

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Complementing smartphone's touch screen data with motion sensors data

Joaquim Oliveira



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Dirk Elias

July 22, 2014

Complementing smartphone's touch screen data with motion sensors data

Joaquim Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Dr. Rui Carlos Camacho de Sousa Ferreira da Silva

External Examiner: Prof. Dr. Jose Manuel de Castro Torres

Supervisor: Prof. Dr. Dirk Christian Elias

July 22, 2014

Abstract

Currently smartphones are the consumer electronic devices with higher growth rate. The wide variety of functionalities that they provide is what makes them so desirable. They can be used them to play, to communicate, to take pictures, to guide us, to work, and more. For that to be possible the smartphones have several built-in sensors, which make them much more interesting.

This project aims to complement the information given by the smartphones touch screen with information given by the accelerometer and gyroscope. In others words, using sensors like the accelerometers and gyroscopes, to provide new ways to characterize the touch. For that the accelerometer and gyroscope signals were studied and several algorithms were create, such as to detect touches using this two sensors, to determine the device position (hand or on surface), the hand and the finger used to touch on device and the touch force.

With this purpose it was used signal processing techniques like filters and sliding windows and data mining techniques (Weka software is used to support the data classification).

In order to evaluate the algorithms and to be able to conclude how the experience using smartphones and the physical conditions affect the performance of the algorithms, it was used two groups with different ages (a group had the mean age of 20.7 years and the other had the mean age of 71.2 years). The results show that the algorithm to determine the smartphone position and the algorithm to detect touch have an excellent performance for both groups (precisions higher than 95%), although the last presents a little problem with the false positives. The other algorithms' results are slightly different. Although results of the younger group are good (hovering around 85%) the results of the older group are slightly lower (hovering around 70%).

Resumo

Actualmente os smartphones são o dispositivo electrónico com a maior taxa de crescimento de vendas. A grande variedade de funcionalidades é os que torna tão desejáveis. Eles podem ser usados para jogar, comunicar, tirar fotografias, orientar-nos, etc. Os vários sensores embutidos que eles possuem são o que os torna tão interessantes.

Este projecto tem como objectivo complementar a informação disponibilizada pelo touch-screen com informação do acelerómetro e giroscópio, ou seja, usando sensores como o acelerómetro e o giroscópio, criar novas formas de caracterizar o toque. Com esse objectivo esta dissertação estuda os sinais do acelerómetro e do giroscópio e desenvolve vários algoritmos para detectar toques usando estes dois sensores, determinar como o smartphone está a ser usado, qual é a mão que está segurando o smartphone, determinar como que dedo foi realizado o toque e determinar a força do toque.

Com esse objectivo, foi utilizado técnicas de processamento de sinal, como filtros sliding windows e técnicas de data mining (WEKA é usado para apoiar a classificação de dados).

Para avaliar os algoritmos e concluir como a experiência no uso de smartphones e a condição física afecta o desempenho dos algoritmos, foi utilizado dois grupos de participantes com idades diferentes (um grupo com a média de idade de 20,7 anos e com a média de idade de 71.2 anos). Os resultados mostram que o algoritmo que determina o modo como o utilizador está a usar o smartphone e o algoritmo de detecção de toques tem um excelente desempenho de ambos os grupos (precisões superiores a 95 %), embora o último apresenta um pequeno problema com os falsos positivos. Os resultados de outros algoritmos são ligeiramente diferentes. Embora os resultados do grupo mais jovem sejam bons (oscilando em torno de 85 %) os resultados do grupo mais velho são ligeiramente mais baixos (oscilando em torno de 70 %). Esta diferença pode ser explicada falta de experiencia dos participantes (para o algoritmo que detecta qual a mão que está a segurar o smartphone) ou pelos facto dos idosos terem os dedos um pouco mais grossos do que a media (para o algoritmo que detecta o dedo).

Acknowledgements

Firstly I would like to express my honored gratitude for Eng. João Cevada, Fraunhofer AICOS supervisor, for the insights and support in the different phases of the project, and for Prof. Doctor Dirk Elias, FEUP supervisor, for his help and recommendations during the development of the project.

Also, I would like to thank to everyone who participated on the data collection, whose contribution was vital for the final outcome.

Finally, I want to, in a very special way, thank my family, especially to my parents, my sister and my grandmother for their infinite patience and support in the good and bad moments as well as for the endless days that they had to hear me.

Joaquim Oliveira

*“Computer games don’t affect kids,
I mean if Pac Man affected us as kids,
we’d all be running around in darkened rooms
munching pills and listening to repetitive music”.*

Marcus Brigstocke

Contents

1	Introduction	1
1.1	Context/Background	1
1.2	Work Description	1
1.3	Motivation	2
1.4	Overview of report	2
2	State of the art	3
2.1	Smartphone sensors	3
2.1.1	Accelerometer	4
2.1.2	Gyroscope	5
2.2	Signal processing	7
2.2.1	Domains	7
2.2.2	Fast Fourier Transform (FFT)	9
2.2.3	Digital filters	10
2.2.4	Kalman filter	11
2.2.5	Sliding windows	11
2.3	Machine Learning	12
2.3.1	Supervised learning	12
2.3.2	Unsupervised learning	13
2.4	Related work	13
2.5	Used Technologies	17
2.5.1	Android	17
2.5.2	Weka	18
2.6	Conclusion	20
3	Implementation	21
3.1	Algorithms	21
3.1.1	Smartphone position	22
3.1.2	Detect touch	24
3.1.3	Hand (right/left)	27
3.1.4	Touch force	28
3.1.5	Finger (thumb/index finger)	29
3.2	TouchSensor service	30
3.2.1	Architecture	31
3.2.2	Hardware communication	31
3.2.3	Sensor data preprocessing	32
3.2.4	Data processing	35

CONTENTS

4	Validation and Evaluation	37
4.1	Sample collection process	37
4.1.1	Participants	38
4.1.2	Scenario	39
4.2	Results	39
4.2.1	Smartphone position	39
4.2.2	Touch detection	40
4.2.3	Hand (Right/Left)	41
4.2.4	Finger (Thumb/Index finger)	42
5	Conclusions and Future Work	45
5.1	Future Work	46
	References	47
A	Decision tree (position)	51

List of Figures

2.1	SmartPhone Reference Frame [Ode]	6
2.2	Classification of techniques applied to sensor signal for fexture extraction [DFC10].	7
2.3	WalkType resulted in higher typing speeds than the control condition, particularly while participants were walking [MGW].	14
2.4	WalkType resulted in lower error rates than Control,especially for walking [MGW].	14
2.5	User interface (TouchLogger) [CC11].	15
2.6	(left) Minimal device rotation in x- and y-axis, and smaller touch size when the user touches nearby with the thumb. (center) Significantly more rotation in x- and y-axi and larger touch size when the far quadrant of the screen is trouched. (right) The shape of the swipe arc in the case of right thumb [GP12].	16
2.7	Block diagrama of the major components of GripSense’s pressure detection module [GP12].	16
2.8	Android components [Kel]	18
2.9	Weka’s interface.	19
3.1	Information flow.	22
3.2	Determine the correct position (state: current position; newState: instantaneous position).	24
3.3	Accelerometer and gyroscope signals of a touch.	26
3.4	Gyroscope signal when someone touches on smartphone (left/right).	27
3.5	Decision tree produced by J48 method	28
3.6	SMV signal when someone touches on the smartphone three times with increasing forces.	29
3.7	System Architecture.	31
3.8	First chart: Raw accelerometer signal; Second chart: Filtrated accelerometer signal. The two charts have different scales.	33
3.9	Determine Window Size.	35
4.1	Data collection application	37
4.2	Comparison of results between the two groups (smartphone position.)	40
4.3	Comparison of results between the two groups (detected touches).	41
4.4	Comparison of results between the two groups (determine the hand).	42
4.5	Picture taken during the data collection (older group).	42
4.6	Comparison of results between the two groups (determine the finger).	43
A.1	Decision tree (smartphone position).	51

LIST OF FIGURES

List of Tables

2.1	Time domain features [DFC10]	8
2.2	Frequency string domain features [DFC10]	9
2.3	Symbolic domain features [DFC10]	9
2.4	Summary of all inferences made by GripSense and when and which features were used for each of them [GP12]	17
3.1	Data mining techniques and their accuracy without data on the training sets	23
3.2	Data mining techniques and their accuracy without data on the training sets (hand detection)	28
3.3	Input and output for each algorithm.	36
4.1	Results of the younger group (smartphone position)	39
4.2	Results of the older group (smartphone position)	39
4.3	Percentages of detected touches.	40
4.4	Results of the younger group (determine the hand)	41
4.5	Results of the older group (determine the hand)	41
4.6	Results of the younger group (determine the finger)	43
4.7	Results of the older group (determine the finger)	43

LIST OF TABLES

Abbreviations

AICOS	Assistive Information and Communication Solutions
API	Application Programming Interface
ARFF	Attribute-Relation File Format
CSV	Comma Separated Values
FFT	Fast Fourier Transform
HMM	Hidden Markov Models
GPS	Global Positioning System
NFC	Near Field Communication
SMA	Signal Magnitude Area
SMV	Signal Magnitude Vector
Weka	Waikato Environment for Knowledge Analysis
Wi-Fi	Wireless Fidelity

Chapter 1

Introduction

This chapter contextualizes this dissertation and it presents its description and objectives. Its motivation and overview of report also are explained.

1.1 Context/Background

This dissertation is inserted in the smartphone sensors field, more precisely the accelerometer and the gyroscope, and how their combination can complement the information about how the user interage with the smartphones. This work will be developed in Fraunhofer Portugal AICOS. This company aims to enhance people's living standards (offering them intuitive and useful technology solutions) and leading to the integration of an increasingly large of the population in the information and knowledge society.

Currently the smartphones are the device with higher growth rate [Dan]. The wide variety of functionalities that they provide is what makes them so desirable. We can use them to play, to communicate, to take picture, to guide us, to work, etc. But how is this possible? We can say that are the sensors which allow this. They are what allow the users interact with the smartphones.

The smartphones have several sensors, such as GPS, speaker, microphone, Wi-Fi, NFC, front and rear cameras, accelerometer, gyroscope, magnetometer, etc. There are many studies about them and about the different ways of combining them. This dissertation study the accelerometer and the gyroscope and which informations we can retire though these.

1.2 Work Description

Currently it only is possible to obtain information about a tap on the smartphone's screen using the touch sensor. The goal of this project is to complement the information given by the smartphones touch screen with information given by the accelerometer and gyroscope and this way to extend the smartphone's touch capabilities with data such as tap strength, smartphone's holding position

(if it is on the users hands or laying on a hard surface), which finger the touches the device (thumb or index finger) or which hand is holding the smartphone.

Using sensors to detect touches is not a new idea, as there is already research which use the smartphone accelerometer to infer which keystrokes were made on a touch screen, and use the accelerometer information to know when there was a tap on the virtual keyboard is made even before the touchscreen detects it, this project aims to use this information to other proposes, getting new information on a touch event and process that information in order to be useful to the user or developer party.

This dissertation is divided into three parts:

1. Develop an algorithm that identify touches with the gyroscope and accelerometer to extract new characteristics of the touch;
2. To evaluate the performance of the algorithms;
3. Develop an API that makes the new touch data easily available (Android Service);

1.3 Motivation

Although there are many studies in this field, using the combination of gyroscope and accelerometer currently is a subject with few approaches and very superficial. This dissertation will go deeper extracting new information about the touch in order to improve the user interaction with the smartphones.

This new information can be very useful to better evaluate the smartphone usage patterns as it complements the touch data already given by the touch screen. It also can enables a host of new functionalities like adding a strength dimension to the touch and additionally as older adults are very prone to input errors this data could give useful to discard unwanted touches.

1.4 Overview of report

This report has six chapters, the Introduction (Chapter 1) where the problem is exposed, as well as motivations and objectives of the developed work.

The Charter 2 reviews the state of the art, it goes trough different signal processing techniques and different classification models. While the chapter 3 presents the used technologies.

The next chapter, Implementation (Chapter 4), describes the system architecture and the several algorithms.

The fifth chapter (Chapter 5) addresses to a description the exercises evaluation and the validation of the algorithms implemented as well as some considerations.

The last chapter (Chapter 6) has a description some potential future work, and the conclusions of this project are presented.

Chapter 2

State of the art

Before starting, it is necessary to know what have been done until now. For that, the state of the art in use of the accelerometer and gyroscope in smartphones is analyzed in this chapter. Additionally some related works in this area will also be reviewed.

2.1 Smartphone sensors

Stepping back 20 years ago, it's worth remembering the vision that begin with General Magic and Apple's Newton: "a dream of improving the lives of many millions of people by means of small, intimate, life support systems that people carry with them everywhere" (General Magic company mission statement, May 1990 – Mountain View, CA). With the emergence of smartphones, we can say that this dream is true. They allow us to have the world with us and everywhere. Everything is so easy with a simple touch or gesture.

But, have you ever thought how your smartphone responds to your movements and gestures so accurately? Or when you are playing a racing game on phone, you simply need to tilt the device in order to steer the car in a particular direction. When you move to a brighter environment, the smartphone display immediately gets brighter. All of these are possible with the help of sensors inside your smartphone.

Sensors are devices which measure the physical energy and converts its into a signal. Later this signal can be read or observed from an instrument or an electronic device connector to the sensor. There are many types of sensors available in market today. There are innumerable applications for sensors of which most people are not aware. This applications can include machines, cars, aerospace, medicine, robotics and robots. In the case of smartphones, the sensors are built into handset using Micro-Electro-Mechanical Systems, or MEMS. It is a technology that can be defined as miniaturized mechanical and electro-mechanical elements that are made using the techniques of micro fabrication. In most cases, a micro sensor reaches a significantly higher speed and sensitivity compared with macroscopic approaches.

