 1 -1
1 0.989397 0.0106026
1 0.85728 0.14272
1 0.984953 0.0150473
1 0.954152 0.0458481
1 0.934602 0.0653981
1 0.990543 0.00945735
1 0.995052 0.00494757
1 0.940266 0.0597344
1 0.973937 0.0260632
-1 0.444752 0.555248
1 0.903133 0.096867
1 0.98761 0.0123903
1 0.952216 0.0477843
1 0.932277 0.0677231
1 0.999254 0.000746434
```

Figure 4.9 - Piece of svm-predict output file

Each row represents a classified sample. The first column is the classification result, the ones labeled with “1” are classified as belonging to the genuine user and those labeled with “-1” are classified as belonging to imposters. The other two columns depict the probabilities, the first of being part of the label 1 class and the second of being part of the label “-1” class.

4.7 Summary

This chapter started by justify the adoption of SVM as the classifier algorithm, used through LIBSVM. After that it described all steps of the classification process.

First, was the construction of the datasets, for training and for testing. After that the datasets were converted from CSV to LIBSVM format so that could be used. The feature values of training datasets were scaled from -1 to 1, and the testing datasets were scaled using the same scaling parameters. The best values to use as parameters to tune the classifier were discovered using a grid-search method with 15-fold cross validation. The number of folds of cross validation was chosen heuristically. The training datasets were trained with those parameters, generating the training model. At last, the testing datasets were compared with the training model, by the classifier, and were generated the score results of that classification.

5.Results and Analysis

In order to proceed to the evaluation of the biometrics and get to the conclusion if it could be used as a continuous authentication mean, were analyzed all classifications made by the classifier. In this chapter are presented the results of the project and consequent analysis. First are exposed the general results and, later, are presented extra tests and their own results.

An authentication system decides if accepts or rejects a user's authentication attempt based on a probability calculation and a preset threshold value. If the attempt has a probability of being genuine higher than the threshold, the system considers that the user is who he claims to be - it is a true match or positive case. By the contrary, if the probability is below the threshold, it considers that he is someone else, an impostor – it is a non-match or negative case.

Under these circumstances, there are two possible situations: the system makes a correct classification or the system makes a wrong classification. Each situation can be provoked two decisions:

1- Correct classification
    a. Accepted genuine sample - true match (TM) or true positive;
    b. Rejected sample of impostor - true non-match (TNM) or true negative;
2- Wrong classification
    a. Accepted sample of impostor – false match (FM) or false positive, known as error type 1.
    b. Rejected sample of legitimate user - false non-match (FNM) or false negative, also referred as error type 2.

By calculating the frequencies of these classifications it is possible to evaluate the performance of the biometric system, in terms of accuracy, sensitivity, specificity and matching error rates.

To help visualize the performance of the classifier is usual to build a confusion matrix, as shown in Table 5.1, matching the actual classes of samples (positive or false) against the predicted classes, making it easier to understand if the classifier is confusing the samples from genuine user with samples from impostors.

		Predicted Class	
		Match	**Non-Match**
Actual Class	**Match (M)**	True Matches (TM)	False Non-Matches (FNM)
	Non-Match (NM)	False Matches (FM)	True Non-Matches (TNM)

Table 5.1 - Confusion Matrix

The accuracy rate reflects the ability of the system to perform a correct classification, that is, the probability of correct classified samples among all.

$$Acc = \frac{TM + TNM}{M + NM} = \frac{TM + TNM}{TM + TNM + FM + FNM} \tag{5.1}$$

A common mistake when evaluating biometric systems is taking accuracy and equal error rate as main performance indicators. The accuracy value, in particular, tend to generalize the performance of the system, however, not all systems have the same purpose. Some are more sensitive, such as identification systems, whilst others are more specific, like authentication systems.

Sensitivity is the ability to recognize legitimate users, in other words, the probability of genuine samples among the accepted ones. It is given by the true match rate.

$$sensitivity = TMR = \frac{TM}{M} = \frac{TM}{TM + FNM} \tag{5.3}$$

Specificity is the ability to recognize impostors, meaning, the probability of rejected samples belong to impostors. It is defined by the true non-match rate.

$$specificity = TNMR = \frac{TNM}{NM} = \frac{TNM}{TNM + FM} \tag{5.3}$$

The matching error rates are the false match rate and the false non-match error, being sometimes referred as 1 – specificity and 1 – sensibility, respectively.

$$FMR = \frac{FM}{NM} = \frac{FM}{FM + TNM} \tag{5.4}$$

$$FNMR = \frac{FNM}{M} = \frac{FNM}{FNM + TM} \tag{5.5}$$

The performance can also be depicted in ROC curves. ROC stands for receiver operating characteristics and it is a threshold independent curve that plots in the x-axis the false match rate and in the y-axis the true match rate, exposing the performance in relation to these two rates. A bit different, but also useful, are the detection error trade-off curves (DET). These curves show the trade-off between both error rates, the false match rate and the false nom-match rate, and are particular appropriate to visualize in which threshold is the equal error rate, EER. The equal error rate is also given by 1 – accuracy.

5.1 System Results

After classifying all samples, the results were analyzed and used to evaluate the biometric system, following the parameters discussed above.

For LIBSVM, the probability threshold from which the samples are classified is 0.5, or 50%, as is possible to understand from Figure 4.9. Once the probability of having label "1" is complementary to the probability of having label "-1", it would be redundant to use both values in the result analysis, therefore, the values used were from the probability of belonging to the genuine class ("1" label), see the 2nd column of Figure 4.9, being interpreted as the classification score. With those scores it was generated a confusion matrix for the threshold of 50%.

		Predicted Class		
		Match	**Non-Match**	
Actual Class	**Match (M)**	815 (TM)	435 (FNM)	1250
	Non-Match (NM)	195 (FM)	1055 (TNM)	1250
		1010	1490	2500

Table 5.2 - Confusion matrix of the system, for a threshold of 50%

Through the confusion matrix is noted immediately that the predicted matches are less than the original matches, having more samples classified as non-matches.

Using the matrix values were calculated the sensitivity, specificity, error rates and accuracy. Those rates were placed on Table 5.3

Evaluation Rate	Value, in %
Sensitivity – TMR	65.2
Specificity – TNMR	84.4
FMR	15.6
FNMR	34.8
Accuracy	74.8

Table 5.3 – Evaluation rates results

As one can see, these values aren't good enough for the authentication system. However, there is no guarantee that the threshold used by LIBSVM was indicated to this data. This way, was performed an analysis for various threshold values, in order to find which gives better results.

The classification results of the both test datasets were clustered in probabilities groups and then counted, generating the following histograms.

Probability	Frequency
0 – 0.05	399
0.05 – 0.10	166
0.10 – 0.15	100
0.15 – 0.20	90
0.20 – 0.25	82
0.25 – 0.30	55
0.30 – 0.35	58
0.35 – 0.40	35
0.40 – 0.45	37
0.45 – 0.50	33
0.50 – 0.55	20
0.55 – 0.60	28
0.6 – 0.65	20
0.65 – 0.70	21
0.7 – 0.75	18
0.75 – 0.80	30
0.80 – 0.85	17
0.85 – 0.90	20
0.9 – 0.95	15
0.95 - 1	6
Total	1250

Table 5.4 - Histogram of attacks

Probability	Frequency
0 – 0.05	66
0.05 – 0.10	34
0.10 – 0.15	27
0.15 – 0.20	27
0.20 – 0.25	35
0.25 – 0.30	38
0.30 – 0.35	51
0.35 – 0.40	51
0.40 – 0.45	49
0.45 – 0.50	57
0.50 – 0.55	40
0.55 – 0.60	59
0.6 – 0.65	51
0.65 – 0.70	61
0.7 – 0.75	67
0.75 – 0.80	70
0.80 – 0.85	88
0.85 – 0.90	88
0.9 – 0.95	132
0.95 - 1	159
Total	1250

Table 5.5 - Histogram of legitimate attempts

In Table 5.4, are grouped the classification results of the attack samples, while Table 5.5 are results of legitimate attempts. The range of each score group is 0.05, or 5 in percentage. To better understand this values, the results distributions for each attempt type are depicted in the graph of Figure 5.1. The values were scaled to the relative frequencies and the score matches are in percentage.

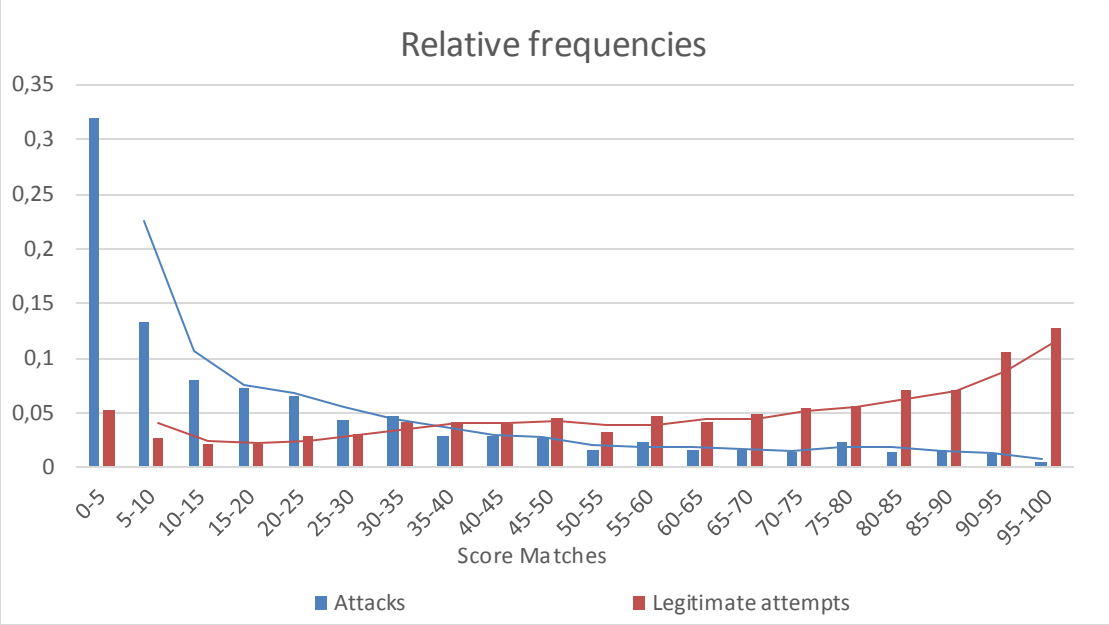


Figure 5.1 – Distribution of relative frequencies graph[4]

With this chart is possible to see that somewhere between 35% and 40% is the turning point, where the legitimate samples start to appear more than the attacks. The attacks, or impostor's attempts, are mostly between 0% and 15%, while the legitimate attempts are more widely distributed, but with more presence in the 90% to 100% interval. However, despite having the bigger concentration in different intervals, these two distributions never stand alone in any score interval, there are always attempts from impostors and legitimate users in every score. This may jeopardize the performance of the classifier in a possible implementation.

The evaluation rates are depicted in Table 5.6, with twenty different thresholds (t).

[4] All graphs and tables concerning the presentation of results were based on [36]

t	TMR	TNMR	FMR	FNMR	ACC
5	0,9472	0,3192	0,6808	0,0528	0,6332
10	0,92	0,452	0,548	0,08	0,686
15	0,8984	0,532	0,468	0,1016	0,7152
20	0,8768	0,604	0,396	0,1232	0,7404
25	0,8488	0,6696	0,3304	0,1512	0,7592
30	0,8184	0,7136	0,2864	0,1816	0,766
35	0,7776	0,76	0,24	0,2224	0,7688
40	0,7368	0,788	0,212	0,2632	0,7624
45	0,6976	0,8176	0,1824	0,3024	0,7576
50	0,652	0,844	0,156	0,348	0,748
55	0,62	0,86	0,14	0,38	0,74
60	0,5728	0,8824	0,1176	0,4272	0,7276
65	0,532	0,8984	0,1016	0,468	0,7152
70	0,4832	0,9152	0,0848	0,5168	0,6992
75	0,4296	0,9296	0,0704	0,5704	0,6796
80	0,3736	0,9536	0,0464	0,6264	0,6636
85	0,3032	0,9672	0,0328	0,6968	0,6352
90	0,2328	0,9832	0,0168	0,7672	0,608
95	0,1272	0,9952	0,0048	0,8728	0,5612
100	0	1	0	1	0,5

Table 5.6 - Rates calculation, with various thresholds (t)

With the lowest threshold, 5 in this case, the quantity of samples predicted as genuine is maximum, consequently, the number of true matches is the biggest because it catches almost every actual genuine attempts. However, the number of false matches is also the biggest because it also classifies most of the actual impostor attempts as genuine. By the contrary, the quantity of true non-matches and false non-matches are the lowest. As the threshold increases, more samples are predicted as impostors and less as genuine, leading to the decrease of true matches and false matches rates and increasing the true non-matches and false non-matches rates.

The threshold that provides the best accuracy value is 35, which corresponds to the turning point seen in the distributions chart. As the accuracy is the complementary of EER, that threshold is also where the EER is lower. In DET Curves chart, in Figure 5.2, is represented the trade-off between the FMR and FNMR, and there it can be seen better the EER point.

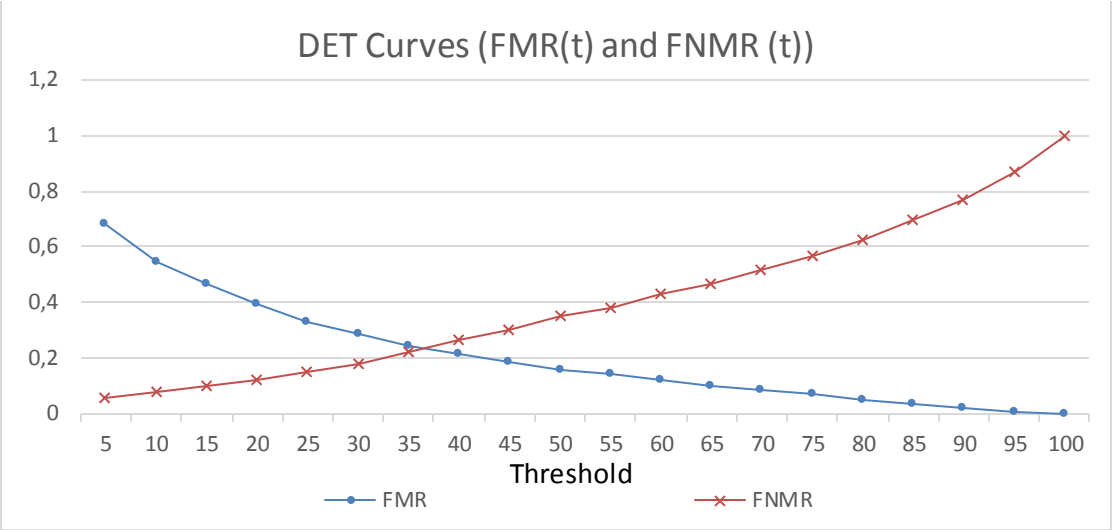


Figure 5.2 - DET Curves

With the DET Curves one can verify that the EER point is at the threshold 35. That is what commercial or not specialized applications are looking for, because this is a point of balance, and the lower the value of it, the better.

