AnDReck: Positioning Estimation using Pedestrian Dead Reckoning on Smartphones

Carlos Filipe Correia Sequeira de Jesus Simões

Thesis to obtain the Master of Science Degree in
Communication Networks Engineering

Examination Committee
Chairperson: Prof. Paulo Jorge Pires Ferreira
Supervisors: Prof. Ricardo Jorge Fernandes Chaves
Prof. José Eduardo da Cunha Sanguíno
Member of the Committee: Prof. António José Rodrigues

November 2013
Acknowledgements

To my Family, especially my Mother and Father: for their patience with my mood shifts across the past year, knowing when to let go and when to show me the path towards my goals.

To my advisors, Ricardo Chaves and José Sanguino: for their patience with my meetings, e-mails and overall stubborness, and especially for their support during these difficult last stages, providing last minute revisions.

André Camões: for being a comrade ever since I got into college to this day, providing great motivational support and being awesome.

Ricardo Leitão and Pedro Luiz de Castro: for their support in testing the early prototype of my application, and showing great interest in my work.

Daniela Ferreira: the logistical support on a field completely out of her scope of study, and motivational support fitting of the best social rehabilitators, without which this thesis might not have progressed past the development stage.

Catarina Moreira: for her lightning quick clarifications in the field of PCA, which was a considered approach at one point.

Prof. Luis Boavida: for his time helping me understand part of the my that dealt with matrix operations, always making sure I understood everything.

Luis Luciano: for giving countless positive perspectives from someone in the same position as I was.

João Lino: for his company in some post-laboral sessions, and for being a jolly good fellow.

Nuno Correia: for never giving up on pushing me towards solutions, always interested in presenting me with solutions and not obsessing over problems, especially in the crucial ending stages.

João Rosa: for the camaraderie, distraction from development woes and crucial help in the last minutes.

Ana Cristina Carreira: for always being understanding of my situation and giving me a huge amount of motivation.
Abstract

In recent years, there was a wide adoption of Geographic Information System (GIS) devices, many of which include Global Positioning System (GPS) technology. Most of the time, these functions are performed by a smartphone, being carried by its user for extended periods of time. Although the common accuracy values for this technology are acceptable for some applications, such as driving guidance and nearby places identification, when it comes to pedestrian navigation they are somewhat lacking. Smartphones include a varied set of inbuilt sensors, which can be leveraged to perform positioning tasks, using the information provided by them. This work analyses existing publications and systems, across different approaches, and proposes a Pedestrian Dead Reckoning (PDR) based solution architecture. An implementation that uses an adapted peak detection algorithm, a step length estimation algorithm, along with its calibration method, and a digital compass orientation estimation method is then described, along with an Android prototype. Tests for this solution are then performed according to a proposed methodology, and the results are then evaluated against those of both systems that were the basis for implementation and related work, in order to validate its applicability.

Keywords: GPS, accelerometer, Pedestrian Dead Reckoning, Step Detection, Step Length Estimation, Android
Resumo

Nos últimos anos, gerou-se uma grande adopção de Sistemas de Informação Geográfica, muitos dos quais incluem tecnologia GPS. Na maioria dos casos, a função desses sistemas é desempenhada por smartphones, que fazem parte do dia a dia das pessoas. Estes apresentam valores de precisão aceitáveis para algumas aplicações, tal como guias de navegação ou identificação de locais de interesse por perto. No entanto, em relação à navegação pedestre, os resultados são pobres. De forma a desempenhar tarefas de posicionamento, os smartphones incluem uma série de sensores embutidos que podem ser de alguma forma aproveitados para a navegação pedestre. Este trabalho analisa publicações e sistemas existentes, atravessando abordagens diferentes, e propõe uma arquitectura de solução baseada em Dead Reckoning Pedestre. Em seguida, é apresentada a concretização dessa arquitetura, que utiliza um algoritmo de deteccção de passos adaptado, um estimador de comprimento de passada, juntamente com o seu método de calibração, tal como um método de estimação de orientação. Por fim, é apresentado um protótipo Android, que concretiza essa arquitectura. A esta solução foram feitos alguns testes segundo a metodologia apresentada, sendo os resultados avaliados em relação a sistemas nos quais se baseia e em relação a outros referidos no trabalho relacionado, de modo a validar a sua aplicabilidade.

Palavras Chave: GPS, acelerômetros, Dead Reckoning Pedestre, Deteccção de Passos, Estimação de Comprimento de Passada, Android
Contents

Acknowledgements iii

Abstract v

Resumo vii

List of Figures xii

List of Tables xiii

Acronyms xiii

List of Acronyms xv

1 Introduction 1

1.1 Motivation 1

1.2 Objectives and Contributions 1

1.3 Requirements 2

1.4 Document Structure 2

2 State of the Art 4

2.1 Related Topics of Research 4

2.1.1 Visually Impaired Guidance Systems 4

2.1.2 Inertial Navigation Systems 8

2.1.3 Other considered techniques 11

2.2 Related Systems and Comparison 11

2.2.1 Nawzad Al-Salih's Real-Time Kinematic (RTK) Visually Impaired Guidance System 11

2.2.2 NavMote 12

2.2.3 Padati 13

2.2.4 AutoGait 14

2.2.5 Comparison 14

2.3 Other Researched Systems 15

2.4 Summary 16
List of Figures

2.1 GPS Trilateration in Action ......................................................... 6
2.2 Plain GPS data vs Combined data (GPS+DR) vs Plain Integrated data [24] ........ 9
2.3 System components, including Mobile Navigation Unit (MNU) and Navigation Service Center (NSC) [5] ...................................................... 12
2.4 NavMote system architecture [13] ................................................. 13
2.5 Components of the Padati system [22] ............................................. 13
2.6 AutoGait high-level components [22] ............................................. 14
3.1 High-level components of the system and information flow ............................. 19
3.2 Calibration components and information flow ........................................ 20
3.3 Positioning Estimation components and information flow ............................ 20
4.1 Example of peaks as step candidates in the acceleration signal ...................... 23
4.2 Example of peaks excluded with the first correction .................................. 23
4.3 Example of peaks and valleys in a pair of steps ..................................... 24
4.4 The averaging of step values (frequency and length) on a segment .................. 24
4.5 Combination of several Orientation values during a Step ................................ 25
4.6 The effect of Singularities in Orientation output [46] .................................. 27
4.7 A UML diagram of the abstract components ........................................ 28
4.8 A UML diagram of the Reading Circular Buffer components ....................... 30
4.9 A UML diagram of the Reading Source components .................................. 31
4.10 Butterworth filter function .......................................................... 32
4.11 Before and After signals (Butterworth Filtered, 10th order, offset to minimize delay) ............................ 32
4.12 A sample signal plot along with its detected steps ................................... 34
4.13 A GPS segment and its Straight Line Identification (SLI) thresholds [22] ........ 35
4.14 Example plots of step length-frequency models of different users [34] ................ 36
4.15 Generation of a new position, triggered by a detected step. .......................... 36
4.16 Storage of AutoGait samples in an Android database ............................... 38
4.17 Retrieval of AutoGait samples stored in an Android database ....................... 39
5.1 Side-view of the Test Site .......................................................... 43
5.2 Front-faced view ................................................................. 43
5.3 Overhead map view of the Test Site .................................................. 44
5.4 Accumulation of Step Length error in a 500 meter test, over time (meters/seconds) .... 47

6.1 Movement Pattern Recognition [27] ................................................... 53

A.1 A UML diagram of the Reading components ...................................... 56
A.2 A UML diagram of the Analyser components ..................................... 57

C.1 Accumulated Distance Estimation Error (Day 1 - 12h41) ....................... 59
C.2 Accumulated Distance Estimation Error (Day 1 - 12h51) ....................... 60
C.3 Accumulated Distance Estimation Error (Day 1 - 13h26) ....................... 60
C.4 Accumulated Distance Estimation Error (Day 1 - 13h28) ....................... 60
C.5 Accumulated Distance Estimation Error (Day 1 - 13h31) ....................... 61
C.6 Accumulated Distance Estimation Error (Day 2 - 12h38) ....................... 61
C.7 Accumulated Distance Estimation Error (Day 2 - 12h46) ....................... 61
C.8 Accumulated Distance Estimation Error (Day 2 - 12h49) ....................... 62
C.9 Accumulated Distance Estimation Error (Day 2 - 12h52) ....................... 62
C.10 Accumulated Distance Estimation Error (Day 2 - 12h56) ...................... 62
C.11 Accumulated Distance Estimation Error (Day 2 - 13h05) ...................... 63
List of Tables

2.1 Comparison table of the systems analysed in this document ................. 15

5.1 Results of the step counting metrics for Walking Tests ....................... 45
5.2 Results of the step counting metrics for Running Tests ....................... 45
5.3 Results of the Distance Estimation Metrics for Walking tests ............... 46
5.4 Results of the Distance Estimation Metrics for Running tests ............... 47
5.5 Results of the Orientation Estimation Metrics for Walking tests (values in degrees, ranged [0; 180], and meters for "Lateral") .................................................. 48
5.6 Results of the Orientation Estimation Metrics for Running tests (values in degrees, ranged [0; 180], and meters for "Lateral") .................................................. 49
5.7 Median and kth Percentile values for both Walking and Running tests (in decimal degrees) 49
List of Acronyms

CM   Central Module
DGPS  Differential Global Positioning System
EGNOS  European Geostationary Navigation Overlay Service
EOA  Electronic Orientation Aids
ERF  Earth Reference Frame
ETA  Electronic Travel Aids
GIS  Geographic Information System
GLONASS  GLObal NAVigation Satellite System
GNSS  Global Navigation Satellite System
GPS  Global Positioning System
IIR  Infinite Impulse Response
IGDG  Internet-based Global Differential Global Positioning System (DGPS)
IGS  International Global Navigation Satellite System (GNSS) Service
INS  International Navigation System
JPL  Jet Propulsion Laboratory
LOS  Line of Sight
MNU  Mobile Navigation Unit
MM  Mobile Module
NASA  National Aerospace Agency
NSC  Navigation Service Center
NRTK  Network RTK
PCA  Principal Component Analysis
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDR</td>
<td>Pedestrian Dead Reckoning</td>
</tr>
<tr>
<td>PPP</td>
<td>Precise Point Positioning</td>
</tr>
<tr>
<td>RSS</td>
<td>Received Signal Strength</td>
</tr>
<tr>
<td>RTK</td>
<td>Real-Time Kinematic</td>
</tr>
<tr>
<td>SLI</td>
<td>Straight Line Identification</td>
</tr>
<tr>
<td>SLL</td>
<td>Step Length Lookup</td>
</tr>
<tr>
<td>TOA</td>
<td>Time of Arrival</td>
</tr>
<tr>
<td>TTFF</td>
<td>Time to First Fix</td>
</tr>
<tr>
<td>WPS</td>
<td>Wi-Fi Positioning System</td>
</tr>
</tbody>
</table>
1.1 Motivation

In recent years, there was a wide adoption of Geographic Information System (GIS) devices, which include Global Positioning System (GPS) technology. Devices that use this technology provide its user with a positioning estimate, whose accuracy is dependent on a number of factors: surrounding environment, atmospheric conditions, satellite line of sight, among others. Although the common accuracy values for this technology are acceptable for some applications, such as driving guidance and nearby places identification, when it comes to pedestrian navigation they are somewhat lacking. This is due to the differences between vehicular and pedestrian navigation: vehicle roads versus pedestrian paths, continuous movement versus cyclic movement, reduced mobility versus high mobility.

Looking at the most common device for this purpose, a smartphone, it can be observed that it also includes other components that may provide means to attain better positioning accuracy, namely acceleration and magnetic sensors. Furthermore, information gathered from these systems may be coupled to GIS data, in order to complement shortcomings between them.

Herein, we propose a system that leverages smartphone technology in order to provide positioning with increased precision. The system should be precise, easy to deploy and portable, taking advantage of the widespread usage of smartphones and its capabilities.

1.2 Objectives and Contributions

Our proposal is to study the research done in related areas, along with relevant systems, and use that to build a system prototype that is capable of outputting relative positions in both indoor and outdoor conditions, in the course of pedestrian locomotion. The positioning techniques used in most systems, which is mostly standard GPS, have poor precision, so alternate approaches such as using relative GPS (RTK and Precise Point Positioning (PPP)) or using dead reckoning will be considered. The system will
also take into consideration the fact that it should be of relatively low-cost, easy to use and portable, in order to have a good public acceptance.

This work developed in this thesis will contribute to future developments by providing investigation topics for similar approaches, the adopted solution, along with the development process associated with it, as well as the results attained from it. As a final topic, another contributing point is present in the form of suggestions for future work, as a guideline to what direction to follow next.

1.3 Requirements

This section will detail the requirements that were set upon this system, detailing on why they are important to the overall development. These requirements may be mentioned throughout the document, highlighting, for instance, why a mentioned technique or technology is relevant.

- **Availability**: increase the opportunities for positioning
- **Accuracy**: improved positioning accuracy and precision
- **Practicality**: system set-up should not be too troublesome
- **Portability**: the system must not hamper pedestrian locomotion

The main objective of this system is to improve the current availability of positioning systems referred in section 1.1. Current systems require a good satellite visibility during positioning. The system should be able to maintain positioning during navigation through terrain with low sky visibility.

Another requirement is to provide an accurate estimation of positioning. Again, satellite visibility conditions current systems in their accuracy levels. Thus, this system should be able to not degrade the precision and accuracy levels in lower visibility conditions.

