May 2014

Multi Sensor System for Pedestrian Tracking and Activity Recognition in Indoor Environments

Juan Jose Marron Monteserin
University of South Florida, juanjomarron@gmail.com

Follow this and additional works at: http://scholarcommons.usf.edu/etd
Part of the Computer Sciences Commons, and the Engineering Commons

Scholar Commons Citation

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Multi Sensor System for Pedestrian Tracking and Activity Recognition in Indoor Environments

by

Juan José Marrón Monteserín

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science Department of Computer Science and Engineering College of Engineering University of South Florida

Major Professor: Miguel A. Labrador, Ph.D.
Sean J. Barbeau, Ph.D.
Yu Sun, Ph.D.

Date of Approval:
March 3, 2014

Keywords: Smartphones, Sensor Fusion, Pervasive Computing, Inertial Navigation, Ubiquitous Localization

Copyright © 2014, Juan José Marrón Monteserín
DEDICATION

This thesis is dedicated to my family for their love, constant motivation, and moral support. Without you and your encouragement this thesis would have never been completed and, do not forget, all my achievements are also yours.
ACKNOWLEDGMENTS

I would like to give my gratitude to the people who helped me to complete the research presented in this thesis. In particular I wish to thank my major professor Dr. Miguel A. Labrador for his insight, guidance and assistance.

I appreciate Dr. Daniel Fernández Lanvin and Dr. Martín González Rodríguez for their comments and suggestions, which have enabled me to make significant improvements to the quality of this work. Similarly, I thank Adrian Menendez Valle who contributed with his time and knowledge.

I would also like to acknowledge the feedback from my committee members, including Dr. Sean Barbeau and Dr. Yu Sun, who provided important input to revise this thesis.

Finally, I wish to thank my parents, family and friends for their love and endless support during these last months.
# TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................................................ iii

LIST OF FIGURES ........................................................................................................................................ iv

ABSTRACT ..................................................................................................................................................... vi

CHAPTER 1: INTRODUCTION ..................................................................................................................... 1
  1.1 Motivation .................................................................................................................................................... 1
  1.2 Aims and Challenges .................................................................................................................................. 3
  1.3 Contributions ............................................................................................................................................ 5
  1.4 Structure of Thesis ................................................................................................................................... 6

CHAPTER 2: BACKGROUND AND LITERATURE REVIEW ....................................................................... 7
  2.1 Global Positioning System ........................................................................................................................ 7
     2.1.1 Overview ............................................................................................................................................... 7
     2.1.2 Components ......................................................................................................................................... 8
     2.1.3 Position Calculation ............................................................................................................................ 9
     2.1.4 Limitations and Constraints ................................................................................................................ 9
  2.2 Inertial Navigation System ....................................................................................................................... 11
     2.2.1 Accelerometer ................................................................................................................................... 11
     2.2.2 Gyroscope .......................................................................................................................................... 12
     2.2.3 Magnetometer ................................................................................................................................... 14
     2.2.4 Barometer .......................................................................................................................................... 15
     2.2.5 Limitations and Constraints ................................................................................................................ 15
  2.3 Mobile Platform ........................................................................................................................................ 17
     2.3.1 Android Operating System ................................................................................................................ 17
     2.3.2 Android Sensor API ........................................................................................................................... 19
  2.4 Related Works .......................................................................................................................................... 21

CHAPTER 3: SYSTEM DESIGN AND IMPLEMENTATION ........................................................................ 26
  3.1 System Design .......................................................................................................................................... 26
  3.2 Architecture ............................................................................................................................................. 29
     3.2.1 Data Collection Module ....................................................................................................................... 31
     3.2.2 Motion Segmentation Module ............................................................................................................ 31
     3.2.2.1 Turn Detection Algorithm ............................................................................................................... 31
     3.2.2.2 Altitude Change Detection Algorithm ............................................................................................ 35
     3.2.3 Activity Recognition Module .......................................................................................................... 38
     3.2.3.1 Elevator (E) ..................................................................................................................................... 39
LIST OF TABLES

Table 4.1  Samsung Galaxy S II SGH-I777 specifications..........................................................57
Table 4.2  LG Nexus 4 E960 specifications. .................................................................................58
Table 4.3  Accelerometer test results...........................................................................................61
Table 4.4  Gyroscope test results. ...............................................................................................62
Table 4.5  Magnetometer test results. ..........................................................................................64
Table 4.6  Barometer test results ...............................................................................................65
Table 4.7  Turn detection test results. ..........................................................................................67
Table 4.8  Altitude changes detection test results .......................................................................69
Table 4.9  Activity recognition test results. ................................................................................69
Table 4.10  Step detection and counting test results. .................................................................70
Table 4.11  Stairs detection and counting test results.................................................................71
Table 4.12  Tracking square motion results ...............................................................................72
Table 4.13  Results of tracking stairs sequence in down direction ...........................................72
Table 4.14  Results of tracking stairs sequence in up direction .................................................73
Table 4.15  Results of tracking floor motion .............................................................................73
LIST OF FIGURES

Figure 2.1 Overview of a typical hybrid positioning system ........................................... 10
Figure 2.2 Calculated speed and distance by integrating acceleration. .......................... 16
Figure 2.3 Android architecture diagram ........................................................................ 18
Figure 2.4 Android sensor API layers .............................................................................. 20
Figure 3.1 Real floor plan example ................................................................................ 27
Figure 3.2 Human gait cycle ......................................................................................... 29
Figure 3.3 Sequence of modules in system’s architecture .............................................. 30
Figure 3.4 Turn detection algorithm signals ................................................................... 34
Figure 3.5 Altitude change detection algorithm signals. ...................................................... 36
Figure 3.6 Example of pedestrians states in indoor trace split by turns. ......................... 37
Figure 3.7 Decision tree to classify the possible indoor states ........................................ 38
Figure 3.8 Transition of magnetic field energy outside and inside an elevator. .............. 39
Figure 3.9 Energy of acceleration values for an up direction elevator travel. .................. 40
Figure 3.10 Air pressure variations for an up direction elevator travel ............................. 41
Figure 3.11 Variance of energy acceleration in walking and stairs states. ...................... 42
Figure 3.12 Values of air pressure in stairs and walking states ........................................ 43
Figure 3.13 Step detection and counting algorithm signals ............................................. 46
Figure 3.14 Stairs detection and counting algorithm signals ........................................... 49
Figure 3.15  System implementation architecture. ................................................................. 50
Figure 3.16  Mobile application screenshots. ........................................................................ 50
Figure 3.17  Server web application screenshots. .................................................................... 52
Figure 3.18  Web server deployment diagram. ......................................................................... 54
Figure 4.1  Coordinate reference system used by the sensor API. ........................................... 59
Figure 4.2  Output signals accelerometer quality test............................................................. 60
Figure 4.3  Output signals gyroscope quality test................................................................. 62
Figure 4.4  Output signals magnetometer quality test. ............................................................ 63
Figure 4.5  Output signals barometer quality test. ............................................................... 65
Figure 4.6  Output signals for sampling rate test. .................................................................. 66
Figure 4.7  Sequences performed for turn detection test. ......................................................... 68
Figure 4.8  Sequences performed for altitude detection test.................................................... 68
Figure 4.9  Motion traces used for testing pedestrian tracking................................................. 71
ABSTRACT

The widespread use of mobile devices and the rise of Global Navigation Satellite Systems (GNSS) have allowed mobile tracking applications to become very popular and valuable in outdoor environments. However, tracking pedestrians in indoor environments with Global Positioning System (GPS)-based schemes is still very challenging given the lack of enough signals to locate the user. Along with indoor tracking, the ability to recognize pedestrian behavior and activities can lead to considerable growth in location-based applications including pervasive healthcare, leisure and guide services (such as, museum, airports, stores, etc.), and emergency services, among the most important ones.

This thesis presents a system for pedestrian tracking and activity recognition in indoor environments using exclusively common off-the-shelf sensors embedded in smartphones (accelerometer, gyroscope, magnetometer and barometer). The proposed system combines the knowledge found in biomechanical patterns of the human body while accomplishing basic activities, such as walking or climbing stairs up and down, along with identifiable signatures that certain indoor locations (such as turns or elevators) introduce on sensing data.

The system was implemented and tested on Android-based mobile phones with a fixed phone position. The system provides accurate step detection and count with an error of 3% in flat floor motion traces and 3.33% in stairs. The detection of user changes of direction and altitude are performed with 98.88% and 96.66% accuracy, respectively. In addition, the activity
recognition module has an accuracy of 95%. The combination of modules leads to a total tracking error of 90.81% in common human motion indoor displacements.
CHAPTER 1: INTRODUCTION

1.1 Motivation

Mobile computing devices such as smartphones, tablets and smartwatches, are nowadays overtaking the popularity of conventional desktop computers. The paradigm of computing has evolved in recent years reducing the prices of these devices, increasing the number of features, processing power and mobility capabilities. As a result, location-aware services and applications that can make use these devices have spurred particularly in the wellness [1] and in the entertainment sector [2]. This rapid growth in people-centric mobile computing applications has led to improvements in localization technologies, not only in terms of localization accuracy, but also across multiple and specific dimensions such as power consumption, energy efficiency and ubiquity.

In outdoor environments localization is successfully solved by traditional Global Navigation Satellite Systems (GNSSs), such as the Global Positioning System (GPS), cell tower localization and Wi-Fi. However, these technologies cannot track the user’s position accurately in indoor environments where humans spend most of our time (offices, home, schools, universities, malls, etc.).

A location service capable of providing accurate positioning in indoor environments could promote the interest in a whole range of new mobile applications based on location-aware services and context interaction. This new functionality could be used in several scenarios, such as:
• Safety: location systems could provide rescue services with an accurate and immediate knowledge of user’s position inside a building in case of emergency.

• Resource-efficiency: smart buildings and homes can utilize the information of where users are to optimize resources such as air conditioning, heating, or lighting.

• Security: location-awareness could permit automatic locking of sensitive resources depending on the owner presence.

• Social networking: allowing users to efficiently find colleagues in indoor scenarios where most of us spend our daily lives.

• Automatic resource routing: creating follow-me applications that allow users with visual impairments to be routed to their goals.

• Leisure: reproduce automatic explanations in museums and gallery arts.

• Navigation: visitors acquainted with an installation could easily navigate to areas or rooms of interest.

• Advertisements: adapting offers and advertising in shopping centers depending on the customer position.

Although several systems and solutions have been proposed to solve the problems related with indoor location, most of them require limitations in the testing environment, or need some form of supplementary hardware devices and additional infrastructure that makes them impractical for most of the common contexts. Additionally, in contrast to outdoor services, indoor location systems require a higher level of accuracy that is not provided by the current methods.

Modern mobile devices are equipped with Micro Electro Mechanical Systems (MEMS) that can be used to overcome these problems and to avoid the dependency on external
infrastructure. Despite the low-performance and noisy sensor data, applying proper signal processing and algorithms to combine the raw data received from the array of inertial sensors (triaxial orthogonal accelerometer, gyroscope, magnetic field detector and barometric pressure sensor), acceptable location accuracy results can be achieved. The use of internal sensors embedded in the latest generation of mobile devices also introduces advantages in terms of power consumption, simplicity of management and reduction of cost of calibration versus external data sources or infrastructures in traditional indoor systems.

1.2 Aims and Challenges

This thesis presents a smartphone-based system for pedestrian tracking in indoor environments that works without any additional infrastructure or external sensors. The system implementation is based on two sequential components: a decision tree for activity recognition which classifies human behavior, and a set of algorithms that provide additional information about the activity, such as motion distance or direction. Altogether the system is able to follow motion displacements and track pedestrians accurately in indoor scenarios.

The system collects real time sensor data to recognize human indoor activities, such as turns (T), stationary times (Sy), use of elevators (E), walking (W) and stairs (St). It applies track-splitting and landmarking strategies to reduce the accumulated error usually inherent in every inertial sensor-based system.

Indoor location systems, like the one being proposed here, are very challenging:

- Sensor’s location: specific high quality sensors are usually mounted in bare functional locations making the system impractical and uncomfortable for the user’s common use.
- Unreliable sensor data: several obstacles in indoor environments, such as machines, walls, corridors, open areas, metals, etc., introduce random noise in the sensor measurements.

- Cumulative errors: low cost sensors embedded in mobile devices are normally low quality devices that would lead to cumulative errors when estimation of new positions are based on previous Pedestrian Dead Reckoning (PDR) calculations.

- Data fusion: the multisensory approach of the proposed system introduces the challenge of combining sensing data collected from sensors of different nature to extract reliable signatures and patterns.

- High level of location accuracy required: because the indoor context varies at fine spatial granularity, most indoor-based applications require a high level of accuracy from the location systems.

- Energy consumption: the continuous use of sensors and processing needs of the location system consume extra energy from already energy-constrained devices like the smartphones.

- Processing power: although hardware resources in mobile devices are in continuous evolution, processing power is still limited. Algorithms and signal processing techniques implemented in these devices should prevent a high consumption of resources.

- Mobile device position: the use of a mobile device as data collector introduces the challenge of multiple possible positions and orientations of the device, such as calling, messaging, swing if the users hold the device in hand while walking or seated in the users pocket or bag. These different actions affect the signal processing algorithms.
• Evaluation tools: the evaluation of positioning systems is a tedious job consisting of the repetition of experiments and comparison of the results with the real scenarios. In most cases experiments are performed manually and individual evaluation is necessary to compute global results.