To improve the perception of the system performance, in Figure 5.3 is a ROC Curve. As said before, this curve doesn't depend of the threshold, it depicts the performance regarding the ability to correctly classify a genuine sample against the error rate in classify that type of samples.

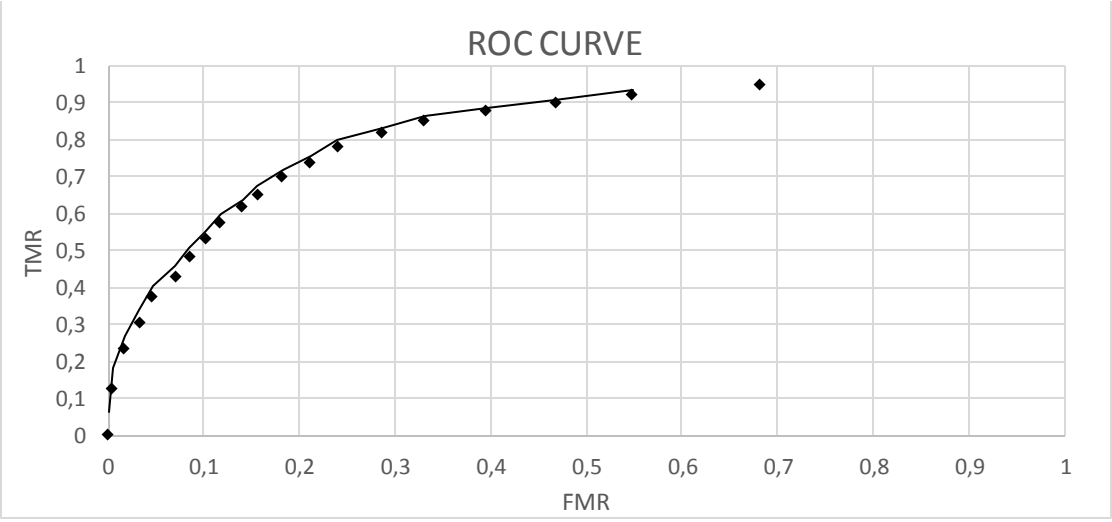


Figure 5.3 - ROC Curve

The ROC curve is as better as closer to the top left, in other words, with maximum TMR and minimum FRM. As expected from the values obtained in the Table 5.6, the curve is a little too far from that position. In fact, despite not being a bad result at all, the evaluation performed suggest that the samples from legitimate user and impostors are not sufficiently distinguishable for the classifier. As a consequence, one may assume that, with these conditions, touch data can't be used for authentication purposes, which doesn't mean that could not be used for continuous authentication, once the decision is made after a sequence of classifications.

5.1.1 Individual Analysis

In consequence of the system global results, the need for an individual analysis became more obvious. Because of the fact that the generated datasets for training and testing of each user were scaled with its unique scaling parameters and the parameters used to train the classifier were optimized for each dataset, it is pertinent to carry an evaluation to each user.

With that in mind, for each user is presented, below, a graph with the distributions of genuine attempts, from its own test dataset, and the distribution of attacks used in the general analysis. After, in Table 5.7 are shown the evaluation rates for the threshold with better accuracy. This way is compared the influence of each user dataset in the classification, using the same attacks. In Appendix B are the complete tables, with calculated rates for every threshold.

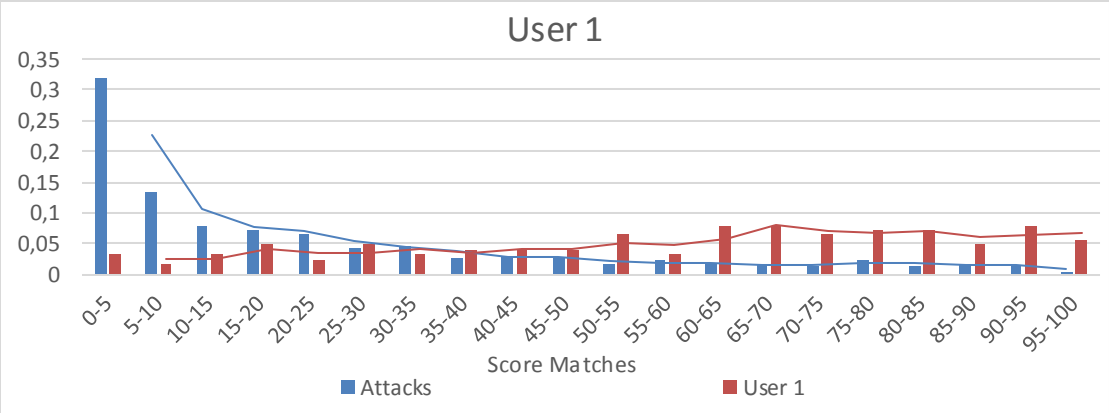


Figure 5.4 - Distribution of probabilities of user 1

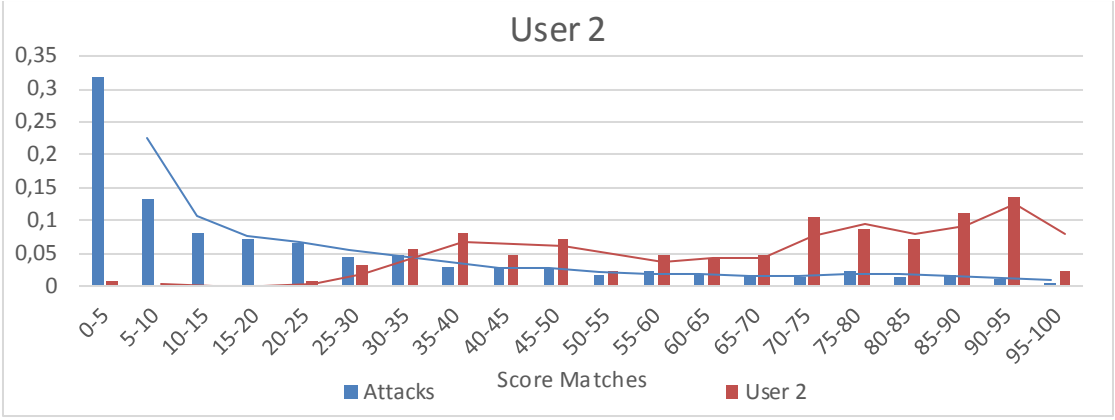


Figure 5.5 - Distribution of probabilities of user 2

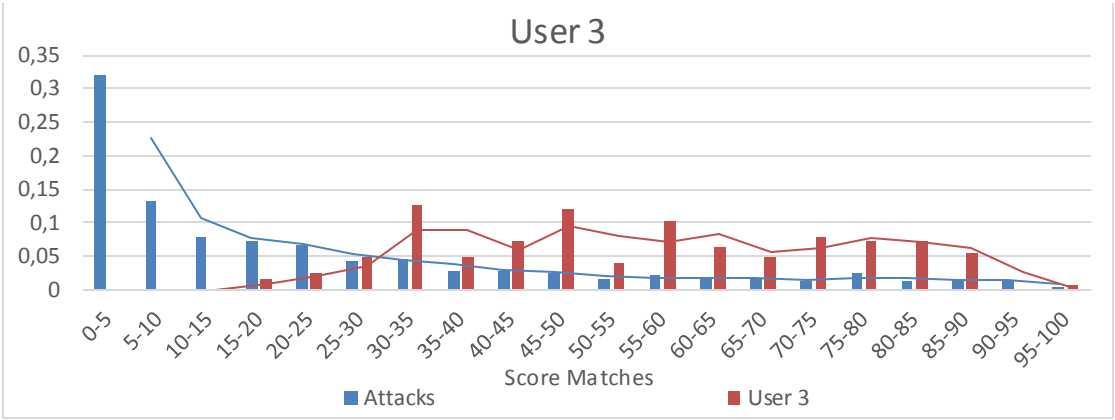


Figure 5.6 - Distribution of probabilities of user 3

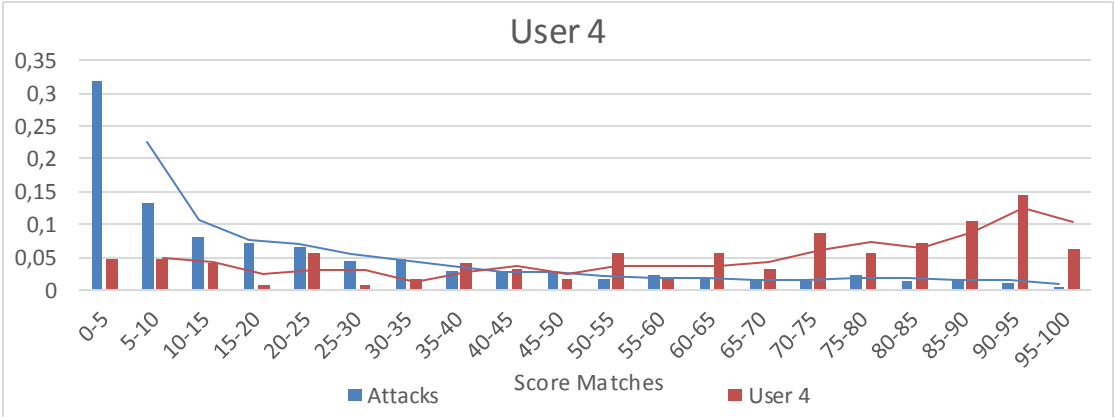


Figure 5.7 - Distribution of probabilities of user 4

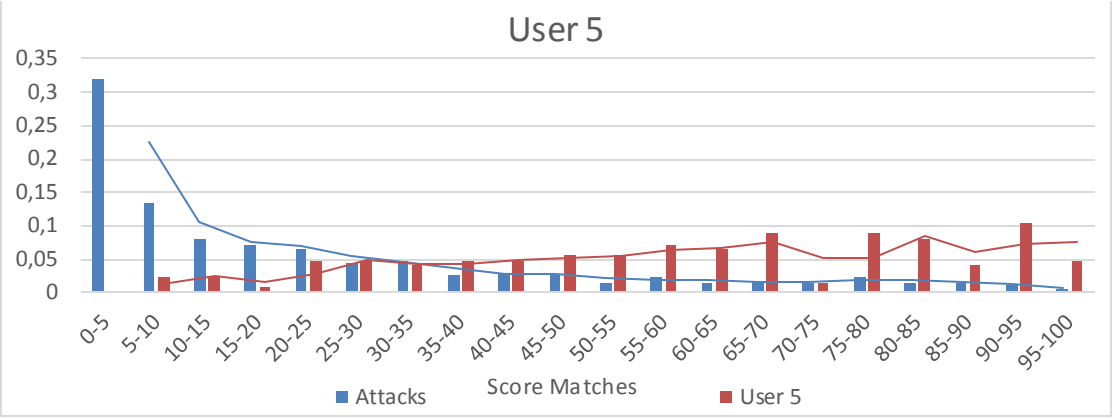


Figure 5.8 - Distribution of probabilities of user 5

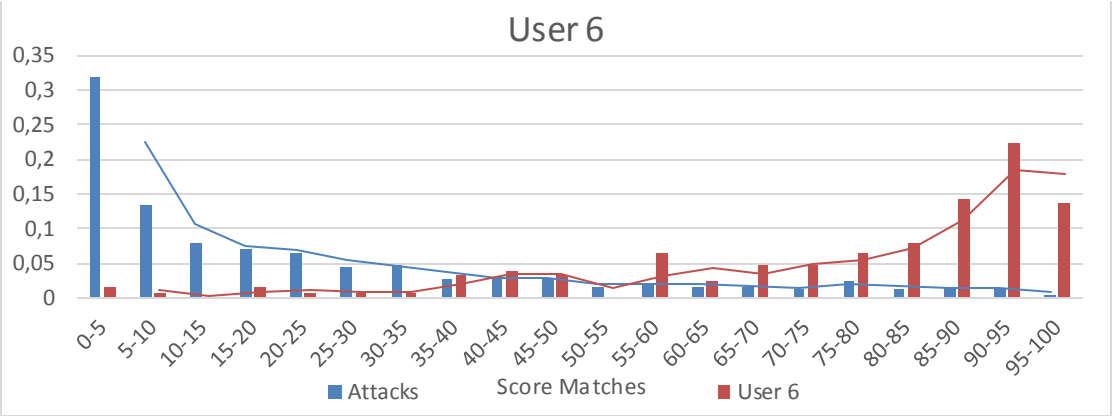


Figure 5.9 - Distribution of probabilities of user 6

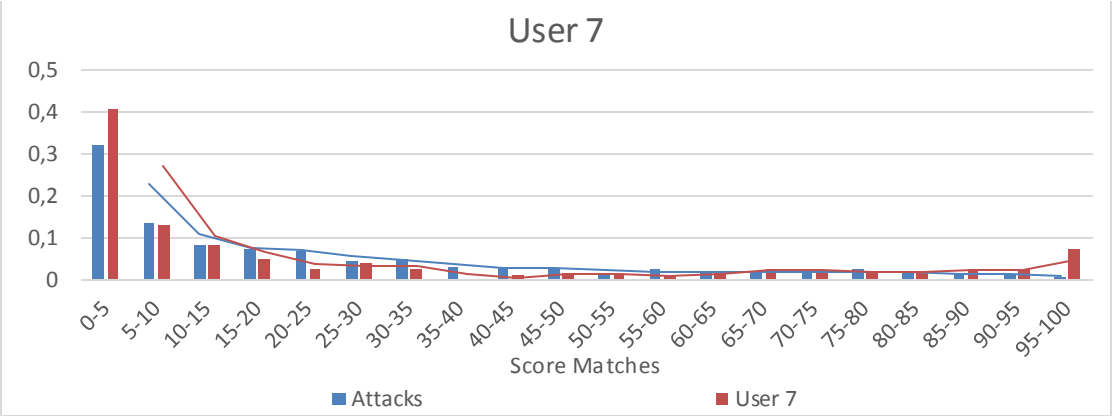


Figure 5.10 - Distribution of probabilities of user 7

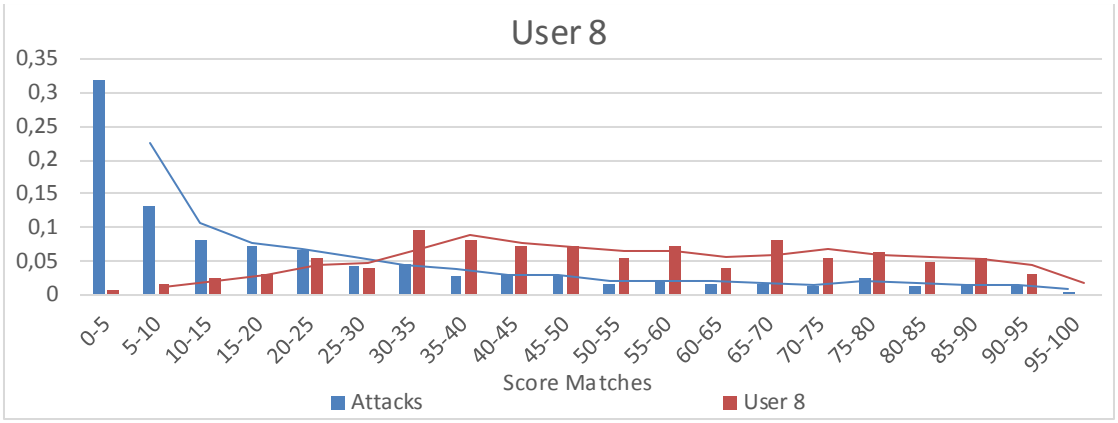


Figure 5.11 - Distribution of probabilities of user 8

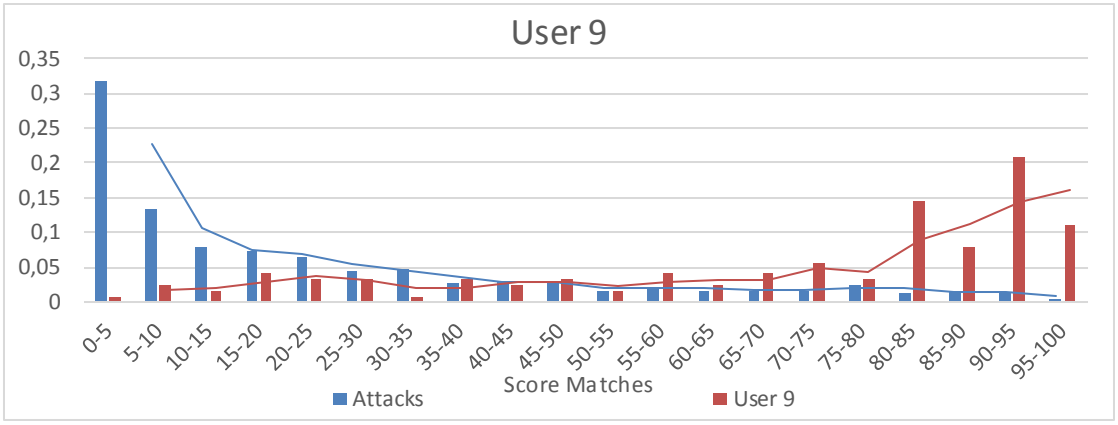


Figure 5.12 - Distribution of probabilities of user 9

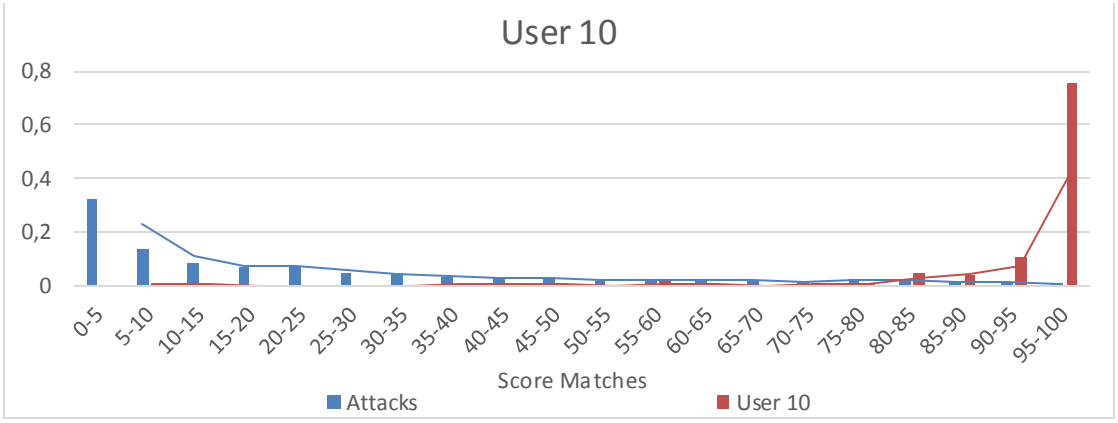


Figure 5.13 - Distribution of probabilities of user 10

Users	Best Threshold	TMR	TNMR	FMR	FNMR	ACC
1	45	68.8	67.2	32.8	31.2	68.0
2	35	89.6	73.6	26.4	10.4	81.6
3	30	91.2	57.6	42.4	8.8	74.4
4	35	77.6	87.2	12.8	22.4	82.4
5	20	94.4	79.2	20.8	5.6	86.8
6	55	83.2	80.0	20.0	16.8	81.6
7	10	46.4	72.8	27.2	53.6	59.6
8	30	82.4	48.8	51.2	17.6	65.6
9	35	84.0	89.6	10.4	16.0	86.8
10	80	94.4	100	0	5.6	97.2

Table 5.7 – Evaluation rates of all users, for the threshold that provides them better accuracy

The individual graphs and the table reveal big differences between some users. First aspect to note is that for some users the best accuracy rate is at a very low threshold, as in case of users 7 and 5.

The best prediction performance belongs to user 10, with 97% of accuracy for a threshold of 80, meaning that most of the genuine samples were classified in the 80% to 100% interval of probability, which can be seen on Figure 5.13.

On the other hand, the worst performances belong, in order, to user 7 with 59.6% of accuracy and to user 8 with 65.6% of accuracy, having both low thresholds, respectively 10 and 30. In the case of user 7 most of the samples were classified in the 0% to 10% range, so, despite having the lowest threshold of all, it has the biggest rate of false non-matches and consequently the lowest of true matches. Both distributions, attacks and legitimate attempts, are very similar as both have the peak of frequencies at the lowest score intervals, so is safe to assume that the classifier wasn't able to set apart genuine samples from impostor samples. Regarding to user 8, there's a different situation, the genuine samples are not aggregated to any particular threshold and the same goes to impostor samples, except for the peak in 0% to 5% interval. As the threshold is relatively low, this results in a high rate of true matches, the problem is that it also results in the highest false match rate of all users.

At last, to compare all the performances the ROC curves of all users were placed in the same chart.

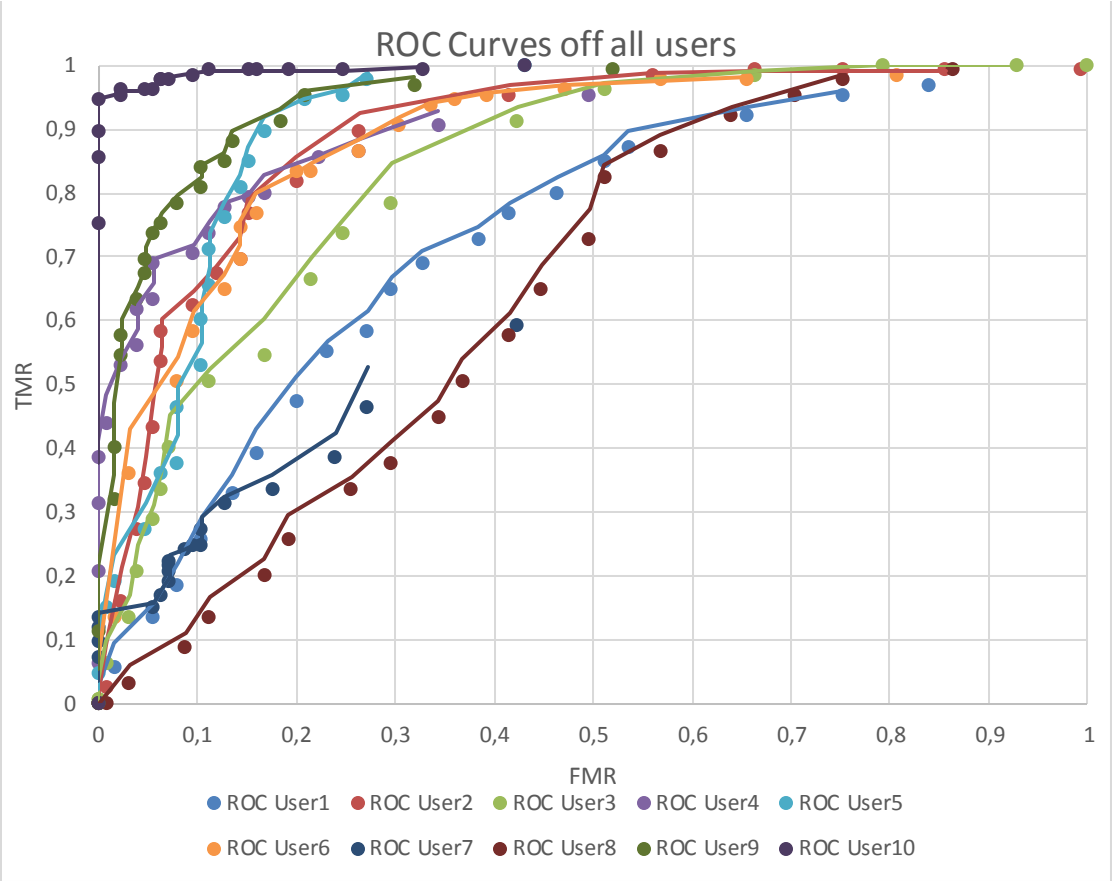


Figure 5.14 - ROC Curves off all users

With so many different performances and distributions of probabilities, there must be some differentiating factors that enable the classifier to work better with some users and worse with the other.

For that reason, was made a further analysis to the captured gestures. Were analyzed all samples from the training datasets of the users with worst and better performances, in relation to false non-matches, they were user 7 and user 10. Using all samples of each gesture, previously scaled in 0 to 1 range, was calculated the mean and standard deviation values of their features, and then was calculated the mean of all mean values and the standard deviation of all standard deviation values, resulting in a pair (mean, standard deviation) for each gesture. After that, gestures were ranked by the standard deviation value, from the highest to lower. This process was made, separately, with samples of each user. With standard deviation is possible