In order to have a wide acceptance, the system should be as ready to use from the start as possible, with minimal set-up steps. This not only enables the system to be useful as soon as possible, but also, by removing complexity, it becomes less susceptible to failures, and thus, more recalibrations.

Finally, the system should be portable during usage. Since the use case is during pedestrian navigation, it is essential that the system hampers locomotion as little as possible. Also, this ties in with the availability requirement, since a portable system will be more easily deployed everywhere.

1.4 Document Structure

Chapter 2 describes the state-of-the-art, introducing some concepts related to these guidance systems in sections 2.1.1 and 2.1.2, and presenting some of the technologies involved in their subsections (2.1.1.1 and 2.1.2.1.1-2.1.2.2). Some examples of these systems will also be described in detail and categorized in section 2.2, with others briefly mentioned in section 2.3, with a summary in section 2.4.

In Chapter 3 the proposed architecture of our solution, which includes an overview of the system requirements from section 1.3, is presented, followed by an overview of the entire system in sections
3.2 and 3.3. High-level details regarding the implementation of each component previously mentioned is present in sections 4.1 and 4.2.

Chapter 4 contains a description of the implementation in high detail, describing all the processes involved, the data that is exchanged, and the context were it is executed.

In Chapter 5, the evaluation of the proposed solution is presented, comparing it both to real results and similar technologies.

Finally, section 6 concludes this document, highlighting what has been accomplished, and topics for future work directions.
State of the Art

This section presents the state-of-the-art of work done in the fields relevant for the proposed development of the system. The first section describes systems with similar requirements and how they relate to the considered problem. It describes both academic and non-academic contributions, as well as theoretical and practical projects, including the involved technologies. Finally, it also points out some of the merits from each of these contributions, along with their shortcomings.

2.1 Related Topics of Research

As a starting point for research topics, systems with similar objectives, and/or even requirements, to that of our own, were considered. In this case, the focus was on systems that require positioning with higher precision and those that employed technology present in smartphones.

The first topic, Visually Impaired Guidance Systems, also has tight requirements regarding positioning, due to the fact their users are disabled in that way, and so rely heavily on the output of the system, that is based in positioning technologies of some sort. The second topic, Inertial Navigation Systems, uses inertial and other sensors to estimate positioning, by measuring their output to calculate movement beginning at a starting point, and then successively after each calculated point.

2.1.1 Visually Impaired Guidance Systems

For Visually Impaired people to achieve some autonomy in their daily lives, they must resort to equipment that helps navigate the environment around them. Some may be supported by more advanced technologies, others more elementary. Some more appropriate to navigating indoors, while others only appropriate for street navigation. It is therefore necessary to categorize these systems, by purpose and employed technologies.

Regarding applicability, these systems can be divided into two categories: micronavigation devices
and macronavigation devices [1]. Micronavigation refers to systems designed to help visually impaired users detect obstacles in their path, in order to make small adjustments to their route (hence the prefix micro). Macronavigation, on the other hand, helps their users find the intended path to a destination, in a larger environment. Note that these two designations have nothing to do with the underlying complexity of the system. In fact, a good example of two low-tech approaches to each of these categories is white-canines and guide dogs, the latter actually being an example of both categories.

While these categories encompass all sorts of technologies used for visually impaired navigational purposes, they do not categorize the main type of technology used to achieve its goal. The type of technology that is the most relevant for this project is electronic, since it can leverage the advantages of information and electromagnetic technologies. We can divide electronic systems into two categories with their corresponding applicability: Electronic Travel Aids (ETA), used for micronavigational purposes, and Electronic Orientation Aids (EOA), for macronavigation.

ETA systems usually work by trying to identify objects located on its users surroundings, with technologies such as ultrasounds, lasers or infrared light, and then alerting the user to their whereabouts [2]. Usually these sensors are installed on the user’s head, neck, a belt or the white cane, in order to have the correct perspective on the scene to analyse, so it can then proceed to guide the user via audio tones, voice or vibrating stimulus. As expected, these systems work both in indoor and outdoor conditions, with ranges varying from 1-15 meters, and viewing angles from 30 to 45 degrees. In the end, these systems end up being more of a complement for macronavigation systems or the traditional white cane.

EOA systems help the user find his path to a destination through a given region, even if he has never crossed it before, by knowing the user’s current location with the best precision possible and guiding him step by step. The key technology here is the positioning system, GPS in most cases, that determines the current position and then uses it to plot a route to the destination. After having the route set, the system only has to guide the user and make sure he stays on course. This is normally done by speech, but sound or vibration may also be employed [3]. The main problems with these systems are the lack of maps adequate for this type of navigation and the lack of positioning presented by the GPS systems. The map inadequacy problem cannot be immediately solved without more thorough cartography work, but there are techniques that may help increase the precision obtained by GPS positioning fixes. Some of them involve using a different approach when calculating the fix [4] [5], others obtain the precise data from a different source, and finally, others complement that fix with data gathered from other sensors [6] [7].

A different approach to calculating positioning that also uses the GPS infrastructure is through DGPS techniques, where the signal received from the several GPS satellites is analysed even further, by comparing it against one from another receiver. This second receiver must be stationary, and is commonly referred to as a base station. A good example of a DGPS system is Real-Time Kinematic (RTK), which provides some of the best improvements over stand-alone GPS [8]. A more detailed explanation is presented in section 2.1.1.1 and in its “Real-Time Kinematic” paragraph, for DGPS and RTK respectively.

Another approach is to fetch some items from a different, and more reliable, source, and use them in the positioning calculations to obtain a more precise result. This distribution is usually done via
terrestrial communication channels, and may include data such as more precise versions of satellite orbit positions (referred to as ephemeris data) and clock values. Such is the case of PPP, a promising technique that follows this approach and presents results similar to those of RTK [9] [10]. PPP has yet to have been explored in the context of visual impaired guidance systems, but its results indicate that it is compatible with our objectives. A more detailed explanation of this system is presented in section 2.1.1.1’s paragraph “Precise Point Positioning”.

Both these last two approaches rely on the usage of dual frequency (L1/L2) GPS receivers, which are expensive, making the system accessible to less users. It is possible to use single frequency receivers, but the accuracy decreases by doing so. This is where the solution of using additional sensors can be useful: by improving positioning and navigation by complementing them [11] [12]. It has been used for dead reckoning [13], a technique that estimates positioning from a starting point, and attitude determination [14], which establishes the orientation of an object.

2.1.1.1 GPS and Differential GPS

The Global Positioning System (GPS) is used in several domains to obtain a location fix, which can then be used for other purposes (such as plotting a course towards a destination, or determining if we have left an intended area). Its working principle is not too complicated: a device that is located on the surface of the earth listens for transmitted GPS satellite signals, registering the receiving time and using it to determine the distance to it. Then, after measuring receive times for three other satellites, one can compute the intersection of four spheres with a radius equal to the determined distance. This intersection results in two points in space, as depicted in figure 2.1, but we can then narrow it down to one by assuming that the correct one is the one closer to the Earth’s surface (only not true for airborne/space borne receivers) [8].

This description assumed that the clocks of both the receivers and satellites are synchronized, which is not completely true, since there are always discrepancies in the sync process. It is also assumed that there is a clear Line of Sight (LOS) between receivers and satellites, and that there are no significant
abnormal atmospheric effects. Real world conditions may not be as benevolent, and as such, positioning errors need to be taken into account. In the case of a hand held stand-alone GPS, with more than 4 satellites in view, 11 meters was the best accuracy measured [8], which may not be enough for some applications.

An augmentation technique used to increase this precision is Differential Global Positioning System (DGPS), which relies on measurements done in a reference base-station to correct the measurements on a roving base-station [8]. These corrections may be applied either in a future time or in real-time, if a channel to the rovers exists. The actual precision obtained from using these techniques depends on the area being served (local, regional or wide), if an absolute or relative position is intended, or if they use either carrier or code based techniques (sometimes even both).

As previously mentioned, there are some DGPS techniques that have shown promising results recently, namely RTK and PPP.

2.1.1.1.1 Real-Time Kinematic  RTK is a DGPS technique that requires a base-station, placed in a known location, to make measurements of the carrier-phase from the several in-view GPS signals available. This data, together with the code measurement and the known location of the base-station, is then transmitted to the interested rover receivers. They can then fix the phase ambiguities present in their own signals and compute its position relative to the base-station. Because they also have the base-station position, they may also compute their absolute positioning, which may be optional for some applications.

The relative positioning obtained this way can have an accuracy up to a few centimetres, within a radius of up to 20 km from the base-station [15], which extends the applicability of GPS positioning to more sensitive domains. There are limitations, however: rover receivers must be located near the base-station (this makes it a local area system), as well as maintain a real-time communication channel with it; a convergence time must be waited before the ambiguities are fixed; and GPS tracking must be continuous, in order to avoid re-initialization. Typically, these types of techniques are used with double frequency GPS receivers, reaching distances of 75 km from the base-station, but single frequency receivers can still reach distances of up to 20 km [16].

2.1.1.1.2 Precise Point Positioning  PPP is a recent wide-area code-based DGPS technique that enables roving receivers to obtain decimetric level positioning by providing them with more precise estimates of ephemeris and clock data to compute it. In a typical implementation of PPP, an external network provides IGS clock and orbit estimates to a double frequency receiver at an arbitrary location [17]. This receiver then estimates the zenith tropospheric path delay and the bias between the pseudo range and carrier-phase measurements (for each satellite), alongside the three-dimensional position and clock bias. After these estimations are made, and if a decimetric level accuracy is desired, the receiver must still account for these error sources: the difference between satellite center of mass and antenna phase center (also known as the satellite lever arm), phase wind-up, solid earth tides and ocean loading [8].
The PPP system may not look like the average DGPS system and, in fact, some have put it in a different category from stand-alone and differential GPS counterparts. The reason for this is that it does not require a base-station in the usual sense: there is a central entity that computes and provides estimates to roving stations, but the proximity required is not as relevant as in other DGPS systems (i.e. RTK). Still, the level of accuracy is quite comparable to other DGPS systems, if not better, as shown in [17]. Two operational examples of PPP systems are the Internet-based Global DGPS (IGDG), from the National Aerospace Agency (NASA) Jet Propulsion Laboratory (JPL), that distributes the data via a dedicated frame-relay link [18], and NavCom Technology’s geostationary satellite links, part of their StarFire network DGPS service [19] [8].

This system still shares a disadvantage with its RTK DGPS counterpart: the system described in [17] requires about 30 minutes to obtain its boasted 5 cm accuracy value. Even if this accuracy is admirable, this makes it somewhat incompatible with the practicality requirement.

2.1.2 Inertial Navigation Systems

Many navigators in the old ages estimated location by determining the vessel's speed and orientation and adding it to the starting point. Using tools to track motion and following it up with positioning derivations is a technique that has been used for quite some time and it is designated as dead reckoning. It has been used as an alternative positioning system in other domains as well (i.e., cattle tracking or car positioning in tunnels) as a way to cope with positional faults from the main positioning system, mainly GPS, by using several types of sensors (like accelerometers and digital compasses).

There are two main approaches to dead-reckoning for pedestrian applications: by integration of the acceleration values or by step detection. The integration method obtains the current acceleration value and the time since the last measurement to update the current speed. Using this speed value, one can compute the amount of movement since the last update in a similar manner. However, applying this technique with low-cost sensors will make it very susceptible to drift errors, due to the fact that the positioning is being obtained by mathematical integration of the successive acceleration values, which also suffers from measurement errors. In the case of running, one can attempt to recognize the gait cycle of the athlete, which then may be used to detect drift errors [7]. With this data, it is possible to correct the accelerometer data, and thus make this dead reckoning technique more effective [7].

The other approach, that avoids these errors, does not use the integrated acceleration values directly to discern distance. This is the case of Pedestrian Dead Reckoning (PDR) systems, which use step events coupled with the step length to determine distance [20], combining it with some form of directional data to obtain positioning. The step length employed in this process may be an averaged value or an estimated value, using step frequency models built from experimental data [21][22]. Such is the case of the AutoGait system, that builds such a model during a calibration phase.

Combining dead reckoning and GPS positioning may involve usage of Kalman Filtering [23], in order to join both the positioning data from each data source into one, in successive iterations. Multiple filters may be used, and results fed from one to the other, as they are calculated. This works out quite well with
low-cost GPS receivers because they usually have a smaller update rate, and thus enable the positioning to be kept updated in-between them, working as a self-contained positioning system. In figure 2.2, we can see an example of the results obtained from 3 sources: a filtered GPS source, one with integrated accelerometer data, and one with both GPS and the integrated data combined [24].

![Figure 2.2: Plain GPS data vs Combined data (GPS+DR) vs Plain Integrated data [24]](image)

We can also use these other sensors to complement the positioning given by GPS, by serving as a spike detector. For instance, if we observe that GPS data shows dramatic movement, by also observing, for example, the accelerometer data, we can verify that it does not match with this same dramatic offset. Since accelerometer data is not subject to the same types of interferences as the GPS data (such as satellite loss or carrier phase cycle slips), and at the same time provides a reasonable amount of reliability, we can then infer which is more accurate at a given point in time.

### 2.1.2.1 Pedestrian Dead Reckoning

As mentioned in section 2.1.2, one of the approaches to inertial navigation for pedestrian locomotion is to analyse the step movement, by detecting step events, estimating distance travelled in each of them and combining that with directional data to obtain a relative position.