1.3 Contributions

The sensor tracking system proposed in this thesis makes several contributions meant to address most of the challenges described before. Among the most important contributions are the following:

• This proposed approach simultaneously harness sensor-based dead-reckoning and environment sensing for localization. The system does not require previous calibration or installation of additional infrastructure.

• This approach leverages the smart phone’s functionality and uses the off-the-shelf sensors embedded in mobile devices to extend its sensing capabilities. No external hardware is needed.

• A set of signal processing algorithms is included to obtain valuable information from the noisy raw sensor data. The algorithms avoid machine learning techniques, which require training tasks, and rely on statistical analysis. Based on filters, peaks detection, thresholds techniques, and internal calibration routines, the algorithms are able to detect and count stairs and steps, detect turns and altitude changes or recognize motion direction and sensing patterns in indoor environments.

• The system shows that adequate points of interests (POI) exist in indoor environments and how the sensor signatures can perceive them. These POIs are applied to dead-reckoning making it practical and reasonably accurate.
• The basic pedometer approach is extended to reduce the accumulation of errors, and the total distance travelled is estimated as the addition of the individual size steps. Furthermore, the complete motion traces are split into independent segments by leveraging indoor points of interest for resetting the errors.
• The system design proposes a practical classification technique by means of a decision tree that recognizes human indoor activities based on sensor measurements and the limitations that indoor obstacles impose on the pedestrian’s indoor actions.
• A complete system for evaluation is developed. It consists of an Android application in the client side in charge of collecting the sensor data, and a server application working as a testbed that simplifies the system’s implementation and the evaluation process.

1.4 Structure of Thesis

The remainder of the thesis is organized as follows: Chapter 2 details the background and the theory considered in this thesis and describes previous research done in this area. Chapter 3 describes the groundwork of this thesis where the key concepts of the system design and its implementation are explained including the modules and algorithms that are part of the system. Chapter 4 includes the evaluation methodology and discusses the performance of the systems and the individual algorithms. Finally, Chapter 5 concludes the thesis and sets forth directions for future research.
CHAPTER 2: BACKGROUND AND LITERATURE REVIEW

This chapter provides the background information about indoor location solutions and describes the most important systems available in the literature. In addition to describe the GPS system, this chapter explains the most important sensors used in indoor location systems as well as the software platform to develop these systems in mobile devices. Finally, the literature on indoor positioning systems is reviewed in detail.

2.1 Global Positioning System

2.1.1 Overview

A Global Positioning System, also known as GPS [3], is a satellite-based navigation system designed to help navigate on the Earth, in the air, and on water. It provides users with positioning, navigation, and timing services.

The GPS system was created by the Department of Defense of the United States and nowadays is still owned and operated by the United States. Originally, it was used exclusively by the U.S. military, but in 1983 an order allowed anyone to use the system. The system was declared fully operational in 1995. Today, GPS is also used for civilian purposes such as surveying, map design, tectonics, and obviously, navigation.

In recent times, other Global Navigation Satellite Systems (GNSS) have been either launched or being developed, such as the Russian GLONASS system, the European Union
Galileo satellite navigation system, the Chinese Beidou satellite navigation system, and the
Indian satellite navigation system.

A GPS receiver included in the mobile devices uses the signal from three or four satellites
to calculate the device’s location. Additionally, the systems presents the speed at which it is
moving, the direction it is going, its altitude and, in some cases, how fast it is going up or down.
Also, many GPS receivers have information about locations and places and show the position
referred to the maps of the area. Most GPS receivers record past locations and can be useful to
plan a journey. While traveling a planned journey, it can predict the time to the destination.

2.1.2 Components

The GPS system is formed of three segments: the space segment (satellites), the user
segment (receivers) and the control segment.

The first of the parts, the space segment comprises 30 satellites located 20,200 kilometers
(12,600 mi) above the Earth orbiting in six orbits which have an inclination of 55° relative to the
Equator. The orbits are arranged in such way that a GPS receiver can detect signals from 6 to 12
satellites at once far from the North Pole and South Pole. GPS satellites send navigation
messages continuously at a rate of 50 bits per second. Information messages consist of a time
stamp, exact orbital information (“ephemeris”), system-wide status information and rough orbits
of all satellites (“almanac”).

The control segment consists of a number of stations and antennas used to control and
monitor the entire system, making necessary corrections when needed.

The user segment consists of military receivers using the GPS Precise Positioning Service
and civil receivers using the Standard Positioning Service. GPS receivers are mainly composed
of an antenna, a very stable internal clock, the software for calculating the user’s location and
speed, and usually a display for providing the information to the user. Nowadays, GPS receivers can be found in navigation devices, mobile phones, wrist watches and other devices. GPS receivers calculate its position several times in one second and estimate its speed and direction by calculating its change in position and change in time during. Most of the basic and inexpensive consumer receivers show an accurate of 20 meters (66 ft) almost anywhere on Earth.

2.1.3 Position Calculation

GPS receivers use geometric trilateration to combine the information from different satellites to predict the correct location. As mentioned above, the information message contains details about the time when message was sent, precise orbital information, health of the system and rough information about the orbits of other satellites. The receiver uses each message to calculate the transfer time and whence, the distance to the satellite. With the use of trilateration, the distances to the satellites together with the satellites locations are useful to estimate the position of the receiver.

2.1.4 Limitations and Constraints

The main functionality of satellite systems is that receivers can compute and estimate latitude, longitude, and altitude with a high degree of accuracy. However, the functioning of these systems requires line of sight (LOS). This disadvantage leads to an inability for using GPS-based systems in most indoor environments where walls, roofs and different obstacles hinder the satellite signals. To detect GPS signals inside a building typically requires a receiver capable of tracking signals with levels of power between $-160dBW$ and $-200dBW$; however a typical commercial receiver has a noise floor of around $-131dBW$. Multipath effects are likely to cause degradation in accuracy even if a receiver is able to track signals from a sufficient number of
satellites. Unlike in outdoor environments, reflected signals are often stronger than those received via direct line-of-sight when indoors. Hence such systems are not accurate enough for indoor location-aware applications, nearly all of which require at least room-level accuracy. Additionally, GPS is an energy hungry technology that consumes the constrained mobile devices battery rapidly [4].

![Diagram of a typical hybrid positioning system](image)

Figure 2.1 Overview of a typical hybrid positioning system.

Classic outdoor tracking systems on the market combine the GPS position calculation with network-based localization techniques such as cell tower signals (GSM) and wireless internet signals (Wi-Fi) to calculate the user’s location. These systems, also called Hybrid Positioning Systems, have been specifically created to overcome the GPS–based system limitations. GPS technology is very exact in outdoors and open areas, but is inaccurate in indoors
or between buildings (the “urban canyon” effect). Compared with GPS signals, cell tower signals are not attenuated by buildings or weather conditions, although they generally allow less precise positioning. Additionally, Wi-Fi positioning systems may provide very exact location in urban areas where a large Wi-Fi network is installed and a comprehensive database of Wi-Fi access points is available.

Figure 2.1 shows an overview of a typical hybrid positioning system. These hybrid systems are increasingly being introduced in location-based services for certain civilian and commercial purposes. Hybrid positioning systems need to work accurate in indoor and urban areas in order to be commercially and functionality viable.

2.2 Inertial Navigation System

INS (Inertial Navigation System) is based on a self-contained navigation technique in which readings are provided by motion sensors (accelerometers), environment sensors (magnetometer and barometer) and rotation sensors (gyroscopes). These sensor measurements are utilized to track position, orientation and context of a mobile object relative to an initial point and velocity of movement without the need for external references.

2.2.1 Accelerometer

An accelerometer is an instrument that measures proper acceleration. It detects the acceleration associated with the phenomenon of weight (suffered by any mass at rest in the frame of reference of the device) instead of the coordinate acceleration (rate of change of velocity). For instance, an accelerometer at rest on the surface of the earth will return an acceleration of $g=9.81 \, \text{m/s}^2$ due to its weight. On the other hand, an accelerometer in free fall will measure an acceleration of $0 \, \text{m/s}^2$. 

11
Accelerometers have multiple uses in science and industry. For example, they are useful to detect and estimate vibration in rotating machinery. More recently, they were integrated in tablet computers, smartphones, and digital cameras and used to always display the images upright on screens.

Among all the possible sources of error in an accelerometer, the most important is the bias. The bias is the systematic distortion of output signal from the real value, in m/s$^2$ in the case of the accelerometer. A constant offset or bias error of $\varepsilon$, can exhibit a quadratic growth error with time in the position calculation, as shown in Equation 1).

$$ S(t) = \frac{\varepsilon t^2}{2} $$

where $t$ is the time of the integration. The bias of an accelerometer can be determined by estimating the long-term average of the sensor readings when it is not experiencing any external acceleration. Uncompensated bias errors are generally the error sources that reduce the accurate of the sensor and they have to be corrected with the aid of calibration methods.

### 2.2.2 Gyroscope

Generally, a gyroscope is a device based on the principles of conservation of angular momentum for measuring or maintaining orientation. A conventional (mechanical) gyroscope is formed of a spinning wheel mounted over two gimbals which allow the wheel to rotate in three degrees of freedom. As effect of the conservation of angular momentum, the spinning wheel will resist heading changes and maintain a constant global orientation. A traditional gyroscope measures orientation, in contrast to MEMS (Micro Electro-Mechanical System) types, which measure angular rate, and are therefore called rate-gyros. MEM gyroscopes consist of vibrating elements that measure the Coriolis Effect. When the gyroscope is rotated, an individual mass is
driven to vibrate along a drive axis and this vibration is induced, as consequence of the Coriolis force, to a secondary vibration axis in perpendicular sense. By measuring this secondary rotation, the angular velocity of the gyroscope can be estimated. The Coriolis Effect, given by Equation 2, states that a mass moving with velocity \( v \) in a reference frame rotating at angular velocity \( \omega \), experiences a force equal to \( F_c \).

\[
F_c = -2m(\omega \times v)
\]  

(2)

An important note to be made is that, whereas the accelerometer and the magnetometer measure acceleration and angle relative to the Earth, gyroscopes measure angular velocity relative to the body.

The bias of gyroscope (i.e., the offset of the output signal from the real value) can be estimated by measuring the long-term average of its output when it is not suffering any rotation. When integrated, a constant bias rate generates an angular error that grows linearly with time, as shown in Equation 3:

\[
\theta(t) = \varepsilon * t
\]

(3)

Once the bias is known, it is simply to compensate for it by subtracting the value from the output.

Another common error arising in gyroscopes is the calibration error. It is related to errors in the linearity, the alignments and scale factors of the gyros. These accuracy problems tend to produce errors while the device is turning. The errors are observed in the magnitude of the drift in the resultant integrated signal. Due to the accumulation of errors, its magnitude is proportional to the rate and duration of the motions. A solution to these accumulative errors consists on splitting the complete trajectory in simpler motion traces and to treat them independently.
2.2.3 Magnetometer

A magnetometer is an instrument which measures the direction and strength of the magnetic field in the surrounding area of the device. Magnetometers are classified in two types: scalar and vector. Scalar magnetometers measure the total strength of the undergoing magnetic field, while vector magnetometers (the type used in this project) measure the magnitude of the magnetic field in a particular direction. The orientation of the component measured is relative to the one of the device.

Additionally magnetometers provide supplementary performances and they also can be used as metal detectors. Although their functionality is limited only to the detection of magnetic ferrous metals, their detection capacity is larger than traditional metal detectors. While a classical metal detector’s range is around 2 meters, magnetometers allow the detection of large ferrous structural objects at tens of meters. More recently, magnetometers have been miniaturized allowing their embedding in integrated circuits at very low cost, hence they are increasingly being included into consumer mobile devices, such as smartphones, tablets and wearable computers.

Magnetometer presents two main sources of errors in its measurements related with magnetic contamination in the sensor. These are: errors in the measurement of the frequency and iron contamination due to the materials containing iron on the operator or in the instruments in the proximity of the sensor. These error sources can generate inaccurate readings, for example, when the magnetometer is rotated while performing a measurement. As will be detailed, calibration and signal filtering routines are applied to the magnetometer’s raw data to reduce and compensate for the eventual sensor errors.
2.2.4 Barometer

A barometer measures atmospheric pressure, usually to determine weather changes or altitude. Pressure tendency can forecast short-term changes in the weather, but additionally pressure variations can also work as detectors of altitude changes.

Because we live at the bottom of the earth's atmosphere, the air pressure decreases as altitude increases. The atmospheric pressure vs. altitude can be expressed as in Equation 4:

\[ P = P_0 \left(1 - \frac{\text{Altitude}}{44330}\right)^{5.255} \]  

where \( P \) is the air pressure at the altitude in mbar, and \( P_0 \), is the standard atmosphere (1 atm = 1013.25 mbar) defined as 29.92 inHg at sea level at a temperature of 59°F. Altitude is the height above sea level in meters.

Barometers have been also used as altimeters alongside radar instruments, helping aviation applications in stabilizing and estimating the vertical position [5] and complementing Inertial Navigation Systems. Using the pressure vs. altitude relation for human motion characterization, a MEMS pressure sensor can be used as a barometer to measure altitude changes and to detect patterns in vertical movements (moving up or down and remaining level) in integrated pedestrian navigation systems.

