UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DEVELOPMENT OF A PEDESTRIAN DEAD RECKONING SYSTEM FOR SMARTPHONE DEVICES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

VICENTE ALBERTO CORTÉS PUSCHEL

PROFESOR GUÍA:
MARTÍN ADAMS

MIEMBROS DE LA COMISIÓN:
RODRIGO MORENO VIEYRA
TOMÁS LUNGENSTRASS POULSEN

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: VICENTE ALBERTO CORTÉS PUSCHEL
FECHA: SANTIAGO DE CHILE
PROF. GUÍA: MARTÍN ADAMS

DEVELOPMENT OF A PEDESTRIAN DEAD RECKONING SYSTEM FOR SMARTPHONE DEVICES

Los sistemas de posicionamiento de personas al interior de edificios no funcionan de manera correcta con GPS, dándose precisiones de bajo orden. El uso de sistemas de posicionamiento basados en Wi-Fi y Bluetooth, han estado al alza en los últimos años, logrando promedios de estimaciones cercanas a los 2 metros en condiciones favorables, solucionando los problemas existentes de GPS. De manera de poder mejorar estos valores, se complementan esos modelos con sistemas basados en los sensores inerciales de los teléfonos móviles. Usando *Pedestrian Dead Reckoning* (PDR), el propósito del presente trabajo es evaluar y comparar entre métodos ya existentes del estado del arte, y modificaciones de estos.

Para calcular la posición de una persona usando PDR, se deben realizar tres procesos: detectar cuándo una persona está dando un paso, determinar el largo de este paso, y estimar la dirección hacia dónde lo está dando. Con esto, la suma acumulada de los pasos permite obtener la posición final de la persona. En cada uno de los tres algoritmos, se analizan los principales inconvenientes asociados a los sensores, ya sea por el ruido presente en estos, o a factores externos.

De manera de poder realizar buenas comparaciones, se genera un dataset de caminatas en diferentes ambientes de oficina, teniendo casos límites con movimientos erráticos en ciertos instantes. Se agrega a esto el dataset de la competencia IPIN 2019, la cual se usa para comprobar caminatas más largas.

Para la detección de paso, se hace uso de métodos clásicos de análisis y procesamiento de señales, empleando los datos filtrados del acelerómetro para realizar tales detecciones. En el caso de la estimación de largo de paso, se usan redes *Long-Short Term Memory* (LSTM), en conjunto con *Denoising-AutoEncoders* (DAE), comparándose estos con modelos tradicionales de estimación. Finalmente, la orientación se estima usando un filtro de Kalman extendido, el cual se complementa con una variación de los sistemas de detección de anomalías magnéticas existentes en el estado del arte.

Evaluando los sistemas desarrollados, se llega a la conclusión que el modelo propuesto puede compararse con otros sistemas a nivel del estado del arte. Los mejores resultados se obtienen usando la red LSTM-DAE para estimación de largo de paso, uso de filtro de Kalman extendido con reprocesamiento de datos para estimación de orientación, uso de *Adaptive Jerk Pace Buffer* en la detección de paso, y caché de las orientaciones dependiendo del grafo utilizado para el sistema en conjunto.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: VICENTE ALBERTO CORTÉS PUSCHEL
FECHA: SANTIAGO DE CHILE
PROF. GUÍA: MARTÍN ADAMS

DEVELOPMENT OF A PEDESTRIAN DEAD RECKONING SYSTEM FOR SMARTPHONE DEVICES

When using positioning systems to determine the location of a person inside a building, the Global Positioning System (GPS) doesn't work as expected, giving an overall low precision in the position estimate. The use of Wi-Fi and Bluetooth based Indoor Positioning Systems (IPS), have been on the rise these last couple of years, achieving averages close to 2 meters of error in favorable conditions, solving the existing problem of GPS based systems. In order to improve these results, the use of Wi-Fi and Bluetooth systems may be complemented with models based on the inertial sensors of mobile devices. Using Pedestrian Dead Reckoning (PDR), the main purpose of this thesis is to create the complementary system of an IPS model, comparing both state of the art and a novel model proposed by the author.

To calculate the position of a person using PDR, three processes must be performed in parallel: the detection of when a person is taking a step, determining the length of this step, and estimating the orientation at which the step was taken. Having done these processes, the cumulative sum of the steps allows the determination of the final position of a person. In each of these three algorithms, the main drawbacks associated with the sensors, are either the noise present in these or external factors such as magnetic perturbations.

In order to make good comparisons, a walk dataset is generated in different office environments, having borderline cases with erratic movements at certain moments. The Indoor Positioning and Indoor Navigation (IPIN) 2019 competition dataset is added, using it to check the overall performance of the model in longer walks.

For the step detection process, classical methods of analysis and processing of signals are used, using the filtered accelerometer data of the mobile device to perform such detections. In the case of the Step Length Estimation (SLE) model, a neural network model that uses Long-Short Term Memory (LSTM) network is generated, coupling it with Denoising-AutoEncoders (DAE) to make the model more robust to noise. This network is compared to classical SLE models, checking the overall performance of both. Finally, the orientation is estimated using an Extended Kalman Filter (EKF), which is complemented by a state of the art magnetic anomaly detection system.

Evaluating the proposed model, we conclude that it can achieve the accuracy of other state of the art systems. The best results overall are obtained using the LSTM-DAE network for step length estimation, the use of an EKF model with data reprocessing for orientation estimation, and the use of an Adaptive Jerk Pace Buffer in step detection, using as well an orientation cache, depending on the graph of the map.

To all those people
who once believed in me.

Acknowledgements

Agradezco en primer lugar a mi familia. A mis padres, que siempre se preocuparon para que no me faltara nada, apoyándome en todo momento. A mi hermano, porque a pesar de la distancia, siempre estará ese lazo que nos une desde niños.

A los integrantes de la comisión, muchas gracias por permitirme trabajar con ustedes, son personas a quienes les tengo mucho respeto y admiración.

A la empresa Arara y cada uno de sus integrantes, habiéndome recibido con los brazos abiertos, y dándome la oportunidad de poder realizar el presente trabajo en conjunto con ellos. El que me hayan permitido participar con ustedes en la competencia IPIN 2019 fue un orgullo gigante.

A aquellos amigos y amigas que me han estado acompañando durante todos estos años. A *Los Bellos*: Javier, Oscar, Feña, Mauro, Cata, Alonso, Nico, Alfredo, Vicho, Cristóbal y Diego, que desde el primer día de la universidad han estado ahí para apoyarme en los buenos y malos momentos. A Valeria, por haberme hecho un hombre muy feliz. A *Las Ñañas*, con quienes compartí tantas risas. A Valentina, amiga desde el colegio.

A todas estas personas les agradezco desde el fondo de mi ser, puesto que no sería quien soy, si no fuera por ustedes.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Motivation

These last couple of years, services based on Location-Based Services (LBS) have become essential in a wide range of industries: Security, mass monitoring, smart storage, marketing, mobile health, and virtual reality are some of the areas where it has managed to flourish exceedingly.

The global satellite navigation system (GNSS) and global positioning system (GPS), play an essential role when using LBS based models, getting results with precision errors of less than 1 meters, but only if the person is in an open area. On the other hand, the performance of GPS and GNSS systems is reduced in a considerable way when wanting to perform the same process in a closed environment.

According to an environmental study of the United States, people spend near 70% to 90% [1] of time in a closed environment. This is why in order to meet the needs of a potential market, a reliable and in real-time IPS system has to be created.

Smartphones have a high number of sensors, which have reached similar technical capabilities, or even better in some cases, to those in a home computer. The technology of these devices provide an excellent tool to develop Indoor Positioning Systems, this being the reason why we tend to use such instruments to carry out the IPS process.

1.2. Context and problem definition

Arara SpA is a Chilean software development startup. It offers different types of products and services, including captive portals and solutions based on Machine Learning; some of the latter corresponds to IPS systems, Data Analytics, and Computer Vision solutions.

Currently, the IPS system provided by Arara makes use of RSS measurements (Received Signal Strength) given by different Access Point locations adjacent to the smartphone device. This means that the phone receives the signal strength for each nearby Acess Point (AP), and based on the values received, the decision is made as to whether the person with the telephone is in one position or another. The main drawback of this system is that incorrect estimates may be made. This can happen since RSS measurements are non-deterministic, with each AP transmitting signals

with a significantly large variance on their measured values, which may generate discontinuities in the position of the person at the moment of estimating it.

It is in this context where the objective of the work falls; there is the need to generate a system capable of correcting such discontinuities in position, using some method not too computationally expensive so it can work in real-time in a smartphone device.