Every day a new smartphone model is being launched into the world with different features. These new features and specifications are possible due to the wide range of sensors and their different combinations. The main sensors are:

Accelerometer

Used to measure the acceleration applied;

Gyroscope

Used to measure the orientation;

Magnetometer

Used for two general purposes - to measure the magnetization of a magnetic material like a ferromagnetic, or to measure the strength;

Front and rear cameras

An instrument that records images. These images may be photographs or movies images.

Barometer

A scientific instrument used to measure pressure;

Microphone

Used to capture songs. It can be used as a voice recorder, for recording personal notes, meetings, or impromptu sounds around us.

Proximity Sensor

Used to detect how far the user from the device is. The primary function of a proximity sensor is to disable accidental touch events.

Light sensor

Sensing light density. It controls display brightness based on how much ambient light is present;

For this dissertation the most relevant sensors are the accelerometer and the gyroscope.

2.1.1 Accelerometer

An accelerometer is a compact device designed to measure non-gravitational acceleration. When the object integrated with it goes from a standstill to any velocity, the accelerometer is designed to respond to the vibrations associated with such movement. It uses microscopic crystals that go under stress when vibrations occur, and from that stress a voltage is generated to create a reading on any acceleration [Eng].

In order to understand how an accelerometer works, we need to recall the Newton's first law "All bodies remain at rest until some external force acts on it". The second law defines the force applied as the product between the body mass and their acceleration. In other words, measuring

the acceleration applied to a body, we discover what is the force applied on it. That way, Accelerometers are important components to devices that track fitness and other measurements in the quantified self-movement.

The most important source of error of an accelerometer is its bias, which is the offset of its output signal from the true value. It is possible to estimate a accelerometer bias by measuring the long term average of the accelerometers output when it is not undergoing any acceleration.

2.1.2 Gyroscope

A gyroscope allows a smartphone to measure orientation, in other words it is a device that uses Earth's gravity to help determine orientation. Its traditional design is a freely-rotating disk called a rotor, mounted onto a spinning axis in the center of a larger and more stable wheel. As the axis turns, the rotor remains stationary to indicate the central gravitational pull, and thus which way is "down" [Hao]. When applied to a smartphone, a gyroscopic sensor commonly performs gesture recognition functions. Additionally, gyroscopes in smartphones are used to help determine the position and orientation of the phone. The gyroscopes used in smartphones are inexpensive vibrating structure gyroscope manufactured with Microelectromechanical system (MEMS). They are making significant progress towards high performance and low power consumption [JEX].

Like all sensors, a gyroscope is not perfect and has small errors in each measurement. The problem with gyroscope is that there are bias and numerical errors. The bias of a gyroscope is the average output from the gyroscope when it is not undergoing any rotation. Integrating the gyroscope bias it's obtained an angular drift, increasing linearly over time. Another error arising in gyros is the 'calibration error', which refers to errors in the scale factors, alignments, and linearities of the gyros. Such errors are only observed whilst the device is turning. Such errors lead to the accumulation of additional drift in the integrated signal, the magnitude of which is proportional to the rate and duration of the motions.

2.1.2.1 Gyroscope mathematical model

A gyroscope is a device used primarily for navigation and measurement of angular velocity expressed in Cartesian coordinates:

$$w = (w_x, w_y, w_z) \quad (2.1)$$

In Android API the device co-ordinate frame is defined relative to the screen of the phone. The X-axis is horizontal and points to the right, the Y-axis is vertical and points towards the top of the screen and the Z-axis points outside the front face of the screen [SR]. The angles definitions are the Pitch (angle of X-axis relative to horizon, also a positive rotation about Y body axis), Roll (angle of Y-axis relative to horizon, also a positive rotation about X body axis) and Yaw (angle of X-axis relative to North, also a positive rotation about Z body axis), as shown in Figure 2.1 .

The angle can be derived by integration of the angular velocity in each direction:

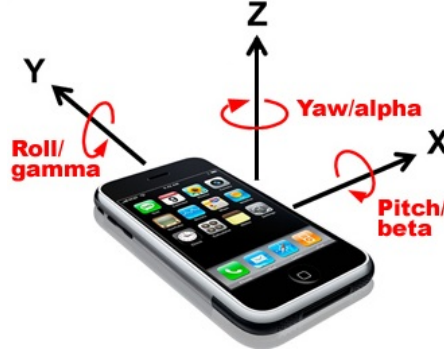


Figure 2.1: SmartPhone Reference Frame [Ode]

$$\theta_p(t) = \int_{t_0}^t w_p(t) dt + \theta_{p0} \quad (2.2)$$

Where p is index which can be (x,y,z), Pitch is θ_y , Roll is θ_x and Yaw is θ_z . θ_{p0} is the initial angle compared to the earth axis coordinates which are defined as follow:

X Axis Positive in the direction of North;

Y Axis Positive in the direction of East (perpendicular to X Axis)

Z Axis Positive towards the centre of Earth (perpendicular to X-Y Plane)

In order to convert from body axes to earth axes we use the following matrix:

$$\begin{bmatrix} \cos(\theta_z) \cdot \cos(\theta_y) & -\sin(\theta_z) \cdot \cos(\theta_x) + \cos(\theta_z) \cdot \sin(\theta_y) \cdot \sin(\theta_x) & \sin(\theta_z) \cdot \sin(\theta_x) + \cos(\theta_z) \cdot \sin(\theta_y) \cdot \cos(\theta_x) \\ \sin(\theta_z) \cdot \cos(\theta_y) & \cos(\theta_z) \cdot \cos(\theta_x) + \sin(\theta_y) \cdot \sin(\theta_x) & -\cos(\theta_z) \cdot \sin(\theta_x) + \sin(\theta_z) \cdot \sin(\theta_y) \cdot \cos(\theta_x) \\ -\sin(\theta_y) & \cos(\theta_y) \cdot \sin(\theta_x) & \cos(\theta_y) \cdot \cos(\theta_x) \end{bmatrix} \quad (2.3)$$

Where θ_x is the Roll around the x-axis, θ_y is the Pitch around the y-axis and θ_z is the Yaw around the z-axis.

The matrix is used to convert the angles from angles referring to the body's movement from its previous position, to angles referring to its movement from the position before it. When it is then multiplied with all the previous matrixes, since each matrix represents one movement, the multiplied outcome represents all the movements since the beginning of the measurements, thus multiplying the body axes coordinates, $[X_b, Y_b, Z_b]$, with the above mentioned matrix, will result in the object's coordinates in earth axes.

2.2 Signal processing

The signal coming from an accelerometer or a gyroscope may require the use of a processing stage in order to extract a set of basic features and in this way allow us to characterize the signal.

2.2.1 Domains

It is possible to classify the sensor signal processing techniques in three broad domains [DFC10], such as time domain, frequency domain and discrete symbolic string domain. It's possible and desirable a combination of them. In the Figure 2.2 is possible is described the most representative techniques in each of these domains.

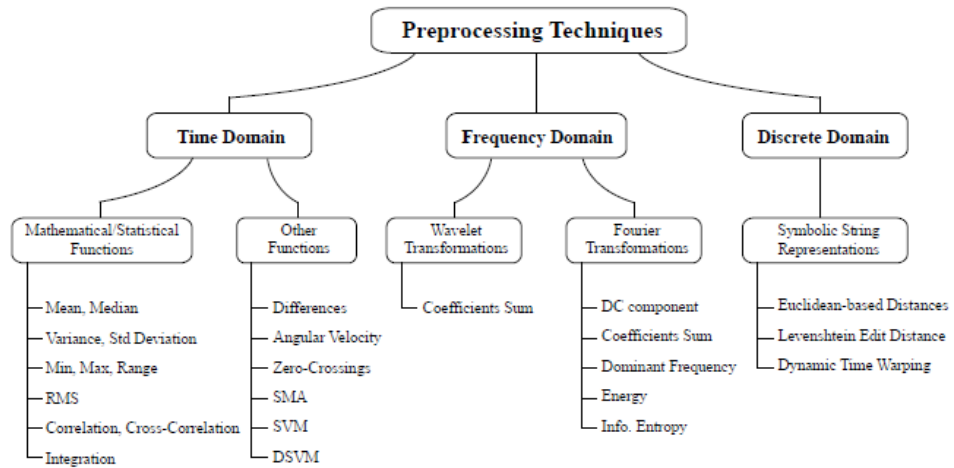


Figure 2.2: Classification of techniques applied to sensor signal for feature extraction [DFC10].

2.2.1.1 Time domain

Simple mathematical and statistical metrics used to extract basic single information from raw data. A time domain graph shows how a signal changes over time. The features obtained are simple to compute because they can be calculated when the data is being read [DFC10].

Table 2.1: Time domain features [DFC10]

Feature	Details
Mean	Identify user posture (sitting, standing or lying) Data smoothing
Median	Separating the higher half of data samples from the lower half
Variance (δ^2)	Average of the squared differences from the mean
Standard deviation δ	Square root of the variance Signal stabilityl
Min, Max, Range	Range is the difference between min and max
Root mean square	$x_{RMS} = \sqrt{\frac{x_1^2 + x_2^2 + \dots + x_n^2}{n}}$
Integration	Measuring the signal area under the data curve
Correlation	Measuring strength and direction of a linear relationship between two signals $\rho_{x,y} = \frac{cov(x,y)}{\delta_x \delta_y}$
Cross-Correlation	Used to search for a known pattern in a long signal
Differences	The difference between signals in a pairwise arrangement of samples
Zero-Crossings	The points where a signal passes through a specific value corresponding to half of the signal range
Angular velocity	It allows to determine orientation
Signal magnitude area	The sum of the area encompassed by the magnitude of each of the three-axis accelerometer signal
Signal vector magnitude	Used to identify possible falls and to monitor and classify behavior patterns $SVM = \frac{1}{n} \sum_{i=1}^n \sqrt{x_i^2 + y_i^2 + z_i^2}$

2.2.1.2 Frequency domain

Frequency-domain techniques capture the nature of the sensor signal. It refers to the analysis of mathematical functions with respect to frequency. The frequency-domain features are obtained using Fast Fourier transform (FFT), which is a spectral representation of the sensor data windows. It's also possible to transformer into frequency domain using the Wavalet Haar Transform, which is based on the decomposition of a set of orthonormal vectors or coefficients [DFC10].

Table 2.2: Frequency string domain features [DFC10]

Feature	Details
DC Component	First coefficient in spectral representation
Spectral Energy	Squared sum of its spectral coefficients normalized by the length of the sample window
Information Entropy	Normalized information entropy of the discrete FFT coefficient magnitude excluding DC component Signal stability
Coefficients sum	Summation of a set of spectral coefficients
Dominant frequency	Frequency value corresponding to the maximal spectral coefficient

2.2.1.3 Symbolic strings domain

It's the transformation of a sensor signal into strings of discrete symbols, using a limited symbol alphabet. Symbolic aggregate approximation (SAX) is a technique used in the transformation process. It uses a piecewise aggregate approximation (PAA) which is a Gaussian equiprobable distribution function to map range values into string symbols [DFC10].

Table 2.3: Symbolic domain features [DFC10]

Feature	Details
Euclidean-related distances	$EuclidianDist(D, T) = \sqrt{\sum_{i=1}^n (ns_i - t_i)^2}$ Distance between the signal values that correspond to each symbol in the string representation
Minimum distance	$MinDist = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^n dist(s_i - t_i)^2}$ Measuring similarity between two sequences
Dynamic time warping	$DTW(S, T) = \min \left\{ \frac{1}{k} \sqrt{\sum_{k=1}^k w_k} \right\}$

2.2.2 Fast Fourier Transform (FFT)

A fast Fourier transform (FFT) is an algorithm to compute the discrete Fourier transform (DFT) and its inverse, in others words, the input signal is transformed into the frequency domain using the DFT [DFC10].

It is simple and fast (computation efficient), because by making use of periodicities in the signs that are multiplied to do the transforms, the FFT greatly reduces the amount of calculation required. The FFT algorithm applies only to signals comprising a number of elements which is equal to 2^m (e.g. $2^8 = 256, 2^{10} = 1024$ etc.). Its main advantage is that it significantly reduces the computation time by a factor of the order $m/\log_2 m$, i.e. more than 100 times for a sample of 1024 elements [Sek].

The FFT return a set of complex number with exception of the spectral components at $f = 0$ and $f = f_s/2$, which are both real. The number of FFT elements is equal to the size of the

time sample. The second half of these complex numbers corresponds to negative frequencies and contains complex conjugates of the first half for the positive frequencies, and does not carry any new information [Sek].

The use of this algorithm is very common in processing of accelerometer and gyroscope data ([MGW]), mainly in the recognition of physical activities ([DFC10]) or in gesture recognition ([JWL09]).

2.2.3 Digital filters

Filtering is a frequency selective process that attenuates certain bands of frequencies while passing others.

Digital filters generally come in two flavors: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. Each one can implement a filter that passes or rejects bands of frequencies, but the mathematics and implementations differ significantly. A FIR filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to IIR filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).

FIR filters have a very useful property: they can exhibit linear phase shift for all frequencies. This means that the time-relation between all frequencies of the input signal is undisturbed; only the relative amplitudes are affected [Wav].

The basic characteristics of Finite Impulse Response (FIR) filters are:

- Linear phase characteristic;
- High filter order (more complex circuits);
- Stability;

The basic characteristics of Infinite Impulse Response (IIR) are:

- Non-linear phase characteristic;
- Low filter order (less complex circuits)
- Resulting digital filter has the potential to become unstable;

2.2.3.1 High-Pass, Band-Pass and Band-Pass Filters

A low-pass filter is a filter that passes low-frequency signals and attenuates (reduces the amplitude of) signals with frequencies higher; a high-pass filter is an electronic filter that passes high-frequency signals but attenuates (reduces the amplitude of) signals with frequencies lower; a band-pass filter is a device that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range. This filter can also be created by combining a low-pass filter with a high-pass filter.

