Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

DISTINGUISHING FREEHAND DRAWING RECOGNITION FOR BIOMETRIC AUTHENTICATION ON ANDROID-POWERED MOBILE DEVICES

Studente:
Alessandro Casasola

Relatore:
Prof. Carlo Ferrari

Anno Accademico 2011/2012

Padova, 15 Ottobre 2012

“Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza”

Dante Alighieri
La Divina Commedia
Canto XXVI - vv. 118-120

Abstract

The goal of this work is to develop a recognition algorithm for biometric authentication using a freehand drawing. There are similar studies about signature verification or keystroke typing[1] but there are not any work about freehand drawing surprisingly.

To achieve this goal we find, analyzed and selected 6 different biometric feature that are sufficient to distinguish two different people drawing the same sketch. First the enrollment and then the recognition algorithms are intentionally not difficult or complex to understand and implement.

The whole work has been developed for and tested on Android-powered mobile devices which has assured an easy-to-use interface and usage and a touchscreen technology ready-to-use.

In chapter 1 we introduce the complete work done and we give an overview about biometrics and the state of the art.

In chapter 2 we give some information about the Android platform and Android Development Tools.

In chapter 3 we analyze the selected features explaining the recognition algorithm in detail.

In chapter 4 we explain our choices, we present and comment the test results.

In appendix A we write a user manual to make usage easy.

Contents

List of Figures

List of Tables

Ad Andrea, Franca e Stefano
Ad Angelo, Antonio, Carmen e Pierina

Chapter 1

The biometrics

Contents

1.1 Main goal

The goal of this thesis is to combine the lastest technologies in the field of biometric authentication and mobile operating systems.

The user authentication based on biometric techniques is having an increasing success in last years because if used alone or as a support to other verification techniques provides a very high level of security. Similarly, the spread of smartphones with touchscreen technology is rising in the population all over the world.

Actually, there are many biometric authentication systems but none of them are about freehand drawing recognition (the closest to this field are handwritten signature and handwriting). For this reason, the challenges to be achieved in this work are totally new and represents the most intresting part in this thesis.

The followings are the goals of this project:

- to build a biometric authentication system that has to grant access to legitimate users and reject intruders;
- to create an algorithm that has to be secure, lightweight (using not much computational resources) at the same time;
- to be user-friendly;
- to prepare the system for the greatest number possible of users.

As discussed in the following chapters, you will see that a normal procedure for authentication based on username and/or password is in some ways worse than authentication based on a drawing:

- the time necessary to draw a sketch is lower than that required to enter a username and a password;
- the possibility to create a drawing totally customized and therefore not bound by characters;
- the extrapolation of features strictly personal and therefore hardly copyable from intruders.

1.2 A short introduction

If we look at biometrics definition on a dictionary we read: the process by which a person's unique physical and other traits are detected and recorded by an electronic device or system as a means of confirming identity[2].

Biometrics offers new perspectives in high-security applications while supporting natural, user-friendly and fast authentication. Biometric identification considers individual physiological characteristics and/or typical behavioral patterns of a person to validate their authenticity. Compared to established methods of person identification, employing PIN-codes, passwords, magnet - or smart cards, biometric characteristics offer the following advantages:

- uniqness: they are significant for each individual,
- universability: they are always available,
- permanence: they do not change over time,
- collectability: they are easily displayable to a sensor,
- they cannot be transferred to another person,
- they cannot be forgotten or stolen,
- they always vary[1].

Biometric systems comprise the following components: data acquisition and pre-processing, feature extraction and coding and computation of reference data and validation. The systems compare an actual recorded characteristic of a person with a pre-registered characteristic of the same or another person. Thereby, it has to be decided between *identification* (1 to many comparison) and *verification* (1 to 1 comparison). Then, the matching rate of the both characteristics is used to validate, whether the person is what he/she claim to be. The procedures seem to be equivalently to the traditional methods using PIN or ID-number. However, the main difference is founded by the fact that in biometrics an absolute statement identical/not identical cannot be given. For instance a credit card has exact that number "1234 5678 9101" or not, contrary, a biometric feature varies naturally at any acquisition.

Biometric technologies will be divided into approaches utilizing physiological characteristics, also referred as passive features, and approaches using behavioral characteristics that are active features. Behavioral characteristics, used e.g. in speaker recognition, signature verification or key-stroke analysis are always variable. On the other hand physiological characteristics employed e.g. in hand, fingerprint, face, retina or iris recognition, are more or less stabile. Variations may be caused by injuries, illness and aging, as well as variations during acquisition.

[1]The presentation of two 100% identical feature sets indicates fraud.

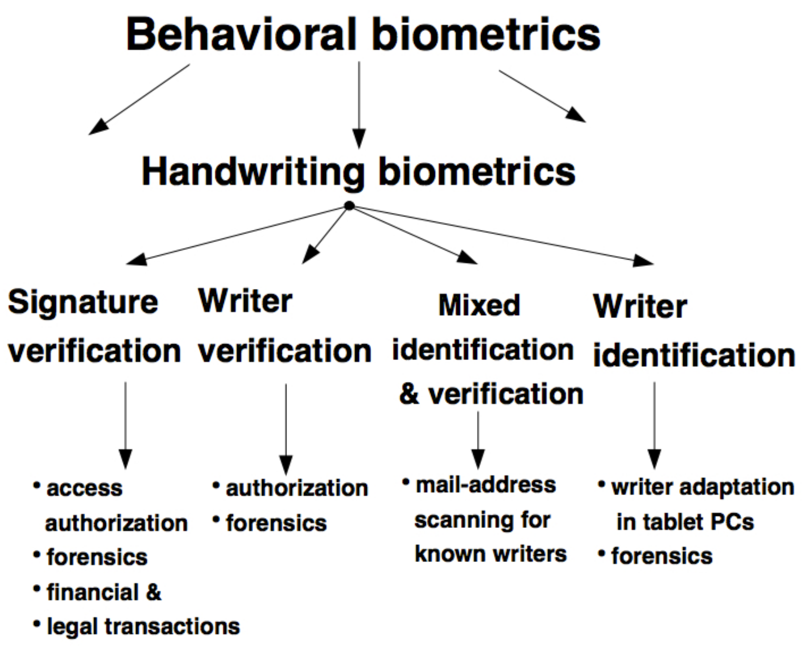


FIGURE 1.1: Branching in handwriting analysis

Biometric approaches have to solve the two-class problem person accepted or person rejected. So, the performance of biometric systems is measured with two basic rates: False Acceptation Rate (FAR) is the probability that the system incorrectly matches the input pattern to a non-matching template in the database; False Rejection Rate (FRR) is the probability that the system fails to detect a match between the input pattern and a matching template in the database[3]. Moreover the followings are used as performance tools for biometric systems: Receiver Operating Characteristic (ROC) plot is a visual characterization of the trade-off between the FAR and the FRR; Equal Error Rate (EER) is the rate at which both accept and reject errors are equal. The value of the EER can be easily obtained from the ROC curve[2].

1.3 About soft-biometrics

A possible solution to the problem of designing a reliable and user-friendly biometric system is to use ancillary information about the user like height, weight, age, gender, ethnicity, and eye color to improve the performance of the primary biometric system. Most practical biometric systems collect such information about the users during enrollment. However, this information is not currently utilized during the automatic identification or

[2]In general, the device with the lowest EER is most accurate.

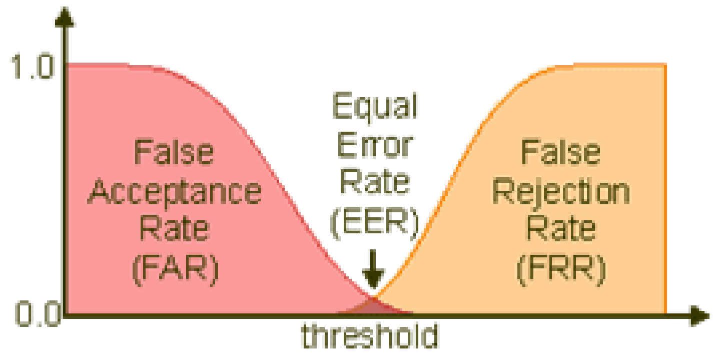


FIGURE 1.2: ROC curve

verification phase. Only when a genuine user is falsely rejected by the system, a human operator steps in to verify the soft biometric traits of the user. If these characteristics can be automatically extracted and utilized during the decision making process, the overall performance of the system will improve and the need for manual intervention will be reduced. The ancillary information by itself is not sufficient to establish the identity of a person because these traits are indistinctive, unreliable, and can be easily spoofed. Hence, we define *soft biometric traits as characteristics that provide some information about the individual, but lack the distinctiveness and permanence to sufficiently differentiate any two individuals*[4].

Traits which accept the above definition include, but are not limited to:

- Physical: skin colour, eye colour, hair colour, presence of beard, presence of moustache, height, weight.
- Behavioural: gait, keystroke.
- Adhered human characteristics: clothes colour, tattoos, accessories.