Each of these components have different approaches, with different results and applications, making them appropriate to several different scenarios. The following paragraphs describe each of them in more detail, highlighting the different solutions available along with their advantages.

#### 2.1.2.1.1 Step Detection

Step detection using embedded systems has been a long running topic of research over the years, mostly in the computer science and signal processing domains. Although several different approaches are employed today with good results, most of them share one common trait: analysis of a signal, very frequently from an accelerometer sensor. This happens because acceleration sensors have become more and more inexpensive and, due to smartphones, ever more ubiquitous.

Acceleration values are indicative of changes in acceleration or lack thereof, and so, direct analysis intuitively makes sense: if during human locomotion there are several moving body parts, using acceleration values to detect this movement is a good guess. Also, it intuitively makes sense that along the
several stages that comprise a step, there are moments where the acceleration rises, followed by a fall. These changes in signal trend can be called peaks, and they are the basis of the peak detection method of step counting, that analyses a window (or buffer) of sensor values and, upon detection of a peak, determines if a step is represented in it [25]. This determination is usually based on threshold values, sometimes fixed, while others averaged across previous acceleration values [26]. This approach can be followed for the several axes of the sensor and/or its vector's magnitude, the latter sometimes denoted "effective acceleration" [27].

Pattern matching is another approach (which in itself may include peak detection), which uses state conditions to extract signal characteristics, and detecting a step only when they match that of an expected step. This improves the false positive and negative count, using characteristics such as the average step period of the last detections, if one of the other axes has had any significant variations in the past, among other criteria. Sometimes several different threshold values may be employed in the different state changes[28], or even the thresholds change themselves during different states. Although peak detection may be employed with these techniques, other types of signal analysis may be applied to the signal to match patterns more closely [29].

2.1.2.1.2 Step Length Estimation In order to determine the distance travelled in a detected step, step length estimations must be performed. Early renditions of these estimations were based in physiological models that mapped and averaged step length to height and sex of a person [30] [31]. Not only should more personal attributes be employed (such as age and weight), using an average step length to estimate the length of every single step will inevitably result in errors accumulating over a number of estimates.

Some dynamic approaches employ a personal coefficient to step characteristics, such as the Weinberg algorithm [32] which relating bounce (calculated from minimum and maximum acceleration values within a step window) to a K value [30], computing step length through the product of average acceleration values and a constant [28] or even both [33].

Another approach relates step length to its period/frequency, based on a studied model that relates them linearly [21][34], and uses other means to calculate step lengths during the training phase. One of these methods involves gathering GPS readings during straight line runs and interpolating them, in order to extract step lengths and frequencies, and inserting those samples into a linear regression model to obtain its coefficients [22].

2.1.2.2 Orientation Estimation

Estimating orientation is similar to step detection in the way that it usually involves manipulation of signals to obtain a value. Most approaches employ some form of magnetometer signal manipulation, often using accelerometers to determine the direction of gravity, and adjusting its values to the Earth Reference Frame (ERF). In order to deal with periodic magnetic disturbances, filtering may be applied to this value in order to smooth it.

Since these magnetic disturbances may be too great in some instances, some approaches involving
other sensors have been considered. One of such approaches is to fuse both gyroscope and compass values, which should compensate the noise from the magnetometers, and the integration errors of an exclusively gyroscopic approach [28]. Smartphones are beginning to integrate gyroscope sensors more frequently, as have approaches that use them [14].

Another approach to manipulating gyroscope and compass values is to employ map matching, coupled with particle filtering [34]. This involves generating a series of objects at start time, called particles, and moving them according to the latest direction and length values. Then, several checks involving a predetermined map are done, such as overshooting turns through walls and balconies, and corrections are performed until a valid particle movement is found [27].

One final approach considered was Principal Component Analysis (PCA), which uses earth reference frame acceleration values to determine the direction of travel [25]. Since these acceleration values are oriented towards the earth frame, applying the referred PCA method over 2 or 3 dimensions of their vector values will return one or more new vectors whose direction is the dominant direction of all of those values, by order. Due to the properties of the method and the values used, it can be said that one of these so-called eigenvectors represents the direction of travel [35].

2.1.3 Other considered techniques

A considered technique was Wi-Fi Positioning System (WPS), which uses an access point location database and Received Signal Strength (RSS) techniques to determine the user’s current location. It has been mainly used as an indoor positioning system, where other conventional techniques were lacking in results. But with the requirement of having access point coverage wherever positioning capabilities are required, along with the lack of precision [36]. As such, this approach will not be herein considered.

2.2 Related Systems and Comparison

An important part of developing a new system is researching for similar work, even if only in specific areas of the system we intend to develop. Relevant areas to look for in similar works are navigational systems and sensor-based positioning augmentations.

A choice of the most relevant systems found on our research is presented in the next sections: the first one is a general purpose, with one using RTK and the other dead reckoning as their main positioning technology; the other two are sports oriented, with the first being a generic approach that uses vibration feedback, and the second for football, with audio feedback.

2.2.1 Nawzad Al-Salihi’s RTK Visually Impaired Guidance System

Nawzad Al-Salihi, a student at West London’s Brunel University, developed a ETA/EOA hybrid system for his PhD thesis. The system was designed to help visually impaired users navigate through urban environments, by using Network RTK (NRTK) as a positioning technique, a video feed to analyse the surroundings, and an operator for guidance [5].
Figure 2.3: System components, including MNU and NSC [5]

The system, entitled "Precise Positioning in Real-Time using GPS-RTK Signal for Visually Impaired People Navigation System", architecture displayed in figure 2.3, has two main components besides the underlying infrastructure: the MNU and the NSC. While the former includes the equipment carried by visually impaired user, the latter is the location where the directions are given from. A MNU uses NRTK to locate itself with precision, and a camera to gather information from the surroundings. All of this information is transmitted via wireless communication networks to an operator in a NSC, who then uses a voice channel to communicate with the visually impaired, giving out instructions on how to proceed to the intended destination.

The fact that this navigation system uses NRTK for positioning and gets good results, even in urban environments, makes it an interesting system to study. One could argue that using NRTK could produce different results than RTK, but studies show that it can perform just as well [37].

2.2.2 NavMote

The NavMote is a lightweight embedded device, featuring a wireless connection and sensor capabilities, designed to gather compass and accelerometer data and communicate with a wireless network of other devices, called NetMotes and RelayMotes, who then receive and process that data, to obtain a dead reckoning estimate of the pedestrian’s trajectory [13].

Dead reckoning is very sensitive to errors, due to its iterative nature, so additional techniques must be employed to reduce (or reset) this cumulative error. The approach taken by this system is to interpret step movement and detect its periodicity. Then, the step distance is estimated and recorded, so it can then be sent to a NetMote for processing, in order to save power. Wireless telemetry and map matching
are also used to augment dead reckoning performance [13].

This dead reckoning approach, along with its step detection technique to reduce errors, is an interesting way to complement GPS navigation (or even be used standalone), even if the wireless and map matching components are ignored, since they require venue specific equipment and calibrations. The precision provided may not be the best, but for the specific application of locating a user inside a known room of a building it serves its purpose.

2.2.3 Padati

Early PDR systems were usually developed around the assumption the device would be carried in a fixed position, often strapped onto a belt buckle or helmet. This ensured acceleration values would be mostly output on the same reference frame, and thus had a fixed angle transformation into Earth Frame Reference values. This approach, however, is misaligned with real world devices, that can not only be carried in multiple places, but also in different positions, depending on the user.

The Padati system [27] is a PDR system that uses step detection, a step length-frequency model and map matching with particle filtering as its approach, while maintaining orientation independence. This enables the system to be implemented in smartphone applications, which can be carried in several spots. Step detection is performed using peak detection on a Butterworth filtered effective acceleration signal, step length parameters are calibrated manually on a per-user basis and orientation is determined gyroscope readings coupled with the already referred particle filtering approach, which has also been used in other approaches [34].
This system presents acceptable accuracy results, without using GPS in any form, only the orientation independent effective acceleration values to detect steps and gyroscope values for direction. Once again ignoring the map matching component, the system presents itself as a very suitable PDR implementation for real-world applications, apart from the fact that its step length model is manually calibrated.

2.2.4 AutoGait

PDR systems, already mentioned in section 2.1.2, require a step length parameter to work, using it as the measure of distance for each step. The most simple way of supplying this parameter is by using a fixed step length, possibly calculated as a function of weight, leg length and possibly other physical traits. This would be fine if the step length was constant, which is only true if pace changes do not occur. With the step length variability coming into play, accumulation of errors may have a significant impact in the total calculated length.

![Figure 2.6: AutoGait high-level components [22]](image)

Taking this into account, the AutoGait system [22] derives a step length model (called Walking Profile) from the user’s GPS and step data, using it to determine the duration of the each step and its length. This model then relates step length with its frequency (which is well validated [21][34]) using a linear least squares fitting of the gathered data. Tests performed by the developers showed 98% accuracy, with differences of up to 26% in error rates, when compared to a fixed step length model [22].

Such a system could be useful due its adaptability to each user, and the fact that it was developed using smartphone technology shows good promise for future similar implementations. Seeing as smartphone step counters are common, combining both could be a good approach, provided only leveled paths are considered, and the accuracy of both the step detector and GPS signal are good [35][26].

2.2.5 Comparison

Out of the 4 systems analysed in depth, one of them uses GPS based technology, while all the others are either PDR implementations or parts of. While Al-Salihi’s system had some real results that were good, simulation scenarios were quite worse, which put DGPS’s accuracy claims into question. The
<table>
<thead>
<tr>
<th>System Name</th>
<th>Positioning Type</th>
<th>Accuracy</th>
<th>Technologies</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Al-Salihi’s RTK VIGS [5] | DGPS | 4.13m and 52.5cm/position (real/sim) | NRTK (GPS and GLONASS) | +Precise GPS positioning | -Big infrastructure 
- Larger user device |
| NavMote [13] | PDR | 2.2cm/meter | Dead Reckoning, Wifi Telemetry and map match. | +Cheap user device | +Uses dead reckoning 
-Weak step len. model 
-Fixed carry position |
| Padati [27] | PDR | 8.7mm/meter | Dead Reckoning, map match. (p. filter) and gyroscope direction | +Dev. orient. independence | +No need for net. or GPS 
- Manual step len. calib. 
- Map matching |
| AutoGait [22] | PDR (no dir.) | 12.5cm/meter | GPS step length estimation, | +Auto. step len. calibration | +Modular w/step detectors or PDRs 
- Needs straight In. and GPS for calib. |

Table 2.1: Comparison table of the systems analysed in this document

NavMote system had better results, with an averaged accuracy result of 2.2cm, from a test base of 20 trials. The Padati system had a similar amount of test subjects, and boasted the best accuracy value of all systems of 8.7mm.

Following the PDR trend of good results, the AutoGait system, while not including a directional component and thus not being a complete dead reckoning system, has an averaged accuracy rating of 12.5 centimeters, averaging results of different walking paces, which shows good adaptability from the system. An additional test employing different users, each calibrated separately, averaged a similar result of 14.3 centimeters, which again shows that the system is adaptable to each user.

The “Accuracy” values in table 2.1 were calculated by: averaging all real and simulation results for the dynamic tests in [5], chapter 7; calculated the estimated distance/real distance average ratio for [13], [27] and [22], with the latter including three 400 meter distinct tests, with slow, moderate and fast walking speeds.

Looking at the advantages of each system, also present in table 2.1, the analysed DGPS offers little more than the increase from regular GPS accuracy, at the cost of more infrastructure and more expensive equipment. PDR systems, even if having different results, have each advantages that solve the others’ disadvantages, which hints that an unification of these approaches could be the best solution.

### 2.3 Other Researched Systems

One considered system is present in Steinhoff’s publication from 2010 [35], studying several approaches to using PCA for directional approaches. A strong point is made regarding the promise of the technique,
trajectory plots are shown in Fig.6 and quantitative results are noted in Table 1, both from the referred paper. The plots show the severity of the effect of using each pocket to carry the device, which hints that some sort of a priori correction can be made. However, the system was ultimately disregarded because of claims in other publications that such techniques are computationally expensive [27], even if in this case there was the benefit of not requiring the device to be in a fixed position according to the user.

The Sound of Football is a system co-developed by Swedish advertising agency Åkestam Holst [38] and technology firms Society 46 [39] and TRACAB [40]. In this case, this system enabled a team of amateur visually impaired athletes to play a match of football with professional players (not visually impaired themselves). In very simplistic terms, it leverages TRACAB’s image tracking technology to track the relevant entities in a football match: the ball, players and pitch limits. With an array of cameras installed on the stadium, a 3D scene of the match can be created and transmitted to each player, equipped with an Apple iPhone smartphone. However promising, this technology, like other venue specific system, requires a complex installation and calibration, and is quite expensive.

2.4 Summary

In section 2.2.5, the accuracy values from Table 2.1 shows good promise for PDR approaches. At the same time, the disadvantages of those systems are complemented by the others’ strong points, which as stated before, suggests that some sort of combination of these approaches is a good solution.