These types of filters are very useful when we are working with accelerometer and gyroscope. The Android API advises users, to when they are using the accelerometer, they implement low-pass and high-pass filters to eliminate gravitational forces and reduce noise, respectively [[Andb](#)].

2.2.4 Kalman filter

The Kalman filter is a mathematical method invented by Dr. Rudolf E. Kalman. The most well-known application where it is used is the GPS receiver itself and later, the integration of GPS with the inertial navigation system (INS). This recursive digital algorithm is used to integrate or fuse GPS measurement with accelerometer and gyroscope data to achieve optimal overall system performance [[Tec](#)].

The Kalman filter algorithm produces estimates of the true values of sensor measurements and their associated calculated values by predicting a value, estimating the uncertainty of the predicted value, and computing a weighted average of the predicted value and the measured value. The most weight is given to the value with the least uncertainty. The estimates produced by the algorithm tend to be closer to the true values than the original measurements because the weighted average has a better estimated uncertainty than either of the values that went into the weighted average [[Tec](#)]. Kalman filter can be considered as core to the sensor fusion scheme.

2.2.5 Sliding windows

Raw data usually need to be pre-processed, in order to be possible we extract its features. Thus, the accelerometer raw data and the gyroscope raw data need to be divided in windows, using sliding windows methods.

A sliding windows [[BZ](#)] model is a case of the streaming model, where only the most recent elements remain active and the rest are discarded in a stream. It reduce the memory usage because it only saves the window with the data.

There are two main types of sliding windows, such as fixed-size windows and bursty windows. The first, also known as sequence bases windows, define a fixed amount of the most recent elements to be active. Fixed-size windows are important for application where the arrival rate of the data is fixed, such as stock market measurements or sensors. The other type, also known as timestamp-based windows the validity of an element is defined by an additional parameter such as a timestamp. This is important for application with asynchronous data arrivals, such as networking or database applications.

There are several algorithms that differ in the storage of the data. For instance, an algorithm that can be used with a sequence-based window is to maintain a reservoir sample for the first data elements in the stream and to stop processing these. When a new element arrives, it causes an element in the sample to expire. The expired element is replaced with the newly-arrived element.

In the case of timestamp-based, an algorithm used by several authors is the “priority sample” [[GL08](#)]. As each date element arrives is randomly assigned a priority which can vary between 0 and 1. The element with higher priority and non-expired is included in the sample.

In the different studies presented in subchapter 2.4, where the authors work with accelerometers and gyroscope signals, they usually use algorithms fixed-size windows type. The windows size, that they define, varies with the sensors characteristics and with the processing methods that they use.

2.3 Machine Learning

After extracting signal features, it's necessary to classify those data and we get conclusions. The classification problem can be approached like a recognition problem.

According to Chen and Nugent [LC09], recognition algorithms can be divided into two major stands, such as the use of machine learning techniques based on probabilistic and statistical reasoning and the use logical modelling and reasoning. Wide [Wil10] went further than, stated that the majority of the literature surveyed utilized machine learning techniques. Its major strength is that they are capable of handling noisy, uncertain and incomplete sensor data. Machine learning techniques include supervised and unsupervised learning methods.

2.3.1 Supervised learning

Supervised learning requires the use of labelled data upon which an algorithm is trained. After this, it is then able to classify unknown data. A supervised learning algorithm has the following steps [LC09]:

1. To acquire sensor data representative of activities;
2. To determine the input data features and its representation;
3. To aggregate data from multiple data sources and transform them into the application-dependent features;
4. To divide the data into training set and a test set;
5. To train the recognition algorithms on the training set;
6. To test the classification performance of the trained algorithm on the test set;
7. To apply the algorithm;

There are a wide range of algorithms and models for supervised learning. These methods are Data Mining methods, such as Hidden Markov Models (HMMs, used in [JLB06]), naive Bayes networks (used in [dS13, NRL05]), decision trees (used in [JRK11, dS13, GP12, NRL05]), K-Nearest Neighbours (K-NN, used in [NRL05]) and Support Vector Machines (SVM, used in [NRL05]).

2.3.2 Unsupervised learning

Unsupervised learning tries to directly construct recognition models from unlabeled data. According to Chen and Nugent [LC09], the basic idea is to manually assign a probability to each possible activity and to predefine a stochastic model that can update these likelihoods according to new observations and to the known state of the system. Such an approach uses density estimation methods to discover groups of similar examples in order to create learning models (used in [CC11]). The general procedure for unsupervised learning includes:

1. To acquire unlabeled sensor data;
2. To aggregate and transforming the sensor data into features;
3. To model the data using either density estimation or clustering methods;

The main difference between unsupervised and supervised probabilistic techniques is that, while the unsupervised probabilistic techniques use a pre-established stochastic model to update the activity likelihood, supervised learning algorithms keep a trace of their previous observed experiences and use them to dynamically learn the parameters of the stochastic activity models [LC09].

2.4 Related work

These two sensors (accelerometer and gyroscope) are used to give a wide range of new functionalities. The most common are:

- Change/Adjust the screen for a proper viewing;
- Control the direction in games (racing cars);
- Hang up a call;
- Advance to the next song;

There are others functionalities that have been studied and applied [Goo]. An example is activity recognition using cell phone accelerometers [JRK11, dS13]. Such studies are significant because the activity recognition model permits us to gain useful knowledge about the habits of millions of users passively. This type the work has a wide range of applications such as determining whether the user is getting an adequate amount of exercise and estimate the number of daily calories expended, automatic customization of the mobile device's behavior based upon a user's activity (sending calls directly to voicemail if a user is jogging) or generating a activity profile to determine if the user is performing a healthy amount of exercise.

Kwapisz, Weiss and Moore in "Activity Recognition using Cell Phone Accelerometers" [JRK11] describe and evaluate a system that uses phone-base accelerometers to perform activity recognition. They collected labeled accelerometer data from twenty-nine users as they performed daily

activities such as walking, jogging, climbing, stairs, sitting and standing, and then aggregated this time series data into example that summarize the user activity over 10-second interval. After preparing the data set (determining for each case the average, the standard deviation, the average absolute difference, average resultant acceleration, the time between peaks and the binned distribution) they used three classification techniques from the WEKA data mining suite [WF] to induce models for predicting the user activities: decision trees (J48), logistic regression and multilayer neural network. They concluded that activity recognition can be highly accurate, with most activities being recognized correctly over 90% of the time.

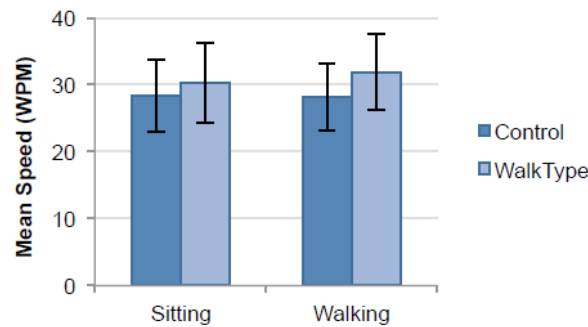


Figure 2.3: WalkType resulted in higher typing speeds than the control condition, particularly while participants were walking [MGW].

In 2012, Mayank Goel, Leah Findlater and Jacob O. Wobbrock [MGW] created WalkType, an adaptive system for mobile touch screen device that leverages the on-device accelerometer to compensate for vibrations and extraneous movements caused by walking. They performed two studies with 16 participants each. One to collect the data for WalkType’s model and another to evaluate the generated models. The results were positive. WalkType increases users’ typing speeds from 28.3 WPM to 31.3 WPM (Figure 2.3), and also reduces the number of uncorrected errors from 10.5% to 5.8% while participants are walking (Figure 2.4).

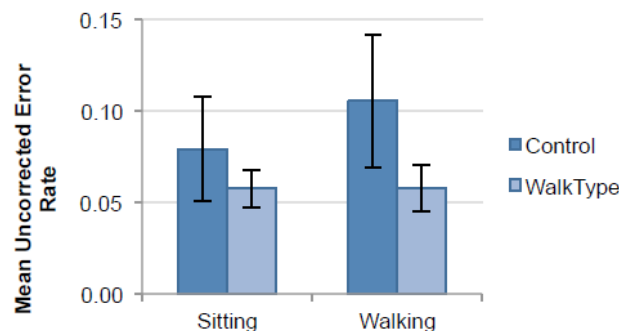


Figure 2.4: WalkType resulted in lower error rates than Control, especially for walking [MGW].

A little earlier, in 2011, Liang Cai and Hao Chen already had conducted studies in this area. In this case they used the smartphone motion to inferring keystrokes on touch screen [CC11]. To this end, they developed TouchLogger (Figure 2.5), an Android application that extracts features (through the accelerometer and gyroscope) from device orientation data to infer keystrokes. When the user types on the soft keyboard on his smartphone (especially when he holds his phone by hand rather than placing it on a fixed surface), the phone vibrates. They discover that keystroke vibration on touch screens are highly correlated to the keys being typed. In our preliminary evaluation, they were able to infer correctly more than 70% of the keys typed on a number-only soft keyboard on a smartphone. In this way they have demonstrated that motion is a significant side channel, which may leak confidential informations on smartphones.

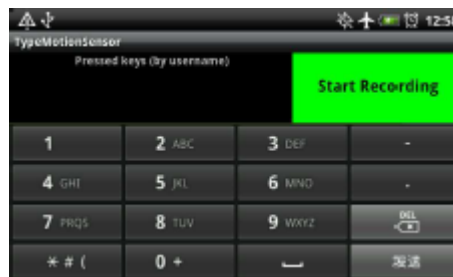


Figure 2.5: User interface (TouchLogger) [CC11].

Other applications of these two sensors in smartphones are to infer hand postures and pressure. A paper produced by Mayank Goel, Jacob O. Wobbrock and Shwetak N. Patel [GP12] describes a system that uses a combination of the touchscreen and the built-in inertial sensors (gyroscope, accelerometer) and built-in actuators (vibration motors) already present on most commodity mobile phones to infer hand postures and pressure (GripSense). It infers postures like the use of an index finger, left thumb, right thumb, which hand is holding the device, or whether the phone is lying on a flat surface. GripSense differentiates between device usage in-hand or on a flat surface with 99.7% accuracy and various hand postures with 84.3% accuracy and, on average, makes a decision within 5 “interaction steps”. GripSense differentiates between three levels of pressure on different areas of the device with 95.1% accuracy.

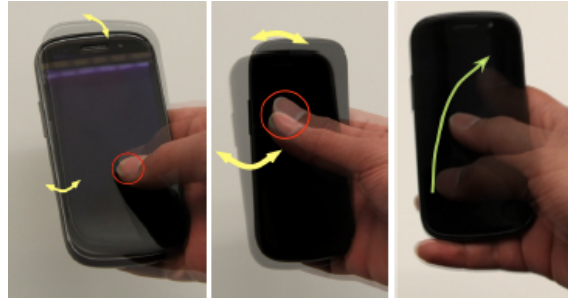


Figure 2.6: **(left)** Minimal device rotation in x- and y-axis, and smaller touch size when the user touches nearby with the thumb. **(center)** Significantly more rotation in x- and y-axis and larger touch size when the far quadrant of the screen is touched. **(right)** The shape of the swipe arc in the case of right thumb [GP12].

To infer the user's hand posture was used a combination of tree features: relative variance in rotation (rotational movement of the device as the user touches the screen), change in touch size (change of size of touch in different regions of the touch screen) and direction of arc for finger swipes [KS05] (an exaggerated arc that the users often draw because of the shape and position of the thumb). These three features can be observed in Figure 2.6.

In the case of the pressure applied to the touchscreen the GripSense uses the gyroscope and vibration motor to classify into three pressure categories: light, medium and heavy. When they trigger the built-in vibration motor when a user touches the screen, the user's hand absorbs a portion of these vibrations. Their experiments show that this vibration absorption is proportional to the amount of pressure being applied to the screen. This damping affect is measured using the on-device gyroscope. They also found that the size of touch and the location of touch on the screen are feature for pressure level classification. Thus, the gyroscope data passes through low pass and high pass filters and appropriate variances and 90th-percentiles are calculated. These features, along with touchscreen features (zone and size), were used to classify to pressure level using the Weka machine learning toolkit (J48 Decision Trees) (Figure 2.7).

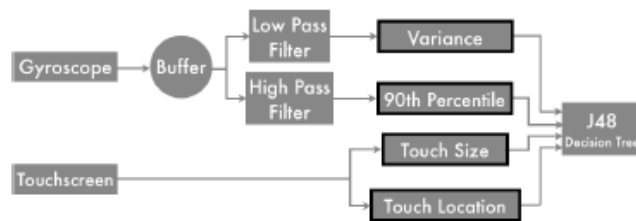


Figure 2.7: Block diagrama of the major components of GripSense's pressure detection module [GP12].

The Table 2.4 resumes the process of detecting hand position and pressure previously presented.

Table 2.4: Summary of all inferences made by GripSense and when and which features were used for each of them [GP12]

Inference	Features Used	Sensor Event
Table vs. Hand	Gyroscope (low frequency in all axes)	Touch Down
Thumb vs Index Finger	Gyroscope (low frequency in x- and y axis)	Touch Down
	Swipe shape	Touch Up
	Touch size	Touch Down
Left Thumb vs. Right Thumb	Gyroscope (Low frequency in y-axis)	Touch Down
	Swipe Share	Touch Up
	Touch Size	Touch Down
Pressure in hand	Gyroscope (Low Frequency)	*Touch Down
	Gyroscope (High Frequency) + Motor	
Pressure on table	Gyroscope (High Frequency) + Motor	Touch Down
Squeeze	Gyroscope (High Frequency) + Motor	Held in Hand

Other relevant scientific paper for this dissertation is “Sensor Synaesthesia: Touch in Motion, and Motion in Touch” [HS11] written by Ken Hinckley and Hyunyoung Song. They explored techniques for hand-devices that leverage the multimodal combination of touch and motion. One of the aspects which they explored was the use of the accelerometer, gyroscope and touch screen in order to they complement themselves. They concluded that the gyroscope and accelerometer signal can be used to localize “touch” interactions to some degree, without any true touch sensing being used at all.