to understand in which gestures the features values vary more, in other words, the gestures that are more distinguishable, influencing more the classifier. Standard deviation is clearly the most important value, however, in combination with the mean value one can also compare the differences in the execution of the gesture by both users. If that pair of values, for the same gesture, is significantly different from one user to another, then it means that it was performed distinctly.

The results are shown in the next tables.

Ranking	Gesture	StdDev	Mean
1	Swipe from top right to bottom left	0,430	0,497
2	Swipe from right to left	0,427	0,475
3	Double tap with swipe from top left to bottom right	0,417	0,512
4	Swipe from top left to bottom right	0,409	0,487
5	Double tap with swipe from top to bottom	0,403	0,530
6	Swipe from left to right	0,398	0,495
7	Swipe from top to bottom	0,397	0,486
8	Zoom in bottom left top right	0,394	0,423
9	Swipe from bottom right to top left	0,393	0,497
10	Swipe from bottom left to top right	0,392	0,485
11	Double tap with swipe from bottom right to top left	0,391	0,514
12	Double tap with swipe from bottom left to top right	0,390	0,423
13	Double tap with swipe from bottom to top	0,390	0,483
14	Swipe from bottom to top	0,383	0,422
15	Double tap with swipe from right to left	0,383	0,453
16	Zoom in bottom right top left	0,382	0,423
17	Double tap with swipe from top right to bottom left	0,375	0,465
18	Double tap with swipe from left to right	0,373	0,386
19	Zoom out bottom left top right	0,373	0,506
20	Zoom out bottom right top left	0,360	0,419
21	Double tap	0,301	0,279

Table 5.8 - Gesture analysis of user 10

Ranking	Gesture	StdDev	Mean
1	Swipe from bottom to top	0,426	0,483
2	Swipe from bottom left to top right	0,413	0,534
3	Swipe from bottom right to top left	0,413	0,494
4	Swipe from top right to bottom left	0,413	0,531
5	Swipe from right to left	0,411	0,522
6	Swipe from top left to bottom right	0,404	0,453
7	Swipe from top to bottom	0,401	0,527
8	Swipe from left to right	0,399	0,376
9	Double tap with swipe from top to bottom	0,393	0,364
10	Double tap with swipe from right to left	0,391	0,459
11	Double tap with swipe from bottom to top	0,388	0,444
12	Double tap with swipe from top left to bottom right	0,384	0,443
13	Double tap with swipe from bottom left to top right	0,383	0,441
14	Double tap with swipe from bottom to top	0,383	0,444
15	Zoom out bottom left top right	0,382	0,500
16	Double tap with swipe from top right to bottom left	0,379	0,413
17	zoom out bottom right top left	0,378	0,464
18	Double tap with swipe from left to right	0,375	0,422
19	Zoom in bottom left top right	0,371	0,465
20	zoom in bottom right top left	0,356	0,445
21	Double tap	0,269	0,314

Table 5.9 - Gesture analysis of user 7

The most variable gesture, executed by user 10, is the *swipe from top right to bottom left*, while in case of user 7 is the *swipe from bottom to top*. There are many differences in the top of the gesture's ranking of each user, suggesting that the train models created from these samples are more influenced by different gestures.

At the same way that the first ranked gestures are the more influent in the classification, the last ranked are the ones with less variation and, consequently, less influence in the classification. With this in mind, and the fact that in both cases the last gesture is *double tap* (static double

tap), one can assume that if the gesture was removed from the datasets, the classification wouldn't differ much.

With this analysis is realized that the same gestures have distinct influence in the training of the classifiers, for different users. That can serve as motivation to a deeper analysis of gestures, evaluating if some gestures could be left aside, or if it must be given more importance to other, or even a combination between both.

5.2 Results of additional tests

Once the results of the project weren't sufficiently strong so that continuous authentication was proven to work with the extracted features, were performed additional. In this sub-section those tests are justified and described. In the end is presented a comparison between the results of each test and the original results, from last section.

In a the classification process, users are distinguished by the features of their gestures, being the distinction as better as more divergent are the feature values in relation to other users. Naturally, there are features that presents more variance and so are the ones which have more influence. By the other hand, there are also features that don't vary much from one user to another and so their inclusion could be counterproductive, this is because, instead of add data for distinguish users, they create common points in the user's profiles. With this in mind, was conducted a study in order to understand which are the more important features and which are the ones that could be removed.

Was selected the training dataset of user 3, chosen by a random function. After the analysis, the less important features were removed from that dataset and from the testing datasets, and the classification process was performed again with the new datasets. In the end was compared the results obtained with the original results of that user.

The analysis was made through Principal Component Analysis [35], using the machine learning suite WEKA [36]. To put it briefly, PCA calculates the directions in which data is more spread and expresses those directions in eigenvectors, one for each direction, with the corresponding eigenvalue, which indicates the weight of their features variance. Inside the

eigenvectors are the features, each one with a given score that represents their influence in that direction. All features score values are in the 0 to 1 range.