The altitude estimation contains an overall noise standard deviation due to a relatively poor resolution of MEMS sensors, making them useful for basic vertical movement detection, but inadequate for tracking highly dynamic movements.

2.2.5 Limitations and Constraints

A common relative navigation technique using inertial sensors is dead-reckoning. It offers an alternative for tracing users inside a building where we cannot use GPS-based
solutions. In dead-reckoning the current location \((X_k, Y_k)\) is estimated based on the previous location \((X_{k-1}, Y_{k-1})\), the displacement \((S)\) and the direction of motion \((\theta)\), as shown in Equation 5:

\[
X_k = X_{k-1} + S \times \cos \theta
\]

\[
Y_k = Y_{k-1} + S \times \sin \theta
\]

Figure 2.2 Calculated speed and distance by integrating acceleration.

where \(\theta\) can be calculated from the gyroscope measurements, while the distance traveled \(S\), can be estimated from the accelerometer readings. The initial position can be fixed a priori.

The distance traveled by an object can be theoretically estimated by double integration of its acceleration with respect to time. Nevertheless, the presence of noise in the accelerometer readings produces the accumulation of errors quickly with time. Also, the presence of the force of gravity on Earth introduces a new component of acceleration and, as consequence, a new source of systematic error or bias. These conditions produce errors in the estimation of the distance traveled that will grow constantly with respect to time. Figure 2.2 shows the linear acceleration component on the y-axis (forward acceleration). The velocity resulted from integrating the acceleration and the walked distance calculated by integrating the velocity over time in an experiment where 13 steps were walked for a distance of 10 meters. The form of the curves
seems correct and it is possible to identify the peaks of acceleration while walking, as well as the increment/decrement of speed with the steps. However, the results obtained (around 6 meters) are approximately half of the total distance actually traveled in the experiment.

In pursuance of successfully apply dead-reckoning technique for pedestrians, a precise estimation of the user’s distance traveled and an accurate detection of the direction of motion are necessary; however, these two aims results are complex to achieve due to the bias introduced by the inertial sensor and the error accumulation (see Section 4.2.2 Inertial Sensors Quality Tests).

2.3 Mobile Platform

2.3.1 Android Operating System

Android is a Linux-based operating system and open-source platform for mobile devices, which is developed and controlled by Google. Android platform includes the operating system, the middleware and the key applications. Several characteristics have promoted the choice of Android as the development environment for this thesis rather than any other mobile platform like iOS, Symbian or Windows Phone.

- Google Inc. is actively developing Android platform, adding new functionalities, services and APIs.
- More popular hardware manufacturers and phone are installing Android on their devices due to its low-cost and customizable features.
- The fact of being an open-source project has appealed to the interest of a large community of developers.
- Its increasingly high popularity has recently become Android the world’s most used platform for smartphones [6].
Android was primarily designed for touchscreen devices and is based on the Linux kernel. Its architecture stack is structured in four complementary layers which involve five groups, as illustrated in Figure 2.3. The layers in the Android architecture are:

- **Application layer**: this layer is the one used by end phone users. Applications can run simultaneously (multitasking) and they are developed in the Java programming language.
- **Application framework**: it is the software framework used for implementing the basic skeleton of an application running in the Android OS.
- **Libraries**: the available libraries are developed in C/C++ and they are instanced by a Java interface.

---

1 Portions of this page are reproduced from work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License [38].
• Android runtime: this environment is formed by two dependent components, a virtual machine and a set of core libraries. The virtual machine, Dalvik VM, operates as translator between the applications and the Android operating system. Different from software in conventional computers, Android applications run in its own process, with its own instance of Dalvik, and an Android device can run multiple instances of this virtual machine. The set of core libraries provides most of the functionalities that are available in the complete Java core libraries.

• The kernel: the base of the Android platform is a Linux kernel. It is used to communicate with the device’s hardware and it handles: device drivers, memory management, process management and networking.

2.3.2 Android Sensor API

Most of latest smartphones devices integrate sensors measuring motion, orientation, and environmental conditions. These set of sensors are useful in applications aiming to monitor positioning and three-dimensional device movement or to detect changes in the context and ambient environment near the device, since they provide raw data with reliable precision and acceptable accuracy.

The Android platform through the Android Sensor frameworks allows access to three wide categories of sensors:

• Motion sensors: sensors in this category measure rotational and acceleration along three components or axes. Accelerometers, gravity sensors, gyroscopes, and rotational vector sensors are classified in this category.

• Position sensors: orientation sensors and magnetometers are commonly included in this category as they provide raw data helping to estimate the physical position of a device.
• Environmental sensors: these sensors measure environmental parameters, such as illumination, humidity, ambient air pressure and temperature. The sensors included in this category are: photometers, barometers and thermometers.

The Android sensor framework lets developers access these three categories of physical components integrated in a handheld or tablet device. These hardware-based sensors obtain their data by directly measuring physical and environmental properties, such as acceleration variations, angular rotation change, magnetic field strength or barometric air pressure. Programmers have access to these sensors with the aid of the Android sensor framework and can manage some of their characteristics to acquire their raw data through various provided classes as `SensorManager`, `SensorEvent`, and `SensorEventListener`. The Android sensor API or framework is part of the general package Android hardware which its architecture layers are shown in Figure 2.4.

![Android sensor API layers](image)

Figure 2.4 Android sensor API layers.
2.4 Related Works

There is a large body of literature on indoor positioning systems. A comprehensive coverage is provided in surveys related to inertial systems [7] and wireless positioning systems [8] [9].

The set of solutions available can be classified as: lateration and angulation systems, proximity systems, radio fingerprint systems, dead-reckoning systems, and hybrid systems.

Among the current technologies, the lateration and angulation methods are probably the most complex to deploy and expensive to maintain in terms of infrastructure. Similarly to the GPS principle for position estimation, the system functionality is based on the computation of distances between the mobile unit (usually requiring specialist location tags to be carried by users) and an array of base stations installed in the building at known locations. Careful choice of the beacon signal can contribute to the difficulties of radio signal propagation that arise due to indoor obstacles. The most common signal types are ultrasonic systems [10], radio frequency-based systems [11], and ultra wideband (UWB) radio systems [12]. Zhao et al. propose AUITS: the autonomous ultrasonic indoor tracking system [13], an ultrasound-based system for locating and tracking mobile objects inside a building. The results of this work show that the coverage area for estimating the location of one device can reach until 65 $m^2$ obtaining a positioning error less than 15 cm with over 90% probability. However, as most of the conventional ultrasonic location systems, it poses some challenges such as manual calibration of the transmitters, high installation cost, antenna mismatch, external interferences from other systems and low power emission.

Proximity systems are another common alternative for indoor positioning. Mobile stations include detectors which recognize signals transmitted by a proximity system. Since the
location of the transmitters is known, true location of the mobile device is easily obtained. Due to the short range of the signals, these systems provide only the nearest room or building area (a coarse location) rather than providing coordinate location. Examples include Bluetooth stations, Radio-Frequency Identification (RFID) systems and Near Field Communication (NFC) infrastructures. Bluetooth is a wireless communication technology for the exchange of data in a short range. The position accuracy is proportional to the number of cells used and it requires high precision receivers [14]. In the case of RFID technology, position accuracy also depends on the amount of tags used and the type of these tags, which can be either active or passive. Proposed RFI-based indoor navigation solutions require an extensive usage of tags to get a reliable position and they are generally based on active RFID tags. Actives tags increase the transmitting distance compared with passive tags, since they include batteries in order to increase the transmitting power. The main disadvantage of the solutions based one active RFID tag is the high cost of the transmitters. Furthermore, studies in this approaches [15] suggest that they do not provide an efficient tracking system. Similarly, NFC solutions [16] have the drawback of requiring a high amount of readers to obtain reliable ubiquitous coverage, except that NFC devices works with little calibration.

According with the survey studies, the most successful indoor systems to date are those based on radio fingerprinting. In these approaches, a radio map of various signal properties such as received signal strength is previously collected and compared to the current measurements. The closest match is searched and identified as the estimated position. Wi-Fi is the most common radio fingerprinting choice due to its ubiquity [17] [18] [19]. The Wi-Fi –based system typically report accuracies of a few meters. However, the time required to install, configure and maintain
these systems together with the expense of access points have so far limited the general deployment of these indoor systems.

Other popular set of systems are independent navigation systems based on Pedestrian Dead Reckoning (PDR) techniques. In contrast with providing coordinate location, a common method for human tracking is to calculate the current position based on the last estimated position, the speed of the item, the route, and the elapsed time between the current and the previous position. Recently, PDR systems used Micro Electro-Mechanical Sensors (MEMS) and inertial sensors embedded in cell phones. These systems are capable of computing their own positions and they require very basic, physical infrastructure to work. Furthermore, MEMs based systems usually offer an additional degree of privacy since the user can choose either share or not its location information with any third party. The most important drawback of PDR-based navigation systems is the need to correct the noise associated with the sensors when the estimation of the new position is based on previous PDR calculation. Previous works on inertial sensor-based user tracking have employed bare functional locations for mounting the sensor or provided low accuracy. Some of them limit the usage of sensors, use additional infrastructures to handle this inherent problem, or require previous knowledge of the indoor map. Robertson et al. [21] proposed the use of accurate foot-mounted inertial sensors for tracking pedestrians in indoor environments. These approaches provide the direction and the displacement of the users but the sensor collocation is limited to an unnatural position in order to obtain a reliable sensor raw data. The inertial sensor’s distance traveled can be calculated from the acceleration signal by double integration with respect to time; however, as consequence of the low accuracy of the accelerometer, the presence of noise and the component of acceleration caused by gravity, error accumulates rapidly with time [22]. An interesting approach shown by Constandache et al. in
[23] takes advantage of digital compass and accelerometer in smartphones to track pedestrians. The system is designed and tested for outdoor environments where map information is available. It compares the estimated path with the true map information without requiring any external extra device. Alzantot et al. [24] show how a step counter for tracking pedestrians can be created using exclusively the inertial sensors built-in a cell phone. They use dead reckoning navigation techniques combined with lightweight finite state machines to obtain an acceptable accurate level.

Additionally, recent developments in Pedestrian Dead-Reckoning systems have demonstrated the ability of these systems in urban sensing and activity recognition. For instance, several inertial sensors worn simultaneously on different parts of the body can detect when a user is walking, turning into a corridor, or climbing up the stairs [25]. Similarly, microphones and magnetometers can be used to detect ambient sounds and magnetic fluctuations [26] [27]. While these signatures have been primarily used for various forms of context awareness, they can contribute to localization purposes as well. The signatures can be treated as landmarks, and are useful for indoor dead-reckoning systems when combined with sensor information in order to recognize indoor points of interest and user’s movements.

All these techniques for indoor positioning systems are not independent and several hybrid systems are also found in the literature. Lateration and angulation systems are often combined with other indoor positioning techniques to improve the global performance. For instance, in [28], Jin et al. propose the use of the digital compass and the accelerometer in a smartphone to track user location in indoor environments. The commercial system, called SparseTrack, trusts on an additional ultrasonic sensor, which is sparsely distributed a priori in the area, to correct the current location and the possible adjust the possible error provided by the
smartphone. Building RF radio maps, similar to Wi-Fi is another common approach implemented to solve the issues of a pedestrian tracking system based on foot-mounted inertial sensors [29]. The typical scheme uses a particle filter and the known radio-frequency map information as one of the parameters for the filter, to track the user and to improve the positioning estimated by the basic INS system. In [30], Tomé et al. present an almost self-deployable solution based on RFID tags inertial Micro Electro Mechanical Sensors. Capelle et al. [31] designed a GNSS–based multisensory system based on the fusion of three different technologies: High Sensitivity GNSS (GPS and the future Galileo), MEMS-based Pedestrian Navigation System and WI-FI. Proposed in [32] Woodman et al. implement a pedestrian localization system for indoor environments using a foot-mounted inertial unit and a localization algorithm using Wi-Fi signal strength to reduce initial complexity. All these hybrid designs partially solve the indoor localization problem. However, all of them require offline training in order to build a radio map or the installation of additional infrastructure. Requiring infrastructure beyond the common mobile phone can make a solution impractical for several kinds of scenarios and it will undermine the use of these solutions in real environments. Comparing this thesis with the current literature, the presented work focuses on the integration of context information collected by sensors embedded in mobile phones only. The sensing capabilities of the smartphones are exploited to provide alternative tracking techniques which improve the classic localization methodologies. No external sensors or infrastructure are needed and the data acquisition is transparent to the user. The cumulative errors, as a consequence of using low quality inertial sensor and PDR techniques, are reduced by splitting and resetting the complete trajectory in small independent traces. Altogether, the proposed method provides an acceptable accuracy at a low cost.
CHAPTER 3: SYSTEM DESIGN AND IMPLEMENTATION

This chapter presents the design of the proposed system, its architecture and the details about its implementation.