Soft Biometrics inherit a main part of the advantages of biometrics and furthermore endorses by its own assets. Some of the advantages include non obtrusiveness, the computational and time efficiency and human compliance. Furthermore they do not require enrolment, nor the consent or the cooperation of the observed subject.

1.4 Aspects in user authentication

The login-password authentication is the most usual mechanism used to grant access because it is low-cost, besides its familiarity to a lot of users. However, this authentication is fragile when there is a careless user and/or a weak password. The purpose of this thesis is to improve or substitute the login-password authentication using biometric characteristics. Biometric characteristics are unique to each person and have advantages as they could not be stolen, lost, or forgotten.

The biometric technology employed in this thesis is different but close to the typing biometrics, also known as keystroke dynamics. Typing biometrics is a process that analyzes the way a user types at a terminal by monitoring the key board inputs in attempt to identify users based on their habitual typing rhythm patterns. The typing biometrics authentication can be classified as static or continuous. The static approach analyzes inputs just in a particular moment, and the continuous one analyzes inputs during all user's session. The methodology of this thesis is unintrusive (using a sketch) and pervasive (almost everyone has a smartphone).

The most important aspects are the followings.

1.4.1 Target string

It is the input string that will be typed by the user and monitored by the system. In some works, four strings (login, password, firstname and lastname) were used as targets. In others, the password itself was employed. String length is a very important issue, considering that it was stated that misclassification increases as the string length drops to fewer than ten characters. However, in this work there isn't a unique traget string because a freehand drawing is used as input.

1.4.2 Number of samples

Samples are collected during the enrollment process to compound the training set. Its number varies a lot, since it was as few as three samples per user in some works and as many as 30 samples per user in others. It was concluded that fewer than six samples is not recommended to obtain good performance.

1.4.3 Features

Two of the most used features are duration of the key, that is the time interval that a key remains pressed and keystroke latency, that is the time interval between successive keystrokes. Sometimes the combination of these features resulted in better results than using them in isolation.

1.4.4 Timing accuracy

As most of the typing biometrics features are time-based, the precision of the key-up and the key-down times have to be analyzed. The timing accuracy varies between 0.1 ms and 1000 ms.

1.4.5 Trials of authentication

It was observed that legitimate users usually fail in the first trial of authentication, but in the second one, a successful authentication was realized. Each user must type his target string two times using a shuffling technique.

1.4.6 Adaptation mechanism

Biometric characteristics change over time. An adaptation mechanism or a re-enrollment could be performed to maintain the templates updated. Most of the researchers did not mention this issue, but an adaptation mechanism was used. This mechanism creates a new updated template every time a successful authentication is performed, including the new sample and discarding the oldest one.

1.4.7 Classifier

In some works a statistical classifier was applied, using known techniques as k-means, Bayes, fuzzy logic, neural networks[5].

1.5 Related work

There are many works on biometric authentication and the closest to freehand drawing recognition is handwritten signature.

1.5.1 Signature verification

In paper [3] there is a comparison between signature verification and other biometric technologies. Signature verification has several advantages as an identity verification mechanism:

- signing is the most traditional and social accepted method of identification.
- signature analysis can only be applied when the person is/was conscious and disposed to write in the usual manner, although they may be under extortion to provide the handwriting. To give a counter example, a persons fingerprint may also be used when the person is in an unconscious (e.g. drugged) state of mind.
- forging a signature is deemed to be more difficult than a fingerprint given the availability of sophisticated analyses.

There are two fundamental approaches for signature validation, which are determined by the manner of signature digitalization. The first, dynamic signature verification, uses a special electronic device (electronic pen and/or an electronic pad) for the acquisition during the signing process. Pen movements are stored as time signals primarily representing the pen displacement in the x,y-writing plane. A person must be present and must explicitly use this special devise for applying signature analysis. For static signature verification, the second approach, the final handwritten signature pattern is handled as a graphical image and used for the analysis. So, static signature verification is able to process signatures written on paper documents as well as signature images generated by the time series of electronic devices. Static signature validation can be applied in a persons absence. Static signature verification is a great challenge among biometrics. Due to the involvement of various writing utensils such as diverse kinds of pen/ink, kinds of paper and kinds of physical texture of the writing pad, static handwriting specimens underlie great variations. The challenge of static signature verification is to employ sophisticated methods of digital image processing and pattern recognition to compensate influences by writing utensils and to validate writer specific characteristics objectively and robustly.

1.5.2 Texture

Angles and curvature in handwriting are to a large extent determined by the degrees of freedom in wrist and finger movement, which in turn depend on the pen grip attitude and the applied grip forces. A stiff hand-pen system will allow for less variation in directions than a compliant agile pen grip. In order to capture the curvature of the ink trace, which is very typical for different writers, it is informative to observe the local angles along the edges. In its simplest form, the histogram of angles along the edges of a handwritten image may be considered. However, while such a feature will capture the variations around the average slant, it will miss the curvature information. Therefore, a related feature with some additional complexity was proposed. The central idea is to consider the two edge fragments emerging from a central pixel and, subsequently, compute the joint probability distribution of the orientations of the two fragments of this "hinge"[6].

1.5.3 Allograph

In the application domain of forensic handwriting analysis, there is a strong focus on allography, i.e., the phenomenon that for each letter of the alphabet, shape variants or allographs exist. Individuals display a preference to produce a particular subset of all the allographs emanating from the population of writers, due to schooling and personal preferences. Figure 1.3 illustrates that allographic style does not concern isolated characters, per se. It is very common that writers emit families of comparable shapes for components of handwriting such as the ascenders and descenders in the given example. Not all writers are equally consistent in their use of allographs, but given a few lines of text there may be enough information to estimate occurrences of shapes. Since there exists no exhaustive and world-wide accepted list of allographs in, e.g., Western handwriting, a codebook must be computed from reference data, which sufficiently captures allographic information. A samples of handwriting can then be described by the histogram of the usage of code-book elements. It was demonstrated that the use of the shape of connected components of upper-case Western handwriting as the basis for codebook construction can yield high writer-identification performance[6].

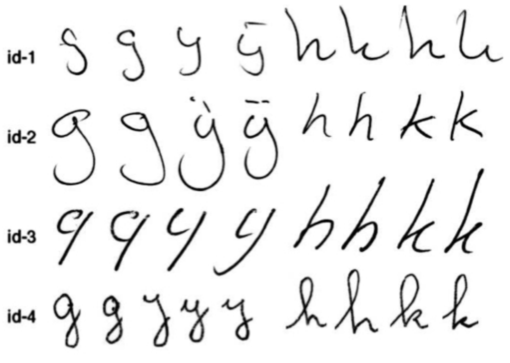


FIGURE 1.3: The letters *g, y, h, k* by four subjects (id1-id4)

Chapter 2

Android operating system

Contents

2.1 A short introduction

Android[1] is the world's most popular mobile platform. With Android you can use all the Google apps you know and love, plus there are more than 600,000 apps and games available on Google Play to keep you entertained, alongside millions of songs and books, and thousands of movies. Android devices are already smart, and will only get smarter, with new features you won't find on any other platform, letting you focus on what's important and putting you in control of your mobile experience. Millions of people already use Android because it makes your mobile device so much more powerful and useful. And because Android is open, you can create a unique mobile experience that's just right for you.

Now, it will be introduced the Gestures pattern which is the most important in this thesis.

[1]Android is a trademark of Google Inc.

2.2 Gestures pattern

Gestures allow users to interact with your app by manipulating the screen objects you provide. The following table shows the core gesture set that is supported in Android.