2.5 Used Technologies

2.5.1 Android

Android is an operating system which powers more than a billion phones and tablets around the world. It is based on the Linux Kernel with a user interface based on direct manipulation, designed primarily for touchscreen mobile devices such as smartphones and tablet computer. The Android SDK provides the tools and APIs necessary to develop applications on the Android platform using the Java programming language.

Android’s source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software.

Android quickly reached the top of the smartphone world. Currently, it is the operator system for mobile devices with the highest market Share (78.9%) [IDC]. With a strong presence within emerging markets and attainable price points for both vendors and customers, IDC (International Data Corporation) expects both a commanding market share as well as prices below the industry average.

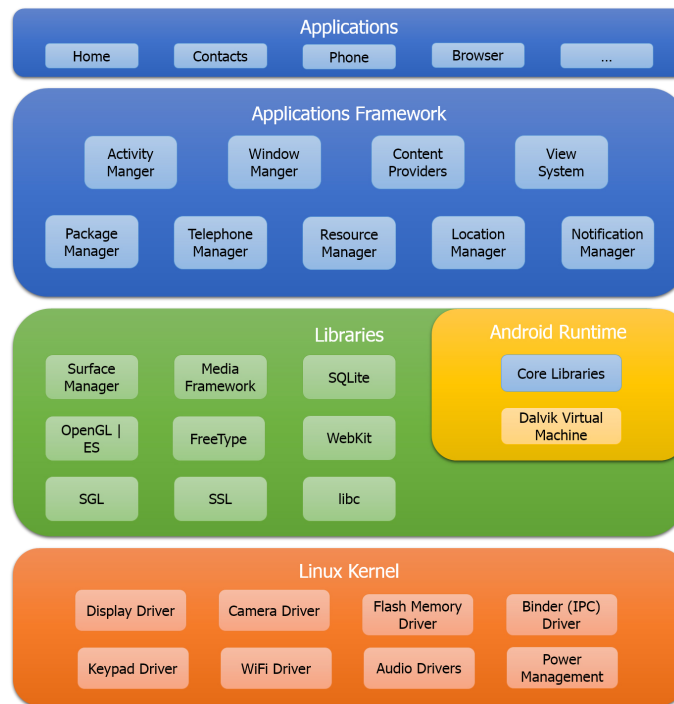


Figure 2.8: Android components [Kel]

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more. The following diagram – 2.8 - shows the major components of the Android operating system.

2.5.2 Weka

Weka is a collection of machine learning algorithms for data mining tasks. Its algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU General Public License.

Weka's main user interface (Figure 2.9) is the *Explorer*, but essentially the same functionality can be accessed through the component-based *Knowledge Flow interface* and from the command line. There is also the *Experimenter*, which allows the systematic comparison of the predictive performance of Weka's machine learning algorithms on a collection of datasets.

State of the art

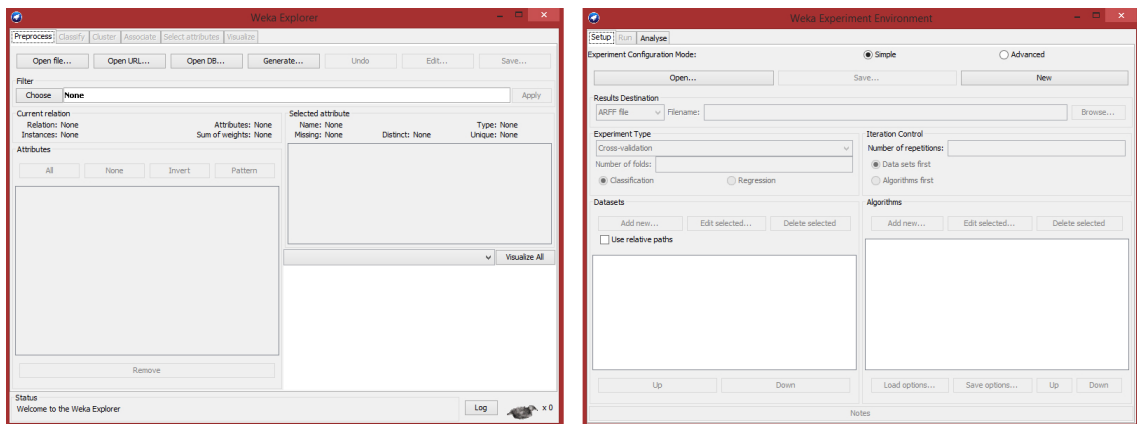


Figure 2.9: Weka's interface.

All of Weka's techniques are predicated on the assumption that the data is available as a single flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Usually this files are ARFF (Attribute-Relation File Format) files.

2.5.2.1 ARFF files

An ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes.

ARFF files have two distinct sections. The first section is the Header information, which is followed the Data information.

The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. An example header on the standard IRIS dataset looks like this:

```
1  @RELATION iris
2
3  @ATTRIBUTE attribute_1  NUMERIC
4  @ATTRIBUTE attribute_2  NUMERIC
5  @ATTRIBUTE attribute_3  NUMERIC
6  @ATTRIBUTE attribute_4  NUMERIC
7  @ATTRIBUTE class        {class_1,class_2,class_3}
```

The attributes can be any of the four types supported by Weka:

- *numeric*
- *integer* is treated as *numeric*
- *real* is treated as *numeric*
- *<nominal-specification>*

- *string*
- date [<date-format>]

where <nominal-specification> are defined listing the possible values: <nominal-name1>, <nominal-name2>, <nominal-name3>,

The Data of the ARFF file looks like the following:

```

1  @DATA
2  5.1,3.5,1.4,0.2,class_1
3  4.9,3.0,1.4,0.2,class_2
4  4.7,3.2,1.3,0.2,class_1
5  4.6,3.1,1.5,0.2,class_3
6  5.0,3.6,1.4,0.2,class_4
7  5.4,3.9,1.7,0.4,class_3

```

2.6 Conclusion

The number of sensors embedded on smartphone keeps increasing, because they are what make the smartphones so attractive. There are many studies about how they can improvement the interaction between humans and smartphones or even our lives.

In this charter we studied different applications of accelerometers and gyroscopes when embedded in smartphones, including to recognize physical activities, to correct keystrokes, to detect the position of a touch on the smartphone's screen, and more.

Other subject that we analyzed was how to work the signal after we obtain it. We present different sliding windows approaches like fixed-size windows and bursty windows, and the different signal domains. The most important techniques of processing signal in sensors field were also analyzed, such as fast Fourier transform, digital filters and Kalman filter. Lastly we showed some classification methods used to we get conclusions.

Chapter 3

Implementation

This chapter describes the implementation of the Android Service, which can be integrated with several applications. Its objective is provide new ways to characterize the touch.

First, it will be described the process of creating algorithms and how they were implemented. Thereafter, it will be presented and explained the architecture service and how the different layers interact. Issues such as the window size and the frequency of touch detection will also be discussed.

3.1 Algorithms

In order to be possible to detect and characterize the touch several algorithms were developed with different functionalities, such as:

1. Detecting where the smartphone is (on surface, in hand or moving);
2. Detecting touch using the accelerometer and the gyroscope;
3. Assess which the hand (right or left) is holding the device;
4. Assess which finger (index finger or thumb) is being used to touch on smartphone;
5. Calculate the impact force of the touch event;

The Figure [3.1](#) summarizes the information flow and how the different algorithms interact.

Implementation

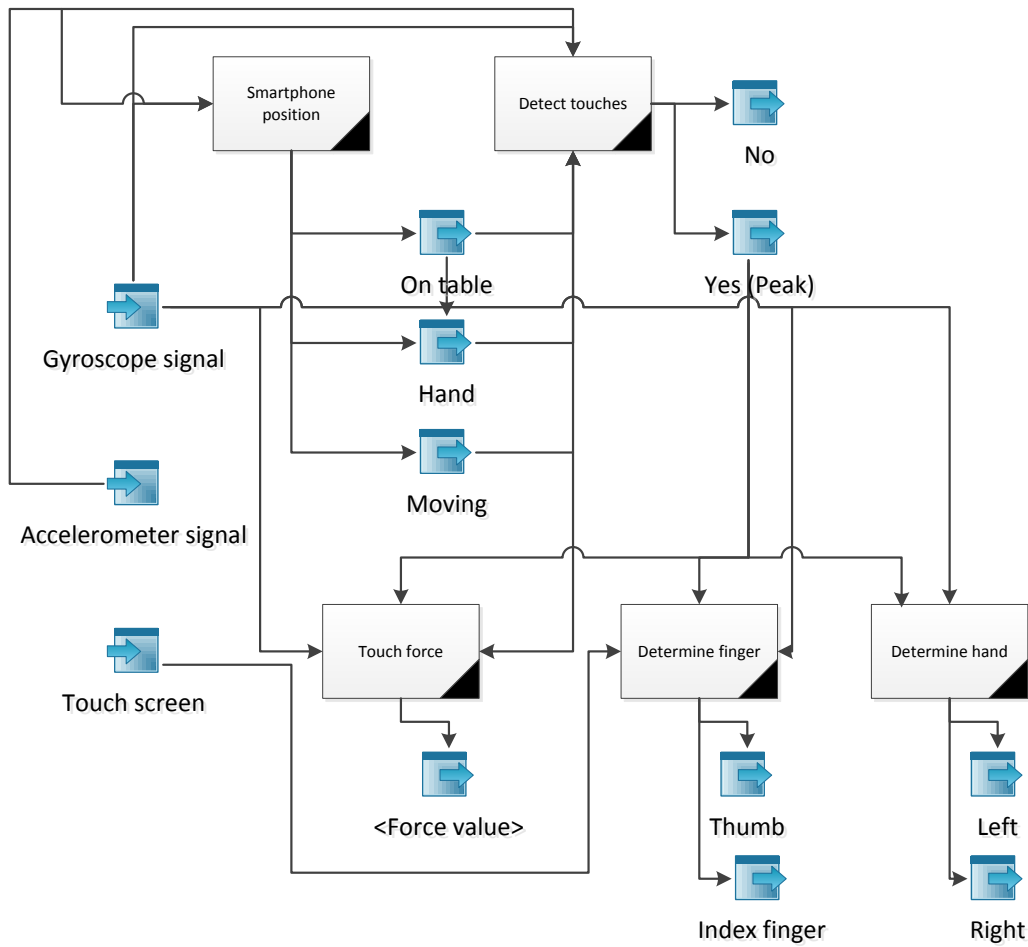


Figure 3.1: Information flow.

3.1.1 Smartphone position

The first algorithm assesses how the smartphone is being used, in other words, it detects if the device is on surface, in user hand or moving. The output algorithm is: ON_TABLE (the smartphone is on surface), HAND (the user handhold the smartphone) or MOVING (the smartphone is moving). This kind of information was very useful because it gave us information about the motion of smartphone, which was used in other algorithms.

As we were working with motion sensors (accelerometer and gyroscope), it was decided to use the smartphone stability to determine how the smartphone is being used. If the stability is high, we can assume that it is being used on surface; if it is in between, the smartphone is in user hand; if it is low, the smartphone is moving. But this approach has one problem: if the user is holding the smartphone with one hand, but the hand is very stable (e.g. the hand is on table) the smartphone stability will be very high. To solve this, the algorithm was divided into two phases: the first determines the instantaneous position and the second phase determines the real position.

It is possible to determine the stability using metrics like mean and standard deviation.

3.1.1.1 Instantaneous position

In this phase the algorithm indicates how the smartphone is being used in each iteration.

In order to create an algorithm to detect the instantaneous position, the different users' data was collected and different approaches were tested, with the purpose of determining the approach with the best results. For that, the application used to evaluate and validate the algorithms (Charter 5) and the Weka were used.

During the several tests it was noticed that the gyroscope is most suitable to detect small movements (e.g. hand vibrations) than the accelerometer. This way, the used metrics to create the ARFF files (files used by WEKA) were the average and the standard derivation of each gyroscope axis, the average and the standard derivation of SMV gyroscope value and the SMA gyroscope value. The Table 3.1 shows the different approaches (data mining techniques) and their results.

Data mining techniques	Results (%)
Best First Tree (BFTree)	98.9
REPTree (Fast Decision Tree Learner)	99.1
J48 Decision Tree	98.8
Hoeffding Tree	95.7
NaiveBayes	96.2

Table 3.1: Data mining techniques and their accuracy without data on the training sets

Although there are three techniques (DFTree, REPTree and J48) with good results, the REPTree (Appendix A) was chosen because it had the smallest size.

3.1.1.2 Real position

Even holding the smartphone in one hand is possible to increase its stability (e.g. leaning against the hand to anything) and this way to simulate that the smartphone is on surface. To prevent this, an heuristic was created: "To change the position of the smartphone, there has to be movement". In other words, whenever we use a smartphone in the hand and we want to put it on table, or vice versa, we have to move it. As the first phase gives us the instantaneous position, this movement is possible to detect. The Figure 3.2 resumes how this phase works.