3.1 System Design

This thesis proposes a new system to track pedestrians in indoor environments by automatically detecting landmarks and pedestrian motion traces where map data are not provided a priori. The system combines data from several sensors to achieve its final goal. It uses accelerometer readings of the mobile phone to record the number of steps/stairs a person has walked/climbed [33] [34] and accordingly obtain the distance traveled by the person. By utilizing the compass, the direction of the heading changes can be tracked [35]. Using magnetometer readings anomaly context behavior can be detected [36] and finally the barometer can perceive vertical movement patterns. Additionally, the proposed technique is based on resetting the accumulation of errors by splitting the complete trajectory into independent motion traces.

In order to model indoor human activities, the possible human actions are reduced to five states which correspond with the limitations that indoor obstacles and floor plans impose to the pedestrian’s indoor actions:

- **Turns (T):** when the pedestrian changes the heading in her/his route.
- **Stationary (Sy):** when the person remains in the same location for some time.
- **Elevator (E):** when the user makes use of an elevator to travel to a different floor.
- Walking (W): moving across the same floor.
- Stairs (St): where the user takes the stairs to go up/down to change the floors.

Consider for example the floor plan of the first floor map of the Fine Arts Hall Building at the University of South Florida shown in Figure 3.1.

![Floor Plan](image)

Figure 3.1 Real floor plan example.

This real location is basically formed by:

- Corridors: where users can walk straight (W).
- Corners: where pedestrians make heading changes (T).
- Stairs: where users climb up/down steps (St).
- Elevators: where user goes up/down floors automatically (E).
Stationary times (Sy) are also considered in which users remain in the same position for a defined period of time.

This set of predictable activities can be translated into identifiable context signatures or landmarks that can be specified using the data gathered by the sensors integrated in the mobile phone. For instance, elevators exhibit a remarkable variation in the magnetic field magnitude added to the pressure variation with the vertical movement. Human walking steps can be identified by a repetitive pattern in the accelerometer raw data, and heading changes can be detected from gyroscope measures. We take advantage of this approach to simultaneously harness sensor-based dead-reckoning and environment sensing.

The proposed system defines a set of rules to detect automatically the activity among the possible states defined above. These rules are based on two key concepts:

- **Indoor points of interest (POI):** multiple tests have shown that certain locations in indoor scenarios present identifiable signatures on one or more sensing dimensions. These signatures can be detected processing the raw data gathered by the sensors and translated into the real indoor points and events. This principle is used in our design to detect elevators and corners.

- **Human body behavior patterns:** pedestrian actions, like walking, generate repetitive and identifiable patterns that are ubiquitously detectable by the inertial sensors. For instance, human gait is defined as the way humans walk. Human walk is a bipedal and biphasic forward propulsion in which there are alternate sinuous movements describing a motion cycle. Different segments of the body are involved in the walk activity, mainly in the lower body. The gait cycle begins with the initial contact of the supporting heel on the ground and ends when same heel contacts the ground for a second time. Thus, it can be
classified in two phases: stance and swing (Figure 3.2). Each cycle begins at initial contact with a stance phase, (defined as the interval of time in which the foot is on the ground, approximately 60 percent of the gait cycle) and proceeds through a swing phase (defined as the interval of time in which the foot is not in contact with the ground approximately 40 percent of the gait cycle) until the cycle ends with the repetition of the initial contact. This idea is used in this system to detect human steps and climbing stairs.

![Figure 3.2 Human gait cycle. Public domain figure from [39]](image)

The approach of navigation based on landmarks and split by activity frames reduces cumulative error of the sensors. Furthermore, processing signal algorithms are applied over every single motion frame to include specific information, such as number of steps/stairs, time in the detected action, distance walked/climbed, elevator direction or turn direction. Thanks to this additional information the system is able to rebuild the user’s motion by a sequence of traces. This functionality is useful for pedestrian tracking, indoor positioning or eventually for participatory floor plan construction.

### 3.2 Architecture

The architecture of the system consists of four main parts or modules: the data collection module, the motion segmentation module, the activity recognition module, and the activity
specification module, as shown in Figure 3.3. This last module applies the specific algorithms (step detection and counting, stairs detection and counting and elevator frame classification).
3.2.1 Data Collection Module

The first module in the architecture of the system is responsible for gathering raw data from the various sensors integrated in the users’ mobile device. The readings collected can be treated locally in the device or buffered and sent lately to the server for processing.

Data collected are measurements from inertial sensors including: accelerometer, gyroscope, magnetometer and barometer. These set of sensors have the advantage of being ubiquitously embedded in most smartphones, having a low-energy footprint, and being always active during the phone operation with the goal of detecting changes in the orientation of the phone or helping in the location of the cell phone. This approach does not require a previous calibration in the module responsible of gathering the data, and the main challenge that needs to be faced is managing the noise that these low cost inertial sensors introduce. For testing the data collection module was developed as an application for an Android OS cell phone. The raw data has been collected every 15ms (66.6 Hz). This duty cycle is good enough to detect user’s activity and motion details as discussed in Section 4.2.3.

3.2.2 Motion Segmentation Module

Two events are considered in this segmentation module as separators to split the complete motion trace into independent segments. Heading changes, when pedestrians perform a turn in a corner, and altitude changes, when the user takes stairs or an elevator willing to change the floor.

3.2.2.1 Turn Detection Algorithm

Corners are a common occurrence in indoor scenarios and they can be used as splitters to force the segmentation of traces. Hence, an important event to be detected in indoor traces is the change on heading directions. Turns can be recognized based on the gyros sensor measurements
applying the algorithm explained below. It is based on significant changes in gyroscope readings. Turns are detected when compass identifies heading changes more significant than random oscillations. The algorithm performs a calibration routine to compensate the bias introduced by the sensor and applies a filter to reduce the background noise. It sets a threshold over the magnitude of the filtered signal to detect high variations and eventual turns. Finally, the turn’s direction is given by the sign of the compass reading with the biggest magnitude. The algorithm is based on the following steps:

1. Calibration routine: during a trace of movement, gyroscope samples are collected in the three axes \([g_x, g_y, g_z]\). Then, using Equation 6, the mean in the compass values for each axis is calculated.

\[
[\overline{Gyro}(x), \overline{Gyro}(y), \overline{Gyro}(z)] = \frac{1}{N} \left[ \sum_{i=0}^{i=N} g_{xi}, \sum_{i=0}^{i=N} g_{yi}, \sum_{i=0}^{i=N} g_{zi} \right]
\]  

(Equation 6)

where \(N\) means the number of samples used for the calibration routine for one a movement trace.

The magnitude of the averages is calculated and considered as bias to compensate and shift the compass variations to zero (Equation 7).

\[
\overline{Gyro}_{calib} = \sqrt{\overline{Gyro}(x)^2 + \overline{Gyro}(y)^2 + \overline{Gyro}(z)^2}
\]  

(Equation 7)

2. Moving average filter: using Equation 8a, the energy of the compass samples \((g_i)\) for every sample \(i\), is computed.

\[
gi = \sqrt{gx_i^2 + gy_i^2 + gz_i^2}
\]  

(Equation 8a)
And Equation 8b applies a moving average filter by estimating the average of energy in a window of size $\omega$ (10 samples) and compensating the bias previously calculated by Equation 7.

$$
\bar{g}_t = \frac{1}{2\omega + 1} \sum_{j=l-\omega}^{j=l+\omega} (g_j - \overline{\text{Gyro}\_\text{calib}})^2
$$

(8b)

3. Threshold: the threshold defined in Equation 9, which generates a square wave to detect the heading changes based on the high and low levels of the signal.

$$
G_T = \begin{cases} 
T & \text{if } \bar{g}_t > T \\
0 & \text{otherwise}
\end{cases}
$$

(9)

After multiple empirical tests, the threshold value $T$ has been fixed to 1.2 rad. Thus, a high level in the signal $G_T$ means that the compass headings changed more than $T$ during an interval of 10 samples (or 150ms with a sampling time of 15ms).

4. Turn detection: a turn is detected when the square signal $G_T$ shows a period of high level followed by a low level. In other words, a transition from low to high level is detected in $G_T$ ($G_{T_{i-1}} < G_{T_i}$) and samples later, a transition from high to low ($G_{T_{i-1}} > G_{T_i}$). The turn sample $i$ is estimated in the center value of this $G_T$ high level period.

5. Turn direction: once a turn is detected for sample $i$, in order to determine its direction it is necessary to study the sign of the compass reading with the biggest magnitude. The function $\text{MainComponent}$ (Equation 10) returns the biggest gyro compass component (maximum absolute value) in a window of size $\omega$ with center in the detected turning sample.
\[ \text{MainComponent}(i) = \max \{ \bar{g}_x, \bar{g}_y, \bar{g}_z \} \text{ with } \bar{g}_i = \frac{1}{2\omega + 1} \sum_{l=j-\omega}^{l=j+\omega} \text{abs}(g_i) \quad (10) \]

According to the sensor's coordinate system in the Sensors Android API (see Figure 4.1) and the standard mathematical definition of positive rotation, a rotation is positive when follows the counter-clockwise direction. It means that, an observer looking at a device positioned on the origin from some positive location on the x, y or z axis would report positive rotation if the device is rotating counter clockwise. Therefore if the MainComponent value for the detect turn sample is positive, a turn to the right has been detected. Otherwise, if the sign is negative, a turn to the left has been performed.

![Figure 3.4 Turn detection algorithm signals.](image-url)
The top Figure 3.4 represents the values of the signals as calculated by Equations 7, 8 and 9 in the turn detection algorithm, and the figure on the bottom shows the components of the compass readings to compute Equation 10 in a basic example. The trace motion in the example includes a turn to the right between samples 60-80, a left turn between samples 230-260, a turn to the right between samples 560-580, and right turn between samples 600-615.

### 3.2.2.2 Altitude Change Detection Algorithm

Altitude changes are critical to detect activities that involve a change of floor. For instance, it is useful to differentiate between walking in flat floors or stairs. Similarly to heading change, altitude change events can be used as splitters to force the segmentation of traces. These changes are analyzed by the altitude change detection algorithm. The algorithm implemented is based on significant changes in the air pressure raw data acquired by the barometer. Altitude changes are recognized when the barometric pressure values vary more significantly than due to random oscillations. The altitude change detection algorithm uses the following condition (Equation 11):

\[
\text{Avg} \left( P(t_{i+1}) \right) - \text{Avg} \left( P(t_i) \right) \geq \text{StDev}(P)
\]  

(Equation 11)

The first term represents the variation of the average between two consecutive intervals,

\[
\text{Avg} \left( P(t_i) \right) = \frac{1}{N} \sum_{j=0}^{j=N-1} \bar{P}_j
\]  

(Equation 12)

where \( \text{Avg}(P(t_i)) \), denotes the average of the low pass filter values for the air pressure readings over a \( t_i \) time period (Equation 12), and \( N \) is the number of samples included in the period \( t_i \). \( \bar{P}_i \), is the low pass filter signal of the pressure readings \( p_i \). It has been implemented using a discrete
implementation of a basic RC low-pass filter, as show in Equation 13, with a smoothing factor of \( \beta = 0.9 \).

\[
\bar{P}_t = LPF[P_] = \beta p_{t-1} + (1 - \beta) p_t
\]  

(13)

The result of applying a low-pass filter is a smoother form of the signal, removing the short-term oscillations and keeping the longer-term trend.

Figure 3.5 Altitude change detection algorithm signals.

The second term in the detection condition of the algorithm (Equation 11) identifies the pressure random oscillations. It is detected based on the average of the standard deviation estimated for the barometric pressure sensor in resting conditions (not moving), 0.05 mbar/sec. This value was obtained from the quality sensors test shown in Section 4.2.2 of this thesis. For the time intervals where a significant variation is detected, the trend of the pressure signal is analyzed to give a sense (up or down) to the pressure change and generate a square wave to
represent the altitude changes (+1 and -1, respectively). It worth noting that, a notable increase in the pressure signal means a decrease in altitude and vice versa.

Figure 3.5 represents the values of the signals in the altitude change detection algorithm for a basic example. The figure on the top left shows the pressure raw data; the one on the top right represents the pressures measurements after being computed by the low pass filter as calculated by Equation 13. The figure on the bottom left shows the difference of averages between two consecutives intervals as computed by Equations 11 and 12. Finally, the figure on the bottom right represents the altitude changes detected. In the example, a positive altitude change was detected in the interval between samples number 60 and 250, and a period of negative altitude, or going down, was detected in the interval between samples 540 and 740.

Once detected, turns and altitude changes events are used in the proposed solution to split the continuous motion traces in a sequence of independent segments. This trace segmentation technique reduces the error accumulation introduced by the inertial sensors. This approach contrasts with classical relative navigation techniques as dead reckoning, where the new location of a user is estimated using the previous location, the distance traveled and the direction of motion.

![Figure 3.6 Example of pedestrians states in indoor trace split by turns.](image)
Figure 3.6 shows an example of an indoor motion trace and sequence of activities split by Turns (T) and Altitude Changes (AC).

### 3.2.3 Activity Recognition Module

The aim of this module is to define a set of rules that automatically allows the system to detect the segment activity among the possible indoor states defined above (stationary, elevator, walking or stairs). This module receives the segments of motion generated by the previous modules and processes the sensor data to classify the segments and estimate the activity performed. Figure 3.7 shows the decision tree that has been defined to classify the possible states.