- **Touch**: triggers the default functionality for a given item;
- **Long press**: enters data selection mode. Allows you to select one or more items in a view and act upon the data using a contextual action bar;
- **Swipe**: scrolls overflowing content, or navigates between views in the same hierarchy;

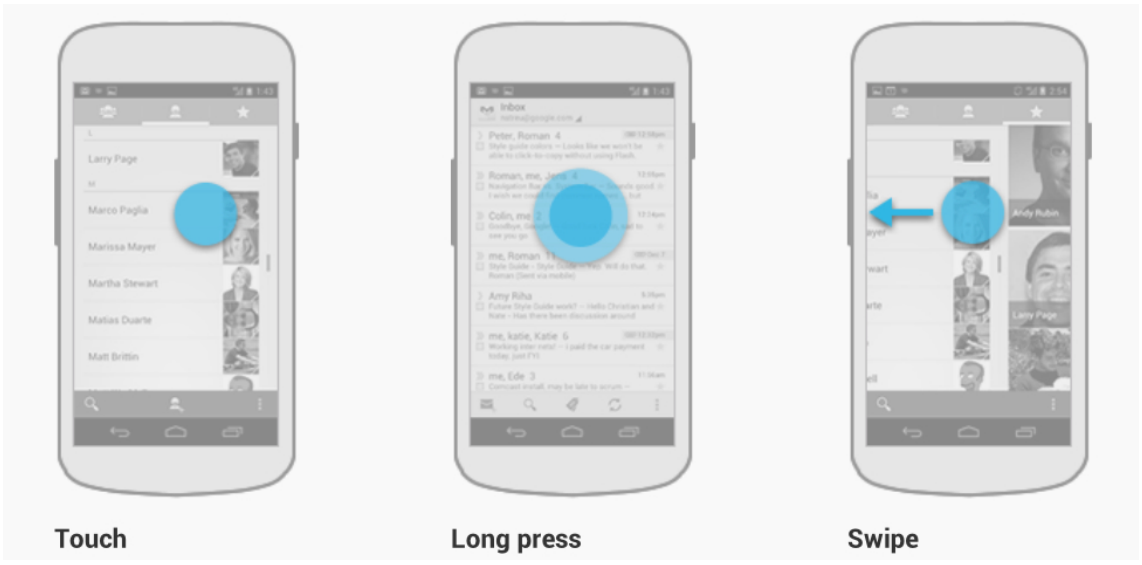


FIGURE 2.1: Touch, long press and swipe demo

- **Drag**: rearranges data within a view, or moves data into a container (e.g. folders on Home Screen);
- **Double touch**: zooms into content. Also used as a secondary gesture for text selection.
- **Pinch open**: zooms into content.
- **Pinch close**: zooms out of content.

2.3 Gestures on Android

Touch screens are a great way to interact with applications on mobile devices. With a touch screen, users can easily tap, drag, fling, or slide to quickly perform actions in their

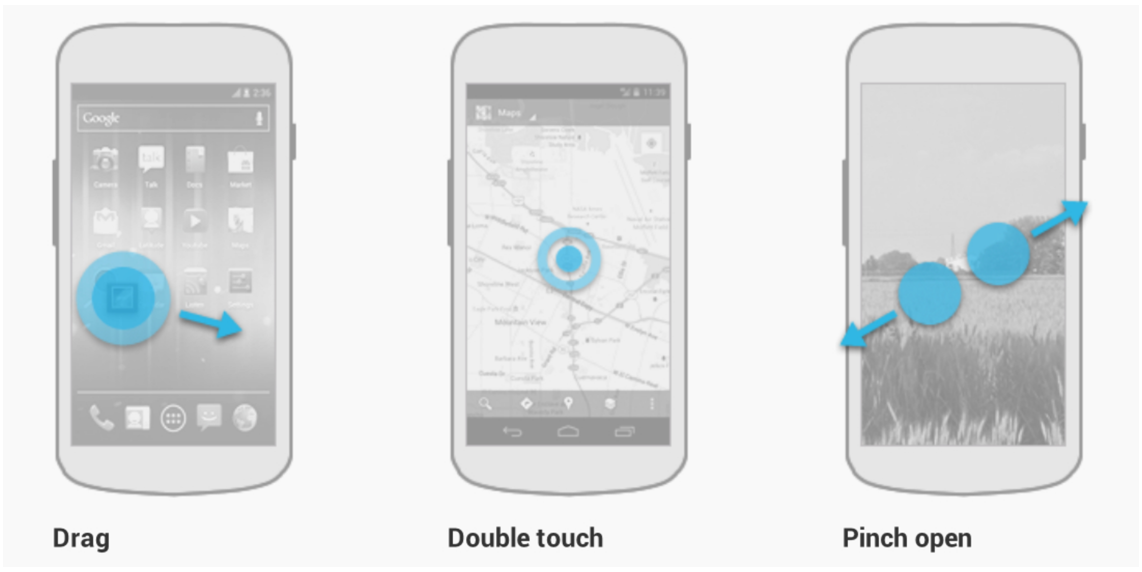


FIGURE 2.2: Drag, Double touch and pinch open demo

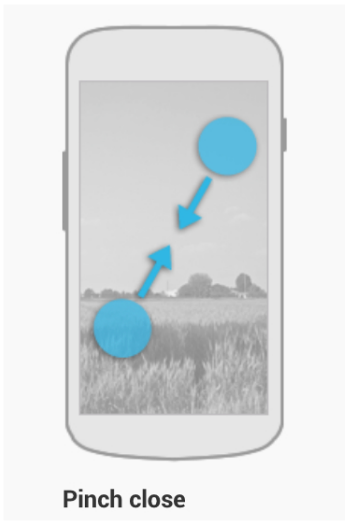


FIGURE 2.3: Pinch close demo

favorite applications. But it's not always that easy for developers. With Android, it's easy to recognize simple actions, like a swipe, but it's much more difficult to handle complicated gestures, which also require developers to write a lot of code. That's why we have decided to introduce a new gestures API in Android 1.6. This API, located in the new package `android.gesture`, lets you store, load, draw and recognize gestures.

2.3.1 Creating a gestures library

The Android 1.6 SDK comes with a new application pre-installed on the emulator, called Gestures Builder. You can use this application to create a set of pre-defined gestures for your own application. It also serves as an example of how to let the user define his own gestures in your applications. You can find the source code of Gestures Builders in the samples directory of Android 1.6. In our example we will use Gestures Builder to generate a set of gestures for us (make sure to create an AVD with an SD card image to use Gestures Builder.) The figure 2.4 shows what the application looks like after adding a few gestures.

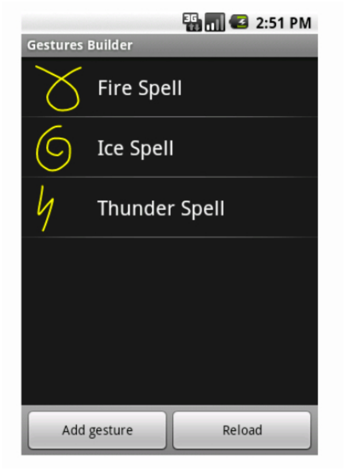


FIGURE 2.4: Gesture Builder screenshot

As you can see, a gesture is always associated with a name. That name is very important because it identifies each gesture within your application. The names do not have to be unique. Actually it can be very useful to have several gestures with the same name to increase the precision of the recognition. Every time you add or edit a gesture in the Gestures Builder, a file is generated on the emulator's SD card, `/sdcard/gestures`. This file contains the description of all the gestures, and you will need to package it inside your application inside the resources directory, in `/res/raw`.

2.3.2 Loading the gestures library

Now that you have a set of pre-defined gestures, you must load it inside your application. This can be achieved in several ways but the easiest is to use the `GestureLibraries` class. You can easily load libraries from files or other sources, like the SD card, which is very important if you want your application to be able to save the library; a library loaded from a raw resource is read-only and cannot be modified. The diagram 2.5 shows the structure of a library.