AutoGait solves Padati’s manual calibration hardships, by performing it automatically, yet without the risk of being incorrect, detecting whether the gathered GPS samples are enough and properly aligned. Padati, however, realises AutoGait as a complete PDR system, providing the missing orientation estimation component, while maintaining the step length-frequency linear model and calibration. The NavMote solution’s limitation of a fixed carrying position is solved by Padati’s approach for step detection of using effective acceleration values, instead of 3D, as input for a peak detection algorithm and not depending on their amplitude [27].
This section presents the architecture of the proposed guidance system. In section 3.1, the requirements for a system of this kind are restated and connected to the systems, analysed in chapter 2, and some additional bibliography. In section 3.2 through to section 4.2, we present an overview of the system architecture and describe the function of each of its components, followed by some additional details taken into account in the design.

3.1 Architectural Considerations

When developing an architecture for a given system, one must keep in mind what objectives were set upon for it to solve, and what restrictions were placed to fulfil those objectives. Recalling section 1.2, the proposal for this work was study work done in related areas, including relevant systems, and use that to build a system prototype that is capable of outputting relative positions in indoor and outdoor conditions, in the course of pedestrian locomotion.". This prototype must then follow these requirements:

- **Availability**: increase the opportunities for positioning
- **Accuracy**: improved positioning accuracy and precision
- **Practicality**: system set-up shouldn’t be too troublesome
- **Portability**: the system must be easily usable during pedestrian locomotion

The first requirement involves looking at the most used positioning systems today and gathering the scenarios where they are not available. When analysing other systems, these scenarios should be considered, and if they are available, this means that positioning availability has improved. The International Navigation System (INS) systems described in section 2.1.2 present, by their nature, a good improvement from GPS, since the required information sources are self-contained, which means
no sky visibility is necessary for it work. This happens with both Integration and PDR positioning that use accelerometer signals, so both of them present themselves as good candidates.

The increased availability of these systems is also related to the "Accuracy" requirement, since the scenarios where GPS loses availability completely are preceded by moments of less accurate readings (i.e. the moments before losing complete sky visibility are the moments of decreasing accuracy). RTK systems, also present good opportunities to increase positioning precision, as demonstrated by the NRTK thesis project from Al-Salihi [5], but these systems are expensive and require more infrastructure, which made set-up less practical (another identified requirement). PDR systems, however, still present very good positioning precision and, as stated above, are not affected by skyline visibility, so they should fulfil both requirements.

In order to be appropriate for everyday usage, the system set-up (including installation and calibration) should be straightforward. GPS systems may have a Time to First Fix (TTFF) of up to 12.5 minutes, which occurs in extreme cases when almanac data needs to be re-downloaded from the satellites, but it requires no input from the user. So, any other set-up that is required from the solution is already a burden in comparison, so it should be the least troublesome possible. Dead-reckoning approaches, since they depend on sensor data, may need calibration, but is only sporadic. The AutoGait system has additional calibration steps, but was designed to run in the background, not requiring user interaction [22]. Set-up for step detectors depends on the implementation, but should not require much more than accelerometer calibration. Finally, smartphone platforms are everyday objects, which makes the installation straightforward.

As a final requirement to be considered, portability imposes that this system should be easily carried and should not hamper movement, due to the fact it will be mostly used during pedestrian locomotion. Again, smartphones were made to be everyday objects, and most of them are very easily carried in their pocket positions. Assuming the systems employed in the solution support this position, they should provide good mobility and portability. There are implementations of step detectors, employed in PDR systems, that can be used in these positions, independent from their orientation, with good results [27]. Other approaches that also use sensor readings may have some noise added in this position, but filtering can solve the problem.

### 3.2 Architecture Overview

As discussed previously in this document, PDR is an appropriate approach to take for a solution to the presented problem. From the two approaches that were examined, the step detection with step length estimation approach will provide good results depending on the step length estimation. On the other hand, the integration with step detection approach has the issue of accumulated errors (commonly referred to as drift), which is even more aggravated with low-cost sensors.

The step length estimation problem occurs due to the fact that the most common way to estimate length is static: either an averaged value for all steps (based in factors such as gender and height [30]) or for each type of activity (walking, jogging or running) [41]. A better way to estimate step length
should involve a scientific model that relates it to another observable value. Such is the case of the step frequency to length linear relation, which has been studied in the past [21][34], and even implemented [22].

![Diagram of system components](image)

**Figure 3.1: High-level components of the system and information flow**

The problem with the integration approach occurs precisely because the drift (accumulated acceleration error) grows indefinitely. This happens even faster if the sensors used are low-cost, since each update accumulates even higher errors. Relevant solutions use a technique called low-velocity updates [7], which consists of resetting the acceleration errors whenever a step is detected. Errors are still present, but they do not accumulate as quickly.

The AutoGait platform explores the former approach, building a step length model, which can relate the length of a step to its frequency. This may then be used during step detection to estimate the length of each movement, using the inverse value of the elapsed time between this and the last step.

Combining the output of these components (distance and direction), using trigonometric expressions, will result in a relative position output. Absolute positioning conversion is possible, as long the initial absolute position is known.

### 3.3 Architecture Detail

The proposed architecture is designed accounting for two different usage scenarios: Calibration and Positioning Estimation. In the Calibration scenario, step and location data are used to calibrate the step length model, while in the Positioning Estimation scenario step and orientation data, along with the calibrated model, are used to generate positions iteratively.

The Calibration scenario records detected steps, while gathering location data, in order to process them into an averaged step frequency and length average. Each of these samples, are then input into a modeller, that builds the step length model as these samples arrive. The step detector is a signal analyser, which receives an input signal and analyses it, be it each instant in isolation or according to past inputs, and registers a step every time the predetermined conditions are fulfilled.

After the Calibration scenario is performed and a step length model is built, the Positioning Estimation scenario can be performed. This scenario also depends on the step detector, described above, as input for the other two components: Distance and Orientation Estimation.
The Distance Estimator component receives the steps generated by the step detector and uses them for two purposes: calibration of the step length model and actual distance estimation. In the first case, location and step information is combined to determine the approximated length of each step, which is then stored. For the second case, only step information is received, which we complement with the information we gathered during calibration, by adding a length to each step.

The Orientation Estimator receives the orientation information and detected steps, combining them as follows: for the duration of each of the received steps, combine all received orientation values into a single one.

Finally, the position estimator receives input from both the distance and direction estimators, combining them into a position using the method described previously.

3.4 Summary

In this chapter, the considerations taken to develop the proposed architecture were detailed, as well as how the objectives are attained. INS systems present themselves as an improved approach regarding the "Availability" objective, when comparing with GPS, since sky visibility does not affect them. "Accuracy" in these low sky visibility scenarios was also considered, where GPS' performance degrades as it decreases, while those of INS do not. The "Practicality" is not too hampered by the calibration steps due
to their periodicity, and the “Portability” requirement is attainable through a smartphone implementation.

An overview of the proposed architecture was then described, detailing the abstract concepts that are involved, along with some examples of related work. The PDR concepts of step detection, distance and orientation estimation were noted as a relevant part of the architecture.

Finally, a high-level description of the components related to those concepts was exposed, as was the interactions between them and the scenarios to which they were relevant. The Calibration scenario required a Step Detection component, together with location data, to generated the needed step length model. The Positioning Estimation scenario also required a Step Detection component, together with orientation data, in order estimate a position.
Implementation

This section describes the implementation developed prototype and its components, starting with an overview section of the approach taken for each component, followed by a section that exposes relevant details for the implementation process. Following sections describe the developed Analysis Library and how it integrates into the Android platform to create the prototype. Finishing this chapter, it also details some of the methodologies involved in the implementation process, and how they influenced it.

4.1 Implementation Overview

The components described in section 3.3 are implemented as a combination of the existing related work and adaptations. The following subsections detail the proposed implementations of those components, the respective related work taken into consideration and the relevant adaptations.

4.1.1 Step Detection

The most common signal analysed to detect steps is the acceleration signal, with several different approaches. One of these approaches involves peak detection which, as the name implies, detects peaks (points in the signal that are preceded by a rise and followed by a slope) and analyses conditions in order to determine if it belongs to an instance of a step. This means that this type of step detector must be able to discern odd peaks in the signal resulting from non-step events, such as equipment shaking, foot contact or involuntary body contact.

An example of a step detection implementation such as this is present in [26], detecting peaks as they appear within a 100 sample buffer (like the ones identified in 4.1), averaging the value of those peaks, and counting only those that rise above a percentage of that value. The proposed implementation is based on this approach, which uses the magnitude of the acceleration signal for the analysis, which is the norm of the 3D vector. However, observed results from this approach were not satisfactory. So, some
adjustments were made in order to improve the step detection accuracy, by selecting the appropriate peaks.

First, it was observed that the signal that reached the analyser had two peaks after odd steps (as described in [42]). Since sometimes these secondary peaks would be high enough to be counted as steps in the original implementation, the following method was devised to exclude them safely: a peak can only be counted as a step, along with the other conditions, if the elapsed time since the last one is bigger than a percentage of the average for previous steps. This is illustrated in figure 4.2, where a portion of the acceleration signal has a set of peak candidates eliminated in this condition. This ensures that these "twin" peaks would be safely excluded, since highly rapid pace changes are unlikely, and maximize the number of properly timed steps that are counted. Other studied approaches in the past also analysed peak distance when validating steps [20].

Secondly, it was observed that before even steps there would be a valley (in contrast to peaks) that was considerably lower than the valleys in that pair of steps, as is shown in figure 4.3. In order to strengthen the analyser, a valley detection process was added, which would count valleys that were below a given value. Then, only two steps could be counted until the next valley was detected, indicating that a pair of steps was taken, as was stated in the beginning of this paragraph.
4.1.2 Distance Estimation

There were fewer approaches for distance estimation, but the AutoGait approach [22] claimed to have promising results. This coupled with the fact that it adapted to each user, was based on an established scientific model [21][34] and was designed for smartphone technology from the get-go (including the calibration step), making it easy to deploy, made it a very promising approach to follow.

The AutoGait system works by building a Walking Profile for each user, through the gathering of straight-line GPS segments, along with its steps. As shown in figure 4.4, at the end of each segment, the step frequency is averaged, along with the corresponding length of each step. This average step length value is calculated by taking the length of the segment, and dividing it by the number of steps. The pair of averaged frequency and length values is then inserted into a Linear Regression model which, with enough samples, would output a linear function defined by its slope and intercept values.

It is this function that represents an approximation of the linear relation between step length and frequency (called Step Length Profile, in the bibliography’s implementation). Once the model is built, it can be consulted during step detection in order to obtain the length of a given step with frequency $f$. 
obtained by calculating the inverse of the elapsed time between this step and the last.

### 4.1.3 Orientation Estimation

Direction estimators are similar to step detectors in the way that they receive sensor signals and perform some sort of analysis on them. Some of these use world coordinated accelerometer signals to perform PCA and discover a dominant directional vector among the samples [43]. The most common approach, however, involves using the digital compass, which is a combination of both acceleration and magnetic signal values, as the direction value [28][14].

![Figure 4.5: Combination of several Orientation values during a Step](image)

Since several compass values are collected up to the point of Orientation Estimation itself, they should be combined into a single one. The method considered for doing this was to average all values during a step, an approach already explored in past implementations of PDR systems [44]. Figure 4.5 shows both Raw and Step Averaged orientation values, displayed on top of the blue plotline, in Radian units.

### 4.1.4 Positioning Estimation

Finally, the Positioning Estimator combines these three sources of information into positions. After each step, a distance estimation is fetched from the step length model, and, combined with the direction during that movement, a set of offset coordinates is generated. These offset values are calculated by multiplying the estimated step distance by the $\cos$ and $\sin$ of the estimated orientation value, for X and Y offsets respectively.

Assuming the user starts in the (0, 0) position, we add the offsets to this value to obtain the new position, and do the same process after each step. These positions are relative to the starting point, but if an absolute position is gathered (using GPS), a new absolute position can be calculated after each update.
4.2 Implementation Details

This section presents some of the finer details that had to be taken into consideration when implementing the proposed solution.

For example, there was no mention to how noisy the input signal could be for both the step detection and orientation estimation. In order to counter this, the signal was filtered before being input to their respective components. For step detection, the guideline present in [26] and [27] was to use a Butterworth filter, due to its flat output response, no ripples and previous successful implementations. It was observed that an order of 10 would suffice, and that the cut-off frequency to be used was 5Hz, since it was over the observed value for the fastest cadence that a human being has been seen running \(^1\), instead of the one used in the referenced implementation. Lower cut-off values may have been used in the past [44], but those systems contemplated only walking step detection and not running.

For orientation estimation, the used filter was a mean filter, with the samples taken from the duration of a step [44]. However, since the orientation is represented by a circular quantity, as in, it is represented by a value inside an interval, whose values loop at the end, the mean between values on opposite ends of the interval will end up averaging to values in the opposite end of the scale. For example, 0.1 and \(2\pi - 0.1\), two angles in that interval that represent a minimal orientation change, have an average value of \(\pi\), which is exactly in the opposite direction. This could be solved by averaging the orientation value including the number of turns, like \(2\pi - 0.1\) and \(2\pi + 0.1\) for the previous example, but there is no precise method of determining a turn based on the orientation value alone.

The implemented method is commonly referred to as ’mean of circular quantities’, and it involves averaging the orientation values in a Cartesian coordinate format, instead of directly. This is done assuming that the angle is a value in the polar form, with argument equal to its value and an absolute value of 1. After averaging the Cartesian values, the value of the average can be converted back into polar form. A similar approach is done in [45], where the \(\sin\) and \(\cos\) values are averaged, which is exactly what is done to convert from polar to Cartesian coordinates.