Implementation

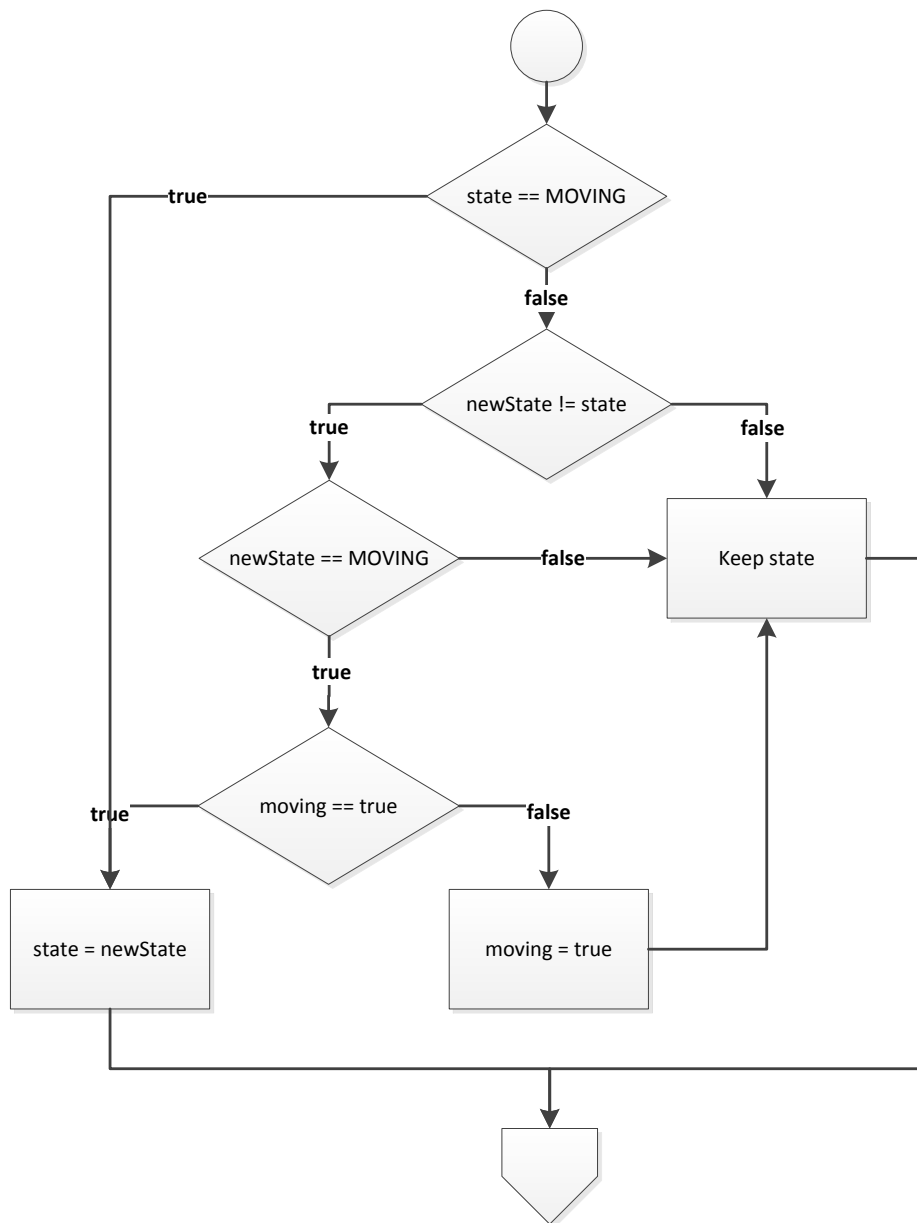


Figure 3.2: Determine the correct position (state: current position; newState: instantaneous position).

3.1.2 Detect touch

The goal of this algorithm is to detect touches using the accelerometer and gyroscope. That is possible because when someone touches on smartphone, a small variation signal occurs (Figure 3.3). It is a variation peak, which can be detected using any algorithm for peak detection in time-series.

Implementation

The algorithm chosen was an algorithm presented by Girish Palshikar [Pal09] because it is very suitable for this type of data. Furthermore it has good performance and good results. The algorithm is:

```

input  $T = x_1, x_2, \dots, x_n, N$     //input time-series of  $N$  points
input  $k$     // window size around the peak
input  $h$     // typically  $1 \leq h \leq 3$ 
output  $O$     // set of peaks detected in  $T$ 
begin
  PHASE I
  |    $O = \emptyset$     // initially empty
  |   for ( $i = 1; i < n; i++$ ) do
  |        $a[i] = S(k, i, x_i, T);$     // compute peak function value for each of the  $N$  points in  $T$ 
  |   end for
  PHASE II
  |   Compute the mean  $m$  and standard deviation  $s$  of all positive values in array  $a$ ;
  |   for ( $i = 1; i < n; i++$ ) do    // remove local peaks which are "small" in global context
  |       if ( $a[i] > 0 \ \&\& \ (a[i] - m') > (h * s')$ ) then  $O = O \cup x_i$ ; end if
  |   end for
  PHASE III
  |   Order peaks in  $O$  in terms of increasing index in  $T$ 
  |   // retain only one peak out of any set of peaks within distance  $k$  of each other
  |   for every adjacent pair of peaks  $x_i$  and  $x_j$  in  $O$  do
  |       if  $|j - i| \leq k$  then remove the smaller value of  $\{x_i, x_j\}$  from  $O$  end if
  |   end for
end

```

This algorithm has three parameters: the data set to analyze (T), the number of neighbors around each point which will be analyzed (k) and a constant (h) used to calibrate the algorithm (the highest value, the highest the average amplitudes of the peaks (O)). It can be divided into three phase:

1. The algorithm creates a set with the average of the maximum among the signed distances of x_i from its k left neighbors and the maximum among the signed distances of x_i from its k right neighbors for each point.

$$S(k, i, x_i, T) = \frac{\max\{x_i - x_{i-1}, \dots, x_i - x_{i-k}\} + \max\{x_i - x_{i+1}, \dots, x_i - x_{i+k}\}}{2} \quad (3.1)$$

2. It filters the points which are real peaks.

Implementation

3. The algorithm retains only one peak out of any set of peaks within distance k of each other.

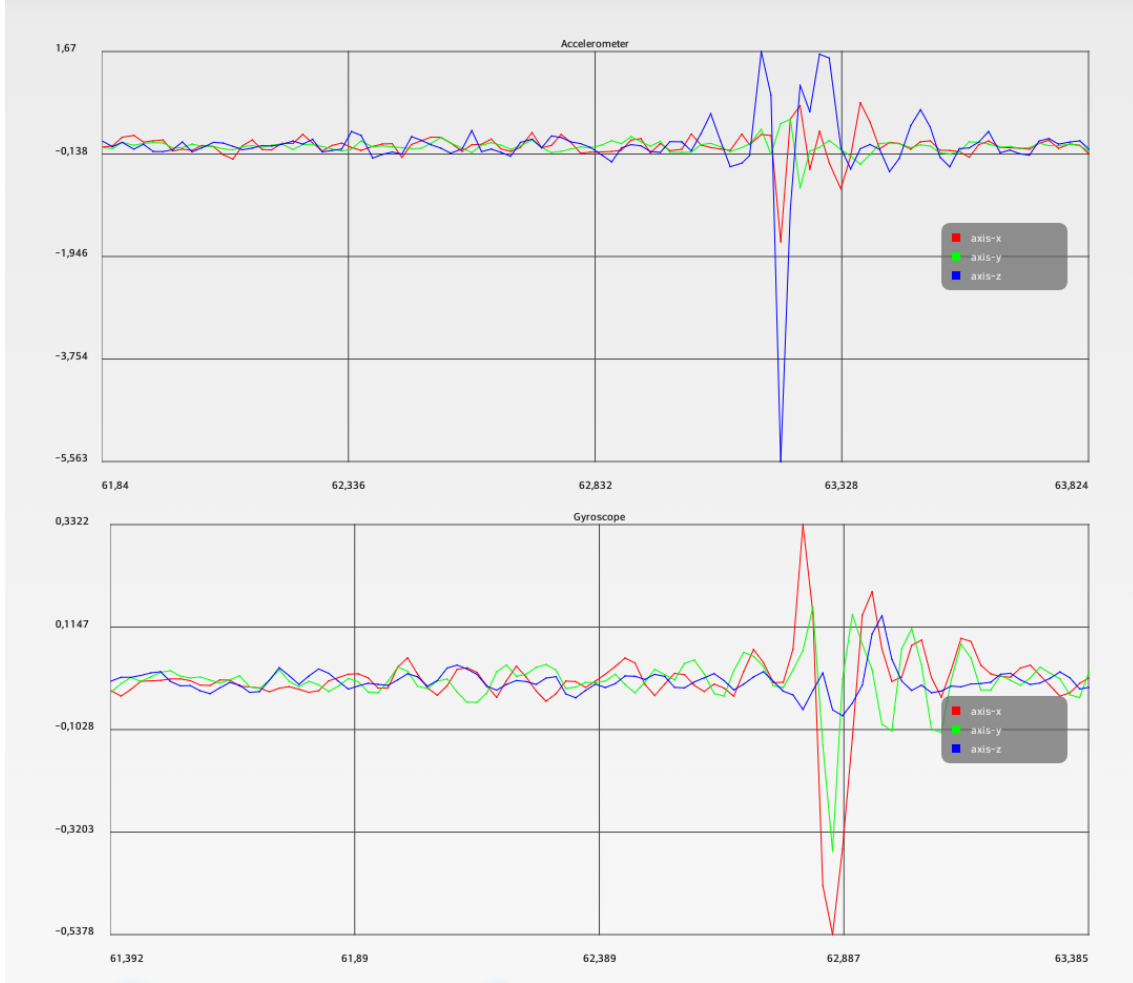


Figure 3.3: Accelerometer and gyroscope signals of a touch.

After we analyzed different sensor signals, we concluded that the sensor signal vary with the smartphone position, the smartphone shape and size, the touch position and the touch force. Different approaches were tested and it was concluded that when the smartphone is on surface the y-axis gyroscope value ($k = 5$ and $h = 1.5$) is more sensible to the touch and when the smartphone is in user hand to use the z-axis accelerometer value ($k = 5$ and $h = 1.5$) presents the best results.

One problem of this approach is the false positives. Some movements (e.g. small hand vibrations) produce peak variations very similar with the peak variations produced by touches. It is possible to decrease its number practically to zero, but the algorithm performance also decrease. This way, to solve this problem, the algorithm optionally can use the touch screen to check is the touch really happened. For that the algorithm used the public `dispatchTouchEvent(MotionEvent event)` method, which detects all touches on touch screen. This option is useful because the output from this algorithm is used in the next algorithms, and the false positives may decrease their performance.

Implementation

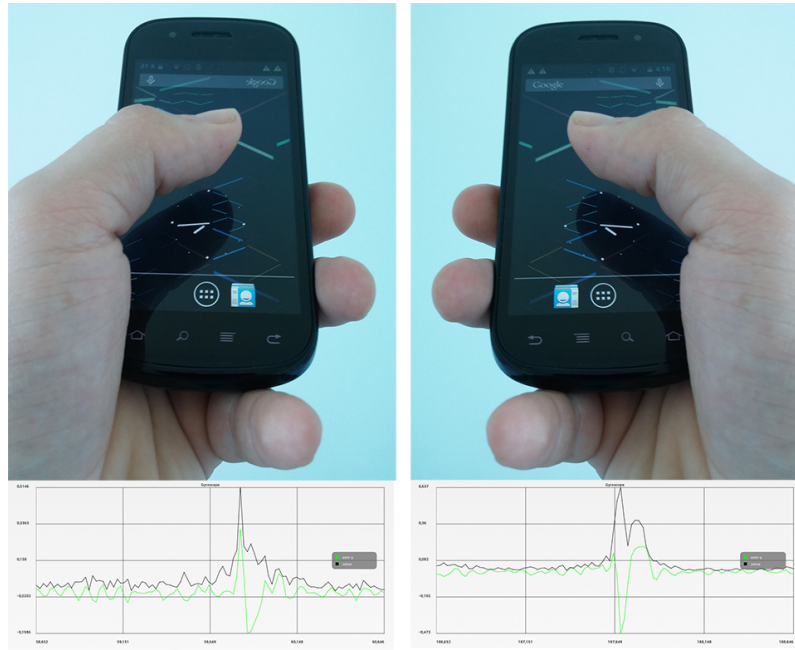


Figure 3.4: Gyroscope signal when someone touches on smartphone (left/right).

3.1.3 Hand (right/left)

Determining which hand the user is using to hold the smartphone is very useful to define user patterns. This algorithm does that; it indicates if the user is using the right hand or the left hand to hold the smartphone. To understand how this algorithm works it is necessary to understand the hand behavior when the touch occurs. The human hand does not have the stability of a table and therefore it always has small motions. When the user touches on smartphone there are small movements which vary from right hand to left hand. One of these movements is a small downward movement of the fingers which hold the smartphone. When someone touches on the smartphone, some force is exerted on the smartphone. This force causes a small move in the same direction force on “the weakest” part of the hand (fingers). If the hand is the right, the smartphone has a small rotation in the opposite direction of the clock; if it is the left the rotation in the direction of the clock (Figure 3.4).

This algorithm was inspired for this movement that can be detected using the cross-correlation (it is a measure of similarity of two waveforms) between the y-axis gyroscope value and the SMV (signal magnitude vector) gyroscope value (Figure 3.4). In order to classify the hand the Weka was used and the results of the different techniques can be seen in the Table 3.2. The results show that the data mining technique with the best results is the J48 Decision Tree. Its results (Figure 3.5) confirms the prediction: if the cross-correlation between the y-axis gyroscope value and SMV gyroscope value is bigger than zero, the smartphone is in the left hand, if it is smaller or equals than zero, the hand is the right.