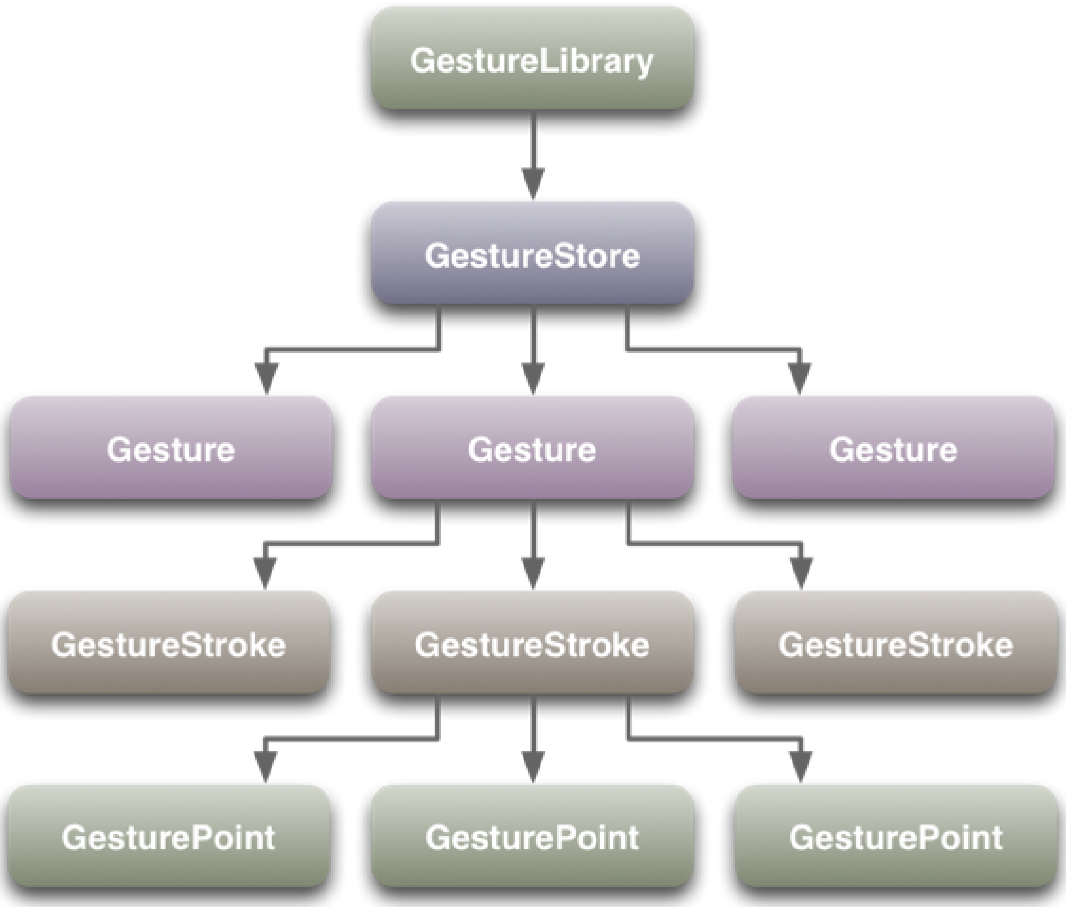


FIGURE 2.5: Gesture Library structure

2.3.3 Recognizing gestures

To start recognizing gestures in your application, all you have to do is add a `GestureOverlayView` to your XML layout. Notice that the `GestureOverlayView` is not part of the usual android.widget package. Therefore, you must use its fully qualified name. A gesture overlay acts as a simple drawing board on which the user can draw his gestures. You can tweak several visual properties, like the color and the width of the stroke used to draw gestures, and register various listeners to follow what the user is doing.

When the listener fires, you can ask the `GestureLibrary` to try to recognize the gesture. In return, you will get a list of `Prediction` instances, each with a name - the same name you entered in the `Gestures Builder` - and a score. The list is sorted by descending scores; the higher the score, the more likely the associated gesture is the one the user intended to draw.

2.4 Developer references

The `android.gesture` package is avaible since API level 4 and provides classes to create, recognize, load and save gestures.

The following list shows the available classes and their function:

- **Gesture**: a gesture is a hand-drawn shape on a touch screen;
- **GestureLibraries** ;
- **GestureLibrary**;
- **GestureOverlayView**: a transparent overlay for gesture input that can be placed on top of other widgets or contain other widgets;
- **GesturePoint**: a timed point of a gesture stroke;
- **GestureStore**: `GestureLibrary` maintains gesture examples and makes predictions on a new gesture;
- **GestureStroke**: a gesture stroke started on a touch down and ended on a touch up;
- **GestureUtils**: utility functions for gesture processing and analysis, including methods for:
  - feature extraction (e.g., samplers and those for calculating bounding boxes and gesture path lengths);
  - geometric transformation (e.g., translation, rotation and scaling);
  - gesture similarity comparison (e.g., calculating Euclidean or Cosine distances between two gestures);
- **OrientedBoundingBox**: an oriented bounding box;
- **Prediction**.

2.5 Eclipse IDE with Android Development Tools

Android is an operating system currenttl primarily developed by Google based on Linux with a Java programming interface. The Android Software Development Kit (Android

SDK) provides all necessary tools to develop Android applications. This includes a compiler, debugger and a device emulator, as well as its own virtual machine to run Android programs.

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give a powerful, integrated environment in which to build Android applications.

ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) *.apk* files in order to distribute your application.

Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives you an incredible boost in developing Android applications. Note that before install or use ADT, you must have compatible versions of both the Eclipse IDE and the Android SDK installed. For details, make sure to read Installing the Eclipse Plugin.

Chapter 3

Strategic planning

Contents

3.1 A short introduction

The main idea behind this project is to autenthicate people using non invasive informations in a secure way. The reader can say that a freehand drawing could not be safe because eveyone can copy another person sketch. The surprising thing is that the main important aspect is not the sketch, but the way you draw it. In order to this sentence there are $4!2^4 = 384$ different ways to wrtite the capital "E" letter because it is possible to type the 4 strokes from left (or top) to right (or bottom) and viceversa (this explains 2^4) and the strokes order could be choosen between 4! options. Actually, just 4 or 5 of this options are currently used but it is not hard to imagine that there is a wide variability even with this easy example[7].

3.2 The features

The features are the project parameters which allows the system to accept or reject an input sketch. For each feature, it will be explained why it was choosen and the type of data used.

1. Strokes number: the first feature represents the strokes count for each gesture. It has an important role in the recognizing preocedure because in necessary to match gestures with the same number of strokes (each gesture can be made by a variable strokes number, but at least one). This value is obtained by counting how many strokes are saved for every gesture.

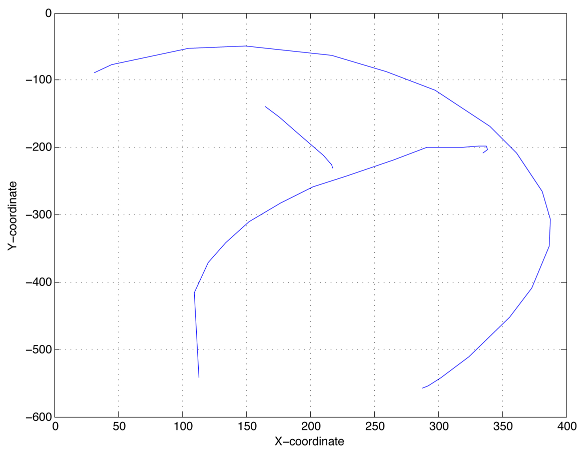


FIGURE 3.1: A 3-stroke drawing example (the y-axis is reversed because the axis origin is the top-left vertex)

2. X-coordinate: this parameter is a sequence of numbers which stores the finger position on X-axis on the touchscreen while gesturing. This is the most important feature because the way the sketch is made distinguishes between users as discussed in the previous section. The coordinates are saved in a float array with a saple period of 23/24 ms.

3. Y-coordinate: this feature is equal as the previous but it refers to the finger position on Y-axis on the touchscreen while gesturing.

4. Time interval: this characteristic represents the time needed to draw the sketch. If the gesture is made only by on stroke the time interval is the difference between the last time value and the first one. In event of a multistroke gesture this feature is measured from the first XY point of the first stroke is typed to the last point of the last stroke is entered. Note that the time between two consecutive strokes must be

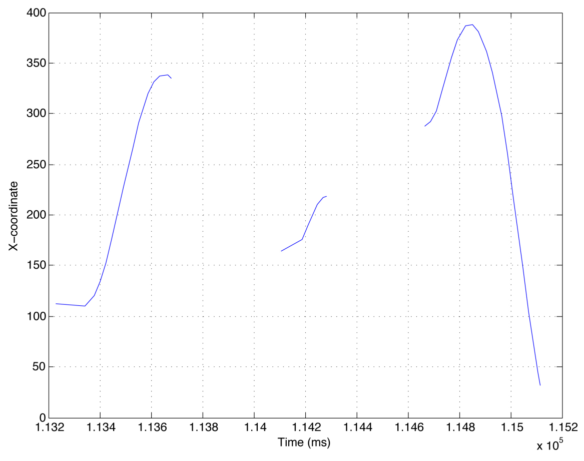


FIGURE 3.2: X-coordinate points

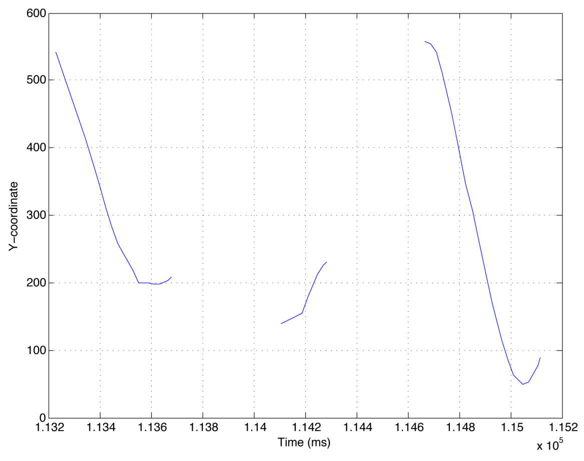


FIGURE 3.3: Y-coordinate points

very small and therefore insignificant due to a design choice. The time interval value is stored as a float variable.

5. Pressure samples: this parameter, like X and Y coordinates, keeps track of the finger pressure on the touchscreen. This value consists of a number between zero[1] (no pressure) to one (maximum pressure).

6. Gesture length: this feature returns the gesture global lenght. This value is the sum of each stroke length and it is important beacause it is used in the input validation phase.

[1]Since the system can store the pressure value only when the finger is moving on the touchscreen it is impossible to get value 0.