PCA returned 12 eigenvectors from the analysis of the training dataset. Weka calculates the proportion of the eigenvectors in relation to the others, based on their eigenvalue, attributes a ranking score (1 - proportion) and ranks them accordingly. It ranks also the features inside each by their score. Were chosen the first four eigenvectors, because together they accumulated nearly 60% of the calculated proportions leaving the remaining 40% to the other eight vectors. In the same way, the first ranked features of each eigenvector are the ones with more variance, so it was decided to choose the first eight features (around half of the total) of the four eigenvectors.

Eigenvector	**1st**	**2nd**	**3rd**	**4th**
Eigenvalue	3.65467	2.56233	1.92649	1.73491
Score	0.785	0.6343	0.521	0.4189
1st feature	Ave. Size 0.468	Max. Size 0.433	Distance Y 0.561	Distance X 0.477
2nd feature	Ave. Pressure 0.461	Velocity X 0.358	Velocity Y 0.506	Initial Time 0.408
3rd feature	Min. Size 0.455	Total Time 0.353	Velocity X 0.435	Total Time 0.365
4th feature	Min. Pressure 0.390	Velocity Y 0.348	Distance X 0.386	Initial X 0.332
5th feature	Final Y 0.266	Max. Pressure 0.334	Total Time 0.141	Velocity X 0.330
6th feature	Max. Pressure 0.264	Initial Time 0.281	Max. Pressure 0.102	Final X 0.320
7th feature	Max. Size 0.220	Distance X 0.280	Min. Pressure 0.099	Distance Y 0.224
8th feature	Distance Y 0.076	Distance Y 0.259	Fling Time 0.098	Initial Y 0.219

Table 5.10 - Best ranked eigenvectors and features

In the table above are presented the four best ranked eigenvectors and their features, in order. As all features are present was not possible to choose immediately which would be removed from the datasets so was necessary to calculate a second ranking score. That value calculation is given by equation 5.6.

$$\sum_{i=1}^{4} Es_i * Fs_i \tag{5.6}$$

In the equation, *Es* is the eigenvector score and *Fs* is the feature score. This way was given more importance to those features that were in the in the first eigenvectors, not forgetting those who appear more than once. After calculation the features were ranked by that value. The order was:

1- Distance Y
2- Velocity X
3- Distance X
4- Velocity Y
5- Max. Pressure
6- Total Time
7- Max. Size
8- Ave. Size
9- Ave. Pressure
10- Min. Pressure
11- Min. Size
12- Initial Time
13- Final Y
14- Initial X
15- Final X
16- Initial Y
17- Fling Time

X and Y coordinates have little influence in the dataset. This is not surprising because the starting and ending areas were indicated in the application and were the same for all users. In contrast, since some users tend to perform gestures quicker than others, and that was observed in the data collection procedure, it is strange having the fling time as the last ranked attribute.

Were realized two different experiments, in the first were left in the datasets only the eight best ranked features, about half of the feature set, and in the second were left the twelve best ranked, roughly three quarters of the set.

Besides this analysis to the influence of the features, was also made one additional test, this time analyzing the influence of a particular category of gestures: zoom. Zoom gestures differ from the other because, each one have two vectors of touch data in the dataset, totaling 40 samples in a dataset of 125, 32% of the data. Because of this were removed all data samples of

zoom gestures of the training dataset and of the testing datasets of the same user 3, in order to see how the classifier reacts behaves.

Next are presented the classification results, for each different analysis is shown the chart with the distributions of relative frequencies. In the end is a table with all rates and a chart with the ROC Curves.

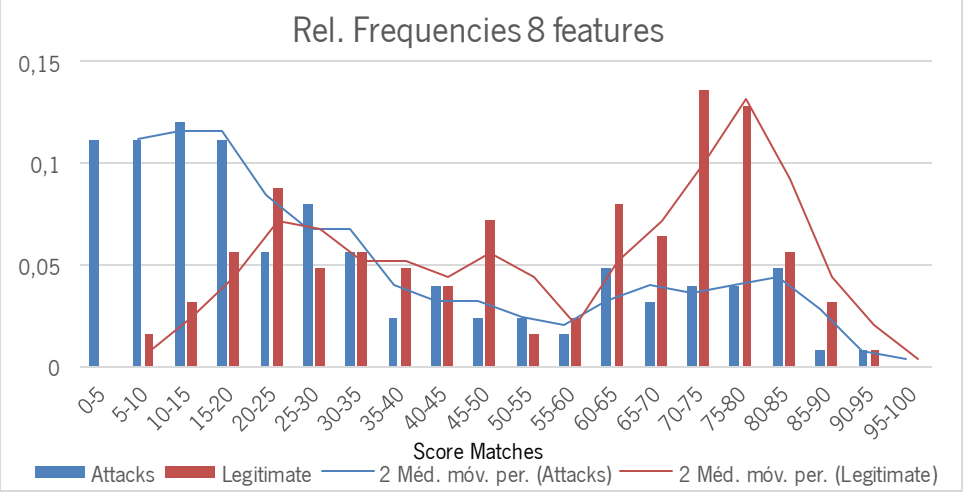


Figure 5.15 - Distributions of probabilities of user 3, using 8 features

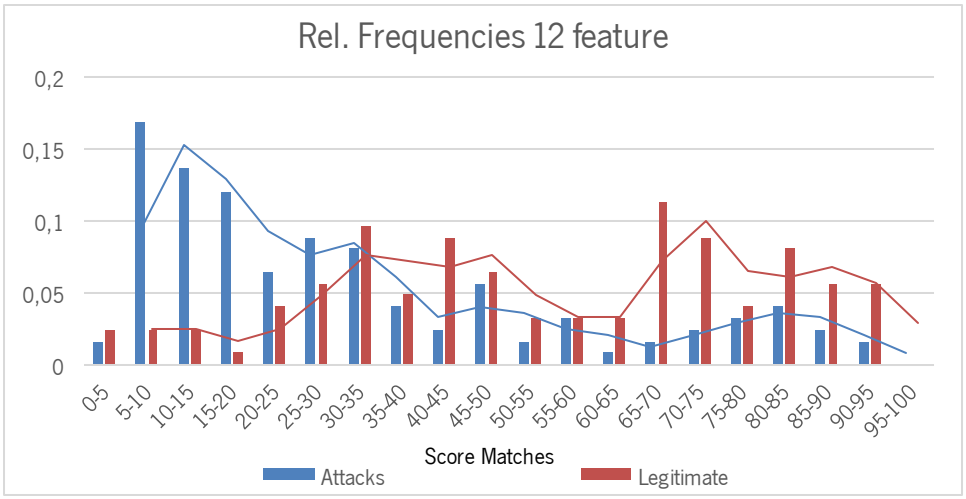


Table 5.11 - Distributions of probabilities of user 3, using 12 features

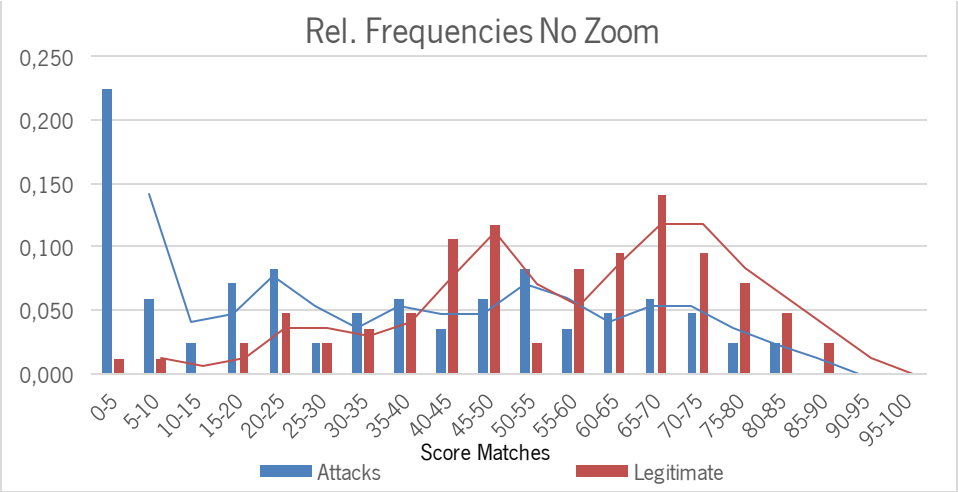


Table 5.12 - Distributions of probabilities of user 3, with no zoom gestures

	Best Threshold	TMR	TNMR	FMR	FNMR	ACC
8 features	20	89.6	45.6	54.4	10.4	67.6
12 features	30	82.4	59.2	40.8	17.6	70.8
No zoom	40	80.0	58.8	41.2	20.0	69.4
Original	30	91.2	57.6	42.4	8.8	74.4

Table 5.13 - Comparison between original results and additional tests results

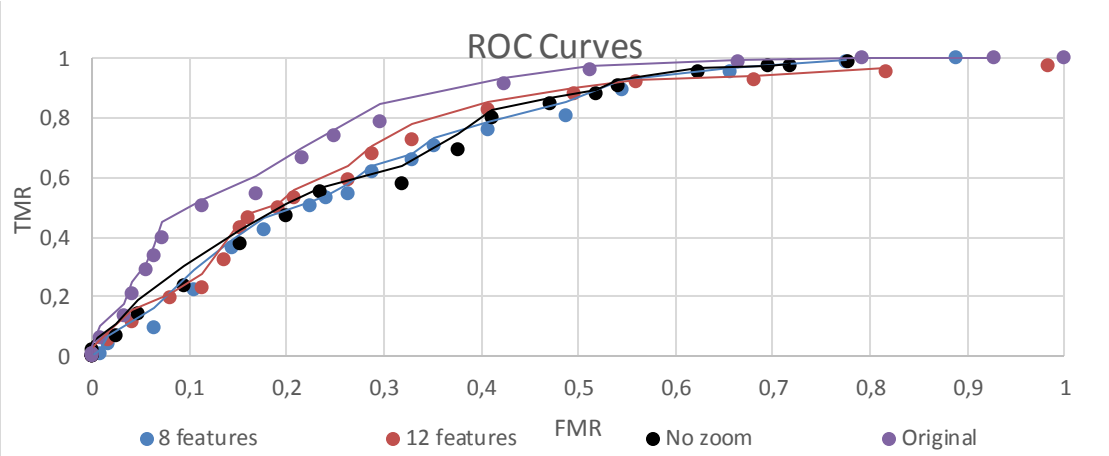


Figure 5.16 - Comparison of ROC curves

None of the additional tests gave better results. In fact, looking at the charts with distributions of relative frequencies is easily seen that they all tend to mistake legitimate attempts with attackers.

5.3 Summary

With the results obtained by the various experiments and analysis, it safe to say that the touch data collected in this project, from the interaction of users in the touchscreen, can't serve as a biometric characteristic, as consequence, an implementation of a continuous authentication system in these conditions is not feasible.

Was performed an evaluation of the proposed system based on the results of the classifier, in terms of true matches, false matches, true non-matches, false non-matches and accuracy rates. These rates were calculated using various thresholds, to find the one in which the system has better performance and to have an insight about how the system behaves to threshold changes. At the best threshold the system presented an accuracy of roughly 77%, with similar rates of true matches and true non-matches. However, was observed that both genuine and impostor attempts are too much scattered through all score values, resulting in high rates of classification errors.

These values are not acceptable because an authentication system needs to have the lowest possible false match rate, and in this case nearly a quarter of the impostor attempts are validated. The ability to reject impostors, the specificity, can be increased by using a higher threshold, which also decreases the sensibility, this is a normal situation in authentication systems. However, in this case that wouldn't work. For example, to increase the specificity to about 90%, the sensibility would decrease to nearly 50%, and that would turn the system useless.

Was conducted an individual analysis of each participant, through which was possible to verify that the system behavior is very different between user, having one user with an accuracy rate just above 97%, while others had an poor performance but for different reasons. That indicates that the system must be optimized to each user.

By performing a study about which gestures had more influence in the classification, was concluded that are some gestures don't vary much in different so they must the ones with less influence.

Additional tests were made, seeking for better results. Were determined the features influence ranking in the dataset of a user, and then was performed the classification process with the first eight features and after with the first twelve. Was also conducted an analysis without zoom gestures. None of these tests provided best performance of the system. This could mean that, even if some features have little influence, it could make the difference between a correct and a wrong classification.

One must not forget that this system is to be implemented in different applications which have, certainly, different demands in terms of user interaction. Considering this, to optimize the system to each user, considering the application, should be made a further study about which are the more appropriate gestures and features to include with a particular user and in a specific type of application.

A last conclusion that must be made is that, due the fact that SVM was the only used classifier, it would be interesting to compare the performance of other classifiers in this system, and perhaps a combination of classifiers. A future work should also be exist in that scope.

6.Conclusions and Future Work

Touchscreen devices have outnumbered the traditional computers. As their number increases, so that happens with the security concerns regarding them. This is because they, progressively, access and store more sensitive information. Is then necessary to protect these devices from unauthorized accesses. As known by now, the traditional means of authentication can't offer the needed degree of security because once unlocked the device is vulnerable to attacks.

In this dissertation was proposed a system to provide continuous authentication of users in mobile devices, based on their interactions on the touchscreen. After study the state-of-art of the biometric system was proposed the architecture and operation of this system. To collect the touch data of the input gestures was developed a mobile application, to Android devices. The application was responsible to collect the data, extract the previous selected features, organize them in one single vector and finally store it in the device storage. To proceed to the classification were created datasets for training and testing for each participant, complementing them with random sample picks from the others. By analyzing the related work was chosen SVM as the classifier. Finally were grouped the results and was made the evaluation of the system.

One of the most challenging aspects was the application development, specially the multitouch gestures processing. Each of the fingers used had to be analyzed individually, but the Android event handler reports samples of all them together in the same event. Zoom events also had a different behavior regarding fling. At first, by experimentation, was seen that all scrolls had a fling presence, so it were coupled scroll and fling as the same gesture, but as zoom events have fingers moved in different directions that probably canceled the fling effect of the gesture, because of that the fling recording was to be made differently

Unfortunately, results is show that is not viable to implement continuous authentication with the biometrics characteristic extracted. The system achieved, for the best threshold, nearly 76.8% of accuracy and error rates of 24% to FMR and 22.2% to FNMR, with a corresponding EER of 23.2%. These results could have multiple interpretation, the problem may be in the selected

features, in the selected gestures, in the parameters used to tune SVM or in the use of SVM at all, which among all is the less likely. In order to confirm that, were made three new classification rounds, excluding some of the less variant features, and in one of them excluding the zoom gestures. After these tests were not obtained better results.

Was made an analysis that shown that the system behavior was too different among the all the users, which can mean that the selected features are best fitted to a particular set of users. Was also made an additional test to compare the variance of the features in each gesture, using the features of the user with the best and worst results, to see which were the most influencing gestures in each one. The result shown that was no similarity in the most influencing gestures in both users, but there was in the less influencing ones, like static double taps, concluding that those probably can be removed from the analysis. This way is also proved that different gestures have different influence in distinct users.