Implementation

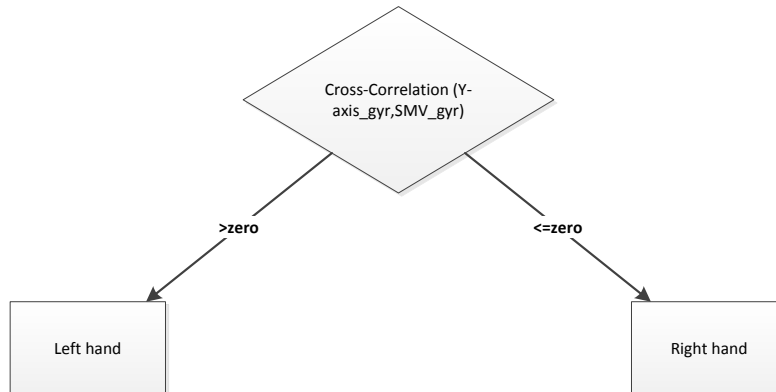


Figure 3.5: Decision tree produced by J48 method

Data mining techniques	Results (%)
Best First Tree (BFTree)	71.2
REPTree (Fast Decision Tree Learner)	73.5
J48 Decision Tree	75.9
Hoeffding Tree	70.5
NaiveBayes	70.32

Table 3.2: Data mining techniques and their accuracy without data on the training sets (hand detection)

To increase the precision value, the algorithm adds the decision tree value to the information of the last touches and considering the last four and the current value the algorithm calculates the belief of each hand. The algorithm's return is the hand with the highest belief value.

3.1.4 Touch force

If the touch pressure value given by Android API (`MotionEvent.getPressure(int)`) is considered, we can conclude this value is proportional to the touch size (`MotionEvent.getSize()`). For instance, if someone double taps on smartphone's touch screen with the same force but with the different touch size (touch area) the Android API considers the touch with the biggest area as the strongest.

This algorithm uses the gyroscope signal and it determines the force of each touch. Strong touches present variation peaks with more amplitude than the weaker touches (Figure 3.6).

This approach works really well when the smartphone is on the user hands, but when a smartphone is on a surface (e.g. a table) there are some problems related with the smartphone shape. For example, if the back of the smartphone is very curve, the gyroscope signal will have a big

Implementation

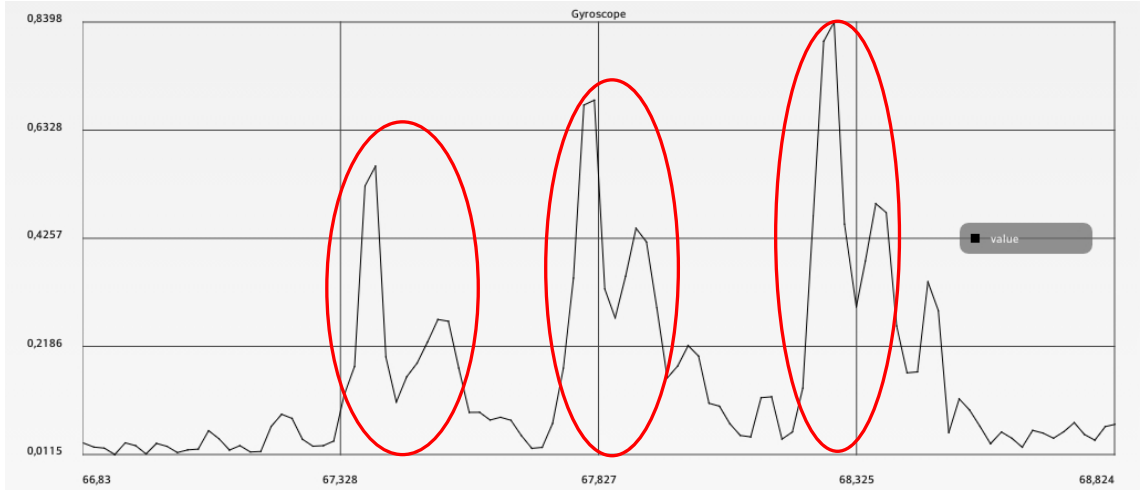


Figure 3.6: SMV signal when someone touches on the smartphone three times with increasing forces.

variation; if it is very plane, the variation of the gyroscope signal will be very small. This way, when the smartphone is in the user hand the touch force is the SMV gyroscope value of the variation peak. When the smartphone is on surface the algorithm processes the SMV gyroscope value in order to normalize it.

In order to normalize the signal, it was necessary to apply the different forces the different touch screen places using the device on table and to repeat the process, but using the device in user hand. For this, a ball was dropped with the same distance between it and the device to apply equal forces and different distance to apply different forces. Thereafter the resultant values for the same place was compared and using mathematical techniques two heuristics was obtained:

$$Force = 8.831 * Value^{0.731} \quad (3.2)$$

$$Force = 0.4352 * \ln(Value) + 1.396; \quad (3.3)$$

where *Force* is the normalized force and *Value* is the value obtained with the device on surface. The first equation (3.2) is used when the touch is in place very plane (it enlarges the value); the second equation (3.3) is used when the touch is in place curve.

3.1.5 Finger (thumb/index finger)

Find out which is the finger (thumb or index finger) used to touch on smartphone is daunting task because the users can use the smartphones in several ways. The device can be used in one hand,

Implementation

and in this cases the used finger is the thumb, or with two hands using one hand to hold the smart-phone and another to touch it (in this case the user can use the index finger (the most common) or the thumb) or holding it with two hand (he just uses the thumb). After to be analyzed several approaches with different features it was concluded that the best features which characterize the touch are:

Rotation of the device The first feature is the rotation movement of the device as the user touches the screen. When the user uses the device only with one hand (using the thumb), he produces a device rotation; when the user uses the index finger holding the device with other with another hand the rotation is smaller;

Touch size Other feature very useful is the size of the touch area. The index finger and the thumb have different sizes and consequently, the touch areas are different;

Touch force The last feature is the touch force, which is supplied by the above algorithm. When the user touches the screen twice with the same finger but applying different forces, the biggest area is from the strongest touch and vice-versa. This way, the touch force is used to complement the touch size;

But this approach has a problem. The finger sizes vary between different users. To solve this problem is necessary that the algorithm learns from the user experience. This way using the Weka to analyze the data set (data collect from ten users) and using the J48 method (it present good performance and the smallest size), a decision tree was created. That tree uses the three previous features, and while the user uses the devise, the decision tree adapts to the characteristics of the user. For that the algorithm analyzes the average of the touch size and it changes the node values. The more you interact with your smartphone, the tree is more adequate for the user's usage pattern.

Like the algorithm which determines the hand, this calculates the belief of each finger. Using the last five outputs from the decision tree it determines which has the highest belief and it returns the corresponding finger.

3.2 TouchSensor service

After the algorithms had been created, it is necessary to develop an API that easily availables the new touch data. For that an Android Service was created. It is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC)

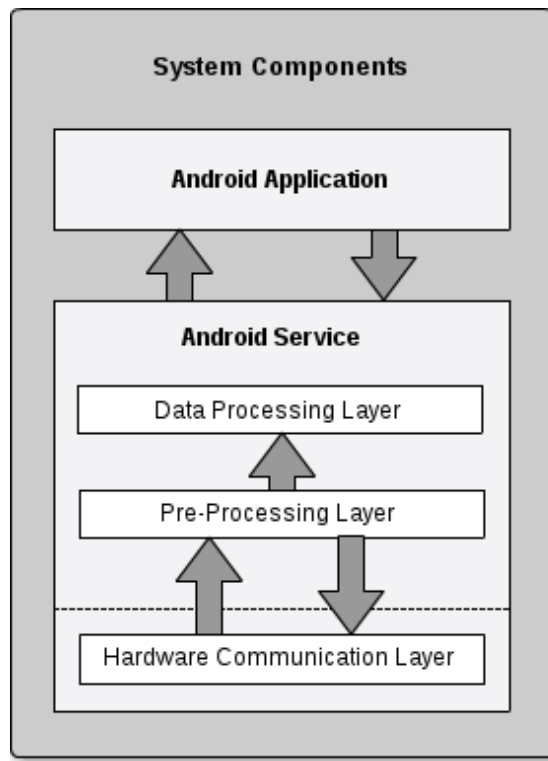


Figure 3.7: System Architecture.

3.2.1 Architecture

The actual solution of this project is based on an Android Service, which is divided into three layers: the hardware communication layer, the pre-processing layer and the data processing layer (Figure 3.7).

The hardware communication layer is responsible for the interaction with the Android API. The communication with sensors and their refresh rates are defined in this layer. Another functionality implemented here is the detection of touch events by the touchscreen, with the purpose of avoiding false positives. When the service is not necessary, the access to the sensors is closed.

The second layer is the preprocessing layer. This is used to apply the filters (low-pass filter and high-pass filter) to the raw data before being processed by the data processing layer. Furthermore, it is in this layer that the data sensors are divided into windows and the window size and the frequency detection are calculated.

The main and the last layer is the data processing layer. It has implemented the different algorithms, which using signal processing techniques, they detect and characterize the touch.

3.2.2 Hardware communication

The device used was an Android-based smartphone. All applications used with the purpose of assisting the realization of this dissertation (to analyze the sensor data and to record the users

Implementation

```
1  sensorManager.registerListener(this, sm.getDefaultSensor(Sensor.TYPE_GYROSCOPE)  
    , SensorManager.SENSOR_DELAY_GAME);  
2  sensorManager.registerListener(this, sm.getDefaultSensor(Sensor.  
    TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_GAME);
```

Listing 3.1: Initialization of the sensor's listeners with the game delay possible.

data) were developed for the Android 4.2.2, API level 17. For the Android development the IDE used was 4.2.0 with the ADT Plug-in, and for testing and signal recording a Google Nexus S smartphone was used.

According to the Android API to identify the sensors that are on a device first it is necessary to get a reference to the sensor service. To do this, it was created an instance of the `SensorManager` class by calling the `getSystemService()` method and passing in the `SENSOR_SERVICE` argument (two listeners, one for each sensor). Another argument is sensor sampling rate. Since different phones have a different sampling rate for their sensors is necessary to choose the best rate for each case. After comparing the different rates from different smartphones was concluded that the most appropriate rate was `SENSOR_DELAY_GAME` (rate suitable for games), because it is the most uniform value in the several smartphones (approximately 100 readings per second) (Listing 3.1).

The Android OS is built so that when a listener for a sensor is registered, every time the values read from that sensor change an event is created. To receive this event there is a method called `onSensorChange`, which is called every time an event originated from a registered sensor is created. Using this method, it is possible to get the sensors values (x-axis value, y-axis value and z-axis value) and their respective timestamps (time in nanoseconds).

As it was explained, one of the problems found on the algorithms development phase was the existence of false positives (the algorithm to detect touches indicated some touches which did not occur). The implemented solution was to use the touchscreen sensor to check if the detected touch by the accelerometer and gyroscope occurred. For that it was necessary to override the method `dispatchTouchEvent(MotionEvent event)`, which according to Android API, detects all touches on the screen, independently of the touch location [Anda].

3.2.3 Sensor data preprocessing

In order to provide correct feedback to the user was necessary to develop algorithms capable of evaluating in real-time the sensor signals. This algorithms should have good performance independently of difference sensor frequencies or external forces. For that it was necessary to work the raw sensor data in order to they were transformed into uniform as possible.

3.2.3.1 Pass Filters

When we are working with sensors like accelerometer or gyroscope, we need to have attention to the signal noise. There can be many different reasons for that, but the main reason identified here

Implementation

was the presence of external forces (e.g. force of gravity). In order to avoid this problem and to unify the signal from different smartphones filters were used (Figure 3.8).

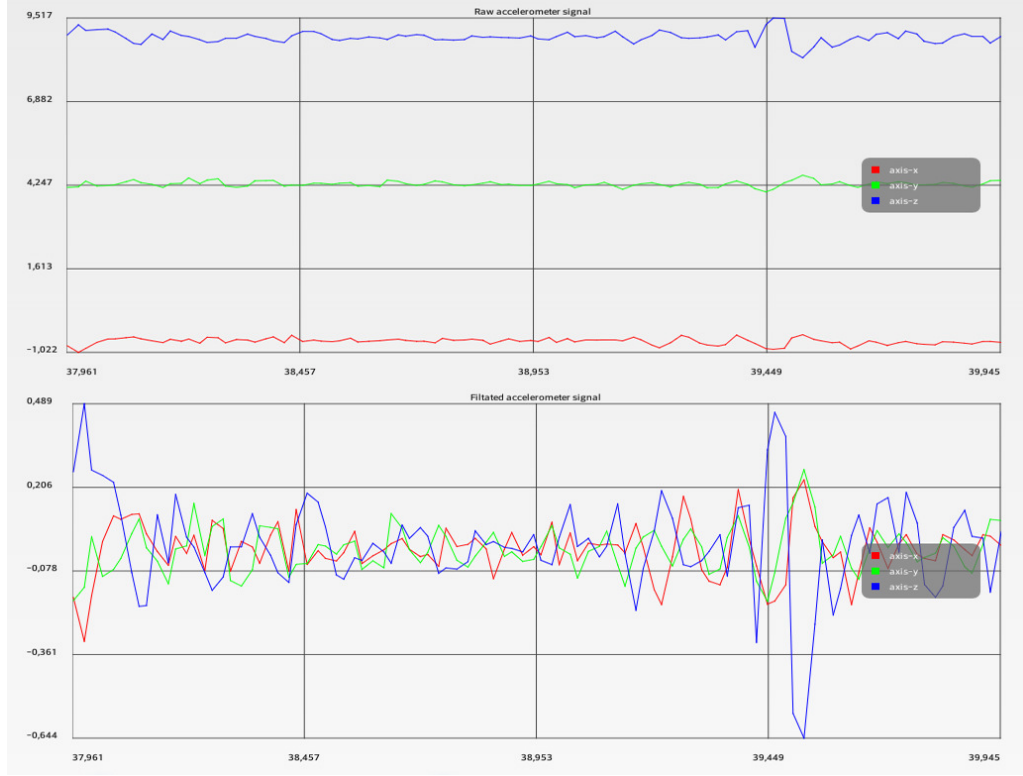


Figure 3.8: **First chart:** Raw accelerometer signal; **Second chart:** Filtrated accelerometer signal. The two charts have different scales.