1.3. General Objective

To generate a system based on Kalman Filters and Machine Learning for the use of Pedestrian Dead Reckoning in the context of smartphone applications.

1.4. Specific Objectives

- To develop a database of different smartphone devices, to compare the walks of different people in various physical environments.
- To formulate and develop a position prediction model for indoor environments.
- To develop and use comparison metrics for both the novel proposed model and the classical methods of IPS. For the step prediction process, we used the difference between the real amount of steps and predicted steps to compare the different models. In the case of the step length estimation, the relative error between the actual step length and the estimated length is used for this same purpose. Finally, the orientation estimation makes use of the mean and standard deviation between the real and predicted steps as a comparison metric.

1.5. Outline

The structure of this document is as follows:

- **Chapter 2. Theoretical Framework:** In this chapter, both the context of the project and the theory necessary to understand this document are formulated.
- **Chapter 3. Methodology:** The databases used in this project are introduced, explaining how they were obtained. Also, the method used to apply Pedestrian Dead Reckoning is formulated.
- **Chapters 4-7. Results and analysis:** Each of the different models to be used is formulated, using the complete system to the different databases. An iterative process of possible changes to the system is made, modifying filter parameters based on a metric to define. Finally, an analysis of the proposed methods and results is performed, verifying if the initial objectives were met.
- **Chapter 8:** Conclusions are presented regarding the project carried out, proposing future work based on the results obtained.

Chapter 2

Theoretical Framework

In this chapter, the key concepts to understand the work in question is presented. Similarly, both systems used in state of the art and their applications in the area are shown.

2.1. Electromechanical systems

The purpose of the present section is to understand how the sensors present in smartphone devices operate, while also comprehending the purpose of each one of them in relation to the Pedestrian Dead Reckoning context.

In the context of smartphone devices, microelectromechanical systems (MEMS) correspond to electronic elements of the range between 1 and 100 micrometers which are capable of exercising mainly as sensors or actuators.

Over the past decades, developers and researchers around the globe have generated such MEMS devices for purposes such as temperature measurements, pressure, inertial forces, presence of unwanted chemical components, levels of radiation, magnetic fields, and many others [2]. Thus, several of these instruments have demonstrated higher performances than those of their respective macroscopic counterparts. For example, the microscopic version of a pressure transducer usually has better results than those of its equivalent counterpart at the macro level.

Thus, a high range of MEMS has been added to smartphone devices, managing to have high-end resolution capacity, allowing the development of applications that were previously not possible to create. Therefore, the present case study makes use of three of the primary sensors found in most smartphone devices.

2.1.1. Accelerometer

This sensor allows measuring the acceleration of gravity in conjunction with the acceleration of the body movement along three orthogonal axes [3] of its device.

In the case of MEMS accelerometers, these are divided into two groups, depending on the way in which acceleration is measured:

- The measurements come from the change in displacement of a mass supported by a hinge.

- The acceleration generates a change in tension of a mobile section, which gets closer or further away from an array of sensors that measure changes in capacitance. This change in capacitance denotes the change in acceleration for a certain axis.

Most MEMS accelerometers used in mobile device nowadays, fall into the second category. An example of these is shown in the Figure 2.1, where the electromechanical principle is shown.

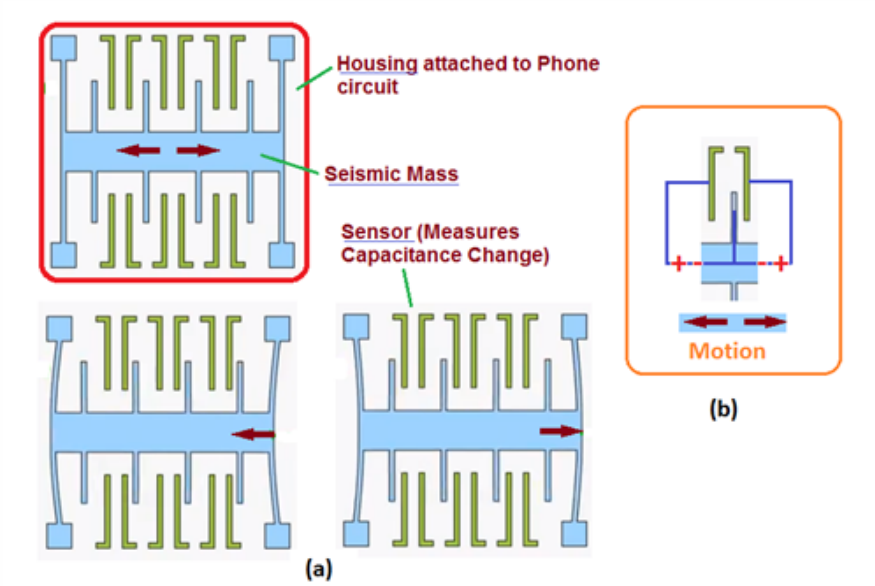


Figure 2.1: Subfigure (a) shows the mechanical principle behind the MEMS accelerometer, having a fixed base corresponding to sensors that measure changes in capacitance. Subfigure (b) shows a change in capacitance as a result of the change in position of the Seismic Mass, when the device is tilted or there is a change in orientation.

The incorporation of this in mobile phones has allowed the development of complex systems, such as the estimation of the number of steps a user has taken in a given time lapse, which is the main use of it in the current work.

2.1.2. Gyroscope

Gyros correspond to devices mounted on a frame, capable of detecting angular velocity on a certain axis, in case that such a frame in which it is located is rotating [4]. There are multiple types of gyros, which vary depending on the physical principle of operation, and the technology involved in its operation. In the MEMS case, the main way to detect angular rotations corresponds to the use of the Coriolis effect in vibrating masses. Thus, these sensors detect the force acting on a mass, which is subject to a linear movement on a fixed reference axis, and it is rotating in an axis perpendicular to the axis of the linear movement already mentioned. The resulting Coriolis acts in a direction that is perpendicular to these two axes. Figure 2.2 shows the mechanical structure of a device based on this physical principle.

These devices can be used both autonomously or included in more sophisticated systems, such as a gyro-compass, inertial measurement unit (IMU), systems of inertial navigation, and reference

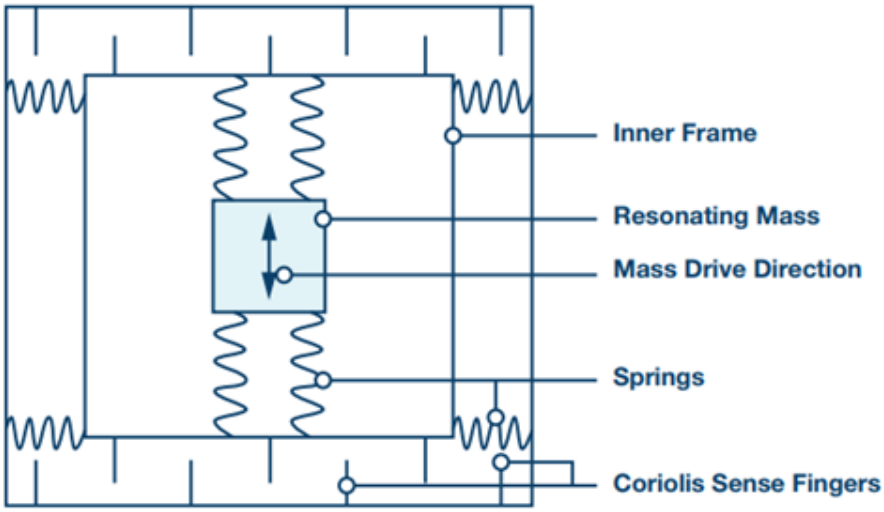


Figure 2.2: Schematic of the mechanical structure of a MEMS gyroscope.

systems for pose estimation. In particular, the Gyroscopes used in MEMS correspond to motion sensors that detect the angular variation of an object in 1, 2, or even 3 axes. This is mainly used to detect changes in the orientation of the device, and also, its user.

2.1.3. Magnetometer

As the name indicates, the magnetometer is intended to measure the adjacent magnetic field to the sensor. Within these, there are five main methods on which the calculation is made for the measurement of the magnetic field. These include the use of Hall effect, giant magnetoresistance (GMR), magnetic tunnel effect sensing (MTJ), anisotropic magnetoresistance (AMR), and the Lorentz Force.

The Lorentz force corresponds to the force applied to a point charge due to both the electric force near it, and the magnetic field when the particle is in movement with respect to it, which is represented in Equation 2.1

$$\mathbf{F} = q\mathbf{E} + q\mathbf{v} \times \mathbf{B} \tag{2.1}$$

where q is the charge of the particle, $\mathbf{E}$ is the adjacent electric field, $\mathbf{v}$ is the speed of the particle with respect to the magnetic field $\mathbf{B}$, and $\mathbf{F}$ is the force applied to the particle.