Though the results were not good as expected, they are far away from the useless point. For continuous authentications purposes, the achieved values are not so bad as for simple authentication, because of the fact that the decision regarding the user's identity is only made after a sequence of classifications. This means that, even with this results, a system might work, but here there is also big behavior differences of the classifier for different users, turning this not viable to implement.

This project was meant essentially to prove that the differences in the behavior of people, when performing gestures in a touchscreen, was distinct enough to serve as a biometric characteristic. One can also not forget that the gestures not free performed, which meant that the differences in the behavior was reduced drastically. This because, this project was thought to be used in application level and not at the operating system level, and application have different need in terms of gestures, and some of them require the same gestures. With these results, there is hope that with a better analysis of gestures and features is possible to increase the accuracy and lower the error rates, making this biometrics characteristic suitable to the proposed system.

Regardless the results there were some accomplishments that are worth to be noted. First the application in developed in way that is possible to collect all gestures inputted in the touchscreen and not only those chosen to be collected, for example is possible to collect the

touch data of multitouch gestures up to ten fingers at the same time, or collect circular or no defined-shaped gestures also. The data extracted is consistent and was rigorously acquired so there is now a reliable set of touch data that can be used in future tests and implementations. Through the scripts built during the project is possible to instantly construct the datasets for training and for testing and scale them in different scales. It is also possible to find the best tuning parameters for use in SVM for all datasets at the same time and use them automatically when training the classifiers. The classification is also made easily through the scripts.

This project was only the first step in towards the implementation of the system that was proposed earlier. That way there is a lot of work to do in the future:

- It is needed a further feature analysis, to determine which of the chosen features are worth to maintain and which are to be deleted. It is also pertinent to study the impact of new features in the dataset;
- The gesture sequence must be analyzes also to determine, in the same way, which are to keep or to delete;
- Take the experiments with a larger population and different handedness. To see if are considerable differences between classes of people that were not possible to see with the ten participants;
- Make the classifications using different classifiers, or with combination of classifiers;
- It would be very important to make a deeper analysis to each gesture, individually, to determine if there are features more important than the others. Equally it is pertinent to study further the individualities of different users. With these two analysis could be achieved important advances towards the optimization of the system to different users and to meet the different requirements of applications.

There are, certainly, other aspects to be studied, but these are the most pertinent to analyze immediately.

References

[1] Gartner, "Forecast: PCs, Ultramobiles, and Mobile Phones, Worldwide, 2010-2017, 4Q13 Update." [Online]. Available: http://www.gartner.com/newsroom/id/2645115. [Accessed: 01-Sep-2014].

[2] CNNMoney, "Mobile apps overtake PC Web usage in U.S." [Online]. Available: http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/. [Accessed: 01-Sep-2014].

[3] IDC, "smart connected device market forecast." [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS24314413. [Accessed: 01-Sep-2014].

[4] J. Angulo and E. Wästlund, "Exploring touch-screen biometrics for user identification on smart phones," *Priv. Identity Manag. Life*, 2012.

[5] Verizon Wireless, "Fingerprint Unlocking, Facial Recognition and Other Security Features." [Online]. Available: http://www.verizonwireless.com/mobile-living/tech-smarts/how-does-fingerprint-scanning-facial-recognition-work-technology/. [Accessed: 12-Sep-2014].

[6] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 136–148, Jan. 2013.

[7] J. Ferreira, H. Santos, and B. Patrao, "Intrusion Detection Through Keystroke Dynamics," in *Proceedings of the 10th European Conference on Information Warfare and Security*, 2011, pp. 81–90.

[8] R. Yampolskiy and V. Govindaraju, "Behavioural biometrics: a survey and classification," *Int. J. Biom.*, vol. 1, no. 1, pp. 81–113, 2008.

[9] Watchful Software, "TypeWATCH - eBiometrics Security." [Online]. Available: https://www.watchfulsoftware.com/en/products/typewatch. [Accessed: 12-Nov-2014].

[10] C. Bo, L. Zhang, and X.-Y. Li, "SilentSense: Silent User Identification via Dynamics of Touch and Movement Behavioral Biometrics," Aug. 2013.

[11] H. Khan, A. Atwater, and U. Hengartner, "Itus: An Implicit Authentication Framework for Android," in *Proceedings of the 20th annual international conference on Mobile computing and networking - MobiCom '14*, 2014, pp. 507–518.

[12] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, 2012, p. 987.

[13] M. Uddin, S. Sharmin, A. Ahmed, and E. Hasan, "A Survey of Biometrics Security System," *IJCSNS*, 2011.

[14] A. K. Jain, A. Ross, and S. Prabhakar, "An Introduction to Biometric Recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 4–20, Jan. 2004.

[15] P. S. Magalhães, K. Revett, and H. D. dos Santos, "Critical aspects In authentication graphic keys." 2006.

[16] K. Delac and M. Grgic, "A survey of biometric recognition methods," in *46th International Symposium Electronics in Marine, 2004. Proceedings Elmar 2004. 16-18 June*, 2004, pp. 184–193.

[17] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of biometrics*. Springer-Verlag New York Inc, 2008, pp. 189–209.

[18] P. S. Magalhães, "Estudo de viabilidade da utilização de tecnologias biométricas comportamentais na autenticação do cidadão perante os serviços electrónicos do Estado." 02-Sep-2008.

[19] P. S. Magalhães and H. D. dos Santos, "Biometria e autenticação." Associação Portuguesa de Sistemas de Informação, 01-Oct-2003.

[20] J. Daugman, "How Iris Recognition Works," in *The Essential Guide to Image Processing*, 2009, pp. 715–739.

[21] R. Moskovitch and C. Feher, "Identity theft, computers and behavioral biometrics," *IEEE International Conference on Intelligence and Security Informatics, 2009. ISI'*, pp. 155–160, 2009.

[22] F. P. R. Colas and F. der W. en Natuurwetenschappen, "Data mining scenarios for the discovery of subtypes and the comparison of algorithms," Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University.

[23] V. Gandhi and J. Prajapati, "Review on Comparison between Text Classification Algorithms," *Int. J.*, vol. 1, no. 3, pp. 1–4, 2012.

[24] F. Lotte and M. Congedo, "A review of classification algorithms for EEG-based brain–computer interfaces," *Journal of Neural Engineering,* Institute of Physics: Hybrid Open Access, 2007, 4. <inria-00134950>

[25] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," vol. 31, pp. 249–268, 2007.

[26] H. Saevanee and P. Bhatarakosol, "User Authentication Using Combination of Behavioral Biometrics over the Touchpad Acting Like Touch Screen of Mobile Device," *2008 Int. Conf. Comput. Electr. Eng.*, pp. 82–86, Dec. 2008.

[27] W. Shi, J. Yang, Y. Jiang, F. Yang, and Y. Xiong, "SenGuard: Passive user identification on smartphones using multiple sensors," in *International Conference on Wireless and Mobile Computing, Networking and Communications*, 2011, pp. 141–148.

[28] T. Feng, Z. Liu, K. A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen, "Continuous mobile authentication using touchscreen gestures," in *2012 IEEE International Conference on Technologies for Homeland Security, HST 2012*, 2012, pp. 451–456.

[29] IDC, "Worldwide Smartphone Shipments Edge Past 300 Million Units in the Second Quarter; Android and iOS Devices Account for 96% of the Global Market." [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS25037214. [Accessed: 14-Nov-2014].

[30] "Android Developers." [Online]. Available: http://developer.android.com/index.html. [Accessed: 14-Nov-2014].

[31] "GestureDetector.SimpleOnGestureListener | Android Developers." [Online]. Available: http://developer.android.com/reference/android/view/GestureDetector.SimpleOnGestureListener.html. [Accessed: 14-Nov-2014].

[32] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, *Top 10 algorithms in data mining*, vol. 14, no. 1. 2007, pp. 1–37.

[33] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, pp. 1–39, 2011.

[34] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," no. 1, pp. 1–16, 2003.

[35] J. Shlens, "A tutorial on principal component analysis: derivation, discussion and singular value decomposition," *Online Note httpwww snl salk edushlenspubnotespca pdf*, vol. 2. pp. 1–16, 2003.

[36] M. Hall, H. National, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software : An Update," *SIGKDD Explor.*, vol. 11, pp. 10–18, 2009.

[37] Santos, H., "Controlo de Acesso e Autenticação", Dpt. Sistemas de Informação, Universidade do Minho, 2014

[38] Khan, H., Atwater, A., & Hengartner, U. (2014). Itus: An Implicit Authentication Framework for Android. In Proceedings of the 20th annual international conference on Mobile computing and networking - MobiCom '14 (pp. 507–518). New York, New York, USA: ACM Press. doi:10.1145/2639108.2639141

[39] Amari, S., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. Neural Networks, 12(6), 783–789. doi:10.1016/S0893-6080(99)00032-5

[40] Revett, K., Jahankhani, H., Magalhães, S. T. De, & Santos, H. M. D. (2008). A survey of user authentication based on mouse dynamics. In Communications in Computer and Information Science: Vol .12. Global e-security (pp. 210–219).

[41] Banerjee, S. P., & Woodard, D. (2012). Biometric Authentication and Identification Using Keystroke Dynamics: A Survey. doi:10.13176/11.427

[42] Bergadano, F., Gunetti, D., & Picardi, C. (2002). User authentication through keystroke dynamics. ACM Transactions on Information and System Security. doi:10.1145/581271.581272

[43] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 167, 121–167

[44] Chang, T.-Y., Tsai, C.-J., & Lin, J.-H. (2012). A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices. Journal of Systems and Software, 85(5), 1157–1165. doi:10.1016/j.jss.2011.12.044

[45] Clarke, N. L., & Furnell, S. M. (2007). Advanced user authentication for mobile devices. Computers & Security, 26(2), 109–119. doi:10.1016/j.cose.2006.08.008

[46] Damopoulos, D., Kambourakis, G., & Gritzalis, S. (2013). From keyloggers to touchloggers: Take the rough with the smooth. Computers & Security, 32, 102–114. doi:10.1016/j.cose.2012.10.002

[47] Gamboa, H., & Fred, A. (2004). A behavioral biometric system based on human-computer interaction. Defense and Security.

[48] Jakobsson, M., Shi, E., Golle, P., & Chow, R. (2009). Implicit authentication for mobile devices.

[49] Jansen, W. (2003). Authenticating users on handheld devices. *Proceedings of the Canadian Information Technology Security Symposium.*

[50] Lau, E., Liu, X., Xiao, C., & Yu, X. (2004). Enhanced user authentication through keystroke biometrics. Massachusetts Institute of Technology.

[51] Magalhães, P. S., Revett, K., & Santos, H. D. dos. (2006). Keystroke dynamics: stepping forward in authentication.

[52] Mccue, R. (2009). A Comparison of the Accuracy of Support Vector Machine and Naïve Bayes Algorithms In Spam Classification, 1–17.

[53] Meng, W. (2013). Design of behavioural biometric based authentication with an adaptive mechanism on mobile phones, (March), 1–16.

[54] Meng, Y., Wong, D. S., Schlegel, R., & Kwok, L. F. (2013). Touch gestures based biometric authentication scheme for touchscreen mobile phones. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 7763 LNCS, pp. 331–350). doi:10.1007/978-3-642-38519-3_21

[55] Monrose, F., & Rubin, A. D. (2000). Keystroke dynamics as a biometric for authentication. Future Generation Computer Systems, 16(4), 351–359. doi:10.1016/S0167-739X(99)00059-X

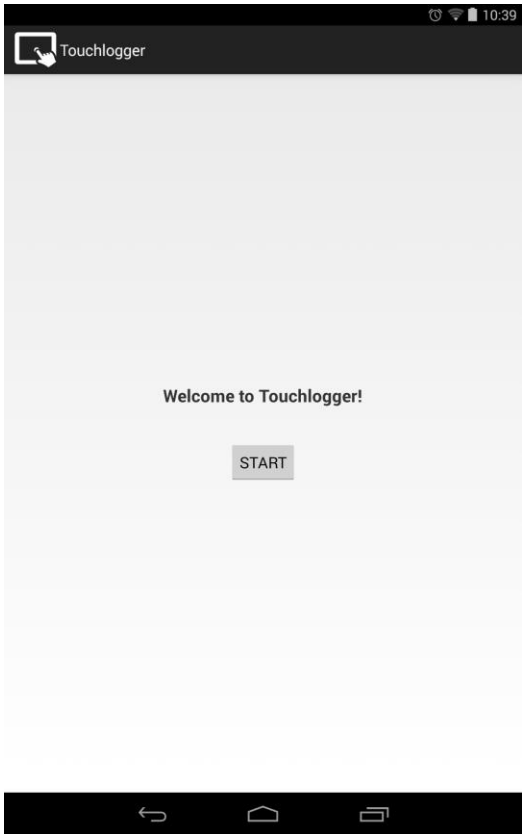
[56] Niinuma, K., Park, U., & Jain, A. K. (2010). Soft Biometric Traits for Continuous User Authentication. IEEE Transactions on Information Forensics and Security, 5(4), 771–780. doi:10.1109/TIFS.2010.2075927

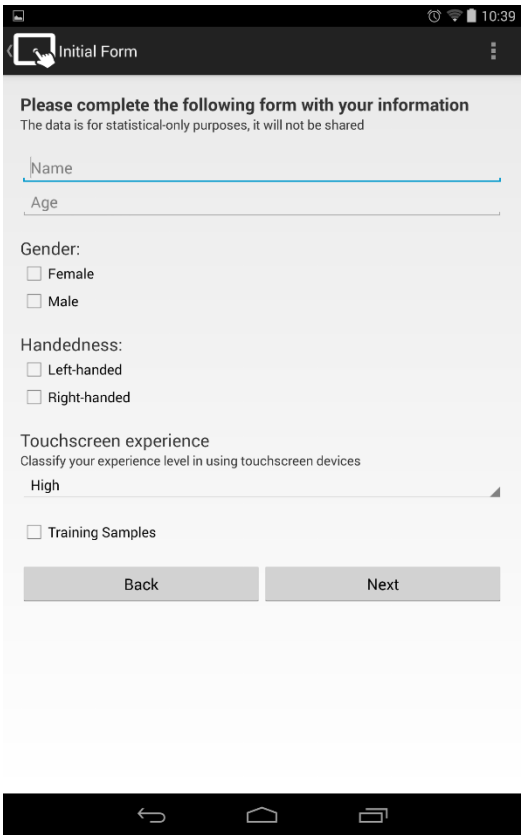
[57] Nonaka, H., & Kurihara, M. (2004). Sensing Pressure for Authentication System Using Keystroke Dynamics. International Conference on Computational Intelligence Instanbul, 578–581.