This implementation uses Euler angles as an orientation vector, whose first value, called the azimuth, maps the direction in relation to the ERF. Using Euler angles to represent orientation is common, but has a limitation: if the azimuth axis of the body becomes parallel to the plane of the earth (displayed in the middle of the plots of figure 4.6), the azimuth and roll orientation values become unstable and unreliable. This happens because ”any three-parameter representation of 3-D orientation is inevitably singular at some point”, quoting [46], in reference to [47]. This phenomenon is sometimes referred to as ”Gimbal Lock”, [14].

In order to have a ground truth during development, both walking and running experiments were performed, recorded with both the platform to use and a video camera. This allowed a near direct mapping of the events in the video with the acceleration signal. It was then possible to determine which parts of the signal belonged to each phase of the gait cycle, pinpointing possible events in the signal that could be used to capture a step.

\(^1\)http://www.drgregwells.com/wells-blog/2012/8/5/100-m-dash-sport-science-preview.html
The original AutoGait implementation assumed that the location framework from the smartphone platform (in this case, the GPS sensor) could be polled whenever a step event occurred. When implementing it in a different platform, which did not have the same type of API available, an adaptation had to be done to the condition to terminate a GPS segment. In the version of the original publication, "if the time interval between two consecutive GPS readings is considerably larger than the values of all of the recorded intervals, it is likely that the mobile user has stopped walking. If the time interval between two steps is greater than the sum of the mean and three times of the standard deviation of the time intervals as a whole, we divide it into different segments".

In practice, this means that the user has taken steps too slowly, or even stood still, before resuming normal movement. So, the equivalent condition, for a system without the original location polling API, would be that if there are no step readings between two consecutive GPS readings (which occur 1 second apart), we consider that the user has not moved enough, and terminate that segment.

A problem faced when determining the frequency of a step occurs when the first step, or the next step after a pause, is taken, and its elapsed time needs to be measured. Since there is no previous step to measure the time from, the only alternative is to make an estimate. The approach taken is to assign a fixed value to this step's duration, which will result in a fixed frequency, and thus a fixed length.

### 4.3 The Analysis Library

With multiple signal transformations and analysis steps, a modular structure of code becomes almost essential. This enables both ease of swapping components, even during run-time, and good separation of filtering and analysing functions. If required by future developers, it also helps new developers integrate a new filter function or analyser into an existing system, provided they have a basic understanding of how the library functions.

This library has two main building blocks you can use to build a new system: Reading Sources and
4.3.1 Sensor Reading

Sensor Readings are the main unit of information for this library, and contains the essential attribute of a recorded sensor reading: a timestamp value. For output formatting reasons, if a string version of this timestamp is stored at creation time, it will be output with the exact same format.

In order to fulfill their specific purpose, several extensions to this Sensor Reading class were made (which is illustrated on Appendix A, figure A.1), each belonging to a specific type of sensor or Analyser (detailed in section 4.3.4):

4.3.1.0.1 Acceleration Reading  An Acceleration Reading (AccelReading in the source code) is used to store the information output of an accelerometer sensor, namely the acceleration values in all three X, Y and Z axes. Upon creation the norm value is also calculated and stored.

4.3.1.0.2 Orientation Reading  An Orientation Reading is used to store the information returned from a Digital Compass sensor. This information, which is also part of the object’s attributes, is the device's Azimuth, Pitch and Roll values. An additional attribute is recorded if it passed upon creation: an aver-
ageable version of the values in this reading, used for compatibility with average filters (more details in section 4.3.3.0.6).

4.3.1.0.3 GPS Reading  A GPS Reading is used to store values from a GPS sensor. The attributes that can be stored are latitude and longitude, and optionally speed and bearing.

4.3.1.0.4 Step Reading  A Step Reading is an extension of an Acceleration Reading, containing two extra attributes: step length and step frequency.

4.3.1.0.5 Relative Position Reading  This reading represents a relative position, merely containing an X and Y value, and GPS Reading, if absolute positioning is desired.

4.3.2 Reading Circular Buffer

A Reading Circular buffer is the system component responsible for storing Sensor Readings for each component, providing an array-like structure and usage. This is achieved internally by behaving like a circular buffer, adding each component to the array position of the oldest element, while maintaining an age index mask for external accesses (both reading and writing). In addition to this, a mechanism is provided to clear all values except the last two, as is a check to assure that the buffer has been filled at least once (the name used to described this state was “warm”, alluding to a warm-up phase).

The Reading Circular Buffer implements this aspect of these components, but its behaviour is extended by two other that may be used for filtering functions: the Average Circular Buffer and the Butterworth Circular Buffer. They are described in the following sections.

Average Circular Buffer

The Average Circular Buffer was created to fulfil the same purpose as the regular Reading Circular Buffer, but instead storing averaged Sensor Readings. This is achieved by having two buffers store values in parallel: one for each calculated average term (the reading divided by the number of desired elements of the average) and the other for the actual calculated values. While the latter is used for history purposes, the former is in place to allow cancellation of older terms, to allow adding a newer one. The "warm" functionality can be used to check if the average value currently present in buffer has the required number of elements to be complete.

Butterworth Circular Buffer

This Reading Circular Buffer was created with a similar intent to that of the Average Circular Buffer: to store ButterWorth filtered Sensor Readings, as well as values needed for future calculations. While for the average case the past values were stored for history reasons, here the ButterWorth filter's Infinite Impulse Response (IIR) behaviour imposes that previous outputs should be stored, as they are used in each calculation. Besides this, previous inputs are also stored, for the same reason, as are
the coefficients needed, computed according to the desired order and cut-off properties (these last properties are stored in a ButterworthData object).

### 4.3.3 Reading Sources

A Reading Source object represents an object that generates readings over time. Once a Reading Source receives a reading, it performs any transformations it has to, stores the value and outputs another reading, usually equal to the transformed reading it generated.

Reading Source objects can either be standalone or connected between themselves. In standalone scenario, readings will be fed manually and the output stored internally. With an interconnected scenario, any readings fed will be pushed to any reading source that is connected, which will then perform similar operations to these pushed readings. This is achieved through the Observer pattern (from Object-oriented programming), where the first reading source is the observable object and the next reading source in the chain is the observer.

Two Reading Source object types exist: a Raw Reading Source and Filters. They will be described in their following sections.
A Raw Reading Source simply receives a reading and stores it in its internal circular buffer, whose size has a size specified at instantiation). No other transformations are performed on the readings before storage, and a history of readings as far back as the circular buffer can go can be provided.

**Step Reading Source**

A Step Reading Source receives step readings, and according to previous input, calculates and sets the step frequencies for each step. As was mentioned in section 4.2, the first step received will have no previous input to base its frequency calculation on, and an hardcoded default frequency value will be set instead.

**Filter**

A Filter object is similar to the Raw Reading Source object, but it performs transformations on the readings it receives. Depending on the actual filter that it is simulating, it may need to store “raw” readings along with the transformed ones, providing further history.

Two implemented filter objects include a Moving Average Filter and a Butterworth Filter, both described below.

**4.3.3.0.6 Moving Average Filter** A Moving Average Filter outputs a value that is equivalent to the moving average value of the last received values, up to a certain length. When first created, this filter receives that length value, indicating that after that number of readings is received, the oldest reading...
value will be discarded and removed from the moving average computation.

This history requires that, along with the collection of computed moving average values, a collection of input readings must be saved as well. The average value is calculated by subtracting the oldest reading’s value from the previous average value (which was stored) and adding in the new reading value. This should be less computationally expensive than computing the whole average every time a new reading is received.

As described in section 4.2, orientation readings are circular quantities and, as such, need to be averaged in a different way. Instead of being processed like regular readings, they are treated like two of them, each containing one of the Cartesian components of each orientation angle, as was described.

4.3.3.0.7 Butterworth Filter  A Butterworth Filter implements an IIR digital version of this originally analog filter. The output of this filter will be a smoothed version of the input, with unwanted frequency components removed. Which frequencies are removed depends on the cut-off values specified during creation of the filter object (one frequency value in Hz for low/high-pass filters, two for band-pass/stop filters) and the sampling frequency.

\[
x'_{i} = \sum_{i=0}^{n} (x_{i} \times a_{i-k}) + \sum_{i=0}^{n} (x'_{i} \times b_{i-k})
\]

\[x', x_i\] filtered output and unfiltered input; \[n, a_i, b_i\] Butterworth filter order and coefficients.

Figure 4.10: Butterworth filter function

Due to the recursive nature of IIR filters, meaning that previous outputs influence future outputs, both input and output readings must be saved. Each output is calculated by multiplying previous inputs and outputs by their respective coefficients, which were calculated at creation according to cut-off and sampling frequencies, as well as the order value.

Figure 4.11: Before and After signals (Butterworth Filtered, 10th order, offset to minimize delay)

Smoothing of the filtered signal is determined by the specified order of the filter, which effectively affects the output by including more past inputs and outputs. It also affects the slope of the frequency response, influencing how close it is to a “perfect” filter.
4.3.4 Analysers

An analyser is a component that also receives readings from reading sources, but does not output transformed readings. Instead, it gathers that received data and performs analysis techniques to extrapolate new data, not necessarily in the form of new readings.

There were three different analysers developed for the system: a Step Analyser, AutoGait Modeller Analyser and a Positioning Analyser. Their design and purpose will be described in the following sections, with a figure illustrating their dependencies on Appendix A, figure A.2.

Step Analyser

A Step Analyser performs the duties of a step detector, described in sections 2.1.2.1.1 and 4.1: it receives an input signal and analyses it, searching for conditions that indicate a step has occurred. As stated previously, this type of step detector analyses peaks in the signal and discerns, from their attributes and previous conditions, if it should be counted as a step.

Algorithm 1 Step Counting pseudocode

```plaintext
while values in buffer do
    Calculate forward and backward slope
    if forward slope < 0 && backward slope > 0 then
        Add peak extremity
    else if forward slope > 0 && backward slope < 0 then
        Add valley extremity
    end if
end while

while extremities in buffer do
    if peak value > Step threshold && No steps while disarmed && Elapsed time > elapsed time avg * coefficient then
        if step is armed then
            Disarm step
        else
            Set step detected in disarm
        end if
        count step and push it through Step Reading Source
    else if valley value < re-arm threshold && step is not armed then
        Re-arm step
    end if
end while
```

To be more specific, this analyser receives acceleration readings, containing 3D acceleration and its norm value, and stores them in a buffer. When the buffer fills, it is processed to detect the steps, and after it is done, the buffer is emptied up to the last two values, in order to keep enough samples to not lose a peak that was "cut-off" or count extra peaks.

Processing is done as follows: the buffer values are analysed for peaks and valleys (both categorized into Extremities in the source code), which are values that are preceded and followed by lower or higher values, accordingly. After the extremities are detected and collected into a list, a set of conditions are verified depending on the type of extremity (peak or valley).
The conditions for a peak to be counted as a step are as follows: its value should be above a predefined threshold value (determined empirically, and also used in [26]), the time since the last step should be higher than a dynamic threshold value and if no step was detected since the step mechanism was disarmed. The mentioned dynamic threshold value is calculated by multiplying the current step period average ("elapsedTimeAverage" in the source code) by a relaxation coefficient that is lower or higher depending on the amplitude value of the extremity.

When a step is detected, the step arming mechanism is consulted: if it is armed, disarm it; if it was disarmed to begin with, count this step as being detected in the disarmed state. Following this, the step value is registered, which includes passing a step reading to the internal Step Reading Source to be outputted, and its period value is averaged with previous ones.

A valley is analysed against the following conditions: if it falls below a lower threshold value and the step mechanism is currently disarmed. If such is the case, the step arming mechanism is rearmed and the "disarmed step" counter is reset, indicating that a pair of steps was taken.

The extremity detection and analysis mechanisms are synchronized separately for better parallelism behaviour, and the previously mentioned step reading source makes sure attached objects are notified of a new step, by passing a step reading.

**AutoGait Modeller Analyser**

An AutoGait Modeller Analyser performs the AutoGait duties during the calibration process, excluding step detection, described in sections 2.2.4 and 4.1: it receives Step and GPS readings, performing an analysis that builds a step length-frequency model (henceforth called Step Length Lookup (SLL), as in [22]). It does this by gathering smoothed GPS readings into a single unit, called Segment, finishing it when certain conditions are met. At the same time, step readings are also collected. When a segment is finished, its GPS readings are analysed to see if a set of them together forms a straight line (a process henceforth referred to as SLI, also as in [22]), and adjusted to include only those readings and the steps within. If the segment included an appropriate straight line portion, the average step frequency and length from that segment is inserted into the SLL model.
Algorithm 2 AutoGait Modeller Analyser pseudocode

```
if received step reading then
  Add it to step buffer
else if received GPS Reading then
  Calculate speed threshold
  if no steps were detected || reading speed > threshold then
    End the segment and create a new one
  end if
  Add step readings in buffer to current segment
end if
```

Upon receiving a reading, the analyser has a different behaviour if it is a step or GPS. In the former case, it simply adds that reading to a buffer. In the latter, the first step is to see if the conditions to finish a segment are met: if the speed threshold for that reading is crossed, or if no steps were detected since the last GPS reading, the segment is finished (or segmented, as in the source code). The speed threshold is referred to in the original implementation as a part of the "Unrealistic Movement Detection", and is calculated as the sum of the average speed value so far and two times its standard deviation.