Most of the magnetometers make use of the Lorentz force as a way to detect the magnetic field, detecting the movement of a small magnetic bar in it. Although the cost of these sensors has allowed its mass production and deployment into mobile devices, it's essential to realize that the resolution of these components is limited by inner structure of the mobile device [5]. In Figure 2.3, you can see both the schematic and a microscopic view of a magnetometer that works on this principle.

It is crucial to realize that magnetometer performance is highly dependent on the environment in which it is located. This is because in the case of using the device inside a building, the magnetic disturbances product of the metallic structure add a bias to the final measurements. Similarly,

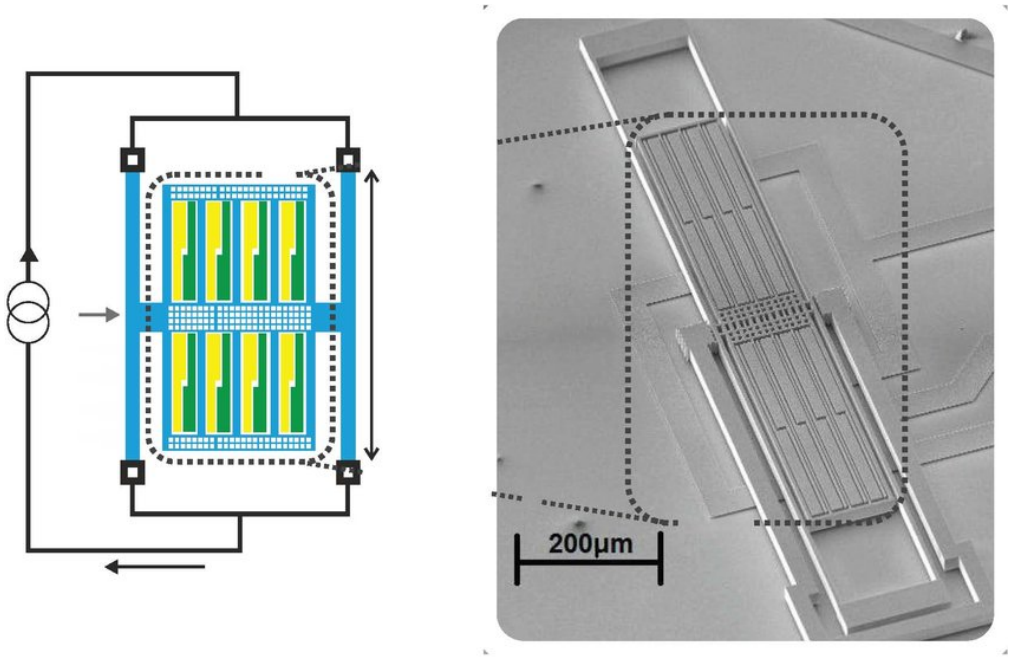


Figure 2.3: Schematic representation of a MEMS magnetometer which acting principle is the Lorentz force, showing also a microscopic view of the implemented device.

this also occurs near devices such as microwaves, or high voltage cables, because the time-varying electric fields induce magnetic fields in the sensor.

The main purpose of the magnetometer in the Pedestrian Dead Reckoning context, is to have a reference point to which we can associate a global coordinate, which corresponds to the geographic and magnetic north. In the case where we did not have this reference point, all the rotations that a person makes, would be with respect to the same sensor, but we would still need an initial guess of the direction to which the person is walking to determine the final orientation.

2.2. Sensor noise analysis

The information provided by sensor manufacturing companies, such as BOSCH, Sensortek, or QTI, is mostly hidden. While this allows these microelectronic device companies to maintain the privacy of their product pipelines, inconveniences are generated when you want to work directly with their sensors. An example of this occurs at the time of modeling the sensors, given that in some cases, it is not well known the types of noise they have, the respective bias, and their accuracies. This is the reason of why it is necessary to analyze each of the sensors in the PDR context, determining the best way to remove unwanted noise from them.

2.2.1. Noise characteristics of inertial sensors

Within the possible types of noise that affect the output of the inertial sensors found on smartphone devices, some are more predominant over others. The types of errors, and how they affect each one of the sensors, are presented below.

2.2.1.1. Constant Bias

The bias of a sensor is the average output of a device, measured over a certain period of time. This must be considered in operating conditions which do not have correlation with sensor rotation (gyroscope), input acceleration (accelerometer), or induced magnetic field (magnetometer). Bias is typically measured in degrees by hour ($^\circ/h$) or radians per second (rad/s) in the case of gyros, in meters per second square (m/s^2) for accelerometers, and microtesla (μT) for magnetometers. The bias consists of two parts: a deterministic part called offset, and a random part. In the case of the deterministic value, the offset bias, corresponds to a constant value which is added to the measurement, and it can be determined eliminated by calibration [6]. On the other hand, the random value is a stochastic process, and refers to the rate at which the error in a sensor inertial accumulates over time.

2.2.1.2. Bias instability

Additionally, there are two features used to describe the bias of a sensor: the asymmetry of the bias, which corresponds to the difference between the bias for positive and negative inputs, and the instability of the bias, which is the random variation in the bias calculated in finite time.

The power spectral density (PSD) rate associated with bias instability,can be seen in Equation 2.2

$$S_\Omega(f) = \left\{ \begin{array}{ll} \left(\frac{B^2}{2\pi}\right)\frac{1}{f}, & f \leq f_0 \\ 0, & f > f_0 \end{array} \right\} \tag{2.2}$$

In this equation, B corresponds to the instability coefficient of the bias, f is the frequency at which the power spectral density is being measured, Ω is the output data of a given sensor, and f_0 is the cutoff frequency, which is the value at which the signal has been attenuated by $3dB$.

2.2.1.3. Random Walk

The output of the sensors tends to be disturbed by thermo-mechanical noise, which fluctuates at a rate greater than that of the sampling rate of the sensor itself. The random walk of a particular sensor, is a noise specification given in units of $^\circ/\sqrt{h}$ for gyroscopes, $m/s/\sqrt{h}$ for accelerometers, and $\mu T \cdot s/\sqrt{h}$ for magnetometers. As a consequence, the obtained samples seem affected by white noise, which is the product of uncorrelated random variables, with a zero mean value. In other words, the random walk describes the average deviation of the error that occurs at the moment that the signal is integrated in time.

For example, in the case of a gyroscope, an angular random walk of $5^\circ/\sqrt{h}$ means that after an hour, the angle deviation corresponds to 5°, and after 2 hours, it's $5\sqrt{2}^\circ \approx 7.07^\circ$.

The PSD for a random walk, can be represented in Equation 2.3, where Q is the random walk coefficient, expressed in $^\circ/h/\sqrt{Hz}$, describing the noise output as a function of bandwidth.

$$S_\Omega(f) = Q^2 \tag{2.3}$$

2.2.1.4. Quantization noise

This noise is introduced into an analog signal as a result of the conversion to a digital signal. It is generated due to the difference between the actual values of the signal analog, and the resolution of the analog-digital converter.

The Equation 2.4 shows the PSD for quantization noise, being Q its main coefficient.

$$S_{\Omega}(f) = \frac{4Q^2}{T}\sin(\pi f T) \tag{2.4}$$

2.2.1.5. Random walk rate

Long-term bias offset changes tend to be distributed randomly, and even become permanent. Although the deviation of a particular sensor respect of the actual value that should be measured cannot be predicted, the time scale over which these changes occur can be defined from the random walk rate, and introduces the need to recalibrate in applications which require an extended life time. The rate of the PSD associated with this noise [7] corresponds to that described in Equation2.5, where K is the coefficient of the random walk rate.

$$S_{\Omega}(f) = \frac{K^2}{2\pi f^2} \tag{2.5}$$

2.2.1.6. Ramp rate

This error is deterministic, corresponding to the monotonous change of the output over a long period of time. Equation 2.6 represents its PSD, with R the ramp rate coefficient.

$$S_{\Omega}(f) = \frac{R^2}{(2\pi f)^3} \tag{2.6}$$

2.2.1.7. Sinusoidal noise

This noise is characterized by having n different fundamental frequencies $f_i\ i \in 1,...n$. Equation 2.7 shows the PSD, being Ω_i the amplitude, and $\delta(x)$ the Dirac delta function.

$$S_{\Omega}(f) = \frac{1}{2}\sum_{i=1}^{n} \Omega_i^2[\delta(f-f_i) + \delta(f+f_i)] \tag{2.7}$$

2.2.2. Allan variance

In order to be able to characterize the types of errors and noise in inertial sensors such as the accelerometer, gyroscope, and magnetometer, the most used techniques correspond to the use of the PSD, and the analysis with Allan variance [8].