![Decision tree](image)

Figure 3.7 Decision tree to classify the possible indoor states.
The simplest way to produce useful data out of the inertial sensor is to calculate the magnitude of the signal considering the energy of three components data vector. For example, in order to estimate the magnitude of the acceleration, the energy represented in each sample provided by the accelerometer is calculated. After filtering the signal to make it smoother, the first decision is to differentiate between movements based on the magnitude level of acceleration. Fixing a threshold over the acceleration energy leads to estimate periods of activity and, complementary, segments on rest.

Analogous, other inertial sensor signals can be used to detect the activity subclasses in the decision tree, such as the magnitude of the magnetic fields, the pressure changes, the variance of accelerations or the correlation of the accelerations.

### 3.2.3.1 Elevator (E)

Similar to cars or planes, elevators behave like a Faraday shield presenting a unique magnetic field pattern that makes them distinguishable with accuracy. Since a typical elevator is a structure formed by conducting material, it blocks non-static electric fields and external static. The different values for the magnetic field energy coming from the outside and the inside of the elevator show a notable difference, thus this transition of states is easily identifiable (see Figure 3.8).

![Figure 3.8 Transition of magnetic field energy outside and inside an elevator.](image)

39
There are two additional details that have to be distinguished in the elevator motion segments: the direction of motion and the estimation of number of floors traveled.

The identification of the elevator direction can be estimated based on the energy of the acceleration measurements when the elevator starts and stops its travel. These events produce a pattern of acceleration peaks in the elevator segment and studying the order of appearance the elevator motion can be classified. If the pattern is a positive peak followed by a negative peak, a travel in up direction was performed (see Figure 3.9). Otherwise, the sequence showing a negative peak followed by a positive one will be detected as a down direction of motion.

![Energy of acceleration values for an up direction elevator travel.](image)

The number of floors traveled can be estimated with the aid of the displacement duration of the travel inside the elevator, considering the number of samples between the acceleration peaks detected.

Besides, for modern smart phones that include a barometric sensor, the analysis of its raw data and the detection of pressure changes helps enormously in the detection of altitude changes, thus in the recognition of elevator travels.
Figure 3.10  Air pressure variations for an up direction elevator travel.

As shown in Figure 3.10, an elevator travelling up means a notable decrease in the air pressure measures. The number of floors the elevator moved can be estimated considering the period of time where the barometer shows an abnormal variation. With these features, the classification of elevator samples and the additional details such as the direction or travel time are simplified.

3.2.3.2 Stationary (Sy)

For an indoor pedestrian segment two states of repose are considered: either the user is travelling in an elevator or stationary (it includes sitting or standing and no displacement performed during a period of time). Since these states are classified as exclusionary final states in the left branch of the tree (see Figure 3.7), if a segment is not detected as elevator it will be considered as a stationary segment.

3.2.3.3 Walking (W) and Stair (St)

Once the rest states have been discarded using the energy of the acceleration to detect active scenarios, it is necessary to differentiate the active segments between stairs and walking
cases. The initial observation is that when the pedestrians are taking the stairs the variance of the acceleration is broader than in the walking case. Figure 3.11 represents the variance of the acceleration’s energy in a Walking (W)–Stationary (Sy)–Stairs (St) motion sequence.

The correlation between the acceleration signals in the motion’s direction (y axis) and the gravity’s direction (z axis) is a good clue to separate stairs scenarios compared with walking. Furthermore, the measurements show that going down stairs (helped by the gravity acceleration) involves higher motion intensity than climbing up.

![Figure 3.11 Variance of energy acceleration in walking and stairs states.](image)

Besides, for mobile devices that include a barometric sensor, the detection of altitude changes can be useful to find the different active states and the direction of movement during stairs periods. As shown in Figure 3.12, the trend of air pressure measurement keeps relation with the type of activity performed. It can also be an accurate signature for the detection of walking down stairs or walking flat.
Combining these three processed signals (variance and correlation of accelerations and pressure changes) and using them as indicators, the three final active states can be detected also including the direction of the user in the stairs segments. On one hand, a segment with high variance in the acceleration, a high $YZ$ correlation in the acceleration and notable increases in air pressure will be detected as going down stairs. On the other hand, a high variance, a medium value of $YZ$ correlation and decreasing changes in air pressure will be detected as going up stairs. Finally, a low level in the variance and the $YZ$ correlation, combined with a stable level in the air pressure readings will classify the segment as walking.

3.2.4 Activity Specification Module

Once the activity recognition module has identified each segment and classified its activity, some of them, like walking or stairs, require additional algorithms to enable a complete tracking of the user. Processing the original signal, based on the accelerometer readings of the mobile phone, by specific algorithms is possible to derive the displacement of the person while walking or climbing the stairs.
In order to reduce the accumulation of errors, the pedometer basic method which estimates the displacement as the addition of the individual step sizes is extended. A step detection algorithm is applied to perceive the pattern that the magnitude of acceleration suffers when a step is performed.

3.2.4.1 Step Detection and Counting Algorithm

Thanks to the detection of cycles in the data gathered by the accelerometer (swing and stance phases) caused by the repeated patterns or events in motion of walking, it is possible to count the number of steps a person has walked, and therefore obtain an estimation of the distance traveled by the person.

Based on the experiments, the effect of walking on the magnitude of the acceleration vector is independent from the phone orientation and tilt. Consequently, our step counting algorithm is designed depending on the magnitude of the acceleration, making this approach for distance estimation independent from the placement of the mobile phone (messaging in hands, calling in user’s ear or swinging in the pocket). The algorithm performs a calibration routine that compensates the bias introduced by the sensor and applies an average filter to reduce the background noise. It sets a double threshold over the magnitude of the filtered signal to detect the stance and swing phases in the human gait. Finally, a step is detected when a transition between stance and swing phases is recognized.

The algorithm implemented for step detection and counting consists of the following steps:

1. Calibration routine: during a segment of movement, samples of linear acceleration $[Acc_x, Acc_y, Acc_z]$ are collected to estimate the mean of the acceleration (see Equation 14)
\[
[\overline{Acc}(x), \overline{Acc}(y), \overline{Acc}(z)] = \frac{1}{N} \left[ \sum_{i=0}^{i=N} Acc_{x_i}, \sum_{i=0}^{i=N} Acc_{y_i}, \sum_{i=0}^{i=N} Acc_{z_i} \right]
\]

(14)

where \(N\) is the number of samples used in the calibration routine for one segment.

The energy of the averages is calculated using Equation 15 and considered as bias to compensate.

\[
\overline{Acc}_{\text{calib}} = \sqrt{\overline{Acc}(x)^2 + \overline{Acc}(y)^2 + \overline{Acc}(z)^2}
\]

(15)

2. Mean of accelerations: compute the energy of the acceleration \(Acc_i\) for every sample \(i\), as shown in Equation 16.

\[
a_i = \sqrt{Acc_{x_i}^2 + Acc_{y_i}^2 + Acc_{z_i}^2}
\]

(16)

Estimate the average of energy in a window of size \(\omega\) (10 samples) and compensate the bias (see Equation 17).

\[
\overline{a_w} = \frac{1}{2\omega + 1} \sum_{j=i-\omega}^{j=i+\omega} (a_j - \overline{Acc}_{\text{calib}})^2
\]

(17)

3. Threshold: a first threshold \(T_1\) is applied following the rule in Equation 18 to detect the high accelerations during the swing phase.

\[
B_{1_i} = \begin{cases} 
T_1 & \text{if } \overline{a_w} > T_1 \\
0 & \text{otherwise}
\end{cases}
\]

(18)

A second threshold \(T_2\), as defined by Equation 19, is utilized for the detection of the walking stance phase.

\[
B_{2_i} = \begin{cases} 
T_2 & \text{if } \overline{a_w} < T_2 \\
0 & \text{otherwise}
\end{cases}
\]

(19)
$T_1$ and $T_2$ are symmetric values respect to 0 and fixed heuristically to ±0.5 m/s², meaning that the acceleration varies more than $T_1$ or less than $T_2$ during an interval of 150 ms or 10 samples.

4. Step detection: a step is recognized in the sample $i$ when a swing phase ends and stance phase starts. For a step detection two sequential conditions must be accomplished:
   1) A change from high to low acceleration $B_{1_{i-1}} > B_{1_i}$, and
   2) there is at least one detection of a low level of acceleration in a window of size $\omega$ ahead of current sample $i$, i.e., $\min (B_{2_{i:i+\omega}}) = T_2$.

5. Finally, the vector with the steps samples is iterated to obtain the average time between steps, which could be useful to estimate the step rate or the velocity of displacement of the person.

![Figure 3.13 Step detection and counting algorithm signals.](image)

Figure 3.13 shows the signals as a result of applying Equations 16, 17, 18 and 19 in the step detection and counting algorithm applied in a basic walking example. Once the number of steps is obtained, the total distance walked can be directly estimated considering the stride length of each step to be constant and with a value of 0.74 m [24].

46
3.2.4.2 Stairs Detection and Counting Algorithm

The algorithm performs a calibration routine to compensate the bias and applies a low pass filter over the energy of the accelerations to reduce the background noise. Finally, it looks for peaks in the signal and applies a guard factor to discard minor variations and detect only the peaks that are eligible as stairs. The algorithm implemented for stairs detection and counting based on peak detection consists of the following steps:

1. Calibration routine: during a segment, samples of linear acceleration \([Acc_x, Acc_y, Acc_z]\) are collected to estimate the mean in the acceleration. The energy of the average is estimated and considered as bias to compensate, similar to the step detection algorithm in Equation 14 and Equation 15.

2. Energy of acceleration: the simplest way to produce useful data out of the three components of the sensor is to take the magnitude of the acceleration vector. It computes the energy of the acceleration \((Acc_i)\) for every sample using Equation 16.

3. Low pass filter and bias compensation: low-pass filters provide a smoother form of the signal, removing the short-term fluctuations, and leaving the longer-term trend. \(LPF[Acc_i]\) is the discrete low pass filter signal of the acceleration energy readings. It has been applied using a discrete-time implementation of a simple RC low-pass filter as show below in Equation 21, with a smoothing factor of \(\alpha=0.9\).

\[
LPF[Acc_i] = \alpha a_{i-1} + (1-\alpha) a_i
\]  

(20)

To compensate the bias, the value \(\overline{Acc_{\text{calib}}}\) is removed for all the energy filtered samples (Equation 21).
\[ Energy[A_{ci}] = LPF[A_{ci}] - \overline{A_{calib}} \quad (21) \]

4. Peak Detection: a peak is detected if during the \( \Omega \) previous samples (\( \Omega = 5 \) meaning 75ms) the backwardSlope of the current sample is positive and in the next sample the forwardSlope becomes negative. These two functions are detailed in Equation 22.

\[ \text{forwardSlope} = Acc[k + 1] - Acc[k] \quad \text{and} \]

\[ \text{backwardSlope} = Acc[k] - Acc[k - 1] \quad (22) \]

5. Stairs detection: the energy signal is traversed by a buffer of a fixed number of samples. In this implementation, the buffer length is 100 samples equal to 1.5 sec. For every set of samples in the buffer, peakMean (Equation 23) is calculated estimating the energy of the detected peaks.

\[ \text{peakMean} = \frac{1}{\#\text{Peaks}} \sum_k \text{Energy}[A_{ck}] \quad (23) \]

where \( k \) is the samples detected as peaks and possible stairs.

After it is multiplied by a guard factor (\( G=0.6 \)) to avoid the detection of eventual secondary peaks, and the new value \( C \) (Equation 24) is the threshold for each set of samples.

\[ C = \text{peakMean} \times G \quad (24) \]

The peaks detected in each buffer have effect on the responsiveness of the algorithm changing the value of the threshold \( C \).

The final step of the algorithm to detect stairs iterates over the \( Energy[A_{ci}] \) and detects the peaks that are above the threshold.
Figure 3.14 shows the signals taking part in the stairs detection and counting algorithm applied in a basic climbing down stairs example.

The total altitude climbed can be estimated considering the number of steps and the maximum stair riser heights. This value is regulated and fixed to 7 inches (178 mm) by the International Building Code (IBC) [37]. The IBC is a model building code which has been adopted throughout most of the United States. It is developed and maintained by a standards organization independent of the jurisdiction responsible for enacting the building.

![Figure 3.14 Stairs detection and counting algorithm signals.](image)

3.3 System Implementation

The system implementation has been divided in two main parts (Figure 3.15): the mobile application (called SensorApp), which is in charge of gathering the data from the mobile device and displays the values on the client device; and the server, which applies the activity recognition decisions and the signal processing algorithms, saves the traces, provides tools to introduce the experiment for testing, and displays the evaluation results.
3.3.1 Client Side: Mobile Application

The data acquisition task from the sensors embedded in the smartphone has been implemented through a mobile Android application using the Android Sensor API.

As shown in Figure 3.16, two possible visualizations are included in the mobile application. Both show the raw data collected from the inertial built-in sensors available in the
device: accelerometer, gyroscope, magnetometer and barometer. The basic layout shows several text views where the values of the raw data collected are displayed. The second view represents the sensor values in a graph where the $x$ axis shows time and $y$ axis can be: $m/s^2$ for the acceleration, $rad/s$ for the rotation, $\mu T$ for the magnetic field or $mbar$ in the case of the air pressure. The $y$ axis scale is adapted to the current values to give a better resolution of the values. Additionally, the mobile application is in charge of transmitting the information packets to the server using a web service when the motion test finishes.