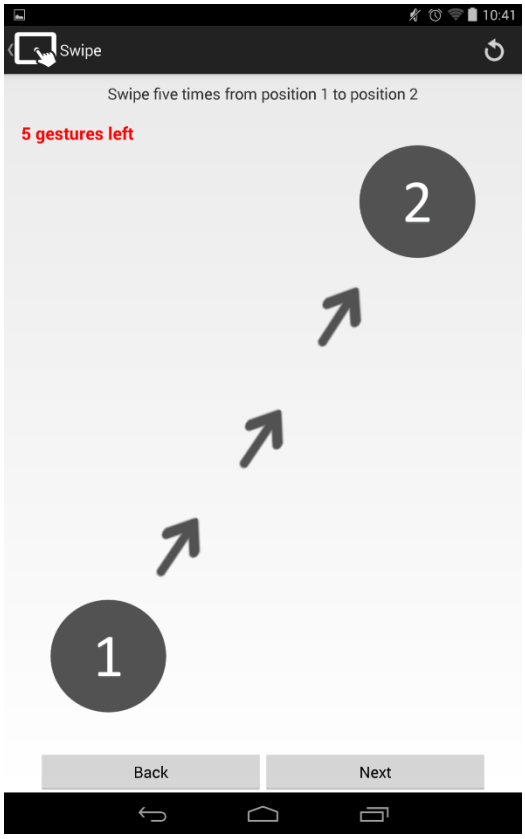
[58] Pedregosa, F., & Varoquaux, G. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830

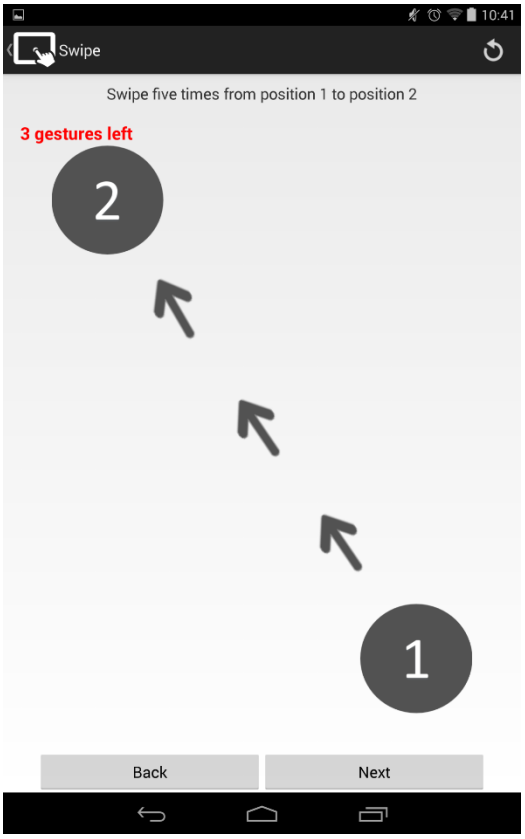
Appendix A

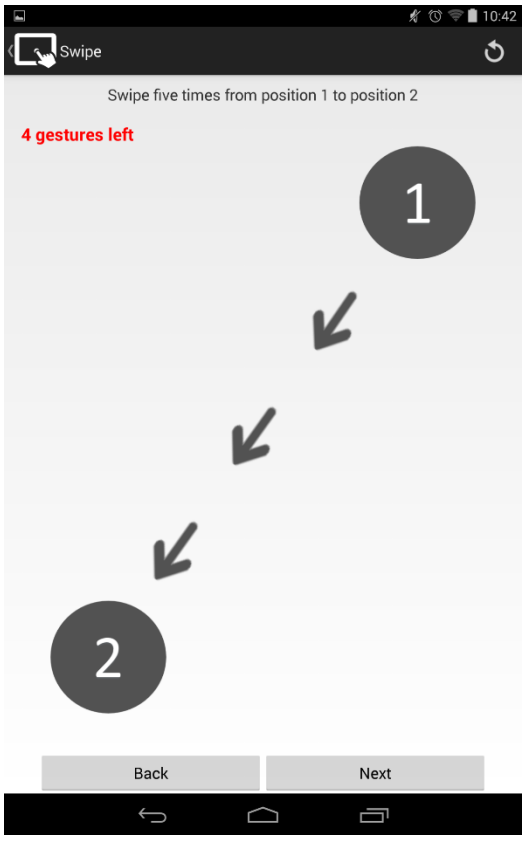
In this appendix are all the user interface provided by the activities, they are in the exact same order used in the data collection procedure.

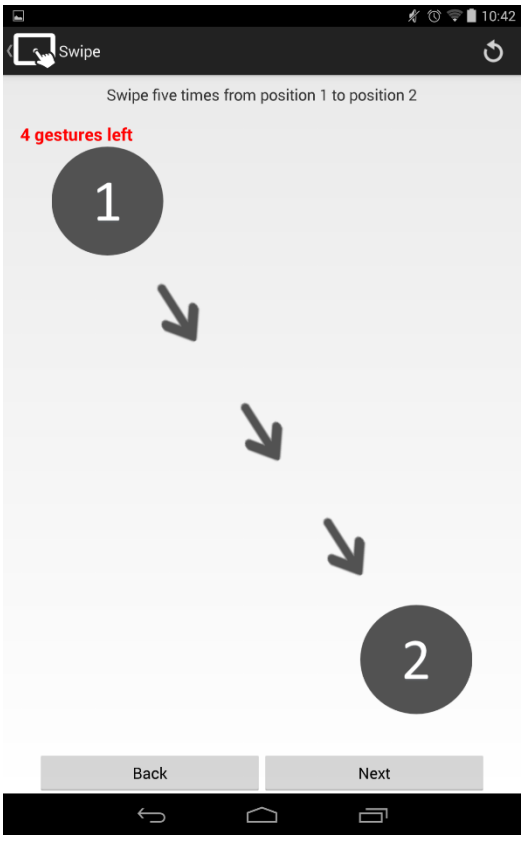


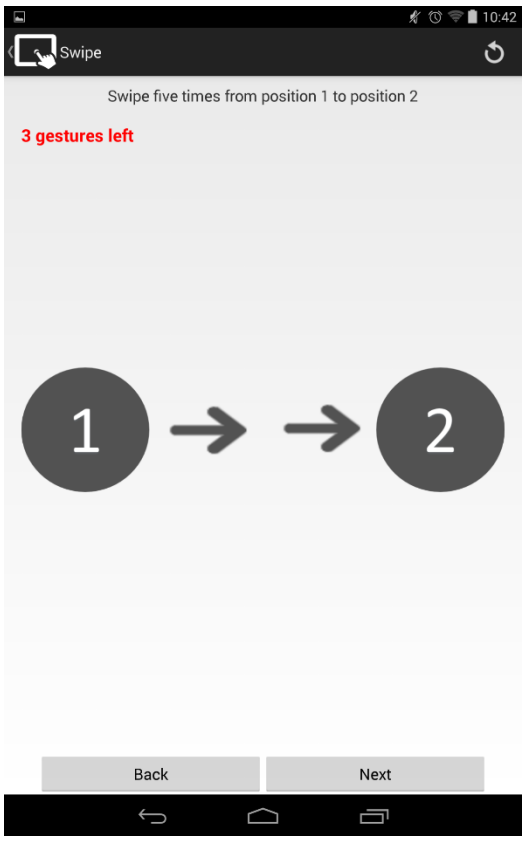


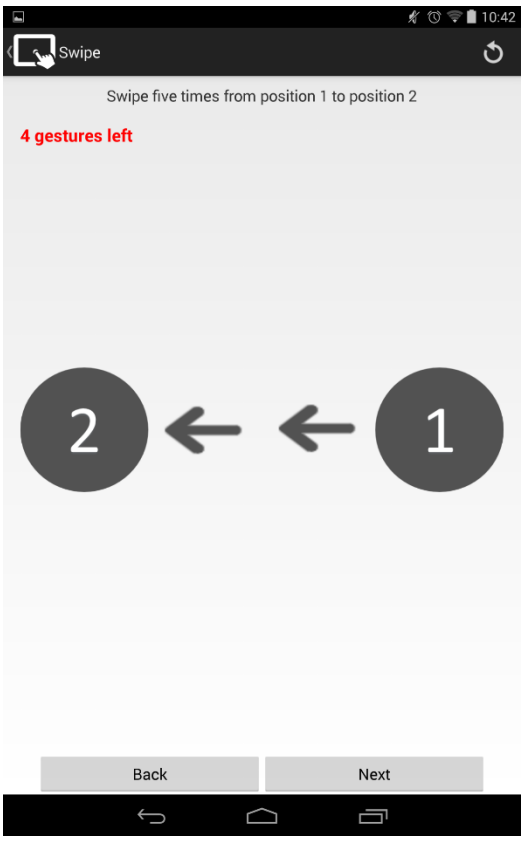

10:41
Swipe
Swipe five times from position 1 to position 2
5 gestures left
2
1
Back
Next

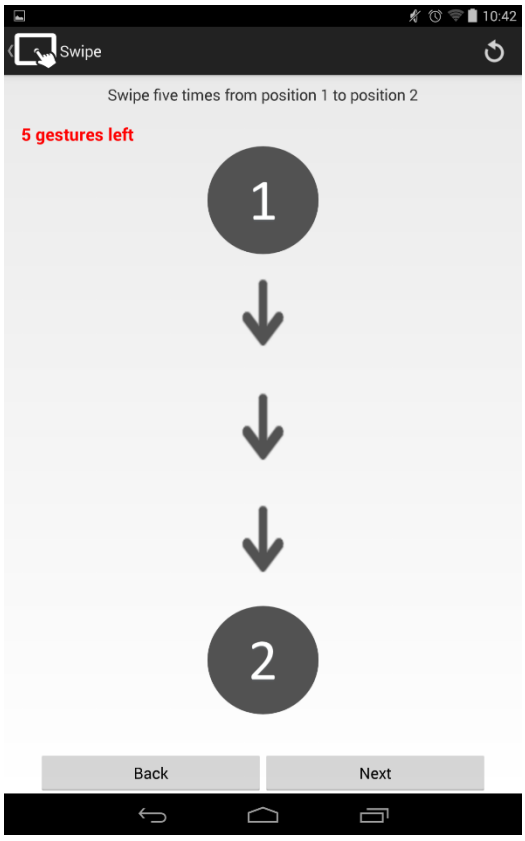

10:41
Swipe
Swipe five times from position 1 to position 2
3 gestures left
2
1
Back
Next

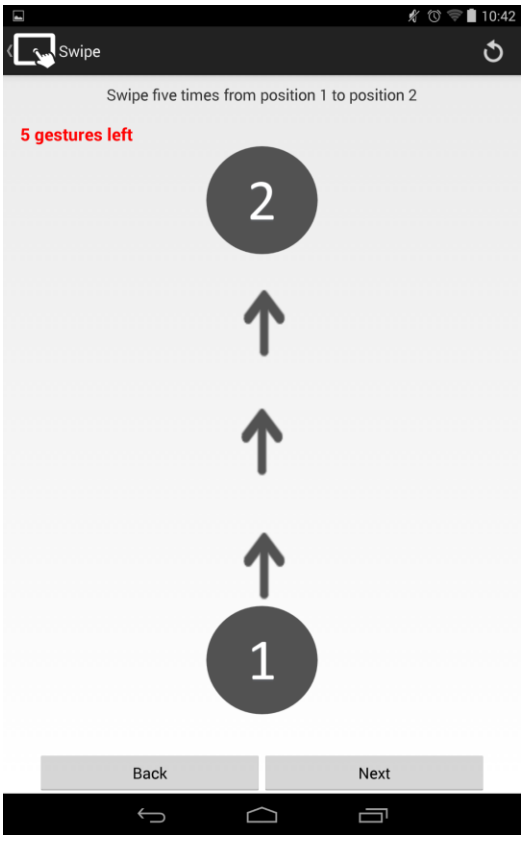

10:42
Swipe
Swipe five times from position 1 to position 2
4 gestures left
1
2
Back
Next

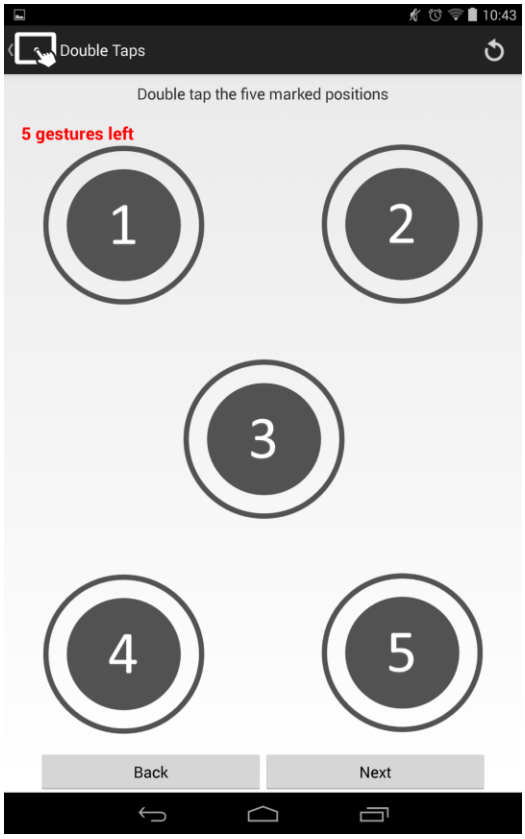

10:42
Swipe
Swipe five times from position 1 to position 2
4 gestures left
1
2
Back
Next

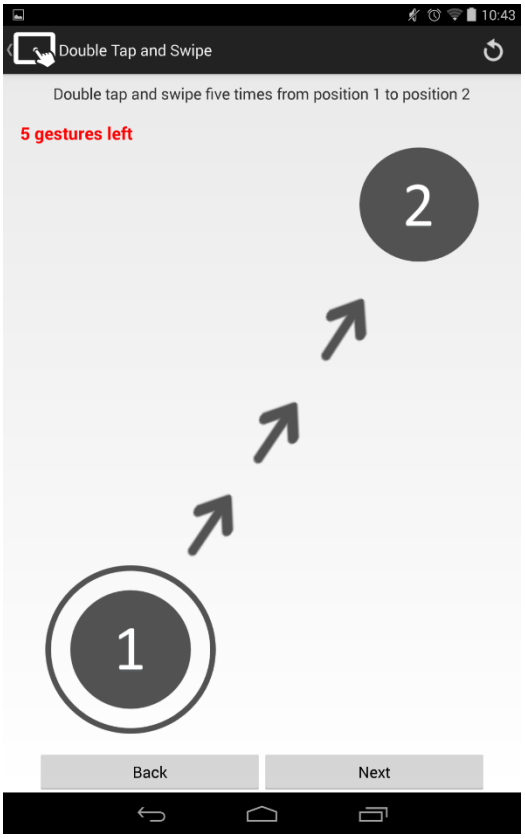

10:42
Swipe
Swipe five times from position 1 to position 2
3 gestures left
1
2
Back
Next

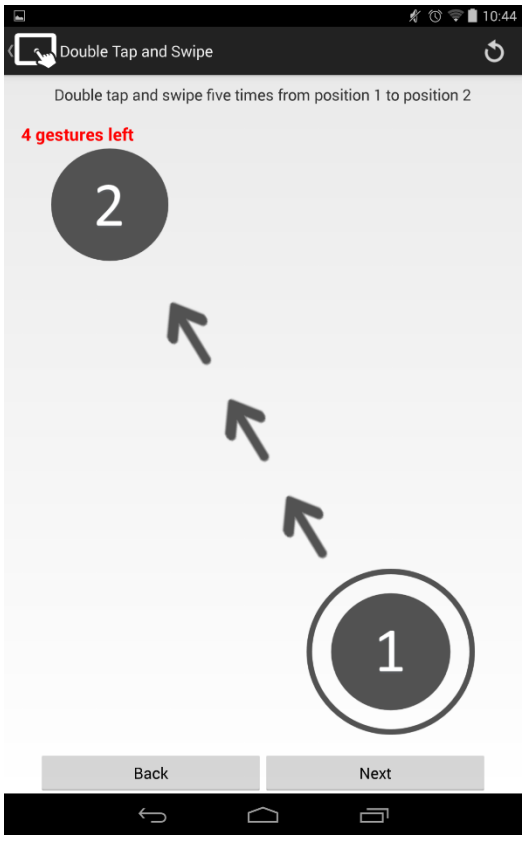

10:42
Swipe
Swipe five times from position 1 to position 2
4 gestures left
2
1
Back
Next

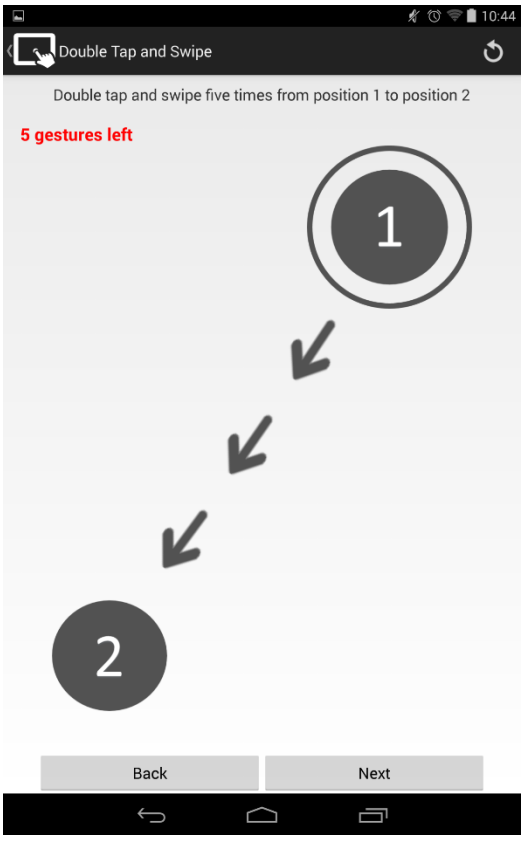

10:42
Swipe
Swipe five times from position 1 to position 2
5 gestures left
1
2
Back
Next

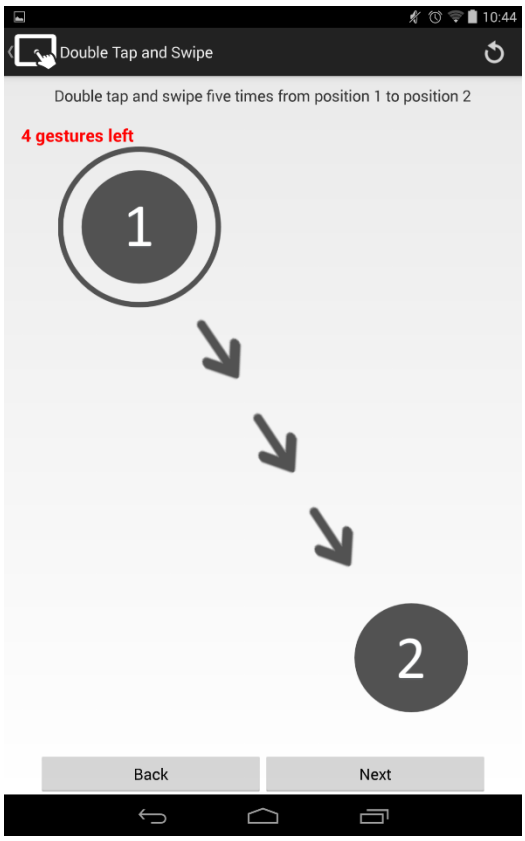

10:42
Swipe
Swipe five times from position 1 to position 2
5 gestures left
2
1
Back
Next

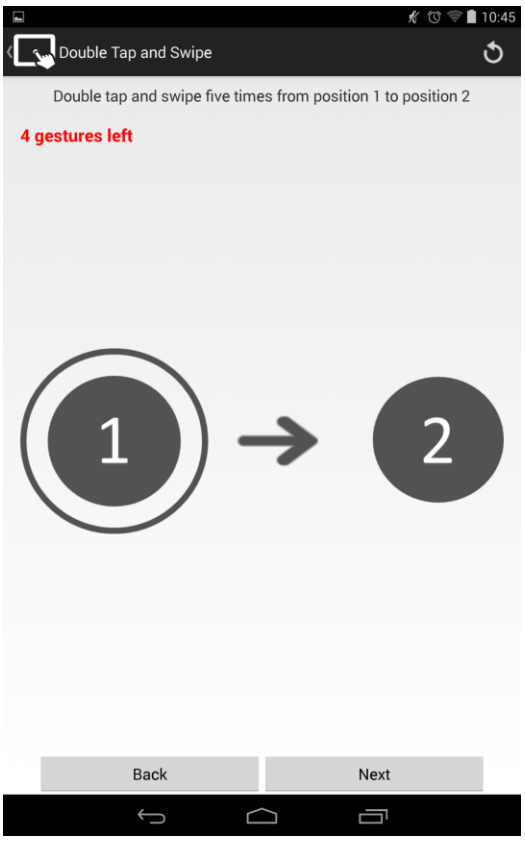

10:43
Double Taps
Double tap the five marked positions
5 gestures left
1
2