This procedure originally corresponds to a method of analyzing a sequence of data in the time domain, in order to measure the frequency stability in oscillators. In just 2003, it was used for the first time in a MEMS context. [9].

The following steps describe how to create a chart that represents the Allan variance of a sensor [10]. In particular, the overlapping method is indicated to calculate the graphic for noise analysis. The following steps show how to do this:

1. The output data $\Omega(t)$ of a given sensor is obtained. It is necessary to keep this device in a stable environment and at constant room temperature. To characterize the model, it is considered that the number of samples are N, and the sampling period is τ_0.

2. The average time is adjusted to $\tau = m\tau_0$, with m being the factor for the average. This value can be chosen arbitrarily, as long as it is met that $m < (N-1)/2$.

3. The temporal signal is divided into finite duration data sets $t = m\tau_0$. This part of the process is show in Figure 2.4.

4. Once generated the sets of data, the variance is calculated from the average of each sampling rate outputs (over each data set).

5. The value of the deviation is calculated for a particular value τ, repeating the process for multiple τ. Having done this, the graph is done, plotting the deviation as a function of τ.

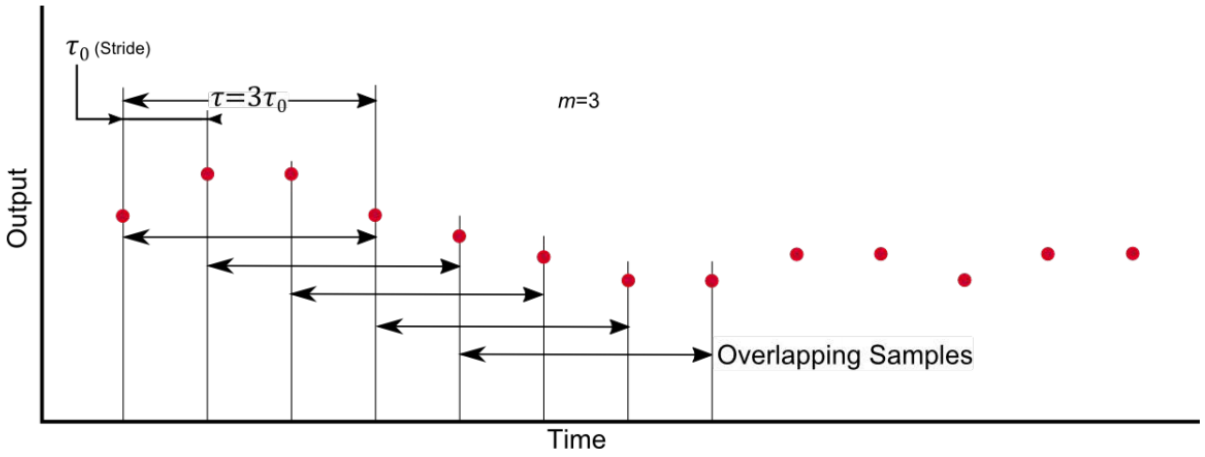


Figure 2.4: Representation of the data overlapping, which is later used for the analysis of the Allan Variance

2.2.3. Noise identification

Depending on the type of random processes that are generated in the sensor to be analyzed, it is possible to identify different slopes depending on the areas in the Allan variance chart. Among the most important noises to take into consideration, are the following:

- White noise/*Random walk*: appears on the graph with a slope of -0.5. The measure of random walk is obtained by drawing a line through the slope, and reading its value at $\tau = 1$.

- Bias instability: Corresponds to the minimum in the graph, where the slope is equal to 0. This value indicates how much the sensor bias changes during a given amount of time, at constant temperature.

In a more general way, in Figure 2.5 the different types of noise can be seen as a function of the slope in the graph

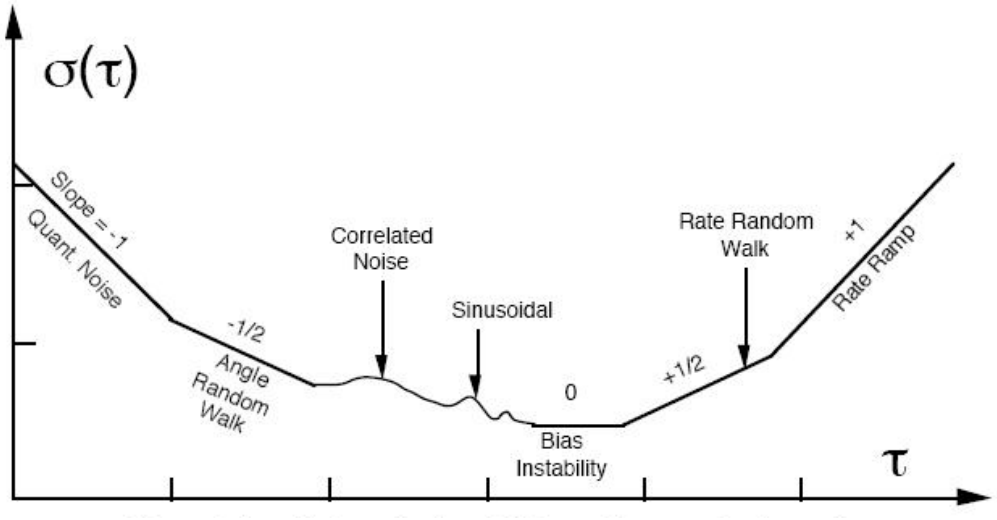


Figure 2.5: Types of noise in sensors for a log-log graph, using Allan analysis of the variance.

2.3. Coordinate system, and use of quaternions

When delivering sensor data, smartphone devices tend to deliver them with respect to the local coordinate system. This is shown in Figure 2.6, where the Z axis of the telephone is not the same as the terrestrial Z axis (indicated by the vector of gravity). That is why it is necessary to carry out a transformation of coordinates between both reference systems.

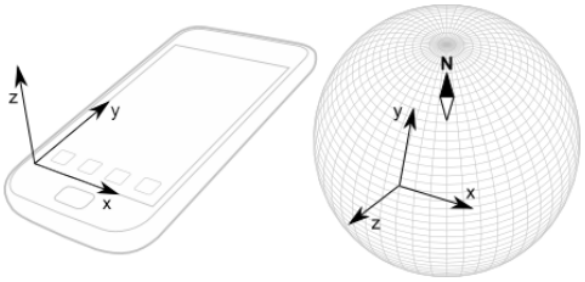


Figure 2.6: Representation of the local and terrestrial coordinate system for a smartphone device

In order to show how to change between one system and another, it is possible to use Euler angles, rotation vectors, or quaternions. This last option tends to be the better, since compared to Euler's angles, they avoid the problem of the Gimbal Lock effect. This effect corresponds to the loss of a degree of freedom in a mechanism which originally has three degrees of freedom, which happens when two of the axis of the mechanism allign in a parallel way. Besides from the fact

that quaternions solve the gimbal effect problem, quaternions are more compact, and more efficient computationally when being compared with rotation matrices.

Given a vector in a Euclidean space (a_x, a_y, a_z), this can be described as $a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}$, being $\mathbf{i}, \mathbf{j}, \mathbf{k}$ the unitary vectors which represent the Cartesian plane. Thus, a rotation of an angle θ along the unitary vector axis (u_x, u_y, u_z) may be described through the quaternion shown in Equation 2.8.

$$\mathbf{q} = e^{\frac{\theta}{2}(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\frac{\theta}{2} + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2} \tag{2.8}$$

Therefore, it can be shown that the rotation applied to the vector $\mathbf{p} = (p_x, p_y, p_z) = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$ from the quaternion $\mathbf{q}$ which was previously mentioned, is given by the Equation 2.9, with $\mathbf{p}'$ the resulting quaternion, and $\mathbf{q}^{-1}$ defined as the multiplicative inverse of the quaternion $\mathbf{q}$, defined in Equation 2.10.

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} \tag{2.9}$$

$$\mathbf{q}^{-1} = e^{-\frac{\theta}{2}(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\frac{\theta}{2} - (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\frac{\theta}{2} \tag{2.10}$$

It is necessary to mention that in order to perform a rotation with the use of quaternions, the quaternion vector associated to the rotation to be performed ($\mathbf{q}$ in the case of Equation 2.9) must be a unit vector, so that when performing multiplications, the norm of the new rotated vector remains constant.