![Figure 4.13: A GPS segment and its SLI thresholds [22]](image)

The segmenting process is composed of the SLI process (as in [22]) and the calculation of the values to be input into the SLL model. The former follows the implementation from [22], analysing cumulative heading change values until all the contiguous groups of values that are below the Middle and End Point Thresholds (denoting a straight lines) are found. Then, several different segments are created and stored, including only the readings within those ranges, as illustrated by figure 4.13. After this, and if straight lines were found, the latter process occurs, obtaining the average step lengths and frequencies for each of them and inserting them into the SLL’s linear regression model. An pseudocode description of the Segmentation and SLI algorithm is present in Appendix B.

After the segment is finished and a new one is put in its place (or even if it was not segmented at all), the GPS reading is inserted into the current segment, being smoothed in the process through a convolution function, and the steps in the buffer are added to the segment. This whole process is repeated until the class is explicitly stopped, unlike in [22], where it would be halted if the model's parameters varied under a certain value, for the last five iterations. When this happens, a similar model to those displayed in figure 4.14 is ready to be used or store for future use.
Positioning Analyser

The Positioning Analyser combines steps (more specifically, their lengths) and direction into a position: it receives step and orientation readings and uses trigonometric functions to calculate X and Y offsets. It then adds those offsets to a previous relative position (or an initial 0,0 position, if there was none) and calculates the new absolute position, if an initial absolute position was recorded.

If an orientation reading is received, it is stored into an average variable, until a step reading is received. Then, when that reading is received, the averaged Azimuth orientation value is retrieved and the average variable cleared. The X offset is calculated from multiplying the step length with the cosine of the Azimuth value, and the Y offset from the same product with the sine of the same value. The new relative position will have its X and Y values containing the sum of the previous relative position’s X and Y value with their respective offsets.

If an absolute position output is desired, an initial absolute position should be recorded. With each
relative position update, the X and Y offset values should also be added to a Cartesian conversion of the previous absolute position (or the initial absolute position, if there was none) and then converted back to the respective format.

4.4 Android Prototype: SensorDirection

The application developed uses the developed library, detailed in section 4.3, in order to perform the duties necessary to prove that it is possible to fulfil the objectives presented in section 1.2. In general terms, it collects sensor data, processes and records it and the result of those operations into a file for later analysis.

This data collection should be performed during pedestrian locomotion, with the smartphone device placed in a place that favors step detection, like a trouser pocket. Beyond its main purpose of data collection, the application can also display the data generated during the calibration mode, also offering the possibility of importing and exporting that data for later use. It is also possible to archive application logs (storing them in an external folder), e-mail or clear them.

The following sections will detail how the application is structured: section 4.4.1 provides the common details between all modes of operation, followed by a detailed description of each mode of operation in sections 4.4.1.1 and 4.4.1.2.

4.4.1 Application Operation Modes

The application has two modes of operation, related to the scenarios detailed in the Architecture section 3.3, to fulfil different purposes: one for the calibration stage and the other for actual positioning estimation. Each of them fulfils tasks which are self-explanatory from their descriptions, and they share a common structure between them.

Part of this is the fact that both of them register listeners to specific types of sensors, and that they take the output of each of those sensors and input them into the specified components of the Analysis library, using a different thread, for processing purposes. One of the components that is present in both modes is the Step Analyser component, which receives the acceleration data from the accelerometer sensor and outputs steps.

All of the sensor data is displayed in the Activities for each mode, being updated every second as the values arrive. This screen is always on (more details about this aspect on section 4.5) and can only be exited by pressing the phones back button twice, in order to prevent accidental termination during collections.

4.4.1.1 AutoGait Model Calibration

This operation mode is used to calibrate the SLL model mentioned in section 4.3.4. It uses an assembly of Raw Reading Sources for both the accelerometer and GPS sensors, attaching the former to a Butterworth Filter of order 10 and a 5Hz cut-off frequency, as was described in section 4.2, which is itself
attached to a Step Analyser. The latter, together with the mentioned Step Analyser, is attached to an AutoGait Modeller Analyser, which builds the model, according to section 4.3.4.

![Diagram of AutoGait samples storage in an Android database]

When finished, the state required to rebuild an equivalent model is stored inside an Android application DB, to be retrieved the next time both modes are started.

### 4.4.1.2 Position Estimation

This operation performs the positioning estimation itself, mentioned in section 4.1.4. The acceleration sensor is attached to a similar assembly of components from the last operation mode, but instead of attaching to an AutoGait Modeller Analyser, the Step Analyser is attached to a Positioning Analyser, together with a Raw Reading Source that receives orientation readings.

When starting, this mode retrieves the samples required to initialize the SLL model, in order for the Positioning Analyser to be able to estimate the distances of each step, necessary for the calculation of each position, as well as the average sample rate of the last run, for filter initialization purposes.

### 4.4.2 Log File Structure

In order to analyse the behaviour of this prototype in the future, both the readings gathered from Android’s Sensor and Location Manager platforms and state changes of the system are recorded into a log file. Each of these events is encoded into a line, with each field separated by a comma character, with the first being the one that identifies the type of event being recorded.
Sensor events being recorded include accelerometer, magnetometer and orientation values (recorded with the letters 'A', 'M' and 'O'). This last type of values are magnetometer values fused with low-passed acceleration values (in order to obtain a gravity vector). While the first two reading types include both a 3D vector with the respective values in each axis and its computed norm value (in m/s² and μT), the latter includes the orientation values with respect to the ERF (azimuth, pitch and roll in radian degrees). Location values (recorded with the letter 'L') are extracted from the GPS events, and include Latitude and Longitude (radian degrees), altitude, accuracy (both in meters), speed (in m/s), bearing (decimal degrees, East of true North) values, as well as the number of satellites that provided it.

Application status change events, that begin with the letter 'I', are also recorded, and include descriptive messages about the events that triggered them: pausing/resuming recording, increase/decrease in the number of satellites in view, event listener attachment status and location services availability.

Finally, all positioning updates are recorded with the letter 'P' in the beginning, at the time of each update event. A line containing the generated relative position is written (with X and Y coordinates in meters), and a line containing its respective absolute position is also written.

### 4.5 The Implementation Process

The implementation process for this application began with an attempt to build initial MATLAB prototypes of each of the components, based on renditions of several of them found during the research for some
of the related systems present in chapter 2 and others [26][20][22].

In order to assess the performance obtainable by these systems, an early prototype of the application was developed, which only collected sensor data without processing it. This prototype was then distributed among several volunteers to gather data, generating enough volume of data to safely assess the capabilities of the developed prototypes. Besides this fact, the test also enabled to create a user-tested baseline for the final application, and allowed an early insight in the Android platforms infamous hardware fragmentation issues [48] as well as some issues with the Java platform.

Some of these issues included the fact that sensor data output (excluding GPS) was either slowed down or completely cut-off if the screen was turned off. This meant that, until a new mechanism is available from the Android platform, all sensor processing operations must be performed with a screen lock attained, in order to keep the screen from turning off. It is possible to attain one of these locks, while dimming the brightness automatically.

Another issue was the fact that some phones had different SD card filesystem paths, which caused the application to undesirably fill the internal memory of the device, as was the fact sensor data listeners should only run enough code to pass the values to another thread 3. Also regarding multi-threading, it was also noted that interface manipulation should be done in the thread specifically created for the task, instead of the code where they were triggered 4. One final issue to note regarding the Android platform was that the re-implementations of some classes from the native Java library may have some differences, despite there being no differences in documentation.

Libraries and functions used to create the various parts of this prototype include the Apache Commons Math library 5 and other URLs present in the source code’s Javadoc, mostly suggested or found in support sites 6.

Another fundamental part of the implementation process was the use of unit testing frameworks 7 and continuous testing plugins 8, coupled with the Eclipse IDE 9 and GIT version control system 10, which enabled agile, yet safe, refactoring of source code, due to frequent changes in requirements and the solution design.

4.6 Summary

This chapter presents the implementation approach for the components required to design the proposed PDR system. The Step Detection component was implemented following an acceleration signal peak detection algorithm, based on related work [26], with the adaptations described in section 4.1.1. Distance Estimation is an implementation from related work [22], which first creates a step frequency length model during the Calibration process, and then consulting it during the Positioning Estimation process.

---

4 https://developer.android.com/training/multiple-threads/communicate-ui.html#Handler
5 http://commons.apache.org/proper/commons-math/
7 http://junit.org/
8 http://infinitest.github.io/
9 http://www.eclipse.org/
10 http://git-scm.com/
for Distance Estimation itself. Orientation Estimation is performed using compass values for the duration of a step [44]. Positioning Estimation is performed combining both Distance and Orientation Estimation using trigonometric functions, without additional processing.

Some of the finer details that have influenced the implementation of the proposed solution were presented in section 4.2, including Euler Angles’ singularities, commonly referred to as “Gimbal Lock” [46], Butterworth filter configurations, how to average orientation angles and some finer adaptations done to both Step Detection and Distance Estimation approaches, namely the change in the stop detection conditions and the initial step frequency.

A description of the Analyser library was presented, describing the main building blocks of Sensor Readings, Reading Sources and Analysers, and the interactions between them. Analyser subtypes, which implement the bulk of the signal processing capabilities of the system, were described, each performing the role of a specific component from the proposed architecture. The Step Analyser implements the peak detection algorithm, with the mentioned adaptations. The AutoGait Modeller Analyser fulfils the purpose of the Distance Estimation component during a Calibration scenario, by computing the step length/frequency averages and inserting them into the Linear Regression model. Finally, the Positioning Analyser takes the Orientation and Step Readings, estimates the distance travelled and the orientation during that period and combines both.

The developed Android Prototype was described, detailing the modes of operation and what Analysis Library components they used. The assembly for the AutoGait Model Calibration and Positioning Estimation was exposed, along with the data stored and loaded from the Android application’s database. Since logfile entries are the main output of the prototype, this structure was also exposed, detailing how to identify each type of entry, and what they contain.
This chapter presents the evaluation done to the developed work and its results, along some comments regarding those same results. A description of the methodology employed in this evaluation is also presented, along with the reasoning behind them.

5.1 Test Methodology and Scenarios

The methodology chosen to validate the solution was to perform multiple tests, of varying characteristics, and examine each of the system’s components separately. This makes comparison with related systems that focus on single components possible, and gives a more granular perspective on the system performance. Also, due its different characteristics, Walking and Running tests were analysed separately, reflecting the different expectations for each case, where appropriate.

Since the system components perform step detection, step length estimation and orientation estimation, each of these functions should be evaluated against the real world values, which means they need to be measured in some form. So the steps should be counted, and step distance and orientation measured. Step counting was done using a chronometer with lap times, counting a lap for each step. This has the added benefit that step times are also recorded, which makes it possible to map accelerometer output to actual steps, as well as measuring the timing accuracy of the step counter.

Step length is not trivial to measure, since it involves individual measuring of each step. A considered option to do this was to perform measurements in a sand floor: footprints would be made for each step, and measured individually afterwards. However, walking on sand significantly alters walking dynamics, which would influence the results. Instead, a previously measured track was travelled, and each of its steps’ estimated length added up. Since it affects step distance, a fairly levelled field should also be considered [22]. The relationship between the measured distance and the real distance should provide a good overview accuracy of the system, while an averaged approach for each step length value
should also be considered to provide an estimate of how much error is incurred during each of them. Since the calibration procedure for this component requires GPS visibility, this should also be taken into consideration. Finally, the orientation estimation component is also tricky to evaluate, since a course may have a certain shape and not be traversed in the same manner by each user, be it due to distractions or other random occurrences. For that reason, a simpler scenario was considered: the track to travel has several lanes, each laid out in a straight-line shape. The straight-line shape is one of the most simple shapes to be travelled, and the floor marking should minimize human error by providing guidance. The output direction is then considered on a step-by-step basis, and this trajectory compared against that of straight line. The method chosen to define this straight line is to gather averaged compass readings at the beginning of each test. This provides the direction to maintain during the whole test, and the value to compare the estimations to.

Grouping these considerations together as a requirement, a suitable location was then chosen. It should have appropriate pavement, have straight line lanes marked, as well as their distance, and should be fairly levelled. Some cycle routes have distance markings along them, but not all of them have significant stretches of straight lines, nor GPS visibility, needed to perform calibration of the step length model.

Such a site was discovered near the "Amadora Este" Metro station (see figures 5.1, 5.2 and 5.3), with ground measurement marks every 25 meters, with a straight line of up to 600 meters. Several runs, of varying distance and cadence, were made this test site. Since the track is a straight line, there are very little differences while travelling forwards and backwards. The distances considered were 100, 400 and 500 meters for walking tests, as well as 100 and 200 meters, for running tests.

The considered tests were performed in their respective step cadences (either walking or running),
in their various distances, with a Samsung Galaxy W I-8150 Android placed in one of the trousers’ pockets, while inspecting the track markings for the travelled distance. In order to help prevent manual miscounting of steps, an additional Android phone with a chronometer was used, performing a lap-time for each step. The result of each test was then stored for later analysis, along with the captures from the main Android phone.

Steps taken during these tests should be as natural as possible, without forcing a specific walking pattern or pace. However, counting the steps induced a rhythm, and thus a constant pace, which could influence the test results. In order to minimize this bias, long runs were performed (400 and 500 meters), which would reduce the rhythm inducing effect of counting process seamless along the way.