In this case it was implemented a low-pass filter and a high-pass filter. The first was used to isolate the external force and the second to remove it and smooth the signal. The low pass filter value can be calculated using the formula:

$$newValue = \alpha * lastValue + (1 - \alpha) * sensorValue \quad (3.4)$$

Where *lastValue* is the last calculated value, *sensorValue* is the raw sensor value and α varies between zero and one and is calculated as follows:

$$\alpha = t / (t + \Delta T) \quad (3.5)$$

where t is the low-pass filter's time constant and ΔT is the sensor delivery rate. To calculate the high-pass filter value was used the value from low-pass filter (*lowvalue*) and the last high-pass filter value (*lastvalue*):

Implementation

$$newValue = lastValue - lowValue \quad (3.6)$$

3.2.3.2 Window size/Frequency of touch detection

When we are working in real-time it is necessary to pay attention to the response time (time delay). We have to play with this value bearing in mind that the smaller the time delay is the better and that the response precision reduces when the speed increases. Another problem is due to different frequencies between sensors from different smartphones. After we analyzed this two problems, we could conclude that they are the same problem: how the sensor frequency influences the response time, in other words, how the window size is calculated.

To solve this problem the sensor data were divided into two windows with different sizes:

Window to analyse: This window contains the data which will be analysed in the iteration. Its size varies with the sensor frequency and to define the number of readings per iteration;

Window with last data: It contains the last analysed data and the data to analyse (the last sixty sensor values). Its size is static;

All the process is divided into interactions. An interaction is composed of reading and loading (for the smallest window) of data and its analysis. As we were working with two sensors at same time, the iteration period (the time between each iteration) the size of window with the new data are directly proportional to frequency of the sensor with more frequency. Thus, working with two windows was possible to decrease the time delay and to keep good results. The smaller windows is used to keep the data which will be analyzed (new data) and the larger window is used to understand the last behaviors and this way to keep stored uniforms metrics, which will be used to analyze the data from other window. This is necessary because the metrics like mean and standard deviation vary with the sensor sensibility. The Figure 3.9 shows all the process.

Implementation

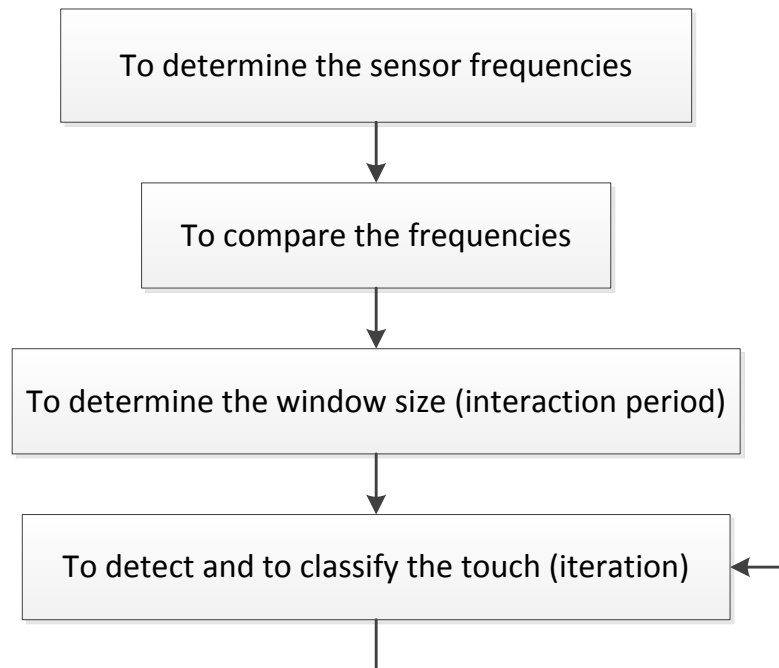


Figure 3.9: Determine Window Size.

3.2.4 Data processing

After being launched, the service analyses the sensor signals and when it detects some touch, it sends to main process their characteristics. First it determines the device position (this information is used to detect touches and to calculate the touch force) and it checks if some touch occurs. If the return is positive the service calculates the touch force and it determines which the hand is (only if the device is in user hand) and which the finger is.

The output from some algorithms is used by other algorithms. The Table [3.3](#) resumes the parameters and the return values for each algorithm.

Implementation

Algorithms	Input	Output
Smartphone position	Gyroscope signal Accelerometer signal	"ON_TABLE" "HAND" "MOVING"
Detect touches	Gyroscope signal Accelerometer signal Smartphone position	No Yes (peak indice)
Touch Force	Peak (detect touches) Gyroscope signal Peak (detect touches)	<Force value>
Determine finger	Gyroscope signal Touch screen	"INDEX_FINGER" "THUMB"
Determine hand	Gyroscope signal	"LEFT" "RIGHT"

Table 3.3: Input and output for each algorithm.

Chapter 4

Validation and Evaluation

On this section, both evaluation methodology and results are detailed. Analyzing these results it is possible to evaluate the final solution, as well as to identify the gaps that need to be improved in order to obtain a more accurate solution.

Additionally it is performed a simultaneously discussion of the results for each case.

4.1 Sample collection process

In order to test and evaluate the implemented algorithms, it was necessary to create a data set with the data from different users. For that, a simple game was used and the players had to catch the mole (Figure 4.1). The mole position is random and it changes whenever the player touches it. The game finished when the player touches on mole thirty times.

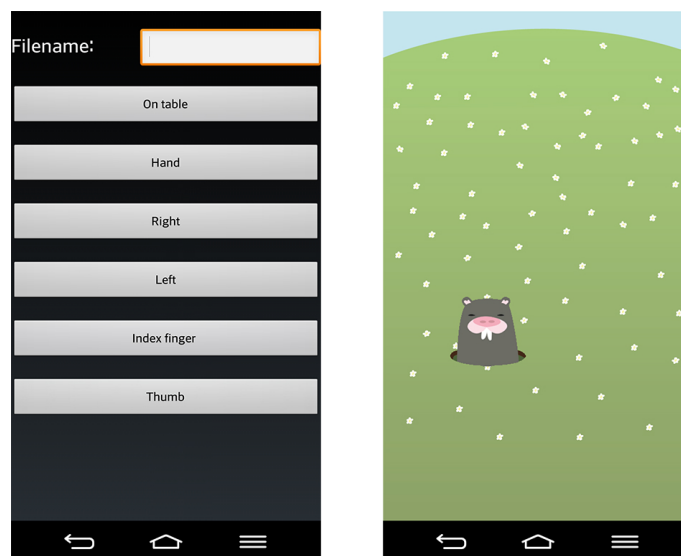


Figure 4.1: Data collection application

Validation and Evaluation

```
1 //INIT -> Information about the device
2 inf,<type_test>,<touchScreen_density>,<acc_frequency>,<gyr_frequency>
3
4 //SENSOR VALUES AND TOUCH INFORMATION
5 acc,<x_acc_value>,<y_acc_value>,<z_acc_value>,<touch>,<touch_size>
6 //or
7 gyr,<x_gyr_value>,<y_gyr_value>,<z_gyr_value>,<touch>,<touch_size>
8
9 //END -> Total number of touches detected by touch sensor
10 end,<touch_number>
11 //
```

Listing 4.1: File structure

While the player is playing the game, the sensor data is being stored in text files in the SD card, in the CSV (Comma Separated Values) format (4.1). These files have information device, such as the touch screen density, the accelerometer and gyroscope frequency, the sensors values and the number of touches detected by touch sensor (to compare with the detected touch by algorithm).

During the data collection the users play the game using the smartphone in six different ways (different positions and different fingers):

- The smartphone is on table;
- The smartphone is in user hand (any hand);
- The smartphone is in right hand;
- The smartphone is in left hand;
- The user touches using the index finger;
- The user touches using the thumb;

In order to analyze the files, it was created a java application, which reads the files and it tests the algorithms. The results and the statistics are shown on screen (using the console),thereby allowing to evaluate the algorithms precisions.

4.1.1 Participants

Two different groups participated on the evaluation study. The first group was a group of elderly with mean age of 71.6 and with ten participants. The other group has the mean age of 20.7 and it was constituted by thirty participants.

The inclusion of these two groups enabled the evaluation of the algorithm performances regardless the individual's physical and heath situation. As these groups have different motor abilities it was possible to have a more complete evaluation. It is also possible to compare the algorithms performance of users with different levels of experience with smartphones, because the participants of the youngest groups have a lot experience and the others have very little experience.

4.1.2 Scenario

In order to make the results as realistic as possible the participants had the maximum of freedom as possible. For that the collect process was divided into six phases and in each phase the participants just had to abide the rule of that phase (smartphone position, hand or finger). They were free to move as they wish.

4.2 Results

In this section presents the result of precision for each algorithm. The data were to collect using the Google Nexus S smartphone.

4.2.1 Smartphone position

In order to analyze the performance of this algorithm, each participant played the game with the smartphone on table and holding it with a hand. The younger group's results are:

	ON_TABLE (%)	HAND (%)
Smartphone on table	95.6	4.4
Smartphone in hand	0.7	99.3

Table 4.1: Results of the younger group (smartphone position)

The older group's results are:

	ON_TABLE (%)	HAND (%)
Smartphone on table	96.2	4.8
Smartphone in hand	4.9	95.1

Table 4.2: Results of the older group (smartphone position)

The results show that this algorithm has excellent performance, being that the accuracy value is always greater than 95%. If the results of both groups are compared (Figure 4.2), we can conclude that the values are very similar. For this reason it is possible to assume that the physical condition and the experience do not affect the performance of this algorithm.

Validation and Evaluation

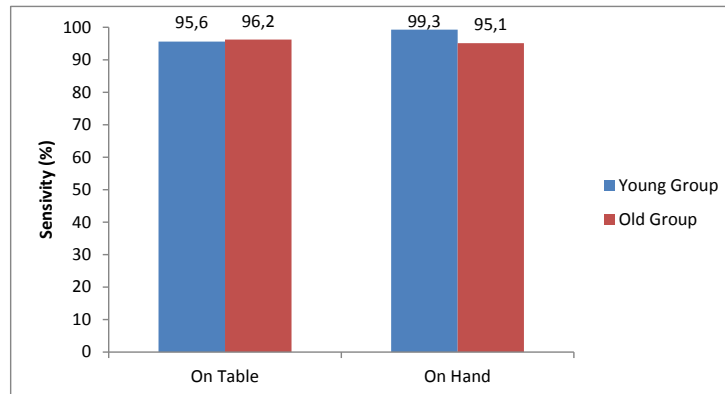


Figure 4.2: Comparison of results between the two groups (smartphone position.)

4.2.2 Touch detection

Using the same data set used by previous algorithm, the algorithm for touch detection was tested. The conditions are the same: to play the game with the smartphone on table and in the hand. The percentages of detected touches for each condition are:

	Young(%)	Old(%)
Smartphone on table	96.2	96.1
Smartphone in hand	93.2	98.0

Table 4.3: Percentages of detected touches.

Like the previous algorithm this results also are goods. The mean percentage of detected touches is 95.36%. One of the reasons why this value is not 100% is due to excess of signal noise (e.g. great movements of the smartphone). In this case there are several variation peaks and the algorithms does not find any peak large enough to distinguish itself from the rest. Another reason can be the touch power is very small and the algorithm can't detect any variation peak.

The results of both groups are very similar (Figure 4.3). As a result of this we can conclude that the algorithm performance is not affect by the physical condition or experience with smartphones.

Validation and Evaluation

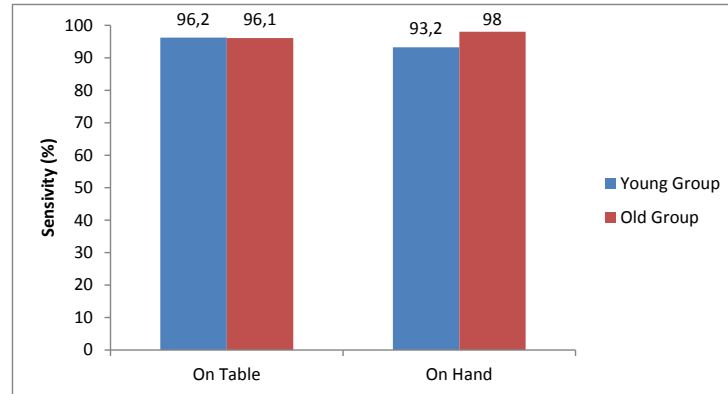


Figure 4.3: Comparison of results between the two groups (detected touches).

4.2.3 Hand (Right/Left)

In order to create the data set, the participants play the game with the right hand and with the left hand. The younger group's results are:

	RIGHT(%)	LEFT(%)
Smartphone in right hand	88.7	11.3
Smartphone in left hand	19.9	80.1

Table 4.4: Results of the younger group (determine the hand)

The older group's results are:

	RIGHT(%)	LEFT(%)
Smartphone in right hand	69	31
Smartphone in left hand	31.7	68.3

Table 4.5: Results of the older group (determine the hand)

If the results are analyzed separately, it can be concluded that the younger group's results are relatively good being that, the values are larger than 80%, while the older group's results are lower (their mean is almost 70%).

Comparing the results of two groups (Figure 4.4) it is visible that there is a decrease of performance on the elder population. It is possible to conclude that this algorithm is good to detect the hand used to hold the device for the young people and it is not so good for the old people. It means that the algorithm is sensible to physical condition or the experience with smartphones. In this

Validation and Evaluation

case we expect that the experience is the more relevant factor than the physical conditional because this algorithm depends on how the users interact with the smartphone.