Finally, the multiplication between a quaternion $\mathbf{q} = (q_0, q_1, q_2, q_3)$ and $\mathbf{p} = (p_0, p_1, p_2, p_3)$, is defined as shown in Equation 2.11.

$$\begin{aligned}
\mathbf{q}\cdot\mathbf{p} &= \begin{bmatrix} q_0p_0 - \mathbf{q}_{1:3}^T\mathbf{p}_{1:3} \\ q_0\mathbf{p}_{1:3} + p_0\mathbf{q}_{1:3} - \mathbf{q}_{1:3}\times\mathbf{p}_{1:3} \end{bmatrix} \\
&= \begin{bmatrix} q_0 & -\mathbf{q}_{1:3}^T \\ \mathbf{q}_{1:3} & q_0I_3 - C(\mathbf{q}_{1:3}) \end{bmatrix} \begin{bmatrix} p_0 \\ \mathbf{p}_{1:3} \end{bmatrix} \\
&= \begin{bmatrix} p_0 & -\mathbf{p}_{1:3}^T \\ \mathbf{p}_{1:3} & p_0I_3 - C(\mathbf{p}_{1:3}) \end{bmatrix} \begin{bmatrix} q_0 \\ \mathbf{q}_{1:3} \end{bmatrix}
\end{aligned} \tag{2.11}$$

Where the matrix cross product $C : \mathbb{R}^3 \longrightarrow \mathbb{R}^{3x3}$ is defined by the Equation 2.12.

$$C(\mathbf{p}) = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix} \tag{2.12}$$

2.4. State estimators

When performing the process of Pedestrian Dead Reckoning, we make use of the sensors in the smartphone device to perform the sub processes associated to PDR (step detection, orientation estimation, and step length estimation). Although it may seem like the sensors output are enough to carry out each of these sub processes, sometimes, the variable of interest (for example the ori-

entation of the person) cannot be measured directly, but has to be estimated indirectly by using state estimators. In most cases where state estimators are used in the PDR context, the state to be estimated corresponds to the orientation of the person, which is the case for the present work.

2.4.1. Kalman Filter

Assuming that you want to know the value of a variable in a process of the form of Equation 2.13, where $\mathbf{x}_k$ corresponds to the state vector at time k; $\mathbf{F}$ is the transition state matrix, assumed to be constant over time, and $\mathbf{w}_k$ is the process noise that comes from a multinormal distribution with known covariance $\mathbf{Q}_k : \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$. The latter corresponds to the non-modeled processes.

In general, $\mathbf{B}_k$ is the matrix that relates the input process $\mathbf{u}_k$ with the state variable, not being necessary to consider this for the purposes of the work in question, since there will be no input variable.

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \tag{2.13}$$

At time k, an observation $\mathbf{z}_k$ is associated with the state $\mathbf{x}_k$ through Equation 2.14, where $\mathbf{H}$ is the observation matrix, which maps the state space to the observed space. Similarly, $\mathbf{v}_k$ is the observation noise which is assumed to come from a multinormal distribution with zero mean, and covariance $\mathbf{R}_k : \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$. In general, this noise is associated with the errors of the sensors, since these have a certain accuracy.

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{2.14}$$

From the above, it is desired to develop an optimal filter according to the mean squared error (MSE), i.e., it is desired to make an estimate $\hat{\mathbf{x}}_k$ of the real value $\mathbf{x}_k$, from the minimization of the expected value of the mean square error. The function to be minimized is represented in Equation 2.15.

$$\begin{aligned} f(\mathbf{e}_k) &= f(\mathbf{x}_k - \hat{\mathbf{x}}_k) \\ f(\mathbf{e}_k) &= (\mathbf{x}_k - \hat{\mathbf{x}}_k)^2 \end{aligned} \tag{2.15}$$

Having defined the cost function, it is desired to obtain the value for $\mathbf{x}_k$ that minimizes the expected value of the error, i.e. $\mathrm{E}(f(\mathbf{x}_k - \hat{\mathbf{x}}_k))$, which will correspond to the estimate $\hat{\mathbf{x}}_k$ already mentioned. This is the main idea behind the linear Kalman Filter.

From that formulation, it is possible to arrive at a recursive formulation, which means that only the estimation of the previous state, and the current observation are needed to know the state estimate at the current timestamp.

The state of the filter tends to be represented by the following variables:

- $\hat{\mathbf{x}}_{k|k}$: the posteriori estimated state at time k
- $\mathbf{P}_{k|k}$: the covariance matrix of the state error. At time k, this is defined by $\mathbf{P}_k = E[\mathbf{e}_k\mathbf{e}_k^T]$

Thus, from the development of the minimization of the MSE, a process that can be separated into 2 phases is done: Predict and Update.

The prediction stage corresponds to the incorporation of the information of the state transition model, while the update includes the information of the measurements made. Thus, the next iterative cycle starts:

Prediction:

1. Estimation of the a priori state: $\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1}$
2. A priori covariance of the state error: $\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{Q}_k$

Update:

1. Innovation computation: $\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}$
2. Innovation covariance: $\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k$
3. Optimal Kalman Gain: $\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1}$
4. A posteriori estimate of the state: $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
5. A posteriori covariance of the state error: $\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$

It is important to note that the Kalman gain represents how much the state and the observations should be weighted. If the covariance of the state error (for a particular $\mathbf{x}_i$ state) is less than the covariance of the innovation (for the same state $\mathbf{x}_i$), it means that at the time of updating the state $\mathbf{x}_i$, the previous state is trusted more than the observations.

2.4.1.1. Extended Kalman Filter

In many cases, the system is non-linear, but both the state transition and the observations are determined from a system of equations as in the Equation 2.16, where the functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are not necessarily linear.

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \\ \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{2.16}$$

It is possible to return to the formulation of the linear Kalman Filter from a linearization of such functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, doing this via Equation 2.17.

$$\begin{aligned} \mathbf{F}_k &= \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k} \\ \mathbf{H}_k &= \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k|k-1}} \end{aligned} \tag{2.17}$$

Thus, the only difference in the processes of Predict and Update with respect to the linear Kalman Filter, is that the prediction of the estimated state is made from the nonlinear function $\mathbf{f}(\cdot)$, and the calculation of innovation comes from: $\tilde{\mathbf{y}}k = \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$.

Now, although the same formulation is reached with the calculation of the Jacobian, the extended Kalman filter does not correspond to an optimal estimate, being possible that the filter differs in the case that the initial estimate is in an unstable area, or if the process is modeled incorrectly.

2.5. Data processing

2.5.1. Signal filters

In the context of signal processing, the filters correspond to devices, or processes, which are used to treat an incoming signal. Usually, filters tend to remove unwanted components from a signal, such as the case of random noise, or to extract elements from it, such as its fundamental frequency. In the case of Pedestrian Dead Reckoning, this is used to extract the main signal of each of the sensors of the smartphone, from the signal corrupted with unwanted noise.

Depending on the application to be used, the generated filters can be separated into the following categories:

- Analog or digital.
- Causal or non-causal.
- In discrete-time, or in continuous time.
- Finite impulse response (FIR), or infinite impulse response (IIR).
- Time-invariant, or time-variant.
- Linear, or non-linear.

An analog filter makes use of electronic components such as resistors, capacitors, or operational amplifiers, to generate the desired effect. These types of filters can be used for noise reduction, video image enhancement, among many other areas.

On the other hand, digital filters use digital processors so that they can perform numerical calculations on a signal with discrete values. These processors can correspond to multipurpose computers, or a specialized Digital Signal Processor (DSP).

2.5.1.1. Types of digital filters

From now on, a single x signal will be taken into consideration for various examples ,which is defined by Equation 2.18. It takes values from a discrete space, and they reach the real number space.

$$x : \mathbb{Z} \rightarrow \mathbb{R} \quad (2.18)$$

Now that we have the idea behind a filter, its mathematical representation can be given from the Equation 2.19. The variable $y(n)$ is the output of the filter at time n, and the function $\mathscr{T}_n\{x\}$ corresponds to the filter itself.

$$y(n) = \mathscr{T}_n\{x(\cdot)\} \tag{2.19}$$

While any form of mapping from a signal to real numbers can be considered a filter, we usually tend to work with filters which have certain types of predefined structures. Within these, the Linear Time-Invariant (LTI) filters are especially useful, since they can be used to perform analyses in the frequency spectrum.

As the name implies, a filter of the type Linear Time-Invariant is both linear and invariant in time. The definitions of each of these two criteria are shown below.