### 3.3.2 Server Side

The server side has been designed as the main module of the system. It aims to be a test environment to support and simplify the system’s implementation and the evaluation process. It allows to create experiments for different real scenarios and to compare them with indoor motion sequences obtained from the tests performed. Thus, it automates much of the testing process and makes simpler the evaluation of the results. The server performs the following tasks:

- **Receiving data:** it offers a web service to receive the set of raw data sent by the mobile devices at the end of every single test.
- **Data processing algorithms:** the different modules explained in the system design are accomplished by several synchronized threads. These algorithms will return the activities detected and their details.
- **Persistence:** the system saves the information returned by the processing algorithms for future references, queries and evaluations. Also, it stores the experiments to be tested.
- **Experimentation:** the server provides a straightforward way to create new motion sequences for the experiments and also an identification to link the experiments with its corresponding tests.
• Presentation of results: it offers a web application to visualize and represent the activity decision for the sequence of motions and the specifications for every single trace tested. Then, it compares the test results with the real traces and computes statistical results for evaluation. Figure 3.17 shows a screenshot of the server application presenting the results of one experiment.

![Server web application screenshots](image)

Figure 3.17 Server web application screenshots.

### 3.3.3 Development and Deployment Tools

The software resources and tools that have been used in the development of the system are presented in this section. All of them are free software.
• Eclipse: it has been the integrated development environment (IDE) used for the server and client development.

• Android Development Tools (ADT) is a well-known plugin for the Eclipse IDE that integrates the environment to build Android applications.

• Apache Struts2: is an open-source framework for developing web applications in Java EE. It adopts the model view–controller (MVC) using and extending the Java Servlet API.

• Bootstrap: is a free front-end framework to create websites and web applications. It contains several HTML and CSS based design templates useful for different interface components.

• Apache Tomcat7: is an open source web server working also as servlet container that includes tools for configuration and management.

• Hibernate: is an object-relational mapping library for the Java language. It provides a framework for mapping object-oriented domain models to a traditional relational databases.

• MySQL5.6 and MySQL Workbench: management system and management tool for databases.

• Additional Libraries: Jersey, jQuery, Google Charts, Gson and GraphView.

Figure 3.18 shows the deployment diagram for the server side of the system. Both, the applications server (Apache Tomcat7) and the database server (MySQL5.6) have been deployed over one single web server. Besides, the web browser to display the web site, and the android client which provides the source data, are connected via TCP/IP to the web server.
Figure 3.18 Web server deployment diagram.
CHAPTER 4: EVALUATION

In this chapter, the performance of the proposed system is evaluated with several basic tests and complete motion traces involving different activities. The chapter starts describing the methodology applied, followed by the details of the mobile devices used for testing. A set quality sensors tests for the inertial sensors are performed and an evaluation of all the modules and algorithms involved in the system is presented. The chapter concludes with the discussion of the tests results.

4.1 Methodology

The client side of the presented system was implemented using the Android platform using two different devices: the Samsung Galaxy SII I777 and the LG Nexus 4 E960, which are equipped with the set of sensors detailed in previous chapters. The set of experiments were carried out in different buildings in the University of South Florida campus, with plenty of corridors, corners, stairs and elevators. To perform an evaluation the system requires three steps:

1. Set the experiment features in the server graphical interface.
2. Perform the test with the mobile application identifying the experiment.
3. Compute the results and show them to the user.

Around 200 motion traces were collected in total where the user was carrying the phone in hand, in texting position. These experiments aim to evaluate the performance of the proposed approach by following the architecture presented in the design of the system. Initially, the data
collection performance is evaluated analyzing the devices and the quality of its sensors. Also the convenience of the chosen sampling frequency is demonstrated. Then, the algorithms detecting turns and altitude changes are evaluated. These algorithms allow the segmentation of original traces. The evaluation continues studying the results of the activity recognition module, in charge of classifying the segments in one of the possible final states. The results for the specification module are also shown. They include the analysis of the algorithms for detecting and counting steps and stairs. Finally, the combined tracking accuracy, when applying the complete architecture of the system, is presented.

4.2 Data Collection Results

4.2.1 Devices

Two Android OS smart phones from different manufacturers has been used in the evaluation of the system: a Samsung Galaxy SII I777 and a LG Nexus 4 E960.

4.2.1.1 Samsung Galaxy SII I777

The AT&T Samsung Galaxy SII I777, also known as Samsung SGH-I777 is the initial release for the Galaxy S II line of smartphones on AT&T's network. This model was released in October 2011. It keeps the 4.3” display seen in the international version. It uses a dual-core Exynos 4 processor clocked at 1.2 GHz. The AT&T Samsung Galaxy S II SGH-I777 uses an 8.0 MP rear camera, as well as a 1.9 MP front-facing camera. The cell phone is running Android version 4.1.2 (Jelly Bean). Table 4.1 details the Samsung Galaxy S II I777 specifications.
Table 4.1 Samsung Galaxy S II SGH-I777 specifications.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimensions</strong></td>
<td>126 x 66 x 8.9 mm (4.96 x 2.60 x 0.35 in)</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>121.9 g (4.27 oz)</td>
</tr>
<tr>
<td><strong>Processor</strong></td>
<td>Exynos C210 Dual-core 1.2 GHz Cortex-A9</td>
</tr>
<tr>
<td><strong>Operating System</strong></td>
<td>Android OS, v4.1.2 (Jelly Bean)</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>16GB storage, 1 GB RAM</td>
</tr>
<tr>
<td><strong>Display</strong></td>
<td>Super AMOLED Plus capacitive touchscreen</td>
</tr>
<tr>
<td></td>
<td>16M colors</td>
</tr>
<tr>
<td></td>
<td>480 x 800 pixels, 4.3 inches</td>
</tr>
<tr>
<td><strong>Connectivity</strong></td>
<td>GSM 850 / 900 / 1800 / 1900</td>
</tr>
<tr>
<td></td>
<td>HSDPA 850 / 1900 / 2100</td>
</tr>
<tr>
<td></td>
<td>Wi-Fi 802.11 a/b/g/n</td>
</tr>
<tr>
<td></td>
<td>Bluetooth v3.0 with A2DP, HS</td>
</tr>
<tr>
<td><strong>Battery</strong></td>
<td>Li-Ion 1650 mA battery</td>
</tr>
<tr>
<td><strong>Sensors</strong></td>
<td>Accelerometer, gyro, proximity, compass, magnetometer</td>
</tr>
</tbody>
</table>

4.2.1.2 LG Nexus 4 E960

The LG Nexus 4 is the fourth generation of Google’s flagship phones. It was manufactured by LG and designed in partnership with Google. The LG Nexus 4 model E960 was available in the market in November 2012. It has a 4.7” display with True HD IPS screen technology. It uses a quad-core Snapdragon chip by Qualcomm running up to 1.5 GHz.

The Nexus 4 includes two cameras: the primary with a sensor of 8.0 MP and the secondary camera in the front with 1.3 MP. It also sports some additional features as inductive charging and NFC chip. This smart phone is running the last Android “pure” version 4.4.2 (KitKat). Table 4.2 details the main LG Nexus 4 specifications.

The main reason for choosing this device is the large number of built-in sensors, which fulfill the hardware requirements of the client side for the system proposed in this thesis. It also offers a set of great features for an affordable price.
Table 4.2  LG Nexus 4 E960 specifications.

<table>
<thead>
<tr>
<th></th>
<th>LG Nexus 4 E960</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>133.9 x 68.7 x 9.1 mm (5.27 x 2.70 x 0.36 in)</td>
</tr>
<tr>
<td>Weight</td>
<td>139 g (4.90 oz)</td>
</tr>
<tr>
<td>Processor</td>
<td>Qualcomm APQ8064 Snapdragon Quad-core 1.5 GHz Krait</td>
</tr>
<tr>
<td>Operating System</td>
<td>Android OS, v4.4.2 (KitKat)</td>
</tr>
<tr>
<td>Memory</td>
<td>16GB storage, 2 GB RAM</td>
</tr>
<tr>
<td>Display</td>
<td>True HD IPS Plus capacitive touchscreen 16M colors 768 x 1280 pixels, 4.7 inches</td>
</tr>
<tr>
<td>Connectivity</td>
<td>GSM 850 / 900 / 1800 / 1900 HSDPA 850 / 900 / 1700 / 1900 / 2100 Wi-Fi 802.11 a/b/g/n dual-band, DLNA Bluetooth v4.0 with A2DP, NFC</td>
</tr>
<tr>
<td>Battery</td>
<td>Li-Po 2100 mA battery</td>
</tr>
<tr>
<td>Sensors</td>
<td>Accelerometer, gyro, proximity, compass, magnetometer, barometer</td>
</tr>
</tbody>
</table>

4.2.2 Inertial Sensors Quality Tests

This section performs an error analysis of the low quality inertial sensors considered as data sources in the system. The inertial sensors are tested to study the long term errors and to consider the necessity of calibration to obtain more reliable outputs. The set of built-in inertial sensors embedded in the Samsung SII are: a 3-axes accelerometer, a gyroscope and a magnetometer, while the LG Nexus 4 also integrates a barometric pressure sensor.

To analyze the accuracy and the behavior of the sensors, stationary tests where the device was laying on a table, were performed. The output of the sensors is relative to the device’s orientation, here referred to the device’s coordinate system, and they are not swapped when the device's screen orientation changes. The coordinate reference system use by the Android Sensor
API defines the default orientation shown in Figure 4.1. The x axis points horizontally to the right, the y axis points up and is vertical, and the z axis points outside from the device screen.

![Figure 4.1 Coordinate reference system used by the sensor API.](image)

In these error analysis tests the sensor samples are recorded during a period of 15 sec with the phone lying flat with its back on the table (z axis pointing to the sky). The phone is immobile in order to prevent any force other than gravity from affecting the output.

### 4.2.2.1 Accelerometer

The accelerometer measures the acceleration in three axes in m/s². It outputs the acceleration applied to the device by measuring forces applied to the sensor. The measured acceleration is always influenced by the force of the earth’s gravity as shown in Equation 25.

\[ a_d = -g - \sum \frac{F}{m} \]

where \( a_d \) is the acceleration applied to the device, \( g \) is the force of gravity, \( F \) is the force acting on the device, and \( m \) is the mass of the device. The sign \( \Sigma \) represents the sum of the x, y and z axes.
As a result, when the device is not undergoing any acceleration, the accelerometer output should read \(0 \text{ m/s}^2\) for \(x\) and \(y\) axes, and negative Earth’s gravity of \(9.81 \text{ m/s}^2\) for the \(z\) axis.

Figure 4.2 shows the phone’s actual acceleration output for the \(x\), \(y\) and the \(z\) axes while being stationary on a table. The figure on the top shows the signals for the Samsung Galaxy SII device and the one on the bottom for the LG Nexus 4. The accelerometer test results (Table 4.3) show that the magnitude of the acceleration measured at stationary position were on average about \(9.46 \text{ m/s}^2\) and \(10.35 \text{ m/s}^2\) respectively and not the expected \(9.81 \text{ m/s}^2\). The standard deviation for the Samsung’s smartphone was \(0.13 \text{ m/s}^2\), it is equivalent to more than one percent of the total acceleration, which, with time, could potentially generate a large error. These errors are compensated in the signal processing algorithms implemented in the system performing initial calibration methods.
Table 4.3 Accelerometer test results.

<table>
<thead>
<tr>
<th></th>
<th>Samsung Galaxy SII</th>
<th>Average</th>
<th>Max value</th>
<th>Min value</th>
<th>Std deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc, m/s²</td>
<td>0.20466</td>
<td>0.38137</td>
<td>-0.05448</td>
<td>0.02999</td>
<td></td>
</tr>
<tr>
<td>Acc, m/s²</td>
<td>-0.27933</td>
<td>-0.16344</td>
<td>-0.43585</td>
<td>0.03228</td>
<td></td>
</tr>
<tr>
<td>Acc, m/s²</td>
<td>9.50077</td>
<td>10.14716</td>
<td>8.85322</td>
<td>0.07012</td>
<td></td>
</tr>
<tr>
<td>Magn. Acc, m/s²</td>
<td>9.42610</td>
<td>10.36508</td>
<td>8.36289</td>
<td>0.13240</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LG Nexus 4</th>
<th>Average</th>
<th>Max value</th>
<th>Min value</th>
<th>Std deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc, m/s²</td>
<td>0.196125</td>
<td>0.20060</td>
<td>0.175613</td>
<td>0.03809</td>
<td></td>
</tr>
<tr>
<td>Acc, m/s²</td>
<td>-0.30517</td>
<td>-0.27171</td>
<td>-0.31099</td>
<td>0.09734</td>
<td></td>
</tr>
<tr>
<td>Acc, m/s²</td>
<td>10.34479</td>
<td>10.36684</td>
<td>10.32756</td>
<td>0.04519</td>
<td></td>
</tr>
<tr>
<td>Magn. Acc, m/s²</td>
<td>10.35168</td>
<td>10.69994</td>
<td>9.88187</td>
<td>0.04543</td>
<td></td>
</tr>
</tbody>
</table>

4.2.2.2 Gyroscope

The gyroscope readings are in radians per second and measure the rate of rotation around the x, y and z axis. Rotation is positive in the counter-clockwise direction. When the device is at rest on a table and not moving, the gyroscope values should read a magnitude of 0 rad/s.