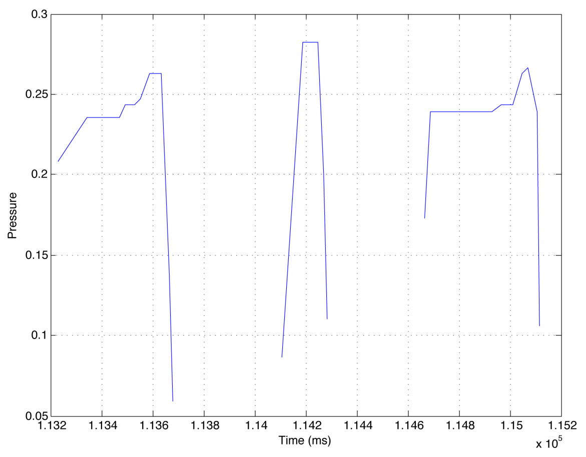


FIGURE 3.4: Pressure values

3.3 The algorithm

3.3.1 Overview

The whole algorithm is not difficult to understand however is made of several complex parts. In this short overview we will explain how they interact themselves while giving an examination in depth in the following subsections.

The enrollment procedure starts displaying a form where the user has to insert his/her name and surname. This step is important because the system stores name and surname in a unique *String* named *personal data* and will display the content of this variable when the corresponding sketch will be drawn in the authentication phase.

Now, the user must draw his/her own sketch on the screen remembering that the same drawing has to be inserted at least 6 times. Once the first gesture is confirmed by pressing the *Done* button, the system validates the figure as described in 3.3.2. If this control has a positive result, the application computes the features and creates the template for the new user.

The algorithm continues asking 5 more drawings and when the user inserts the sixth sketch the system decides if the quality of the samples inserted is good enough or if additional drawings are required to complete the enrollment process. The control is made in this way: the system tries to authenticate the user with the last drawing inserted and looks at the authentication response. If the *personal data* obtained is equal to the user's *personal data* and the minimum threshold is exceeded then the procedure ends inserting the last sample in the user's template and saving the lower layer data (see section 3.3.3) in memory. Otherwise the application asks you five more drawings and repeats this procedure when you insert the last one. If the two controls are successful the procedure ends, otherwise you must enter five more samples and so on.

The authentication procedure is not treated in this overview because is sufficiently described in subsection 3.3.6.

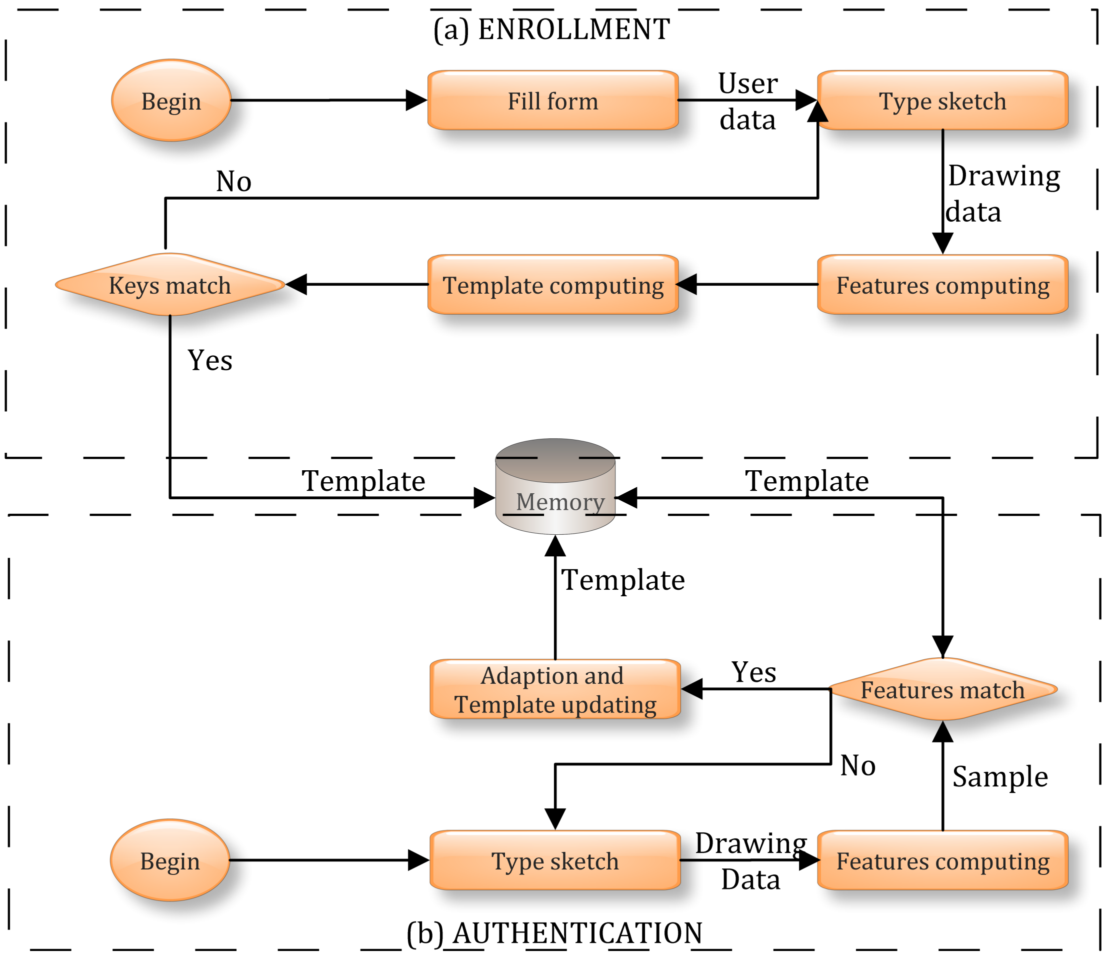


Figure 3.5: Flowchart of the methodology: (a) Enrollment (b) Authentication

3.3.2 Input validation

The flowchart in figure 3.5 shows the main steps in user authentication and enrollment. Each time a new enrollment procedure starts it will be displayed a form where the user must insert his/her personal data. This step has the function to insert all the information linked to the user once for all.

After this short preamble, the user has to draw a sketch which can be made of one or multiple strokes. There isn't an upper bound on the strokes number, however a gesture with more than 10 strokes is difficult to draw and to reproduce several times.

The validation phase is the first check procedure in this work and its importance is due to ensure a greater security. Every sketch has to overcome two tests:

- Length: if the drawing total length is lower than length threshold (1200 points) the enrollment procedure will refuse the gesture and signals an error message. This first check is necessary to avoid too shoort drawings which can be easily exploitable by intruders.
- Duplicate: if a user has already choosen the same gesture a generic error message appears to the user asserting that the gesture is not valid. This control avoids ambiguity in the authentication procedure because two people could not use the same gesture and using a generic error message, the user who reads the error message, doesn't kwon this information and doesn't authenticate in the wrong manner.

3.3.3 Data format: lower layer

A significant aspect in this project is data management. Working with data which is difficult to manipulate or no immediately available will make complicate work and a worst source code. The approach followed in this thesis is a two layer approach: the first named lower and the second called higher.

The lower layer contains the raw data: personal data and all gestures drawn are stored for every user. At this point we refer at all gestures drawn as all the gestures inserted by the user, each of them made of one or more strokes. Every gesture stroke is made of XY-points, pressure and timestamp values not yet standardized.

3.3.4 Input sampling

The next step in the enrollment procedure is drawing sampling. This phase is crucial for two reasons:

- Standardization: for each stroke drawn in the enrollment procedure, the system returns a list of XY-coordinates, pressure values and a timestamp with a sample period of 23 ms. If the drawings were performed with the same speed[2] a short sketch will have less samples than a long sketch. For this reason a standardization method is provided to work with a constant points number: 25 points per X-coordinate, Y-coordinate and pressure value.

[2]This assumption is generally true.

User #1	Gesture #1	Stroke #1	XY-points	$[x_1, y_1, x_2, y_2, ..., x_n, y_n]$
			Pressure values	$[p_1, p_2, ..., p_n]$
			Timestamps	$[t_1, t_2, ..., t_n]$
		Stroke #2	XY-points	$[x_1, y_1, x_2, y_2, ..., x_n, y_n]$
			Pressure values	$[p_1, p_2, ..., p_n]$
			Timestamps	$[t_1, t_2, ..., t_n]$
	Gesture #2	Stroke #1	XY-points	$[x_1, y_1, x_2, y_2, ..., x_n, y_n]$
			Pressure values	$[p_1, p_2, ..., p_n]$
			Timestamps	$[t_1, t_2, ..., t_n]$
		Stroke #2	XY-points	$[x_1, y_1, x_2, y_2, ..., x_n, y_n]$
			Pressure values	$[p_1, p_2, ..., p_n]$
			Timestamps	$[t_1, t_2, ..., t_n]$
	Gesture #3	Stroke #1	XY-points	$[x_1, y_1, x_2, y_2, ..., x_n, y_n]$
			Pressure values	$[p_1, p_2, ..., p_n]$
			Timestamps	$[t_1, t_2, ..., t_n]$
		Stroke #2	XY-points	$[x_1, y_1, x_2, y_2, ..., x_n, y_n]$
			Pressure values	$[p_1, p_2, ..., p_n]$
			Timestamps	$[t_1, t_2, ..., t_n]$

TABLE 3.1: Lower layer data format for user #1 having 3 Gestures made of 2 strokes each

- Matching: having input samples with the same points number allows to compare each gesture to all others. The match is not among the whole gesture, but it is made between the gestures strokes because they are made of arrays with the same length.