Definition 2.1 (Linear Filters) A filter $\mathscr{T}_n$ is said to be linear if for any pair of signals $x_1(\cdot), x_2(\cdot)$ and for any constant gain g, the Equations 2.20 are fulfilled for any instant of time $n \in \mathbf{Z}$:

$$\text{Scaling: } \mathscr{T}_n\{g\,x(\cdot)\} = g\,\mathscr{T}_n\{x(\cdot)\}, \quad \forall g \in \mathbf{C},\ \forall x \in \mathscr{Z} \tag{2.20}$$

$$\text{Superposition: } \mathscr{T}_n\{x_1(\cdot) + x_2(\cdot)\} = \mathscr{T}_n\{x_1(\cdot)\} + \mathscr{T}_n\{x_2(\cdot)\}\ \forall x_1, x_2 \in \mathscr{Z}, \tag{2.21}$$

Definition 2.2 (Time-invariant filters) A filter $\mathscr{T}_n$ is said to be time-invariant, if for each input signal x, with the shift operator defined by $Shift_N\{x(n)\} = x(nN)$, the equation is fulfilled 2.22

$$\mathscr{T}_n\{Shift_N\{x\}\} = Shift_N\{\mathscr{T}_n\{x\}\} = Shift_N\{y\} \tag{2.22}$$

With the concept of LTI filters already delivered, it is necessary to mention the idea of causality in the filters. The definition 2.3 shows the idea of this from a rational point of view. This type of filter makes sense to be used when making applications in real-time. But this is not a must since the filtering of a signal can be done with the values which were obtained in previous time steps, delaying the signal in a fixed manner.

Definition 2.3 (Causal filter) A filter is said to be causal if its output does not depend on any future input.

2.5.1.2. Filter frequency analysis

As already mentioned, LTI filters are useful since it is possible to perform analyzes in their frequency spectrum.

Since any signal with a discrete sampling time can be represented in the frequency domain, there is the possibility of attenuating or amplifying certain frequencies associated with it through filters. Depending on the frequency components to be attenuated (or maintained in the same range), filters can be separated into the following categories:

- Low pass filter: Low frequencies of the incoming signal stay nearly the same, while high frequencies are attenuated.

- Filtro pasa alto: Las frecuencias altas de la señal entrante pasan, mientras que las frecuencias bajas son atenuadas.
- High pass filter: High frequencies of the incoming signal stay nearly the same, while low frequencies are attenuated.
- Filter rejects band: Only frequencies within a certain range of frequencies are attenuated.

Generally, when making a filter, it is desired to generate a filter as ideal as possible, i.e., that allows cutting the desired frequencies at a specific point. Although, in theory, this may be possible in some cases, when analyzing the transfer function of such filters, it is possible to realize that an infinite amount of CPU is required to generate such filters. That is why, in many cases, filters are generated, which best approximate the ideal filters, meeting certain criteria to maximize.

Examples of these filters just mentioned can be seen below, with Figure 2.7 showing visual examples of these filters in the normalized frequency spectrum:

- Butterworth filter: It has a maximum flat frequency response.
- Chebyshev filter: It has the best approximation to the ideal response for any specific degree.
- Bessel filter: It has a maximum flat phase delay.
- Elliptical Filter: It has the steepest cut point than any other filter for a specific order and ripple.

2.5.2. Interpolation

The process of deriving a function from a discrete set of points is called interpolation. This is done so that other intermediate points in the signal which do not correspond to the original signal may be estimated.

2.5.2.1. Linear interpolation

In linear interpolation, each estimated point is assumed to lie on the straight line that joins the points closest to the point to be estimated. Thus, let two points be $p_0 = (x_0, y_0)$, $p_1 = (x_1, y_1)$ defined in $\mathbb{R}^2$, and with $x_1 > x_0$, the line defined between these two points are given by the equation 2.23.

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \tag{2.23}$$

2.5.2.2. Spline interpolation

This type of interpolation corresponds to the process of obtaining a defined by parts polynomial, which is called spline. Without loss of generality, it is assumed that we have $n+1$ points $x_0, x_1, ...x_n$, which meet that $x_0 < x_1 < ... < x_n$. Thus, a k degree spline function corresponds to a S function which meets the following restrictions:

- For each interval $[x_{i-1}, x_i]$, S is a polynomial of degree $\leq K$.

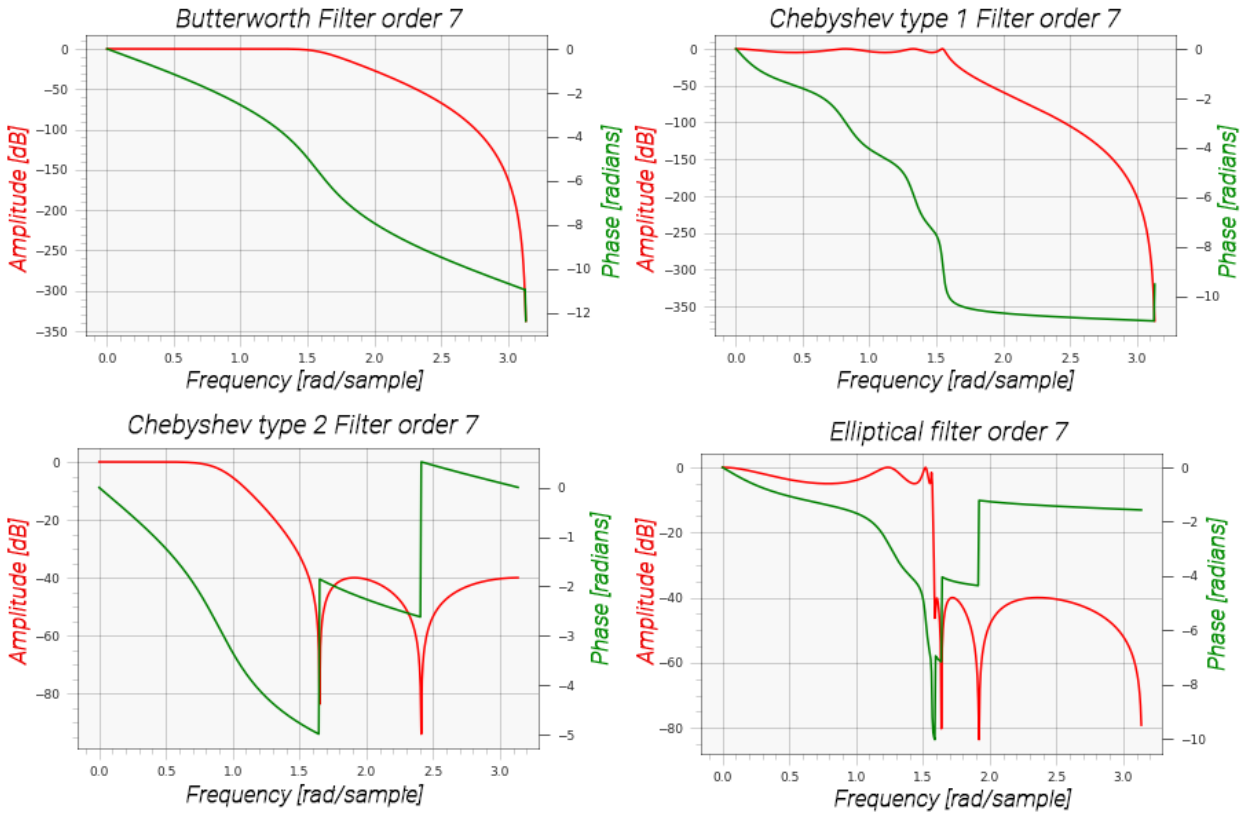


Figure 2.7: LTI Filter Examples

- The S function has a derivative of degree (k-1), which is continuous in $[x_0, x_n]$.

An example of interpolation by spline can be seen in Figure 2.8, with splines of grade 2, and 3.

2.6. Machine Learning and regression

In order to describe processes or phenomena from a mathematical point of view, studies tend to be conducted on their behavior in an environment of certain conditions, seeking to analyze the different effects on the functionality of these systems. Carrying out such studies allows us to generate complex models that are adjusted to the data used. Such models are normally dependent on one or more intrinsic system properties. Examples of this can be seen in the variable α of the Equation 2.24, which describes the heat equation in a medium of temperature $u(x,y,z,t)$, representing the heat flux in a medium with diffusivity of value α. In certain cases, finding the exact relationship that relates the independent variables of the dependents can become an arduous task, product of the number of input values of the system, or the randomness of the system.