5.2 Test Results

Each of the following sub-sections describe the evaluation done to the components identified in subsection 5.1: Step Counting, Step Length Estimation and Orientation Estimation.

Each of these sections begins with a description of the aspects that are intended to be evaluated for that particular component, followed by an explanation of the formulas employed to do so. It ends with a brief overview of the obtained results, while the conclusions are presented in section 5.3.

5.2.1 Step Counting

As discussed in section 5.1, the step counting component should be evaluated on both count and time accuracy, in order to assess, at the same time, that no steps are being missed and that the steps counted are properly measured. Since each step represents a positioning event, every miscount represents an error in this regard. Also, since each step’s duration affects the distance value estimated for that step, any difference will represent an error in that component.

Equation 5.2 defines the number of steps miscounted, by calculating the difference between the number of steps counted manually and the ones detected by the system. This value is then used in equation 5.2, where the step counting accuracy ratio is calculated, by dividing that value by the expected steps to obtain the error ratio, whose complement value is the accuracy ratio. The last equation (5.3) takes each of the absolute difference between the detected and the real step times and averages all of
\[ \text{StepError} = \text{abs(StepCount} - \text{RealStepCount}) \]  
(5.1)

\[ \text{StepAccuracyRatio} = 1 - \frac{\text{StepError}}{\text{RealStepCount}} \]  
(5.2)

\[ \text{AvgStepTimeAccuracy} = \text{Avg}_{i=0}^{n}(\text{abs(CountedStepTime}_i - \text{RealStepTime}_i)) \]  
(5.3)

\[ \text{abs}(x) \quad \text{Absolute Value of } x; \]
\[ \text{Avg}_{i=0}^{n} \quad \text{Average of } n \text{ elements with index } i; \]

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Steps Counted</th>
<th>Steps Detected</th>
<th>Step Error (steps)</th>
<th>Step Counting Accuracy</th>
<th>Step Time Offset Average (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12:41</td>
<td>647</td>
<td>640</td>
<td>7</td>
<td>98.92%</td>
<td>0.059</td>
</tr>
<tr>
<td></td>
<td>12:51</td>
<td>734</td>
<td>717</td>
<td>17</td>
<td>97.68%</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>13:26</td>
<td>131</td>
<td>132</td>
<td>1</td>
<td>99.24%</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>13:29</td>
<td>130</td>
<td>125</td>
<td>5</td>
<td>96.15%</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>13:31</td>
<td>131</td>
<td>127</td>
<td>4</td>
<td>96.95%</td>
<td>0.046</td>
</tr>
<tr>
<td>2</td>
<td>12:38</td>
<td>610</td>
<td>580</td>
<td>30</td>
<td>95.08%</td>
<td>0.027</td>
</tr>
<tr>
<td></td>
<td>12:56</td>
<td>641</td>
<td>622</td>
<td>19</td>
<td>97.04%</td>
<td>0.020</td>
</tr>
<tr>
<td></td>
<td>13:05</td>
<td>609</td>
<td>595</td>
<td>14</td>
<td>97.70%</td>
<td>0.039</td>
</tr>
</tbody>
</table>

Table 5.1: Results of the step counting metrics for Walking Tests

The results for the walking tests are presented in table 5.1, compiled using the gathered data from the system, the manually counted steps and the already presented equations. The Step Error average value was 12 steps, with a standard deviation value of 10 steps. Step Counting Accuracy had an average value of 97.34% with a standard deviation value of 1.37%. Finally, the average value of Step Time Offset was 36.4 milliseconds, with standard deviation of 16 milliseconds.

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Steps Counted</th>
<th>Steps Detected</th>
<th>Step Error (steps)</th>
<th>Step Counting Accuracy</th>
<th>Step Time Offset Average (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12:46</td>
<td>74</td>
<td>72</td>
<td>2</td>
<td>97.30%</td>
<td>0.100</td>
</tr>
<tr>
<td></td>
<td>12:49</td>
<td>146</td>
<td>144</td>
<td>2</td>
<td>98.63%</td>
<td>0.028</td>
</tr>
<tr>
<td></td>
<td>12:52</td>
<td>145</td>
<td>133</td>
<td>12</td>
<td>91.72%</td>
<td>0.050</td>
</tr>
</tbody>
</table>

Table 5.2: Results of the step counting metrics for Running Tests

Table 5.2 presents the similar results for the running tests. Step Error is averaged in 5 steps, deviated by 6 steps, Step Counting Accuracy is averaged 95.88%, with a deviation of 3.66%, and the average Step Time Offset was 60 milliseconds, with deviated by 37 milliseconds.

### 5.2.2 Distance Estimation

In order for the distance estimation component to be properly evaluated, both a total and a step-by-step (even if averaged) accuracy figure should be included, in order to give both detailed and overview perspectives of its performance. As was mentioned in section 5.2.1, the accuracy of the step counting
component also affects the distance estimation accuracy, so this was also considered, both for averaged and accumulated cases.

\[
\text{Distance Accuracy Ratio} = 1 - \frac{\text{abs}(\text{Distance Estimated} - \text{Real Distance})}{\text{Real Distance}}
\]  
(5.4)

\[
\text{Avg Est Step Length Error} = \text{Avg}_{i=0}^{n}(\text{Est Length} (\text{Counted Step Frequency}_i) - \text{Est Length} (\text{Real Step Frequency}_i))
\]  
(5.5)

\[
\text{Accum Distance Error} = \sum_{i=0}^{n}(\text{Est Length} (\text{Counted Step Frequency}_i) - \text{Est Length} (\text{Real Step Frequency}_i))
\]  
(5.6)

\[
\text{abs}(x)
\]
Absolute Value of \(x\);

\[
\text{Avg}_{i=0}^{n}
\]
Average of \(n\) elements with index \(i\);

\[
\text{EstLength}(f)
\]
Estimation of step length of frequency \(f\), using model.

Equation 5.4 describes how the distance accuracy ratio was calculated, by calculating the complement of the ratio between the absolute difference between the estimated and the actual travelled distance. The Average Step Length Estimation error is calculated using equation 5.5, by taking the difference values of the estimated lengths using actual and counted step times and averaging them. Equation 5.6 also uses the difference values between actual and counted step length estimations, but sums them, providing the Accumulated Distance Error for that test.

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Distance</th>
<th>Distance Error (meters)</th>
<th>Distance Accuracy (total)</th>
<th>Avg. Estimated Length Offset (meters per step)</th>
<th>Accumulated Step Length Error (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12:41</td>
<td>400</td>
<td>10,89</td>
<td>97,28%</td>
<td>0,014</td>
<td>-0,782</td>
</tr>
<tr>
<td></td>
<td>12:51</td>
<td>500</td>
<td>33,04</td>
<td>93,39%</td>
<td>0,021</td>
<td>-1,859</td>
</tr>
<tr>
<td></td>
<td>13:26</td>
<td>100</td>
<td>8,79</td>
<td>91,21%</td>
<td>0,028</td>
<td>-0,367</td>
</tr>
<tr>
<td></td>
<td>13:29</td>
<td>100</td>
<td>2,01</td>
<td>97,99%</td>
<td>0,034</td>
<td>-1,280</td>
</tr>
<tr>
<td></td>
<td>13:31</td>
<td>100</td>
<td>4,88</td>
<td>95,12%</td>
<td>0,034</td>
<td>-0,840</td>
</tr>
<tr>
<td>2</td>
<td>12:38</td>
<td>500</td>
<td>16,92</td>
<td>96,62%</td>
<td>0,060</td>
<td>0,653</td>
</tr>
<tr>
<td></td>
<td>12:56</td>
<td>500</td>
<td>10,68</td>
<td>97,86%</td>
<td>0,077</td>
<td>4,628</td>
</tr>
<tr>
<td></td>
<td>13:05</td>
<td>500</td>
<td>14,01</td>
<td>97,20%</td>
<td>0,068</td>
<td>1,448</td>
</tr>
</tbody>
</table>

Table 5.3: Results of the Distance Estimation Metrics for Walking tests

The results regarding the walking tests are presented in table 5.3, which were generated by using collected system data, the equations described previously and the actual distance of each test. The average Distance Accuracy value was 95.83%, deviated by 2.42%. The Estimated Step Length Offset had an average value of 4.2 centimeters per step, deviated by 2.2 centimeters. Finally, the average Accumulated Step Length Error across walking tests was 20 centimeters, with a 2 meter deviation.

Table 5.4 presents the results for the running tests. The average Distance Accuracy value was 66.35%, deviated by 2.25%. Estimated Step Length Offset had an averaged value of 14.1 centimeters per step, deviated by 23 centimeters. The average value for the Accumulated Step Length Error was -8 centimeters, with standard deviation of 2.59 centimeters.
Table 5.4: Results of the Distance Estimation Metrics for Running tests

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Distance</th>
<th>Distance Error (meters)</th>
<th>Distance Accuracy (total)</th>
<th>Avg. Estimated Length Offset (meters per step)</th>
<th>Accumulated Step Length Error (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12:46</td>
<td>100</td>
<td>33.95</td>
<td>66.05%</td>
<td>0.120</td>
<td>-3.011</td>
</tr>
<tr>
<td>2</td>
<td>12:49</td>
<td>200</td>
<td>58.72</td>
<td>70.64%</td>
<td>0.138</td>
<td>1.877</td>
</tr>
<tr>
<td>2</td>
<td>12:52</td>
<td>200</td>
<td>75.31</td>
<td>62.34%</td>
<td>0.165</td>
<td>0.895</td>
</tr>
</tbody>
</table>

Figure 5.4: Accumulation of Step Length error in a 500 meter test, over time (meters/seconds)

The Accumulated Step Length Error results contemplate that step length can be both under and overestimated. Another way of looking at step length estimation errors is by considering that the error is an absolute value. Accumulating these values along the experiment gives an overview of the quantity of error incurred thus far. Depicted in figure 5.4 is the plot of this accumulated value, accumulating approximately 4 meters of error every 100 meters. The plots for each test are presented in Appendix C.

5.2.3 Orientation Estimation

Section 5.1 detailed how the test procedure should be travelling in a straight-line, because it is a feasible for users to execute with acceptable accuracy. Another reason for this is that it enables the comparison of orientation estimation values against one single reference value: if the trajectory is roughly a straight line, then there should be little skew between the estimation values and this single value. This reference value is calculated by taking the first moments of each test, when the subject is in standstill and the smartphone is in position, and averaging its raw orientation values.

Equation 5.7 calculates the absolute difference between each orientation value and the reference value, converting both angles to Cartesian coordinates. While using the $\text{atan2}$ function to convert the Cartesian value back into an angle, the smallest possible difference between the two angles is calculated. Equations 5.8 and 5.9 are used to evaluate maximum and average orientation estimation offset from the reference value, and are calculated for three types of orientation values: raw, collected directly from the sensors; averaged, using a sliding mean of 71 samples\(^1\); and the averaged value for the duration of each step. While the first focuses on the worst case scenario, denoting the upper limit of the

---

\(^1\)This number of samples is based around the average sample rate of the system (71Hz); this was done to include all the maximum number of values a step can contain, due to the timespan limit considered for a step (1 second)
\[ \text{AngleDiff}_i = \arctan(\text{OrientX}_i - \text{AvgStraightOrientX}, \text{OrientY}_i - \text{AvgStraightOrientY}) \]  

(5.7)

\[ \text{MaxOrientationOffset} = \max_{i=0}^{n}(\text{abs}(\text{AngleDiff}_i)) \]  

(5.8)

\[ \text{AvgOrientationOffset} = \frac{\sum_{i=0}^{n}(\text{AngleDiff}_i)}{n} \]  

(5.9)

\[ \text{LateralOffset}_i = \text{EstimatedDistance}_i \cdot \sin \theta_i - \theta_{i-1} \]  

(5.10)

\[ \text{AccumulatedLateralOffset} = \sum_{i=0}^{n}(\text{LateralOffset}_i + \text{LateralOffset}_{i-1}) \]  

(5.11)

\[ \text{MedianOrientationOffset} = \text{Median}_{i=0}^{n}(\text{AngleDiff}_i) \]  

(5.12)

\[ \text{kthPercentileOffset} = \text{Percentile}_{i=0}^{n}(k, \text{AngleDiff}_i) \]  

(5.13)

- \( \arctan(x, y) \): Arctangent angle of \( x \) and \( y \) components, ranged \([-\pi; \pi]\);
- \( \max_{i=0}^{n} \): Maximum value in group of \( n \) using index \( i \);
- \( \text{Avg}_{i=0}^{n} \): Average of \( n \) elements with index \( i \);
- \( \text{LateralOffset}_i \): Lateral Offset at index \( i \), from the reference orientation;
- \( \text{AccumulatedLateralOffset} \): the difference between the first orientation and the reference;
- \( \text{Median}_{i=0}^{n}(x_i) \): The median value of the sample containing \( n \) elements;
- \( \text{Percentile}_{i=0}^{n}(k, x_i) \): The \( k \)th percentile value of the sample containing \( n \) elements.

offset between the expected and the actual values, the seconds provides an averaged result all-around, joining further and closer results alike into a single measure. Equation 5.10 calculates the lateral offset value from the straight line in a given position, and equation 5.11 sums these values until the last point. Finally, equations 5.12 and 5.13 calculate the median and percentile values of the orientation offset values. This value includes, respectively, 50% and \( k \)% of the sample, and is used to be compared against a related system that also evaluates this component [35].