3
4
5
Back
Next

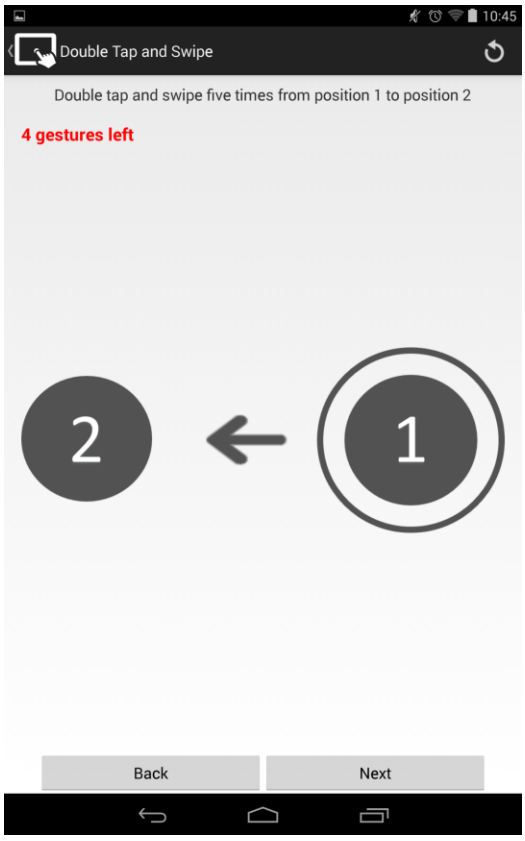

10:43
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
5 gestures left
2
1
Back
Next

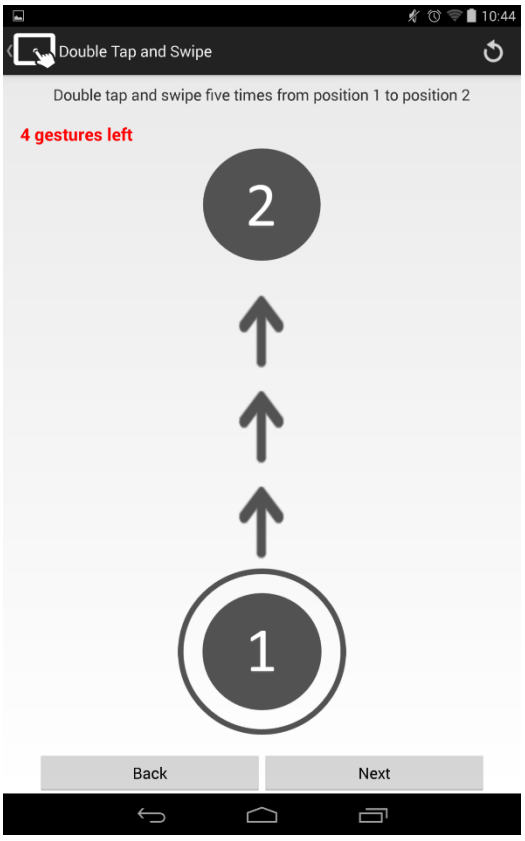

10:44
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
4 gestures left
2
1
Back
Next

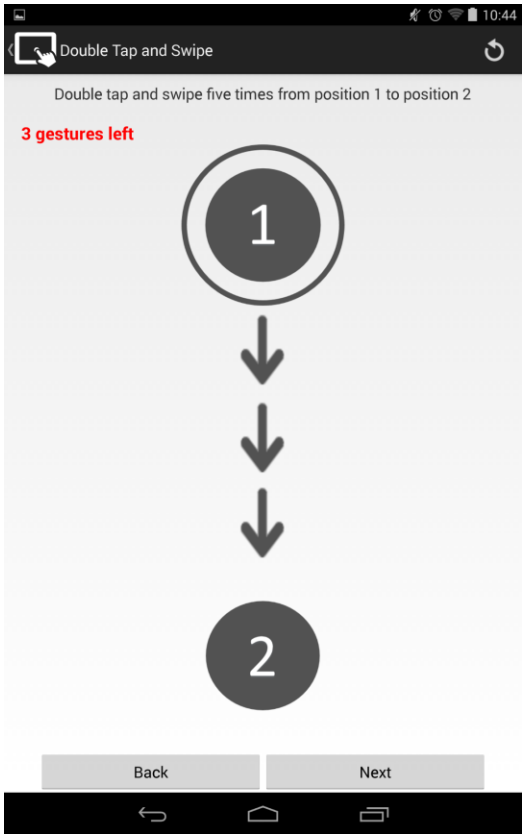

10:44
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
5 gestures left
1
2
Back
Next

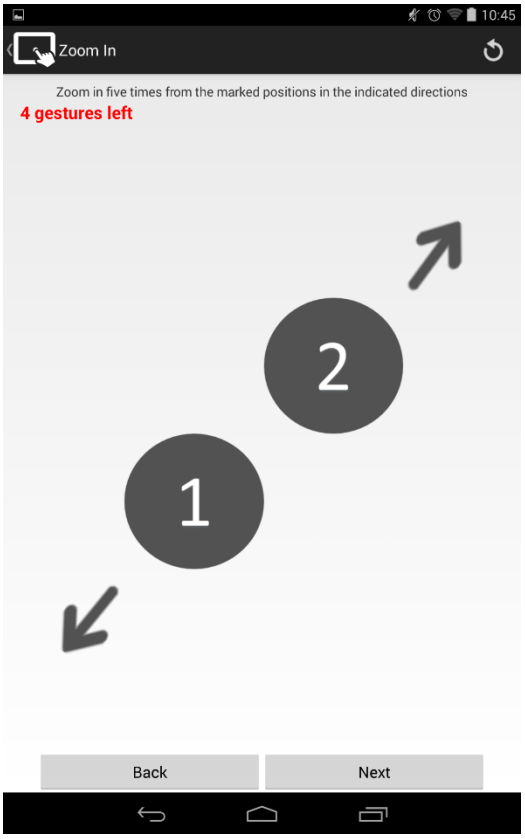

10:44
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
4 gestures left
1
2
Back
Next

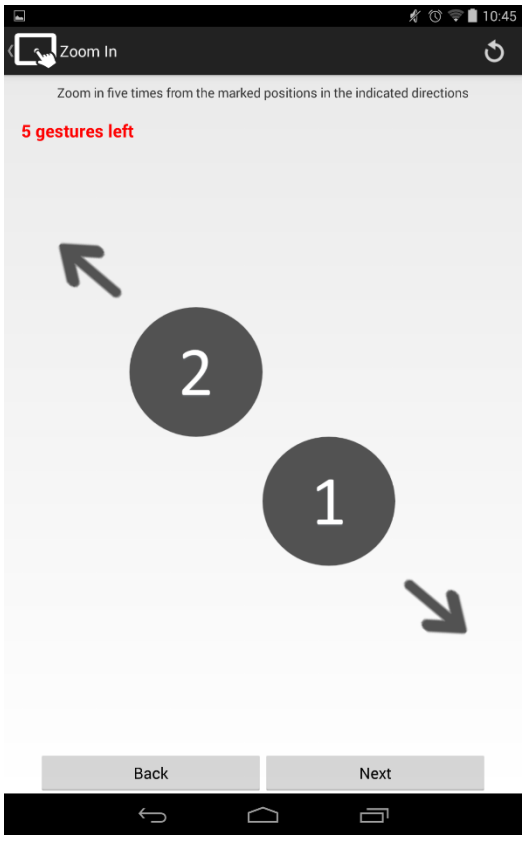

10:45
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
4 gestures left
1
2
Back
Next

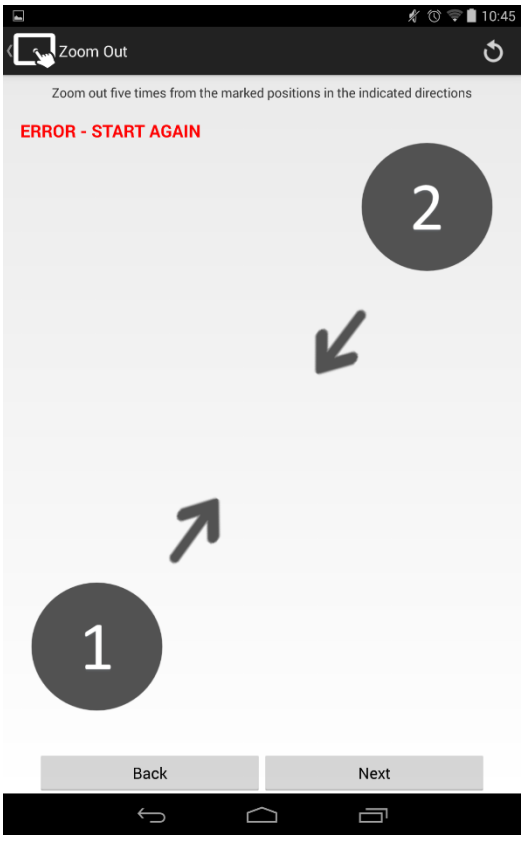

10:45
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
4 gestures left
2
1
Back
Next

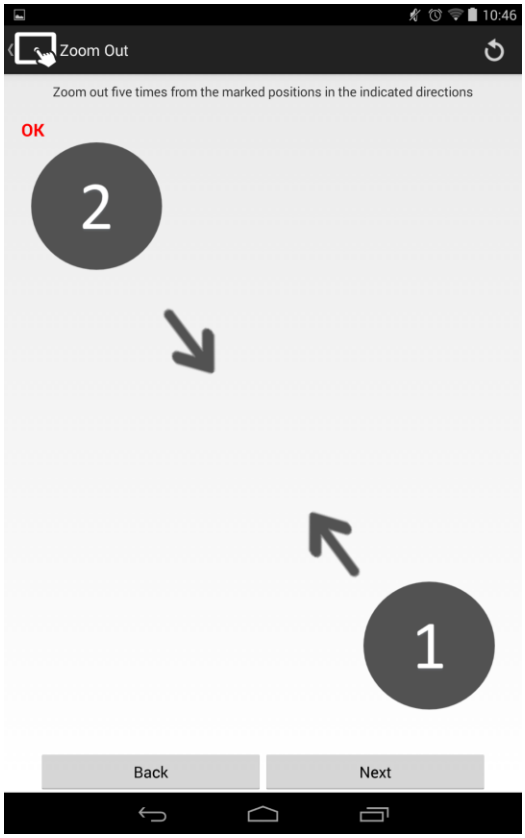

10:44
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
4 gestures left
2
1
Back
Next

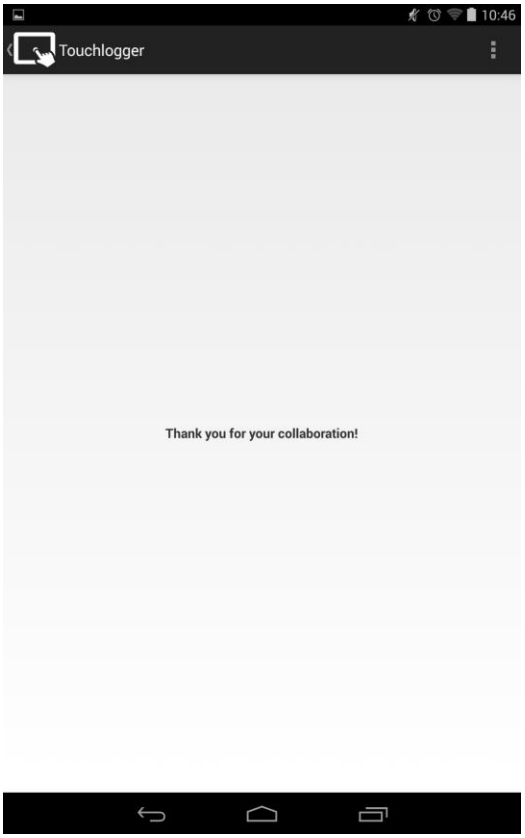

10:44
Double Tap and Swipe
Double tap and swipe five times from position 1 to position 2
3 gestures left
1
2
Back
Next


10:45
Zoom In
Zoom in five times from the marked positions in the indicated directions
4 gestures left
2
1
Back
Next


10:45
Zoom In
Zoom in five times from the marked positions in the indicated directions
5 gestures left
2
1
Back
Next


10:45
Zoom Out
Zoom out five times from the marked positions in the indicated directions
ERROR - START AGAIN
2
1
Back
Next


10:46
Zoom Out
Zoom out five times from the marked positions in the indicated directions
OK
2
1
Back
Next


10:46
Touchlogger
Thank you for your collaboration!