Figure 4.3 (top figure for the Samsung device and the one on the bottom for the LG) shows the measured angular speed around the x, y, and z axes when the devices are stationary on the table. Analyzing the results in Table 4.4, the rotation sensor introduce an offset or bias, calculated as the standard deviation of the gyroscope outputs magnitude when it is not undergoing any rotation.
Table 4.4 Gyroscope test results.

<table>
<thead>
<tr>
<th>Samsung Galaxy SII</th>
<th>Average</th>
<th>Max value</th>
<th>Min value</th>
<th>Std deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_x$ [rad/s]</td>
<td>0.02209</td>
<td>0.03787</td>
<td>0.00305</td>
<td>0.00549</td>
</tr>
<tr>
<td>$\omega_y$ [rad/s]</td>
<td>-0.00231</td>
<td>0.04184</td>
<td>-0.08247</td>
<td>0.00563</td>
</tr>
<tr>
<td>$\omega_z$ [rad/s]</td>
<td>-0.00119</td>
<td>0.01191</td>
<td>-0.01466</td>
<td>0.00464</td>
</tr>
<tr>
<td>Magn. $\omega$ [rad/s]</td>
<td>0.01858</td>
<td>0.09163</td>
<td>-0.09407</td>
<td>0.01577</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LG Nexus 4</th>
<th>Average</th>
<th>Max value</th>
<th>Min value</th>
<th>Std deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_x$ [rad/s]</td>
<td>-0.00024</td>
<td>0.00195</td>
<td>-0.00232</td>
<td>0.00060</td>
</tr>
<tr>
<td>$\omega_y$ [rad/s]</td>
<td>0.00006</td>
<td>0.00851</td>
<td>-0.00643</td>
<td>0.00076</td>
</tr>
<tr>
<td>$\omega_z$ [rad/s]</td>
<td>0.00007</td>
<td>0.00383</td>
<td>-0.00151</td>
<td>0.00064</td>
</tr>
<tr>
<td>Magn. $\omega$ [rad/s]</td>
<td>0.00099</td>
<td>0.00875</td>
<td>0.00048</td>
<td>0.00068</td>
</tr>
</tbody>
</table>

To calculate the angle of rotation $\alpha$ (rad), the deviation for the angular speed of the gyroscope $\omega$ (rad/s), should be integrated over time $t$ as shown in Equation 26.
\[ \alpha = \sum (\omega \cdot \Delta t) \]  

During a 15 seconds test, an error of $0.236 \text{ rad}$ was estimated for the Samsung Galaxy SII and $0.0102 \text{ rad}$ for the LG Nexus 4. Since the expected error average is zero, the calculated values are approximately equal to the bias error of the sensor. This error is compensated in the gyroscope-based algorithms in the system.

### 4.2.2.3 Magnetometer

The magnetometer measures the strength of the ambient magnetic field in micro-Tesla ($\mu T$), in the $x$, $y$ and $z$ axes. Ideally, a magnetometer completely isolated should measure a magnitude of $0 \mu T$. To estimate the bias error introduced by this sensor, a study of the standard deviation long term errors is performed in this test.

![Output signals magnetometer quality test.](image-url)
Figure 4.4 shows the $x$, $y$ and $z$ axis magnetic field signals collected during a test of 15 sec on repose state for both analyzed devices (Galaxy SII on the top and Nexus 4 on the bottom). Statistics results from the test are shown in Table 4.5.

Since the measurements are not centered in $0 \mu T$, because the magnetometer is not isolated, the average standard deviation is the only value useful to analyze the sensitivity of the sensor. It results in a rather high standard deviation with an estimated value of $0.402 \mu T$ and $0.275 \mu T$ for the Galaxy SII and the Nexus 4, respectively. Due to this deviation, calibration and signal filtering routines are applied to the magnetometer raw data to reduce and compensate the eventual sensor errors in the system.

Table 4.5 Magnetometer test results.

<table>
<thead>
<tr>
<th>Samsung Galaxy SII</th>
<th>Average</th>
<th>Max value</th>
<th>Min value</th>
<th>Std deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Mag_x[\mu T]$</td>
<td>-3.99611</td>
<td>-1.08</td>
<td>-5.52</td>
<td>0.916693</td>
</tr>
<tr>
<td>$Mag_y[\mu T]$</td>
<td>-7.52068</td>
<td>-6.42</td>
<td>-8.76</td>
<td>0.413968</td>
</tr>
<tr>
<td>$Mag_z[\mu T]$</td>
<td>-50.32660</td>
<td>-49.21</td>
<td>-51.36</td>
<td>0.39095</td>
</tr>
<tr>
<td>Magn. Mag[\mu T]</td>
<td>-51.05188</td>
<td>-49.93</td>
<td>-52.13</td>
<td>0.40299</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LG Nexus 4</th>
<th>Average</th>
<th>Max value</th>
<th>Min value</th>
<th>Std deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Mag_x[\mu T]$</td>
<td>-4.02902</td>
<td>-3.17993</td>
<td>-4.85992</td>
<td>0.27377</td>
</tr>
<tr>
<td>$Mag_y[\mu T]$</td>
<td>-16.24118</td>
<td>-15.53955</td>
<td>-16.97998</td>
<td>0.25678</td>
</tr>
<tr>
<td>$Mag_z[\mu T]$</td>
<td>-53.91173</td>
<td>-53.21960</td>
<td>-54.65851</td>
<td>0.27249</td>
</tr>
<tr>
<td>Magn. Mag[\mu T]</td>
<td>-56.45019</td>
<td>-55.73744</td>
<td>-57.23139</td>
<td>0.27505</td>
</tr>
</tbody>
</table>

4.2.2.4 Barometer

The barometric air pressure sensor or barometer is classified in the category of environmental sensors. It measures the ambient air pressure in $mbar$. Recognizing changes in the barometric air pressure can be useful in weather forecast and also in the detection of altitude.
changes. Similar to the rest of the sensors, a study of the standard deviation error was performed in this test to estimate the bias error introduced by the barometric sensor integrated in the LG Nexus4. The Samsung model used does not integrate this kind of sensor.

Figure 4.5 shows the air pressure readings collected during a test of 15 sec with the device stationary.

The value for the average of standard deviation estimated in this barometer sensor quality test (see Table 4.6) was used in the altitude change detection algorithm of the systems to fix a threshold to the random pressure oscillations.

![Output signals barometer quality test.](image)

**Figure 4.5** Output signals barometer quality test.

<table>
<thead>
<tr>
<th>Barometer test</th>
<th>Average</th>
<th>Max value</th>
<th>Min value</th>
<th>Std deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure[mbar]</td>
<td>1014.69</td>
<td>1014.88</td>
<td>1014.54</td>
<td>0.050086</td>
</tr>
</tbody>
</table>

### 4.2.3 Sampling Rate Test

For querying the sensors, a sampling rate of 15ms was used. After several tests, this value was considered the minimum to obtain a good resolution of the user’s activity, while no over consuming energy in the data collection task for the trace generation. This test studied the effect of changing the sampling rate and the convenience of the rate chosen. An experiment performing
the same walking activity is studied with four different sampling rates of: 7ms, 15ms, and 30ms. Walking activity is the most demanding final state in terms of sample resolution among the ones considered in the system. Thus, it imposes the maximum value for the sampling parameter.

Figure 4.6 shows the output signals for a walking segment of 15 meters sampled at different rates. As the sampling value increases, the resolution and magnitude of the energy signals decreases, making harder the step detection and counting task.

As shown, the value of 15ms offers an accurate resolution of the original signal without oversampling the processing algorithms.
4.3 Segmentation Results

This section evaluates the algorithms for detecting the events considered as separators in the segmentation module of the system: heading changes, when pedestrians perform a turn in a corner, and altitude changes, when the user takes stairs or an elevator to change the floor. These turn detection and altitude changes algorithms allow the segmentation of the original complete motion traces into independent segments.

4.3.1 Turn Detection Test

Table 4.7 summarizes the results of the heading changes detection algorithm for six different sequences of two, three and four heading changes. L means turn to the left in a corner and R means turn to the right. The sequences of turns are detailed in Figure 4.7. Each experiment was repeated 5 times for a total of 30 experiments and 90 turns.

<table>
<thead>
<tr>
<th>Turn detection test</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Test 5</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence 1 (LL)</td>
<td>LL</td>
<td>LL</td>
<td>LL</td>
<td>LL</td>
<td>LL</td>
<td>0</td>
</tr>
<tr>
<td>Sequence 2 (RR)</td>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>0</td>
</tr>
<tr>
<td>Sequence 3 (LRL)</td>
<td>LRL</td>
<td>LRRL</td>
<td>LRL</td>
<td>LRL</td>
<td>LRL</td>
<td>1</td>
</tr>
<tr>
<td>Sequence 4 (RLR)</td>
<td>RLR</td>
<td>RLR</td>
<td>RLR</td>
<td>RLR</td>
<td>RLR</td>
<td>0</td>
</tr>
<tr>
<td>Sequence 5 (LRRL)</td>
<td>LRRL</td>
<td>LRRL</td>
<td>LRRL</td>
<td>LRRL</td>
<td>LRRL</td>
<td>0</td>
</tr>
<tr>
<td>Sequence 6 (RLLR)</td>
<td>RLLR</td>
<td>RLLR</td>
<td>RLLR</td>
<td>RLLR</td>
<td>RLLR</td>
<td>0</td>
</tr>
</tbody>
</table>
4.3.2 Altitude Changes Detection Test

The altitude changes detection algorithm was evaluated using six different altitude change sequences, involving one, two and three changes respectively. The sequences of altitude variation are shown in Figure 4.8, where U means an interval of time going up, and D a time going down. Each experiment was repeated 5 times, for a total of 30 tests and 60 altitude changes.

Table 4.8 details the results of the experiments and the number of errors.

Figure 4.8 Sequences performed for altitude detection test.
Table 4.8 Altitude changes detection test results.

<table>
<thead>
<tr>
<th>Altitude changes detection test</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Test 5</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence 1 (D)</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>Sequence 2 (U)</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>0</td>
</tr>
<tr>
<td>Sequence 3 (DD)</td>
<td>DD</td>
<td>DD</td>
<td>DD</td>
<td>DD</td>
<td>DD</td>
<td>0</td>
</tr>
<tr>
<td>Sequence 4 (UU)</td>
<td>UU</td>
<td>UU</td>
<td>UU</td>
<td>UUU</td>
<td>UU</td>
<td>1</td>
</tr>
<tr>
<td>Sequence 5 (DUU)</td>
<td>DUU</td>
<td>DUU</td>
<td>DUU</td>
<td>DUU</td>
<td>DUUU</td>
<td>1</td>
</tr>
<tr>
<td>Sequence 6 (UDD)</td>
<td>UDD</td>
<td>UDD</td>
<td>UDD</td>
<td>UDD</td>
<td>UDD</td>
<td>0</td>
</tr>
</tbody>
</table>

4.4 Activity Recognition Results

Table 4.9 shows the confusion matrix for the activity classification tests, where the first column shows the real experiment performed and the first row the activity recognized. A set of 10 experiments were accomplished for every single activity considered in the decision tree. In total 40 activity recognition tests were performed.

Table 4.9 Activity recognition test results.

<table>
<thead>
<tr>
<th>Activity recognition test</th>
<th>Stationary (Sy)</th>
<th>Elevator (E)</th>
<th>Walking (W)</th>
<th>Stairs (St)</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stationary (Sy)</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Elevator (E)</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Walking (W)</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Stairs (St)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

4.5 Activity Specification Results

Once the activity performed is recognized, some of them require additional algorithms to enable a complete tracking of the user. In the case of the walking activity, the step detection and
counting algorithm returns the number of steps walked and the stair detection and counting algorithm performs similar with stairs. The results of these two algorithms are analyzed in this section.

4.5.1 Step Detection and Counting Test

The step detection and counting algorithm was tested executing three walking sequences with different distance of 5, 10, and 15 steps. Each sequence was repeated 10 times, for a total of 30 walking traces and 300 steps. Table 4.10 shows the results, where the first column is the real number of steps walked and the first row indicates the number of steps detected relative to the right value. The content in the table show the number of experiments for each possible number of mistakes.

Table 4.10  Step detection and counting test results.

<table>
<thead>
<tr>
<th>Step detection test</th>
<th>-2 steps</th>
<th>-1 steps</th>
<th>Correct</th>
<th>+1 step</th>
<th>+2 steps</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance 1: 5 steps</td>
<td>-</td>
<td>3</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>Distance 2: 10 steps</td>
<td>-</td>
<td>2</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>Distance 3: 15 steps</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>

4.5.2 Stairs Detection and Counting Test

Similar to the previous step tests, the stairs and counting algorithm was evaluated testing 2 stair sequences of 6 and 12 stairs respectively. Each sequence was repeated 10 times, making a total of 20 motion traces and 180 steps. Table 4.11 shows the results following the same format that the previous evaluation for steps.
Table 4.11 Stairs detection and counting test results.