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Ref.</th>
<th>Orientation Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Raw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max</td>
</tr>
<tr>
<td>1</td>
<td>12:41</td>
<td>-1,179</td>
<td>179.997</td>
</tr>
<tr>
<td>12:51</td>
<td>1,922</td>
<td>161,835</td>
<td>63,294</td>
</tr>
<tr>
<td>13:26</td>
<td>-0.799</td>
<td>179,632</td>
<td>29,338</td>
</tr>
<tr>
<td>13:29</td>
<td>2,392</td>
<td>179,499</td>
<td>34,196</td>
</tr>
<tr>
<td>2</td>
<td>12:56</td>
<td>-1,602</td>
<td>135,951</td>
</tr>
<tr>
<td>13:05</td>
<td>1,134</td>
<td>179,294</td>
<td>21,680</td>
</tr>
</tbody>
</table>

Table 5.5: Results of the Orientation Estimation Metrics for Walking tests (values in degrees, ranged \([0; 180]\), and meters for "Lateral")

The results for the walking test are presented on table 5.5, for each of the orientation value types ("Ref." is the considered reference value for the orientation). The average for Orientation Offset values is 35.509, 22.663 and 14.037 degrees, for Raw, Slide and Step value types, respectively, standard deviated by 15.158, 11.516 and 9.052 degrees, respectively. The average Accumulated Lateral Offset was -25.4 centimeters, standard deviated by a value of 1.417 meters.

Table 5.6 presents the results for the running tests (again, "Ref." is the considered reference value for
Table 5.6: Results of the Orientation Estimation Metrics for Running tests (values in degrees, ranged \([0; 180]\), and meters for "Lateral")

Table 5.7: Median and kth Percentile values for both Walking and Running tests (in decimal degrees)

The average Median, 75 and 95 Percentile values for the walking tests were 11.14, 17.94 and 25.24 degrees, respectively, with a standard deviation by 5.74, 11.84 and 18.75 degrees. For the running tests, these same components has averaged values of 5.74, 11.84 and 18.75 degrees, deviated by 1.78, 2.94 and 5.26 degrees.

5.3 Result Discussion

The average results of the Step Detection component for the walking tests were above 97.34%, which amounts to an undetected step every 33 that occur. Subtracting the standard deviation to this value (resulting in a 95.97% accuracy value), results in missing a step every 25 taken. Assuming an hipotetical scenario of a constant 50 centimeter step length, this results in miscounting that same value in a total of 12 meters.

Still analysing the walking tests, the average value of the number of step errors was -12, deviated by 10 steps. The confidence interval in this case is \([-22;-1]\). When comparing to the interval the similar scenario (trousers’ front pocket, 500 meter walking) presented in the publication which was the basis for the implemented step counting algorithm [26], which was \([-11;29]\), there is an intersection in values, hinting that both results are close. Since the interval is smaller (because of a smaller standard deviation value) this indicates that the system is more precise.

Finally, step time offsets were averaged 36.4 milliseconds, with an interval of \([20;52.4]\). In the worst case scenario, with the step model considered for for this evaluation (with an \(\alpha\) parameter of 0.16, and
a $\beta$ parameter of 0.52), this value will result in an error of 8 millimeters, which for every 20 steps would result in a 16 centimeter error.

Running tests present results that are partly worse. An average Step Count Accuracy value of 96%, with a standard deviation of 3.66% means that the system will lose a step every 12. Assuming the same scenario of a constant step length of 1 meter, this means losing that same meter after covering 12 meters.

Step Count Errors for these tests are smaller than those of the walking test, with a resulting interval of [-11.1;0.44]. This indicates a good count performance in terms of step count results.

The Step Time Offset interval was [23;97] in milliseconds, bigger than that of the walking tests, indicating a poorer performance in accurately detecting step times in these conditions. Considering the same step length model for the walking test, the worst case error would result in an error value of 2 centimeters. Every 10 steps, the system would accumulate an error of 20 centimeters.

The Average Distance Accuracy value of 95.83%, with a deviation of 2.42% are below those in AutoGait’s publication (98% on average) [22], but considering that for their results a different type of step counting technologies was used and did not miss any steps to be counted, they are satisfactory. Other systems showed 96% [44] and 97.5% [13] results for similar conditions, which also reinforces this statement.

These results are still much better than those of a fixed step length model, as shown in AutoGait’s publication, with errors ranging from -25.7% and 14.6%. This proves that using this step length estimation model is better than using no model at all.

Observing the Distance Error values, it can be noted that the error does not seem to grow with distance, with different distance tests showing similar results. This is different from what happened with acceleration integration PDR systems, which quickly accumulated estimation errors.

The average Estimated Step Length Offset of 4.2 centimeters provides a look at how much error is generated per step. For example, after taking 130 steps in a 100 meter path, 5.46 meters of error will have, on average, been incurred.

The average Accumulated Step Length error of 20 centimeters, is suspicious, considering the standard deviation value is 2.08 meters. A possible explanation for this is that some tests present an high value for this parameter, due to the fact that initial step times are not estimated correctly. This happens because the step time average value, described in section 4.1.1 as the first adaptation to the step counting algorithm, might not have been calculated with enough samples yet, resulting in a threshold value too restrictive to allow legitimate steps to be counted, instead counting longer steps.

Looking at the results for the running tests, the poor performance of the step length estimation model for running steps is confirmed. Distance Estimation Accuracy Averaged at 66.35%, with a standard deviation value of 2.25%, along with a an average Estimated Step Length Offset of 14.1 centimeters deviated by 2.25 centimeters, the performance is much weaker than that of the walking tests. This had already been hinted in AutoGait’s publication itself [22] and others [13].

The Orientation results are presented in the form of a comparison between the raw orientation values of a digital compass, the same values smoothed through a sliding mean process and the implemented
process of calculating the average of the orientation values during a particular step. The metrics considered are mostly around angular offsets from a reference orientation, but also in the form of metric distance for the Step Mean estimation method.

Comparing average Orientation Offset values between orientation offset value types, it is clear that the Step Orientation estimation method is superior to the others, strengthened by the fact that its standard deviation values are also lower, showing less spread. For these same values, running tests perform better than their walking counterparts, a fact that can only be explained by the fact that these experiments are performed in shorter distances (coinciding with lower values for 100 meter walking tests).

In order to expose the impact of these values, the lateral distance offset from the predicted trajectory was also analysed, with the worst results being 1.2 and 1.5 meters and the best 2.5 and 3.7 centimetres, which are good results. Since there are weak results in both running and walking tests, as well as long and short, no correlation was found between the test characteristics and their outcome.

When looking for the results of related work, it was hard finding some that measured orientation estimation error to evaluate that platform. Steinhoff’s work [35], performs this analysis, presenting results for median, 75th and 95th Percentile orientation errors. Its best solution performs better than the one proposed, beating even the results of the running tests. This indicates that this component could use some adjustments in the future, in order to perform to its best potential. One of the considered adjustments to do is to perform low-pass filtering on the raw orientation values before computing the step orientation average, adjusting the weight according to the current step frequency [44].
Conclusion

Given the increasing need of location capabilities with both high availability and accuracy, a system was proposed, compiling several related solutions and joining them together into a single prototype. Systems with augmented GPS technology were considered, but preference was given to accelerometer sensor approach. In this category, both Acceleration Integration and Pedestrian Dead Reckoning (PDR) approaches were analysed, and due to their characteristics, preference was given to PDR. Different PDR systems were analysed, some of them focusing on improving different components that composed their solutions (be it orientation [28][27] or distance estimation [22]).

An architecture for the proposed solution was presented, defining the several components: Step Detection, Distance and Orientation Estimation, which are then combined into Positioning Estimation. The interactions between these components was described, both for the calibration and data collection scenarios, highlighting what data is exchanged between them in each of those scenarios.

While developing the architecture for this solution, a signal processing framework was created, in order to modularize its several components. The components developed were Sensor Readings, Reading Sources, which contained Reading Circular Buffers, and Analysers. Sensor Readings encapsulated the values collected from the sensors, in order to be pushed through Reading Sources (some of them Raw Reading Sources, some of them Filters) to be buffered, smoothed and guided into Analysers, which perform signal processing tasks, and extract the expected conclusions.

Using this framework, an Android prototype was developed, taking advantage of the ample development documentation, API capabilities and portability. Two scenarios for the prototype were chosen: calibration and data collection. Each of them used different assemblies of the library components, but both of them shared similar usage. Other factors taken into account for choosing this platform were its wide adoption and ease of use, even if some of the expected functionality was not as usable as expected, prompting the adoption of a set of workarounds.

After development, the prototype was evaluated across its step detection, distance and orientation estimation capabilities. Positive results were obtained for the Step Detection component, when com-
pared to the publication on which it was based on[26], in particular with a smaller standard deviation.

The Distance Estimation results were mostly positive, and while they were not on par with the AutoGait publication [22], other systems had closer results [44][13], and a clear improvement over a fixed length step model was still clear. It was also noted that this component’s results could have been affected by the implemented Step Detection component, and that there were disparities between distance estimation errors, perhaps hinting at the need for more testing.

Finally, Orientation Estimation results for the Step Averaging method showed an improvement over raw and sliding mean filtered orientation signals, with less average orientation offset and standard deviation. When compared to related work, however, the results were not so positive, indicating that the approach followed could be improved.

The following section details other aspects where this solution could be augmented in the future, or how other projects working towards the same objectives may approach their own solution.

### 6.1 Future Work

The developed solution is based on the assumption that the phone rests in the users’ pockets. Some of the included bibliography mentions the possibility of alternate placements, and this option could be explored in the future to expand usage scenarios. Another avenue to further positioning opportunities may come from activity detection, by analysing the patterns shown by the acceleration signal (as illustrated by figure 6.1)[27].

![Figure 6.1: Movement Pattern Recognition](image)

According to [44], a good way to estimate direction in a smartphone PDR system is to take averaged low-pass filtered orientation value during a step. This is based on the assumption that orientation doesn’t change mid-step, due to the physiological characteristics of human walking. To offer further smoothing,
the position calculated using step detection and the orientation is then input into a Kalman Filter, along with the GPS readings. Another approach that uses this is seen in [49], where accelerometer step detection is used to minimize compass orientation noise, resulting in errors of $6.4^\circ$, from $45^\circ$. Any of these approaches should be considered to improve orientation estimation results.

The current application provides step detection capabilities using the device’s sensors, processing the readings from these elements. This not only puts load on the device’s processor, and also generates a bigger battery drain. Given the modularity that the Analysis Library provides, it is possible to detect steps using an external equipment, connected to the device via wireless technology, which should alleviate the referred concerns. It may also provide the opportunity to improve the existing system. Another way of saving battery power is to apply the AutoGait algorithm to the readings in an offline platform.

As mentioned in section 4.2, the first step taken during collection will always contain an estimated frequency, due to the fact that there were no previous step readings to compute the timestamps from. A future approach to consider might be to delay the position output until the next step, and use that value instead, since it is likely that there are no sudden cadence shifts during the first couple of steps.

Finally, further tests to the distance estimation component should be performed, in order to confirm the positive results, and minimize the spread in error. Further testing with running scenarios could also hint if a separate step length model, calibrated specifically for them, could improve the results.
Sensor Readings are the objects that are traded between Reading Sources and Analysers, encapsulating the sensor values that were collected. Its design and dependencies are detailed in figure A.1.

Analysers are the components that receive Sensor Readings and process them, in order to extract features and conclusions from one of several signals. Its design and dependencies are detailed in figure A.2.
Figure A.1: A UML diagram of the Reading components
Figure A.2: A UML diagram of the Analyser components
Segmentation Functions

The following algorithm describes, in more detail, both the Segment() function and its intermediate step function PerformSLI(), mentioned in section 4.3.4.

**Algorithm 3 Segmentation Algorithm Pseudocode**

```plaintext
function Segment()
    PerformSLI()
    for all straight lines identified do
        Calculate average step length and frequencies
        Add values to Linear Regression model
    end for
    Create new segment
end function

function PerformSLI()
    Calculate cumulative Heading Change values
    while there are cumulative HC values in the list to be analysed do
        Register all the contiguous cumulative HC values between MT and ET as a straight line
        Remove the straight line cumulative HC values from the list
        if no straight lines were found then
            Remove the first value from the HC value
        end if
    end while
end function
```
Accumulated Distance Error

The following plots show the Absolute Accumulated Distance Estimation Error across all experiments, illustrating the different error densities of each case.

Figure C.1: Accumulated Distance Estimation Error (Day 1 - 12h41)
Figure C.2: Accumulated Distance Estimation Error (Day 1 - 12h51)

Figure C.3: Accumulated Distance Estimation Error (Day 1 - 13h26)

Figure C.4: Accumulated Distance Estimation Error (Day 1 - 13h28)
Figure C.8: Accumulated Distance Estimation Error (Day 2 - 12h49)

Figure C.9: Accumulated Distance Estimation Error (Day 2 - 12h52)

Figure C.10: Accumulated Distance Estimation Error (Day 2 - 12h56)
Figure C.11: Accumulated Distance Estimation Error (Day 2 - 13h05)
Bibliography


[38] Holst, A.k.: Åkestam Holst: The Sound of Football


[40] TRACAB: TRACAB: The Sound of Football


[45] Ladetto, Q., Gabaglio, V., Merminod, B.: Combining Gyroscopes, Magnetic Compass and GPS for Pedestrian Navigation. 205–212