Appendix B

In this appendix are the tables of each participant, with all rates calculated. In the first column are the threshold values.

User 1	TMR	FMR	TNMR	FNMR	ACC
5	0,968	0,84	0,16	0,032	0,564
10	0,952	0,752	0,248	0,048	0,6
15	0,92	0,656	0,344	0,08	0,632
20	0,872	0,536	0,464	0,128	0,668
25	0,848	0,512	0,488	0,152	0,668
30	0,8	0,464	0,536	0,2	0,668
35	0,768	0,416	0,584	0,232	0,676
40	0,728	0,384	0,616	0,272	0,672
45	0,688	0,328	0,672	0,312	0,68
50	0,648	0,296	0,704	0,352	0,676
55	0,584	0,272	0,728	0,416	0,656
60	0,552	0,232	0,768	0,448	0,66
65	0,472	0,2	0,8	0,528	0,636
70	0,392	0,16	0,84	0,608	0,616
75	0,328	0,136	0,864	0,672	0,596
80	0,256	0,104	0,896	0,744	0,576
85	0,184	0,08	0,92	0,816	0,552
90	0,136	0,056	0,944	0,864	0,54
95	0,056	0,016	0,984	0,944	0,52
100	0	0	1	1	0,5

User 2	TMR	FMR	TNMR	FNMR	ACC
5	0,992	0,992	0,008	0,008	0,5
10	0,992	0,856	0,144	0,008	0,568
15	0,992	0,752	0,248	0,008	0,62
20	0,992	0,664	0,336	0,008	0,664
25	0,984	0,56	0,44	0,016	0,712
30	0,952	0,416	0,584	0,048	0,768
35	0,896	0,264	0,736	0,104	0,816
40	0,816	0,2	0,8	0,184	0,808
45	0,768	0,152	0,848	0,232	0,808
50	0,696	0,144	0,856	0,304	0,776
55	0,672	0,12	0,88	0,328	0,776
60	0,624	0,096	0,904	0,376	0,764
65	0,584	0,064	0,936	0,416	0,76
70	0,536	0,064	0,936	0,464	0,736
75	0,432	0,056	0,944	0,568	0,688
80	0,344	0,048	0,952	0,656	0,648
85	0,272	0,04	0,96	0,728	0,616
90	0,16	0,024	0,976	0,84	0,568
95	0,024	0,008	0,992	0,976	0,508
100	0	0	1	1	0,5

User 3	TMR	FMR	TNMR	FNMR	ACC
5	1	1	0	0	0,5
10	1	0,928	0,072	0	0,536
15	1	0,792	0,208	0	0,604
20	0,984	0,664	0,336	0,016	0,66
25	0,96	0,512	0,488	0,04	0,724
30	0,912	0,424	0,576	0,088	0,744
35	0,784	0,296	0,704	0,216	0,744
40	0,736	0,248	0,752	0,264	0,744
45	0,664	0,216	0,784	0,336	0,724
50	0,544	0,168	0,832	0,456	0,688
55	0,504	0,112	0,888	0,496	0,696
60	0,4	0,072	0,928	0,6	0,664
65	0,336	0,064	0,936	0,664	0,636
70	0,288	0,056	0,944	0,712	0,616
75	0,208	0,04	0,96	0,792	0,584
80	0,136	0,032	0,968	0,864	0,552
85	0,064	0,008	0,992	0,936	0,528
90	0,008	0	1	0,992	0,504
95	0,008	0	1	0,992	0,504
100	0	0	1	1	0,5

User 4	TMR	FMR	TNMR	FNMR	ACC
5	0,952	0,496	0,504	0,048	0,728
10	0,904	0,344	0,656	0,096	0,78
15	0,864	0,264	0,736	0,136	0,8
20	0,856	0,224	0,776	0,144	0,816
25	0,8	0,168	0,832	0,2	0,816
30	0,792	0,152	0,848	0,208	0,82
35	0,776	0,128	0,872	0,224	0,824
40	0,736	0,112	0,888	0,264	0,812
45	0,704	0,096	0,904	0,296	0,804
50	0,688	0,056	0,944	0,312	0,816
55	0,632	0,056	0,944	0,368	0,788
60	0,616	0,04	0,96	0,384	0,788
65	0,56	0,04	0,96	0,44	0,76
70	0,528	0,024	0,976	0,472	0,752
75	0,44	0,008	0,992	0,56	0,716
80	0,384	0	1	0,616	0,692
85	0,312	0	1	0,688	0,656
90	0,208	0	1	0,792	0,604
95	0,064	0	1	0,936	0,532
100	0	0	1	1	0,5

User 5	TMR	FMR	TNMR	FNMR	ACC
5	1	0,432	0,568	0	0,784
10	0,976	0,272	0,728	0,024	0,852
15	0,952	0,248	0,752	0,048	0,852
20	0,944	0,208	0,792	0,056	0,868
25	0,896	0,168	0,832	0,104	0,864
30	0,848	0,152	0,848	0,152	0,848
35	0,808	0,144	0,856	0,192	0,832
40	0,76	0,128	0,872	0,24	0,816
45	0,712	0,112	0,888	0,288	0,8
50	0,656	0,112	0,888	0,344	0,772
55	0,6	0,104	0,896	0,4	0,748
60	0,528	0,104	0,896	0,472	0,712
65	0,464	0,08	0,92	0,536	0,692
70	0,376	0,08	0,92	0,624	0,648
75	0,36	0,064	0,936	0,64	0,648
80	0,272	0,048	0,952	0,728	0,612
85	0,192	0,016	0,984	0,808	0,588
90	0,152	0,008	0,992	0,848	0,572
95	0,048	0	1	0,952	0,524
100	0	0	1	1	0,5

User 6	TMR	FMR	TNMR	FNMR	ACC
5	0,984	0,808	0,192	0,016	0,588
10	0,976	0,656	0,344	0,024	0,66
15	0,976	0,568	0,432	0,024	0,704
20	0,96	0,472	0,528	0,04	0,744
25	0,952	0,392	0,608	0,048	0,78
30	0,944	0,36	0,64	0,056	0,792
35	0,936	0,336	0,664	0,064	0,8
40	0,904	0,304	0,696	0,096	0,8
45	0,864	0,264	0,736	0,136	0,8
50	0,832	0,216	0,784	0,168	0,808
55	0,832	0,2	0,8	0,168	0,816
60	0,768	0,16	0,84	0,232	0,804
65	0,744	0,144	0,856	0,256	0,8
70	0,696	0,144	0,856	0,304	0,776
75	0,648	0,128	0,872	0,352	0,76
80	0,584	0,096	0,904	0,416	0,744
85	0,504	0,08	0,92	0,496	0,712
90	0,36	0,032	0,968	0,64	0,664
95	0,136	0,016	0,984	0,864	0,56
100	0	0	1	1	0,5

User 7	TMR	FMR	TNMR	FNMR	ACC
5	0,592	0,424	0,576	0,408	0,584
10	0,464	0,272	0,728	0,536	0,596
15	0,384	0,24	0,76	0,616	0,572
20	0,336	0,176	0,824	0,664	0,58
25	0,312	0,128	0,872	0,688	0,592
30	0,272	0,104	0,896	0,728	0,584
35	0,248	0,104	0,896	0,752	0,572
40	0,248	0,096	0,904	0,752	0,576
45	0,24	0,088	0,912	0,76	0,576
50	0,224	0,072	0,928	0,776	0,576
55	0,216	0,072	0,928	0,784	0,572
60	0,208	0,072	0,928	0,792	0,568
65	0,192	0,072	0,928	0,808	0,56
70	0,168	0,064	0,936	0,832	0,552
75	0,152	0,056	0,944	0,848	0,548
80	0,136	0	1	0,864	0,568
85	0,12	0	1	0,88	0,56
90	0,096	0	1	0,904	0,548
95	0,072	0	1	0,928	0,536
100	0	0	1	1	0,5

User 8	TMR	FMR	TNMR	FNMR	ACC
5	0,992	0,864	0,136	0,008	0,564
10	0,976	0,752	0,248	0,024	0,612
15	0,952	0,704	0,296	0,048	0,624
20	0,92	0,64	0,36	0,08	0,64
25	0,864	0,568	0,432	0,136	0,648
30	0,824	0,512	0,488	0,176	0,656
35	0,728	0,496	0,504	0,272	0,616
40	0,648	0,448	0,552	0,352	0,6
45	0,576	0,416	0,584	0,424	0,58
50	0,504	0,368	0,632	0,496	0,568
55	0,448	0,344	0,656	0,552	0,552
60	0,376	0,296	0,704	0,624	0,54
65	0,336	0,256	0,744	0,664	0,54
70	0,256	0,192	0,808	0,744	0,532
75	0,2	0,168	0,832	0,8	0,516
80	0,136	0,112	0,888	0,864	0,512
85	0,088	0,088	0,912	0,912	0,5
90	0,032	0,032	0,968	0,968	0,5
95	0	0,008	0,992	1	0,496
100	0	0	1	1	0,5

User 9	TMR	FMR	TNMR	FNMR	ACC
5	0,992	0,52	0,48	0,008	0,736
10	0,968	0,32	0,68	0,032	0,824
15	0,952	0,208	0,792	0,048	0,872
20	0,912	0,184	0,816	0,088	0,864
25	0,88	0,136	0,864	0,12	0,872
30	0,848	0,128	0,872	0,152	0,86
35	0,84	0,104	0,896	0,16	0,868
40	0,808	0,104	0,896	0,192	0,852
45	0,784	0,08	0,92	0,216	0,852
50	0,752	0,064	0,936	0,248	0,844
55	0,736	0,056	0,944	0,264	0,84
60	0,696	0,048	0,952	0,304	0,824
65	0,672	0,048	0,952	0,328	0,812
70	0,632	0,04	0,96	0,368	0,796
75	0,576	0,024	0,976	0,424	0,776
80	0,544	0,024	0,976	0,456	0,76
85	0,4	0,016	0,984	0,6	0,692
90	0,32	0,016	0,984	0,68	0,652
95	0,112	0	1	0,888	0,556
100	0	0	1	1	0,5

User 10	TMR	FMR	TNMR	FNMR	ACC
5	1	0,432	0,568	0	0,784
10	0,992	0,328	0,672	0,008	0,832
15	0,992	0,248	0,752	0,008	0,872
20	0,992	0,192	0,808	0,008	0,9
25	0,992	0,16	0,84	0,008	0,916
30	0,992	0,152	0,848	0,008	0,92
35	0,992	0,112	0,888	0,008	0,94
40	0,984	0,096	0,904	0,016	0,944
45	0,976	0,072	0,928	0,024	0,952
50	0,976	0,064	0,936	0,024	0,956
55	0,976	0,064	0,936	0,024	0,956
60	0,96	0,056	0,944	0,04	0,952
65	0,96	0,048	0,952	0,04	0,956
70	0,96	0,024	0,976	0,04	0,968
75	0,952	0,024	0,976	0,048	0,964
80	0,944	0	1	0,056	0,972
85	0,896	0	1	0,104	0,948
90	0,856	0	1	0,144	0,928
95	0,752	0	1	0,248	0,876
100	0	0	1	1	0,5