3.3.5 Data format: higher layer

Before introducing the second layer we focus the attention how to produce the higher lever data. The problem observed in this phase is how to obtain, for each user profile, a set of feature values which have to represent all the enrollment drawings and have to be easily and quickly comparable with a authentication sketch. The answer to this question is a template built on data stored at first level which is elected as representative of all the input drawings.

The procedure to compute the second layer data for XY-coordinates and pressure values is based on two statistical concepts: weighted average and variance. Weighted mean refers to a unique value which describes a data set where, instead of arithmetic mean where each data point contributes equally to the final average, some data points contribute more than others. Why was weighted average choose instead of the arithmetic one? Because more weight is assigned to recent drawings because they are more recent and therefore closer to the user's drawing idea than the oldest (This will be explained in subsection 3.3.7). Lastly, the variance is a measure of how far a set of numbers is spread out.

The algorithm to compute the second layer template runs for every user profile. It starts with an *HashMap* data type representing data at lower layer which is made of two fields: a string which contains user's personal data (name and surname e.g.) and an *ArrayList* of *Gest* type which can contain at most 60 gestures.

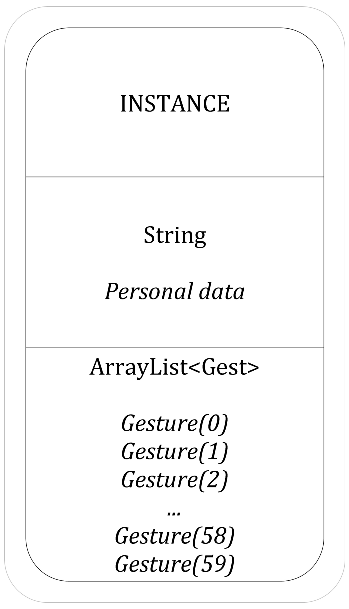


FIGURE 3.6: Instance of HashMap data type

This arraylist follows a FIFO (First In First Out) policy in order to save each new gesture and discard the oldest one. Now, for each gesture, the XY-coordinates array and the pressure values array are sampled according to input sampling rules defined in subsection 3.3.4. Then the template creation can start calculating XY-average array and XY-variance array using the following weights:

Gesture number	Weight
0 to 11	1
12 to 23	2
24 to 35	3
36 to 47	4
48 to 59	5

TABLE 3.2: Weights used in weighted average calculation

where the 0^{th} gesture is the oldest (or first inserted) and the 59^{th} gesture is the latest drawn (or last inserted). The following formula describes how to compute the weighted average for the $j-th$ point in the $i-th$ gesture. Notice that n refers to the number of gesture inserted by the user: this number is zero when the user creates his own profile and increases during the enrollment procedure.

$$\bar{x}_j = \frac{\sum_{i=1}^{n} w_{ij} x_i}{\sum_{i=1}^{n} w_i}$$

Once the average value is available the method continues with variance calculation to measure of how far the set of numbers is spread out. For each data point we first calculate its absolute deviation making a subtraction between the point value and the average one. Then, we square this partial result, we sum all the squared deviations and we divide by n: the number of gestures we consider.

$$\sigma^2(x_j) = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2$$

With this procedure we obtain the second layer data for XY-coordinates and pressure values and time average and variance values. All this data will form the second layer described before representing the user template.

User #1	Stroke #1	XY-ponts	Average	$[\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, ..., \bar{x}_n, \bar{y}_n]$
			Variance	$[\sigma^2(x_1), \sigma^2(y_1), ..., \sigma^2(x_n), \sigma^2(y_n)]$
		Pressure values	Average	$[\bar{p}_1, \bar{p}_2, ..., \bar{p}_n]$
			Variance	$[\sigma^2(p_1), \sigma^2(p_2), ..., \sigma^2(p_n)]$
	Stroke #2	XY-points	Average	$[\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, ..., \bar{x}_n, \bar{y}_n]$
			Variance	$[\sigma^2(x_1), \sigma^2(y_1), ..., \sigma^2(x_n), \sigma^2(y_n)]$
		Pressure values	Average	$[\bar{p}_1, \bar{p}_2, ..., \bar{p}_n]$
			Variance	$[\sigma^2(p_1), \sigma^2(p_2), ..., \sigma^2(p_n)]$
	Time value		Average	$\bar{t}$
			Variance	$\sigma^2(t)$

TABLE 3.3: Higher layer data format for user #1 having 3 Gestures made of 2 strokes each

3.3.6 Recognizing procedure

As shown in figure 3.5 the application is divided in two parts: one is the enrollment and the second is the authentication. In the last subsections it was descibed the enrollment procedure, how to get the input drawings and how to manage raw data. This subsection is focused about the recognizing method: how to compare the autentication drawing to stored templates.

The algorithm starts with a request for a sketch which represents the user demand for authentication. Once the drawing is complete by the user the system makes the following actions:

1. Sketch sampling

2. Comparison 1 to N
3. Score computation
4. Prediction production
5. Prediction sorting.

Like explained in subsection 3.3.4, the system samples the sketch drawn by the user in order to be able to compare the last inserted drawing with all the stored templates. The sampling procedure is the same used for the enrollment drawings to assure no error in sampling action.

The most important step in this thesis is the comparison method between the authentication drawing and the user templates stored in memory. For each template, the procedure begins with data retrieval to get user features ready to use. The comparison is made of four in series sub-procedures and to return a true boolean value, which represents a positive match between authentication drawing and template considered, the sketch must get a positive response in all these controls.

The following four checks are based on feature comparison:

1. Strokes number: this control is based on the number of strokes each drawing has. If the stokes number of the authentication drawing differs from the strokes number of the template currently in use the check fails. Vice versa, if both values are the same the sub-procedure the procedure continues.
2. XY coordinates: there is a short preable before this control: the procedure begins with standard deviation computation from the XY-variance array. The operation is quite simple because it is necessary to compute the square root for each variance value. Now, the system is able to create an interval, centered on point average value and bounded by $\bar{x} - CONST * \sigma(x)$ and $\bar{x} + CONST * \sigma(x)$. The control works in this way: if the XY sketch point belongs to the interval the sistem goes through the next step, otherways it stops returning a false value.
3. Time value: the time check works as illustrated in the previous step. However there is a difference on the number of points compared: in XY-control there were 25 points for X-coordinate and 25 points fot Y-coodinate. Now there is a unique time average and time variance value. In other words, if a gesture is drawn to fast or to slow this check will return a false value, vice versa, the main comparison procedure contiues with next template.

4. Pressure values: this is the last control made during the comparison procedure. It works the same as XY-coordinate giving a positive response in case of point within the precomputed interval and a negative value otherwise.

A more formal way to summarize the procedure is:

- In order to get a valid match between the authentication sketch and the currently template analyzed all controls has to return a true value since the logic operation calculate is

$$RESULT = CHECK1 \wedge CHECK2 \wedge CHECK3 \wedge CHECK4.$$

- If a control returns a false value, which means check failed, the procedure discards the current template and continues with the next one or it stops if no more templates are stored in memory. With this trick no work is wasted and the application execution will proceed faster.

In both cases it is calculated a score, which measures the distance between the sketch and the template, useful to establish a ranking between the various templates. The one which gets a lower score corresponds probably to the user who has entered the drawing.

The score computation it is stored in a new data type named *Prediction*. This is made by two fields: the first is a String and contains the personal data information while the second is a Double vaiable which stores the score value. The score calculation procedure is achieved in two different manners depending on result of the comparison method:

- in case of a negative match between sketch and template the template score will be set at 1000;
- vice versa, in case of positive comparison the score is the worst average absolute deviation between all the template strokes[3].

When all the templates have been compared with the authentication drawing a simple method sorts the computed *Predictions* in order to obtain a final ranking. The *Prediction* ranked at first position contains the personal data of the user who typed the drawing. However, if the first *Prediction* score is greater or equal than 1000 the application returns an error message indicating that the authentication drawing doesn't fit in any template in this case the authentication is rejected.

[3]It was observed that this value is never greater than 1000.

3.3.7 Updating mechanism

The adaptation mechanism consists of creating a new updated template, including the new sample and discarding the oldest one. This mechanism is performed after a successful authentication with the goal to keep the template up-to-date. As the adaptation mechanism is performed, the average and variance for each feature is modified and the thresholds are modified.

Chapter 4

Results and conclusions

Contents

4.1 Find the best parameter

As discussed in chapter 1, the most important parameters to evaluate the recognizing system are False Acceptance Rate and False Rejection Rate. To achieve this goal the application should be permissive (accepting those drawings that are enough similar to the template) and strict (rejecting intruders attempts) at the same time. In practice, this constitutes an engineering compromise between FAR and FRR.