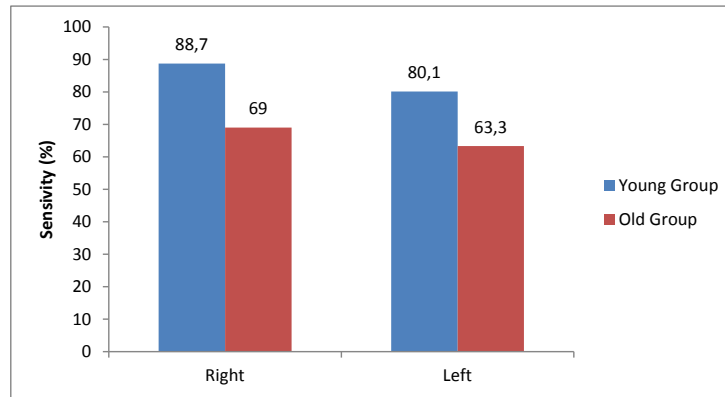


Figure 4.4: Comparison of results between the two groups (determine the hand).

A reason for that is the users have few use experience with smartphone, and because that they support the smartphone on top of the palm (as it is possible to see in the Figure 4.5). When they touch the device, it has a small rotation opposite to that expected by the algorithm. This rotation deceives the algorithm and as result their output is wrong.

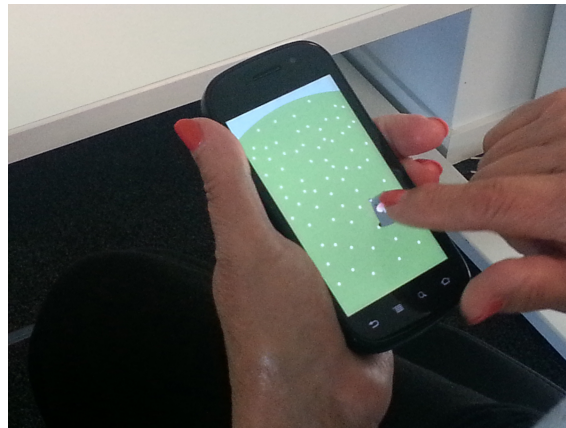


Figure 4.5: Picture taken during the data collection (older group).

4.2.4 Finger (Thumb/Index finger)

In this case the participants play the game using the thumb and the index finger to touch the mole. The younger group's results are:

Validation and Evaluation

	THUMB(%)	INDEX FINGER(%)
Touch with the thumb	75.1	14.9
Touch with the index finger	4.6	95.4

Table 4.6: Results of the younger group (determine the finger)

The older group's results are:

	THUMB(%)	INDEX FINGER(%)
Touch with the thumb	75.3	24.7
Touch with the index finger	27.1	72.9

Table 4.7: Results of the older group (determine the finger)

The results show that the algorithm presents good performance when the user is young (the mean value is 85.25%). With increasing age the performance for the index finger decreases (Figure 4.6). This can be associated with the fact that the fingers thicken with age because the main feature used in this algorithm is the touch size. When the user uses the thumb, the algorithm performance does not vary with the age. The precision value keeps constant.

Generally speaking, it is possible conclude that the algorithm has a good performance for young users and when they are using the index finger to touch the device. In the others cases it is possible to conclude that the output algorithm can have a small margin of error. We believe that the accuracy values increase with increasing of the touch number and if the user alternates the finger with touch the smartphone because this algorithm learns with the experience.

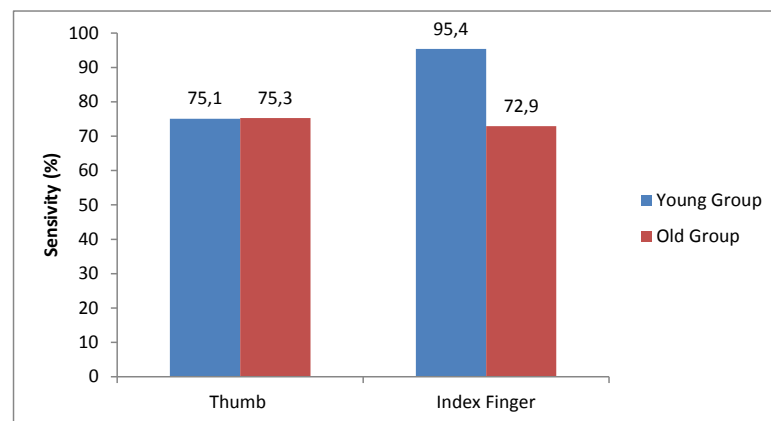


Figure 4.6: Comparison of results between the two groups (determine the finger).

Chapter 5

Conclusions and Future Work

Currently the sensors embedded on smartphones are responsible for the human-device interaction. The increase in their number is one of the main factors responsible for the success of smartphones. This dissertation explores the possibility of using the embedded sensors to other functionalities other than the traditional ones. Specifically if the accelerometer and gyroscope can complement the touch screen information.

These two sensors are motions sensors. The accelerometer is an electromechanical device that measures acceleration forces. The gyroscope is a device for measuring or maintaining orientation, based on the principles of angular momentum. They are usually used in smartphones to control the direction in games, to detect user falls or to change/adjust the screen for a proper viewing.

We started with a study of the sensor's signal to determine what new data could be associated to a touch event. It was concluded that it is possible to detect touches using the accelerometer and the gyroscope; to determine where the user is using the device (on table or in hand) used when the touch happens; to identify if the user is holding the device with the right or the left hand (if it is applicable); to identify the finger used to touch the device; and to determine the force touch.

For this purpose signal processing techniques were used like filters, sliding windows and data mining techniques. In the last case the Weka software was used to support the data classification. Thereafter five algorithms were created, one for each new feature.

In order to evaluate the algorithms it was used two groups with different ages. A group had the mean age of 20.7 years and the other had the mean age of 71.2 years. The reason for this difference was to be able to compare the results from participants with different experience using smartphones and different physical conditions.

The results show that the algorithm that assesses how the smartphone is being used and the algorithm of touch detection have an excellent performance for both groups (precisions higher than 95%), although the last presents a little problem with the false positives (which was solved using the touch sensor). The other algorithms' results are slightly different. Although results of the younger group are good (hovering around 85%) the results of the older group are slightly

Conclusions and Future Work

lower (hovering around 70%). This difference can be explained for participants' inexperience with smartphones (for the algorithm that detect the hand) because they use the smartphone on the top of the hand or for the thick fingers of the older group's participants (for the algorithm which detects the finger).

5.1 Future Work

The main improvements of this project and future work identified during its developing are:

- Try different approaches with the prupose of compare their performances;
- Improve the algorithm that detects where the device is, with the purpose of distinguish when the user is holding the device with one or two hands;
- Find a better solution for the false positives of the detect touches algorithm;
- Research for a better machine learning approach to the algorithm that determines which the finger is used to touch the smartphone, in order to improve its performance;

References

- [Anda] Android. Android api. Available in <http://developer.android.com/reference/android/view/View.html#dispatchTouchEvent%28android.view.MotionEvent%29>, last accessed on May 25, 2014.
- [Andb] Android. Motion sensors. Available in http://developer.android.com/guide/topics/sensors/sensors_motion.html, last accessed on January 29, 2014.
- [BZ] Rafail Ostrovsky Braverman, Vladimir and Carlo Zanioloi.
- [CC11] Liang Cai and Hao Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. *USENIX Association*, 2011.
- [Dan] Tony Danova. Smartphone market update: The markets, manufacturers, and devices that will drive future growth. Available in <http://www.businessinsider.my/smartphone-market-update-2014-2014-1/#.U3N08Yb0HpQ>, last accessed on May 14, 2014.
- [DFC10] Diogo R. Ferreira Davide Figo, Pedro C. Diniz and Joao M. P. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, pages 645–662, 2010.
- [dS13] Joana Raquel Cerqueira da Silva. *Smartphone Based Human Activity Prediction*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2013.
- [Eng] Dimension Engineering. A beginner’s guide to accelerometers. Available in <http://www.dimensionengineering.com/info/accelerometers>, last accessed on January 23, 2014.
- [GL08] Rainer Gemulla and Wolfgang Lehner. Sampling time-based sliding windows in bounded space. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008.
- [Goo] Ryan Goodrich. Accelerometer vs. gyroscope: What’s the difference? Available in <http://www.livescience.com/40103-accelerometer-vs-gyroscope.html>, last accessed on January 29, 2014.
- [GP12] Jacob Wobbrock Goel, Mayank and Shwetak Pateli. Gripsense: Using built-in sensors to detect hand posture and pressure on commodity mobile phones. *ACM*, 2012.
- [Hao] Lim Pei Hao. Gyroscope in smartphone. Available in <http://www.mobile88.com/news/read.asp?file=/2012/4/21/20120421165938>, last accessed on January 23, 2014.

REFERENCES

- [HS11] Ken Hinckley and Hyunyoung Songi. Sensor synaesthesia: touch in motion, and motion in touch. *ACM*, pages 801–810, 2011.
- [IDC] IDC. Despite a strong 2013, worldwide smartphone growth expected to slow to single digits by 2017, according to idc. Available in <http://www.idc.com/getdoc.jsp?containerId=prUS24701614>, last accessed on May 26, 2014.
- [JEX] Roberto De Nuccio Jay Esfandiyari and Gang Xu. Introduction to mems gyroscopes. Available in <http://electroiq.com/blog/2010/11/introduction-to-mems-gyroscopes/>, last accessed on January 28, 2014.
- [JLB06] T. Choudhury J. Lester and G. Borriello. A practical approach to recognizing physical activities. *Lecture Notes in Computer Science*, pages 1–16, 2006.
- [JRK11] Samuel A. Moore Jennifer R. Kwapisz, Gary M. Weiss. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12:74–82, 2011.
- [JWL09] Daqing Zhang Guande Qi Jiahui Wu, Gang Pan and Shijian Li. Gesture recognition with a 3-d accelerometer. *Ubiquitous intelligence and computing*, pages 25–38, 2009.
- [Kel] Amey Kelkar. Android architecture: A brief introduction. Available in <http://blog.ameykelkar.com/2014/01/android-architecture-brief-introduction.html>, last accessed on May 26, 2014.
- [KS05] Bederson B.B. Karlson, A.K. and John SanGiovanni. Applens and launchtile: two designs for one-handed thumb use on small devices. *ACM*, 2005.
- [LC09] Chris Nugent Liming Chen. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, pages 410–430, 2009.
- [MGW] Leah Findlater Mayank Goel and Jacob O. Wobbrock. Proceedings of the sigchi conference on human factors in computing systems. *ACMr*, pages 2687–2696.
- [NRL05] P. Mysore N. Ravi, N. Dandekar and M. Littman. “activity recognition from accelerometer data. *Proceedings of the National Conference on Artificial Intelligence*, 2005.
- [Ode] Den Odell. Sense and sensor-bility: access mobile device sensors with javascript. Available in <http://mobiforge.com/design-development/sense-and-sensor-bility-access-mobile-device-sensors-with-javascript>, last accessed on May 14, 2014.
- [Pal09] G. Palshikar. Simple algorithms for peak detection in time-series. *Business Analytics and Intelligence*, 2009.
- [Sek] Michael Sek. Frequency analysis fast fourier transform, frequency spectrum. Technical report, Victoria University.
- [SR] Maayan Zadik Shay Reuveny. 3d motion tracking with gyroscope and accelerometer. Available in <http://www.cs.huji.ac.il/~gaddi/3D%20Motion%20tracking%20with%20Gyroscope%20and%20Accelerometer.pdf>, last accessed on January 29, 2014.
- [Tec] Solid State Technology. Sensor technology.

REFERENCES

- [Wav] WaveMetrics. Digital filters. Available in <http://www.wavemetrics.com/products/igorpro/dataanalysis/signalprocessing/digitalfilters.htm>, last accessed on January 29, 2014.
- [WF] Ian H. Witten and Eibe Frank. Data mining: Practical machine learning tools and techniques. Technical report.
- [Wil10] Adriana G. Wilde. An overview of human activity detection technologies for pervasive systems, 2010.

REFERENCES

Appendix A

Decision tree (position)

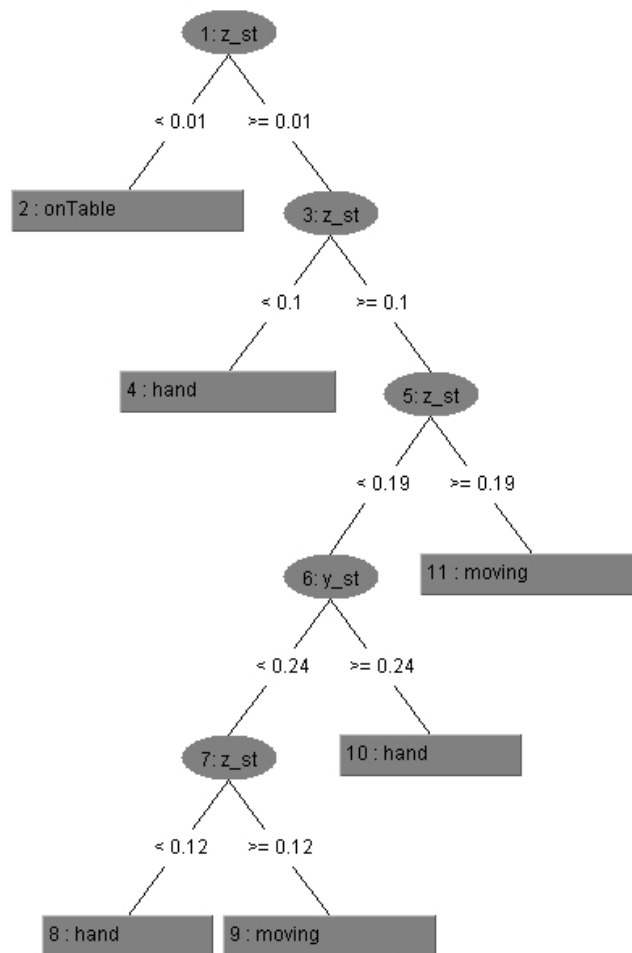


Figure A.1: Decision tree (smartphone position).