<table>
<thead>
<tr>
<th>Stair detection test</th>
<th>-2 stairs</th>
<th>-1 stair</th>
<th>Correct</th>
<th>+1 stair</th>
<th>+2 stairs</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion 1: 6 stairs</td>
<td>-</td>
<td>2</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>Motion 2: 12 stairs</td>
<td>-</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>

4.6 Pedestrian Tracking Results

This section presents the combined tracking accuracy when the complete architecture of the system is applied. Three motion traces were used for testing, including the main indoor obstacles and all the activities involved in this approach (see Figure 4.9).

The first experiment (the one on the left in Figure 4.9) is a square and it includes 4 walking segments and 4 turns. Each walking section involves 7m (9 steps approximately). Five independent tests were performed in both, clockwise (tests 1 to 5) and counter-clockwise (tests 6 to 10) directions. Table 4.12 shows the results of the walking segments since the corners were detected correctly.

![Motion traces used for testing pedestrian tracking.](image-url)
Table 4.12 Tracking square motion results.

<table>
<thead>
<tr>
<th>Tracking square motion</th>
<th>W 1</th>
<th>W 2</th>
<th>W 3</th>
<th>W 4</th>
<th>Total steps</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>37</td>
<td>1</td>
</tr>
<tr>
<td>Test 2</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>36</td>
<td>2</td>
</tr>
<tr>
<td>Test 3</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>37</td>
<td>1</td>
</tr>
<tr>
<td>Test 4</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>34</td>
<td>2</td>
</tr>
<tr>
<td>Test 5</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>35</td>
<td>3</td>
</tr>
<tr>
<td>Test 6</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>38</td>
<td>2</td>
</tr>
<tr>
<td>Test 7</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>Test 8</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>Test 9</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>35</td>
<td>3</td>
</tr>
<tr>
<td>Test 10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td>8.6</td>
<td>9.3</td>
<td>8.9</td>
<td>9</td>
<td>35.8</td>
<td>1.6</td>
</tr>
</tbody>
</table>

The second experiment is a sequence of walking, stairs and turns. It was performed in down and up direction, five times each. The sequence, from up to down, consists of the following segments: walking $2m$ (3 steps approx.), 12 stairs, turn, walking $3m$ (4 steps approx.), turn, 7 stairs and $2m$ walking. Table 4.13 summarizes the results for this experiment in down direction and Table 4.14 in up direction. Turns were correctly detected.

Table 4.13 Results of tracking stairs sequence in down direction.

<table>
<thead>
<tr>
<th>Tracking stairs sequence</th>
<th>W 1</th>
<th>St1</th>
<th>W 2</th>
<th>St2</th>
<th>W 3</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Test 2</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Test 3</td>
<td>2</td>
<td>13</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Test 4</td>
<td>3</td>
<td>12</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Test 5</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>2.4</td>
<td>11.2</td>
<td>2.8</td>
<td>7.8</td>
<td>3.2</td>
<td>4.2</td>
</tr>
</tbody>
</table>
Table 4.14 Results of tracking stairs sequence in up direction.

<table>
<thead>
<tr>
<th>Tracking stairs sequence motion</th>
<th>W 3</th>
<th>St2</th>
<th>W 2</th>
<th>St 1</th>
<th>W 1</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>2</td>
<td>7</td>
<td>2</td>
<td>12</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Test 2</td>
<td>3</td>
<td>9</td>
<td>2</td>
<td>12</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Test 3</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>12</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Test 4</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>11</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Test 5</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Average</td>
<td>3</td>
<td>7.6</td>
<td>2.8</td>
<td>11.6</td>
<td>3</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Last tracking sequence combines stationary, walking and elevator segments (see right sequence at Figure 4.9). The sequence consists of the following segments: stationary, walking 1.5m (2 steps approx.), turn, 2 floors in the elevator (5 sec approx.), and 1.5m walking. Table 4.15 summarizes the results for this experiment. Tests 1 to 5 show down direction and 6 to 10 up direction. Turns and stationary periods were correctly detected.

Table 4.15 Results of tracking floor motion.

<table>
<thead>
<tr>
<th>Tracking floors motion</th>
<th>W 1 (steps)</th>
<th>E (sec)</th>
<th>W 2 (steps)</th>
<th>Total steps</th>
<th># Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Test 2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Test 3</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Test 4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Test 5</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Test 6</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Test 7</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Test 8</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Test 9</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Test 10</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td>2.1</td>
<td>5</td>
<td>2.5</td>
<td>4.4</td>
<td>1</td>
</tr>
</tbody>
</table>
4.7 Discussion

The previous set of experiments and the result tables evaluates the performance of the proposed system. Initially, the data collection module was studied analyzing the quality of the sensors embedded in the devices by calculating the long term average of the output signal when it is not undergoing any movement. The accelerometer sensor showed a standard deviation in the magnitude of the acceleration of $0.0889 \text{ m/s}^2$ on average for the two devices. The gyroscope measurements of the angular velocity returned a standard deviation of $0.00825 \text{ rad/s}$. The magnetometer standard deviation was in average of $0.33902 \text{ µT}$ and the barometer standard deviation was estimated in $0.05008 \text{ mbar}$. These results obtained for the low quality set of MEMSs, shows the necessity of compensate the errors introducing filtering and calibration routines in the processing signal algorithms. Also, the convenience of choosing $15\text{ms}$ as the sampling rate was demonstrated studying the effect of its change over walking traces.

Then, the segmentation module and its algorithms detecting turns and altitude changes were evaluated. The system detects turns with proper heading information with an accuracy of $98.8\%$, with only 1 mistake out of 90 turns performed in the experiments. The tests to detect changes in altitude and the direction of these changes returned 2 errors in 60 tests, for an accuracy of $96.6\%$. In both cases the errors were false positives, returning additional detections.

The evaluation continued studying the results of the activity recognition task. The overall classification accuracy for this module is $95\%$, with 2 mistakes out of 40 classifications. The most common confusion was between elevator and stairs activities when performing elevator traces. The movements inside the elevator and the acceleration peaks detected when the elevator starts and stops its travel were the main causes for wrong activity classifications.
The results of the specification module included the analysis of the algorithms for detecting and counting steps and stairs. Both showed reliable performances with an error of less than 4%. Only 9 errors were detected in 300 steps for the step counting algorithm and 6 errors out of 180 steps in the stairs detection algorithm. Thus, the proposed techniques are able to detect and to track accurately the number of walking steps and stairs steps performed by the users.

Finally, a combined tracking evaluation was carried out for complete pedestrian traces testing the entire architecture of the system. Three motion traces were used for testing, including main indoor obstacles and all the activities considered in this approach. The overall pedestrian tracking accuracy was on average 90.81%, obtaining the lowest percentage in the stairs sequence experiment with 86.89%. Combining the individual modules added complexity to the tracking problem. The separators used to segment the original trace (turns and altitude changes) can split the motion in wrong samples that lead to errors in the final activity specification. Also, the quick transition among activities in pedestrian movements, as consequence of the granularity of the indoor environments, requires a high accuracy, not only in the classification of the event, but also in the detection of the exact moment when the action was performed.

While a more rigorous experimentation is necessary (across more buildings, users, phone and models), the results obtained from these experiments are promising to justify the proposed multisensory indoor system.
CHAPTER 5: SUMMARY

5.1 Conclusion

In the global scenario of Navigation Systems the ability to accurately track people in indoor environments has particular interest for many ubiquitous computing and location systems. This thesis provides a technique to create automatically accurate indoor pedestrian traces based on the noisy inertial sensors included in today’s commodity smart phones. The presented work extends the smart phones functionality using harness sensor-based dead-reckoning for tracking pedestrians, and environment sensing for detecting sensing signatures in indoors environments. A set of signal processing algorithms are presented for recognizing users’ motion in scenarios where map data are not provided a priori and in a transparent manner to the users. Corners (using a gyroscope), stairs, steps, and corridors (making use of an accelerometer combined with a barometer), and elevators (considering a magnetometer and barometer readings) are detected and detailed in constrained indoor environments. Additionally, a complete development that simplifies the system’s evaluation process was implemented.

The evaluation of the system shows a high tracking accuracy with less than 9.19% error. The activity recognition module studied independently obtains an accurate of 95%. The step counting and stair counting algorithms achieve an accuracy of 97% and 96.66%, while the detection of user changes of direction and altitude are performed with 98.8% and 96.6% accuracy, respectively. Performance results suggest strong motivation to pursue the goal of real deployment and to expand the system working on new different directions.
5.2 Future Works

The proposed work can be extended and improved in several directions.

First, the presented evaluation was performed using the phone in one position, when it is in messaging position. A complete solution should be able to track the user regardless of the position, or at least using the very common positions of the device such as, calling, swinging (refers to the position in which the user holds the device in a hand while walking) and pocket (the device sits in the user’s pocket/bag, which is the most common position when the user is walking). Since most of the algorithms use the magnitude of the signal they should potentially work in any of the described cell phone’s location. However, this needs to be evaluated.

Second, the current design of the system tracks pedestrian motions offline, meaning that the sensor data are initially gathered by the mobile application and, once the motion is finished, they are sent to the server, which applies the different algorithms and returns the results. Another possible approach would consist of tracking pedestrian movements in real time, collecting and processing the data in the smartphone.

Third, the current version of the system the data collection module is initiated manually by the user when entering in an indoor location. A useful improvement would be an additional module responsible of changing automatically from outdoor location services to this indoor location system. For instance, a smart implementation could activate the indoor location part considering the fact that building entrances are characterized by a visible drop in the GPS confidence when the user moves from outdoors to indoors.

Finally, in most of the cases indoor location is aimed for battery-dependent devices which are limited in size and capacity. Thus, an investigation of the power consumption of the system would be useful.
REFERENCES


APPENDIX A: COPYRIGHT PERMISSION FOR FIGURE 2.3

Content License

For the purposes of licensing, the content of this website is divided into two categories:

- Documentation content, including both static documentation and content extracted from source code modules, as well as sample code, and
- All other site content

The documentation content on this site is made available to you as part of the Android Open Source Project. This documentation, including any code shown in it, is licensed under the Apache 2.0 license, the preferred license for all parts of the of the Android Open Source Project.

Apache 2.0 is a commercial and open-source-friendly software license. The majority of the Android platform and documentation is licensed under the Apache 2.0 license. While the project strives to adhere to the preferred license, there may be exceptions, such as for documentation (code comments) extracted from a source code module that is licensed under GPLv2 or other license. In those cases, the license covering the source code module will apply to the documentation extracted from it.

Third-party components of this site such as JavaScript libraries are included in the Android Open Source Project under the licenses specified by their authors. For information about these licenses, refer to the source files in the Android Open Source Project.

All other content on this site, except the license documents themselves and as otherwise noted, is licensed under the Creative Commons Attribution 2.5 license.

You may use the content of this site in any way that is consistent with the specific license that applies to the content, as described above. For content licensed under Creative Commons Attribution 2.5, we ask that you give proper attribution.

Terms of Use

We are pleased to license the Android documentation and sample code under terms that encourage you to take, modify, reuse, re-purpose, and remix the content as you see fit. Except as noted in the Restrictions section below, you are free to use the documentation content in your own creations. For example, you could quote the text in a book, cut-and-paste sections to your blog, record it as an audiobook for the visually impaired, or even translate it.
Restrictions

- While the documentation itself is available to you under the Apache 2.0 license, note that proprietary trademarks and brand features are not included in this license.
- Google's trademarks and other brand features (including the Android styled typeface logo) are not included in the license. Please see Brand Guidelines for information about this usage.
- In some cases, a page may include content, such as an image, that is not covered by the license. In that case, we will label the content that is not licensed.
- In addition, content linked from a page on this site is not covered by the license unless specifically noted. For example, pages may link to videos or slide decks that are not covered.
- The use of sample source code provided in the SDK or shown in this documentation is subject to the conditions detailed in the Apache 2.0 license.

Attribution

Proper attribution is required when you reuse or create modified versions of content that appears on a page made available under the terms of the Creative Commons Attribution license. On this site, the requirement for attribution applies only to the non-documentation content, as described earlier in this document. The complete requirements for attribution can be found in section 4b of the Creative Commons legal code.

In practice we ask that you provide attribution to the Android Open Source project to the best of the ability of the medium in which you are producing the work. There are several typical ways in which this might apply.

Exact Reproductions

If your online work exactly reproduces text or images from this site, in whole or in part, please include a paragraph at the bottom of your page that reads:

Portions of this page are reproduced from work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.

Also, please link back to the original source page so that readers can refer there for more information.

Modified Versions

If your online work shows modified text or images based on the content from this site, please include a paragraph at the bottom of your page that reads:

Portions of this page are modifications based on work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.

Again, please link back to the original source page so that readers can refer there for more information. This is even more important when the content has been modified.

Other Media

If you produce non-hypertext works, such as books, audio, or video, we ask that you make a best effort to include a spoken or written attribution in the spirit of the messages above.
ABOUT THE AUTHOR

Juan José Marrón Monteserín received his MSc. in Telecommunications Engineering in 2010 and MSc. in Web Engineering in 2013 from the University of Oviedo (Spain). He has two years of work experience as System and Network Administrator for Computer Sciences Corporation Spain. He is currently studying his MSc. in Computer Science at University of South Florida (USA). His research interests include: Participatory Sensing, Indoor Location and Pedestrian Tracking. His e-mail is jmarronm@mail.usf.edu.