This results in choosing the width of the interval within the values are considered accepted neither too small nor too large. To achieve this goal we had to find which value best fits in $CONST$ parameter used in subsection 3.3.6. This is a value which affects the interval bounds: if it is too high the interval will be too large, vice versa it will be to short. The default value is 6 and the choice is light strict because we prefer to have an higher False Rejection Rate instead of an higher False Acceptance Rate.

This value is unique for each feature analyzed (XY-coordinates, time and pressure values) because it was found that there is no gain using a different value for each feature, and also would have no sense to do it.

The consequences of this choice are:

- the main aspect concerns FAR and FRR parameters. Programming a strict algorithm brings to a very low False Acceptance Rate, however the False Rejection Rate could be significant.
- despite 3σ in statistics covers $99,7\%$ of data points, in this algorithm 6σ are just enough to authenticate each user with his/her drawing without having a too high FRR.

4.2 Results

In this section we present the results obtained during the test phase and explain how we collect the experimental data.

The people involved in tests are:

- males and females;
- from 18 to 50 years old;
- with a different experience level of Android OS and touchscreen technology.

The test procedure is now described:

1. I start showing my own authentication drawing so the tester can see how to draw sketches on a touchscreen display;
2. when the user completely understands how to reproduce my personal sketch (in terms of xy-points, time and pressure), he/she makes few authentication tries to get the hang of it.
3. in case of multiple tester the *i*-th tester explains to the *(i+1)*-th person how to reproduce his/her own sketch.
4. now I ask the tester to authenticate 20 times using my sketch or the previous user sketch in order to measure the False Acceptance Rate.
5. the next step is necessary to calculate FRR. The tester should enroll using the enrollment procedure to gain a template within the application.
6. the tester tries to authenticate 20 times and every time the application refuses the sketch a false rejection is marked. In this way the FRR can be computed.

User #	Gender	Rejections	FRR		Acceptance	FAR
1	M	3 of 20	15%		0 of 20	0%
2	M	2 of 20	10%		0 of 20	0%
3	M	1 of 20	5%		0 of 20	0%
4	F	6 of 20	30%		0 of 20	0%
5	M	6 of 20	30%		0 of 20	0%
6	M	0 of 20	0%		0 of 20	0%
7	M	2 of 20	10%		11 of 20	55%
8	M	0 of 20	0%		4 of 20	20%
9	M	6 of 20	30%		0 of 20	0%
10	F	1 of 20	5%		0 of 20	0%
11	M	4 of 20	20%		0 of 20	0%
12	M	1 of 20	5%		0 of 20	0%
13	F	7 of 20	35%		0 of 20	0%
14	F	2 of 20	10%		0 of 20	0%
15	M	0 of 20	0%		0 of 20	0%
16	M	0 of 20	0%		0 of 20	0%
17	F	6 of 20	30%		0 of 20	0%
18	F	0 of 20	0%		0 of 20	0%
19	M	4 of 20	20%		0 of 20	0%
20	F	0 of 20	0%		5 of 20	25%
21	M	6 of 20	30%		0 of 20	0%
22	F	3 of 20	15%		0 of 20	0%
23	M	2 of 20	10%		0 of 20	0%
24	M	1 of 20	5%		0 of 20	0%
25	M	5 of 20	25%		0 of 20	0%
		FRR average	13,6%		FAR average	4%

TABLE 4.1: Experimental data

The following table resumes the data collected.

The first colum denotes the user progressive number, the second refers to the gender, the third is the number of times the user failed to access to his/her own account and in the fourth the relative FRR. The fifth column reports how many times the tester authenticate with my account and in the last column it is reported the False Acceptance Rate.

The False Rejection Rate is about 14% and now we give some details:

- this is due to the strategical choice to develop a strict application.
- moreover, this has to be referred to the fact that not every person has experience with touchscreen devices.
- analyzing where the users fail to authenticate half of the mistakes are about xy-points and the other half about pressure. The time feature is never decisive.

The average FAR value is close to 5% (1 mistake in 20 tries) and we consider this as a very good value for many reasons:

- if I didn't explain which drawing I insert in the system the FAR was almost surely 0%;
- the FAR obtained is the result of a well-explained sketch, so only in this case the intruder has some possibilities to steal someone's else credentials;
- despite what just said is not sure that the intruder can log in with different credentials because many features are analyzed: in case of username and password once the intruder knows this data he/she gets a 100% FAR;
- drawings made of 2 or more strokes are totally secure (0% FAR) while gestures of one stroke are the only which you can cheat.

4.3 About results

In this section we analyze the results presented in the last section. The following are algorithm strengths that are advantages introduced by this application:

- **Universality**: this application can be used on any Android-powered mobile devices which owns the touchscreen technology and pressure sensor;
- **Security**: this application has a different level of security rather than typing username and password. As a matter effect, when the intruder knows user credentials he can access to user's profile for sure. However, using the system developed in this thesis the intruder must know many more information to draw the user sketch so the access is not always granted. Just think of different cases:
  - if the intruder knows the user drawing he cannot kwon how to draw it in terms of strokes sequence unless he has not seen the user while gesturing;
  - at the same time the intruder can make some mistakes about drawing positioning, correct shape and right size, unless the intruder observed the designer.
  - once again, the intruder cannot know in how much time he has to insert the sketch;
  - finally, if the intruder knows all the informations he cannot know anything about pressure imprinted by the user.

- **Simplicity**: the application can be considered simple for two main aspects: it is easy-to-use and the algorithm behind this application is complex but not difficult. The first is about user side and guarantees an easy user interface, many informations during the enrollment procedure and guided messages in case of not sufficient good drawing. On the other side, the algorithm is quite easy because it uses weight average and variance concepts instead of neural networks, hidden Markov model, support vector machine and so on.

- **Scalability**: the application can work from one to unlimited users and there are no constraints except SD card capacity in the mobile device. This because every information needed by the software is stored in the device memory storage.

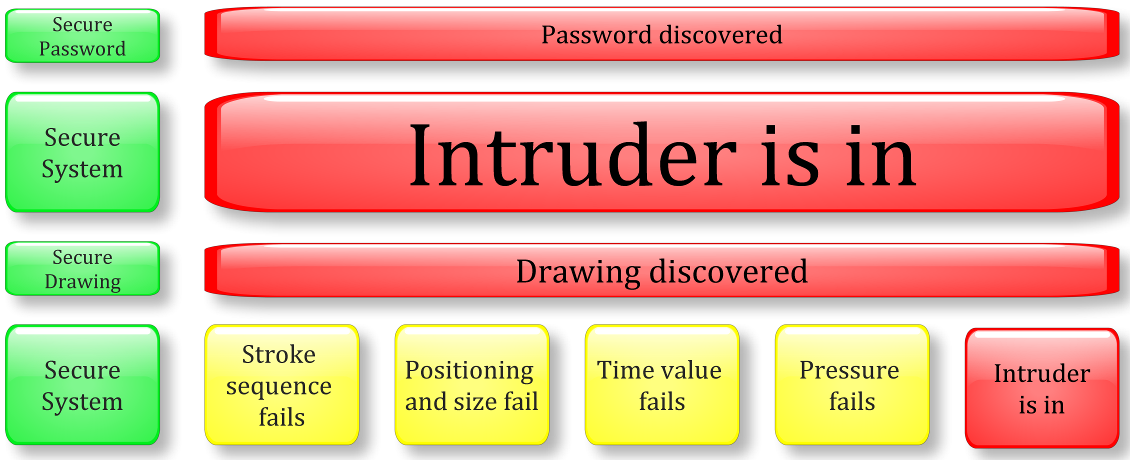


FIGURE 4.1: Comparison between password security system (top) and drawing security system (bottom)

Analyzing the data collected during the test phase we can say more:

- we obtain a higher False Rejection Rate than False Acceptance Rate, so the algorithm is considered strict;

- drawings made of 2 or more strokes as more secure than ones constitutes of only one stroke. The only cases when a false acceptance was made by the system are about sketches made of one stroke.

- **Flexibility**: changing just one project value the algorithm can be more permissive or stiffer. Notice that changing a project constant will increase or decrease acceptance intervals and causing a different FAR and FRR values.

4.4 Conclusions

This work is something totally new in the biometric field. I think that this algorithm joined with this simple application could become a solid base for future developing. I imagine that this application can overtake the traditional username and password procedure in two different ways: the first is less invasive where once the biometric authentication is complete, the system sends the linked username and password to the remote server to authenticate the user as the traditional manner. The second one replaces traditional credentials: the authentication is based only on the drawing recognition. Each device sends the raw data to the remote server which runs the algorithm and find the best match and, if there is one enough good, it will authenticate the user. This needs a different code implementation and a sufficient storage capacity on the server side.

This work shows that a simple freehand drawing can distinguish people one from the other with no apparent differences, however there are concrete differences on strokes number, XY-points and time and pressure values. Moreover, due to its flexibility, this application can be calibrated for every user need yielding a more or less strict authentication procedure.