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \tag{2.24}$$

To overcome the problem that was just mentioned, Machine Learning (ML) techniques have been developed, which allow a system to learn to act in a certain way, without being explicitly programmed. Moreover, given a set of data, ML algorithms use groups of rules to detect patterns

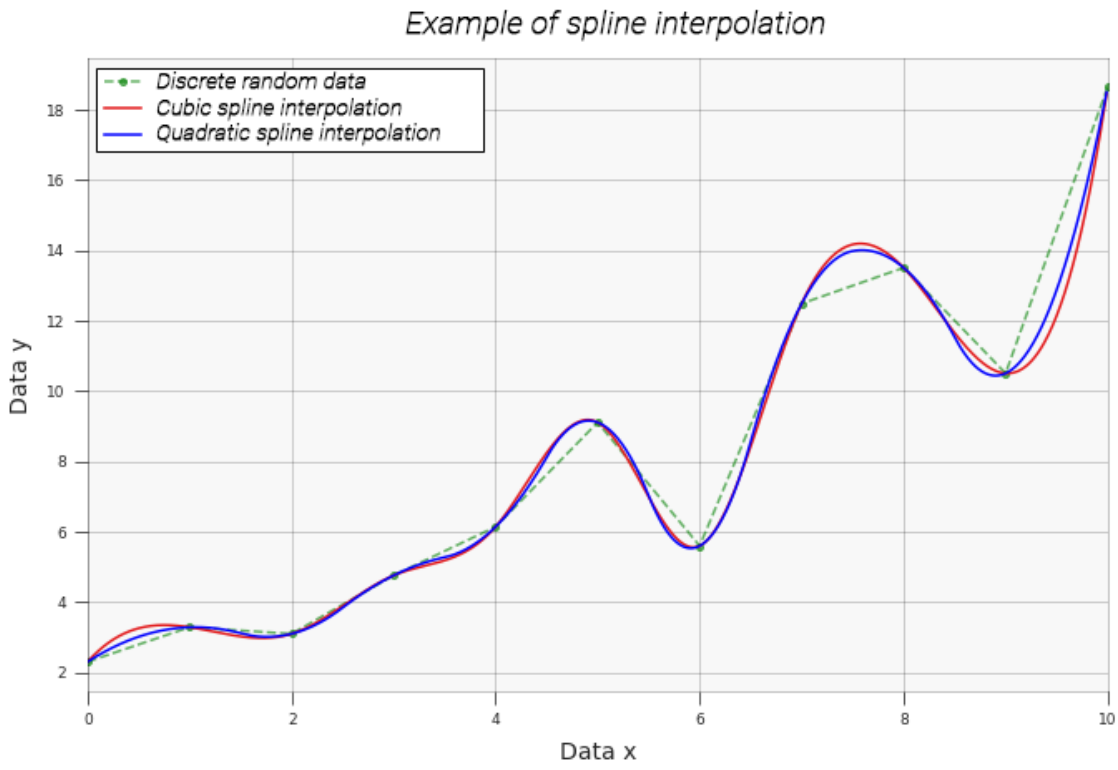


Figure 2.8: Example of spline interpolation

in the data, which are then used to make decision-making processes. The advantage of these ML techniques is that they are able to automatically learn the data, without having to explain the rules of the system on which they are acting, as would be the case in Equation 2.24. This makes such algorithms ideal for problems which are too complex for classical approaches.

Although there is a very large number of ML algorithms, they tend to be separated into different groups, depending both on the type of data that is delivered to the algorithm during the training process, and the purpose for which it is being used. The first of these criteria separate ML mainly in the supervised and unsupervised training branches, separating supervised training into classification and regression problems.

In the context of Pedestrian Dead Reckoning, Machine Learning is used as a mean to create an estimate from labeled data. In the present work, the value to be estimated corresponds to the length of the step of the smartphone's user at each timestamp.

2.6.1. Regression problem

A regression algorithm is trained in order to estimate a continuous variable $\boldsymbol{y} = f(\boldsymbol{x})$, with $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{R}^m$. In other words, we want to find a function f which is capable of mapping an independent variable $\boldsymbol{x}$ to an estimate $\boldsymbol{y}$. The criteria of how good the results are, depends on the person who develops the algorithm, often giving the minimization (or maximization) of a particular functional. An example of this type of model can be seen in Figure 2.9, which shows a polynomial regression of order 3, and a linear one.

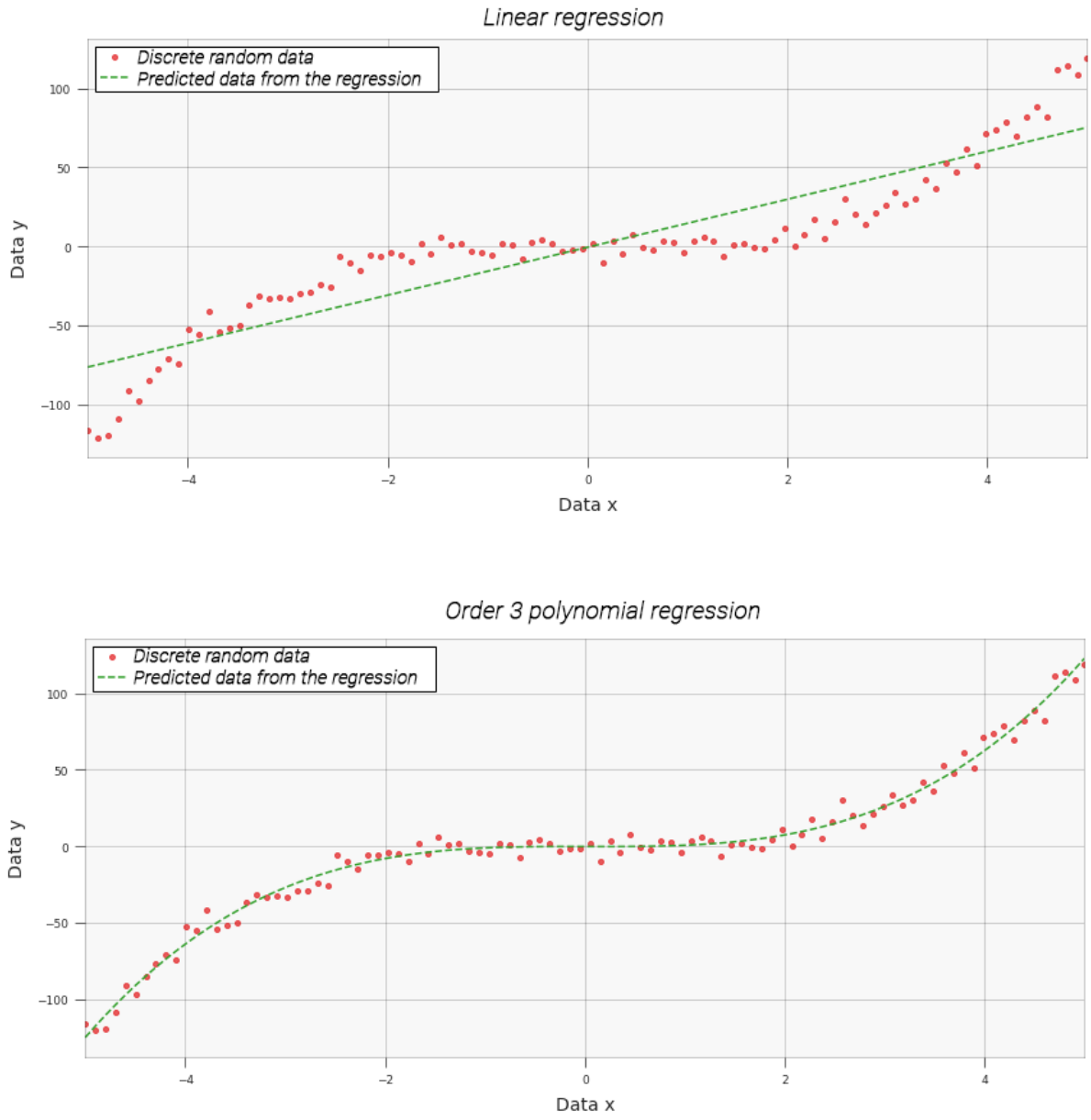


Figure 2.9: Regression example for randomly sampled data within a defined range

2.6.2. Artificial Neural Networks

Artificial neural networks (ANN) correspond to a non-linear mapping system, which attempts to simulate the structure and functional aspect of a neuron.

The idea is that when generating groups of processing units, united in proper ways, they are able to generate and model behaviors that are much more complex than that of their on their own.

In neuroscience, a neural network describes a collection of physically connected neurons, whose inputs define a particular circuit. Thus, a neuron is the basic processing unit of the central nervous system, with communication between several neurons involving electrochemical pro-

cesses.

2.6.2.1. Perceptron and multilayer perceptron

It is desired to have a mathematical model which correctly represents the biological functions of neurons. In order to do this, the biological process is simplified, identifying three main parts of a neuron on which the model is based. These parts correspond to synapses, adders, and activation functions.