Appendix A

User Manual

Contents

A.1 Main screen

In this appendix we will explain how to use the *Smart Login* application. The main goal of this work is not the production of an application for Android operating system, but the realization of an algorithm for the recognition based on a drawing. For this reason, the user interface is intentionally left simple and functional for the user without the introduction of aesthetic embellishments. The following screenshots are taken from an *HTC Desire* smartphone[1].

The main screen (see figure A.1) allows the user to choose with a simple tap between the enrollment procedure, which has to be performed by each user at first access, and the authentication mode.

A.2 Enrollment procedure

The enrollment procedure begins with the insertion of the user's personal data in a form where only first and last name of the person are required. If the authentication system is

[1]HTC is a trademark of HTC Corporation.

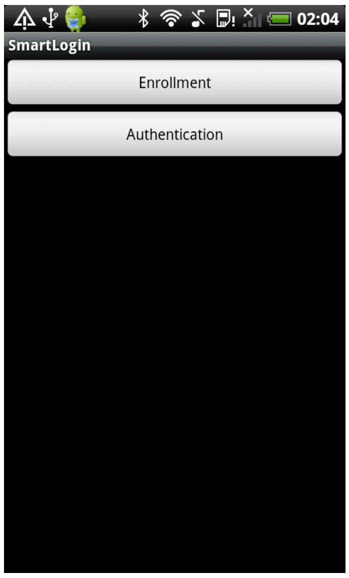
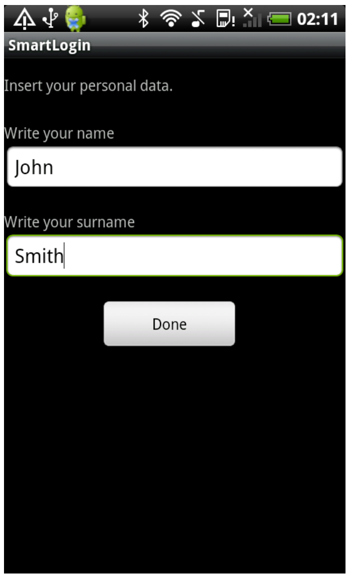


FIGURE A.1: Main screen.

developed for an area other than the current one, you can change the form fields without too much work and require user name and password, for example. In this way the application, after recognizing the user, can authenticate him/her to a website without having to enter user name and password each time.



Insert your personal data.

Write your name

John

Write your surname

Smith

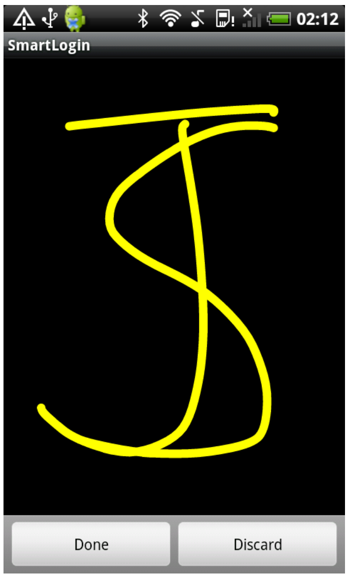
FIGURE A.2: The enrollment form

After entering your data press *Done* button.

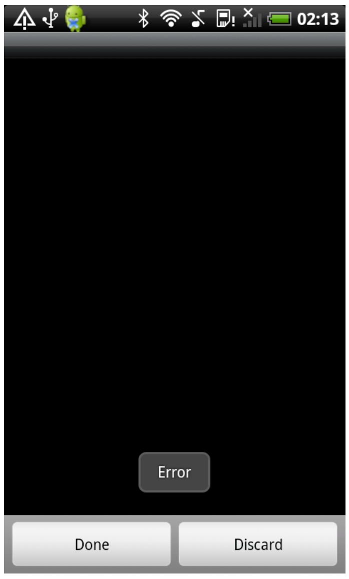
Now, is the time to enter your own drawing within the application. Please note that you can enter a multi stroke sketch and that the total length should be long enough. Press

Done button at the end. If a mistake has been detected (e.g. too shoort length) an error message will be displayed as shown in figure A.3(b).

The *Discard* button has the purpose of eliminating the input drawing in case of mistake. Notice that this resets the whole enrollment procedure and the user has to start again from the beginning.



(a) An example of three strokes drawing



(b) Error message

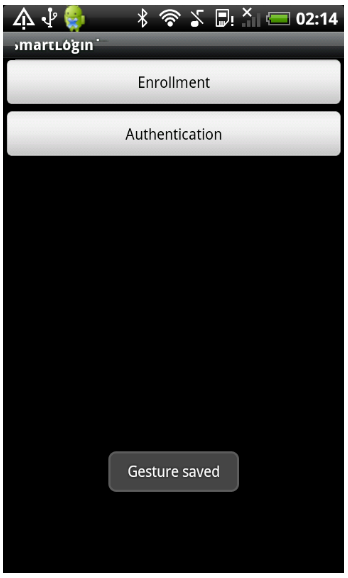
FIGURE A.3: Enrollment screenshots

If the first 6 drawings inserted are sufficiently similar to each other the enrollment procedure ends after with the message *Gesture saved* in main screen. If not, the procedure continues asking 5 more drawing with the message *5 more times*.

A.3 Authentication procedure

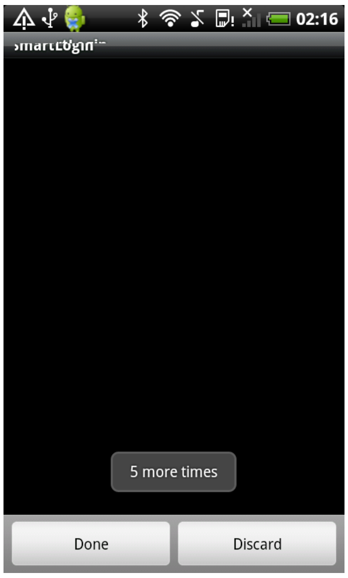
The authentication procedure begins when the user taps the authentication button in main screen. In the next screen, the user has to enter his/her sketch to be authenticated. The layout displayed is the same used in the enrollment procedure for a choice of conformity and reference points for the user. The drawing now you have to draw should be as similar as possible to those inserted in to the training phase both in terms of size, orientation and pressure with your finger.

If everything goes right a message with the user's personal data will be despalyen in main window, otherwise a *Gesture not recognized* message will assert that the comparison failed.



Gesture saved

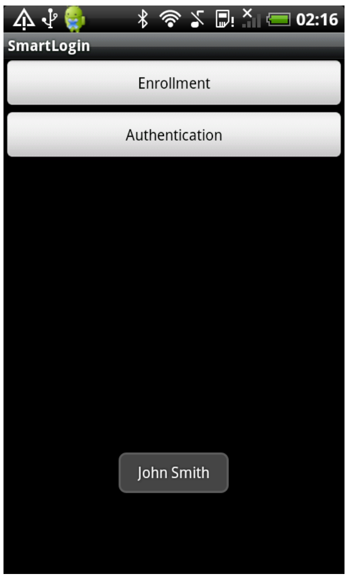
(a) Successful enrollment



5 more times

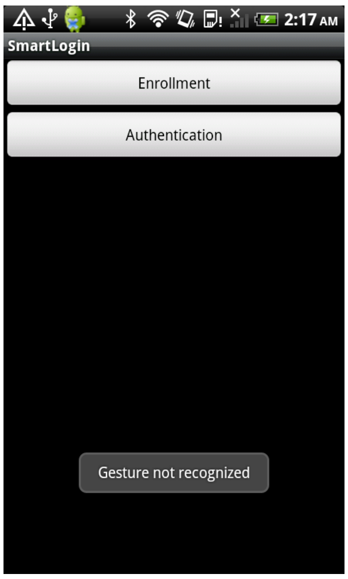
(b) Continuation message

FIGURE A.4: Enrollment phase messages



John Smith

(a) Successful authentication



Gesture not recognized

(b) Authentication failure

FIGURE A.5: Authentication phase messages

Bibliography

[1] A. V. Rubin F. Monrose. Keystroke dynamics as a biometric for authentication.

[2] URL http://dictionary.reference.com/browse/biometrics?s=b.

[3] Mario Koeppen Katrin Franke, Javier Ruiz-del-Solar. Soft-biometrics: Soft-computing for biometric-applications.

[4] S.C. Dass A.K. Jain and K. Nandakumar. Soft biometric traits for personal recognition systems.

[5] M.G. Lizarraga L. L. Ling L. C. F. Araujo, L. H. R. Sucupira Jr and J. B. T. Yabu-Uti. User authentication through typing biometrics features.

[6] L. Schomaker. Advances in writer identification and verification.

[7] Fabio Masarin. Un sistema per la verifica di identità basato sull'analisi del disegno di simboli grafici. Master's thesis, Università degli Studi di Padova, 2011.