Each of the synapses is modeled as a weighted value. Thus, if there is a signal x_j at the synapse input j, which is connected to the neuron k, then w_{kj} corresponds to the weight by which x_j is multiplied. This weight w_{kj} represents the strength of a synapse between two neurons. Hence, negative weights reflect inhibitory connections, while positive weights designate values of excitatory connections. [11]

The next two components model the current activity in the cell body. As the name says, the adder is used to sum all the inputs x_j multiplied by their respective weights w_{kj} for a particular neuron k. Finally, an activation function σ_k is used to control the amplitude of the output of the neuron k. In Figure 2.10, it is possible to see the graphic representation of the concepts just mentioned in a neuron.

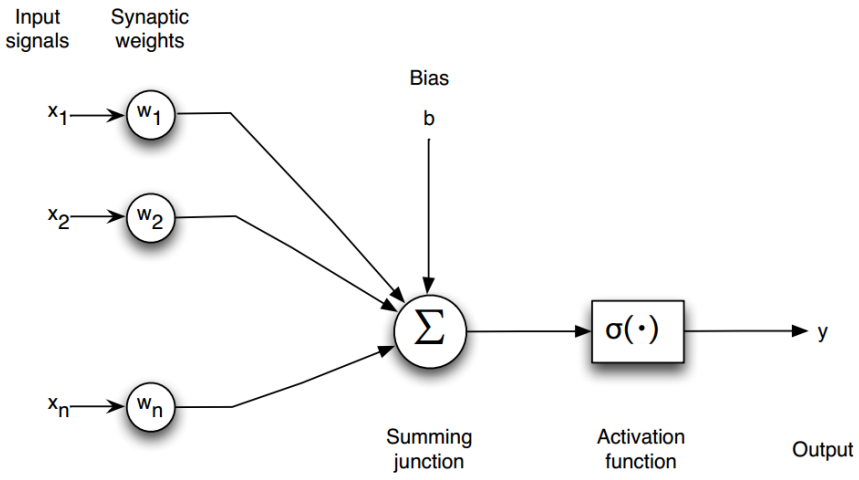


Figure 2.10: Graphical representation of a perceptron

An associated bias b_k term is also added, which is used to represent a limit for a certain neuron k. Thus, having a vector of entries $\boldsymbol{x} \in \mathbb{R}^n$, a vector of synaptic weights $\boldsymbol{w} \in \mathbb{R}^n$, a bias $b \in \mathbb{R}$, and the activation function σ, the formulation of the output of a neuron is given by the Equation 2.25.

$$y = \sigma\left(\sum_{j=1}^{n} w_j x_j + b\right) \tag{2.25}$$

This neuron modeling is known as a perceptron, which comes from the original ideas of the McCulloch-Pitts model, being similar to that of Frank Rosenblatt's in 1957.

2.6.2.2. Denoising autoencoder

An autoencoder corresponds to an unsupervised Machine Learning method, which reconstructs the network input at its output, without having to learn the irrelevant parts of the data.

As we can see in Figure 2.11, in the simplest way possible, an Autoencoder contains a single input layer, a hidden layer, and an output layer, which attempts to replicate the input.

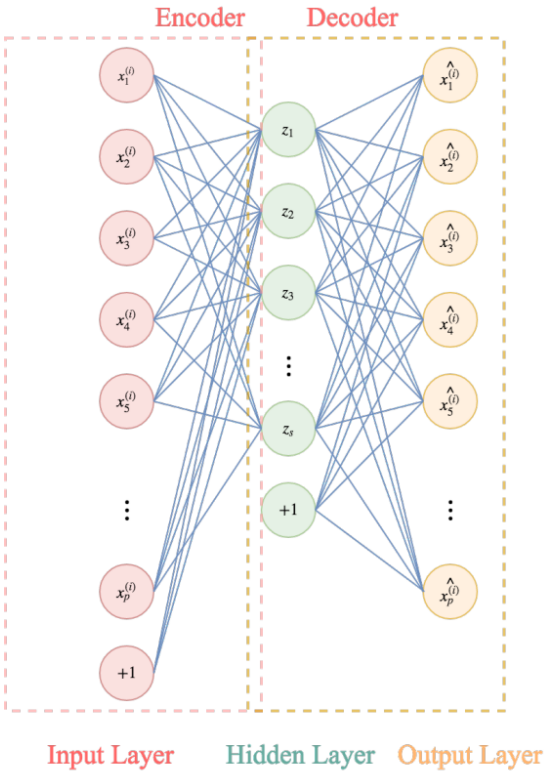


Figure 2.11: Graphical representation of an autoencoder.

An autoencoder takes an entry $x \in \mathbb{R}^d$ and performs an encoding process, bringing such data to a latent representation $z \in \mathbb{R}^{d'}$, with d' a subspace of d. The idea behind this is that the z data is able to capture the coordinates along with the main variation factors of the input data. This vector in the latent space is taken to the original representation of the d dimension using a *decoder*. For the simplest case, the two stages of an autoencoder can be represented in the Equation 2.26.

$$\begin{aligned} \boldsymbol{z} &= s(W\boldsymbol{x} + \boldsymbol{b}) \\ \boldsymbol{o} &= s(W'\boldsymbol{z} + \boldsymbol{b}') \end{aligned} \tag{2.26}$$

By increasing the number of hidden layers and using non-linear activation functions, the latent state representation is able to capture complex characteristics of the input values.

Already having the basic structure associated with neural networks, the topology of a feed-forward neural network is introduced. This consists of a set of perceptrons positioned in an array of layers, connecting the outputs of the neurons with the next layer, with no feedback in between them. An example of this can be seen in Figure 2.12, where the outputs of each of the neurons can affect more than one input in the next layer.

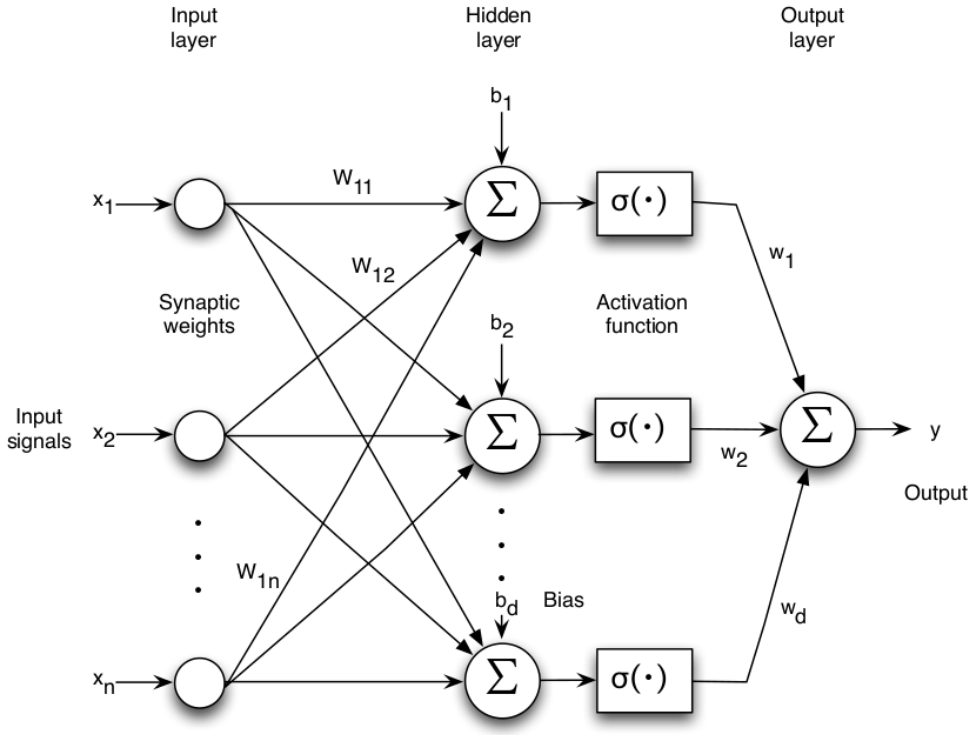


Figure 2.12: Artificial neural network with a feed-forward topology

Given this particular topology, it is possible to add more intermediate layers (hidden layers), as seen in Figure 2.12. Thus, the final output of the model corresponds to a composition of functions from the previous layers.

The Denoising Autoencoder is an extension of the Autoencoders. When corrupting the input data with noise (not necessarily Gaussian or additive), the network is forced to reconstruct the clean input values, causing the hidden layer to learn characteristics in a robust manner.

2.6.2.3. Deep Learning

Deep Learning (DL) corresponds to a branch of Machine Learning based on a statistical learning class, coming from the previous ideas of Artificial Neural Networks. Within this, the same basic logic of the neuron is used as the fundamental unit, to which a bias value and a respective weight are associated.

Regardless of the model to be used, it is necessary to pass input data so that it is mapped correctly according to the function associated with the model. In many cases where multilayer models are used, such as Multilayer Perceptrons, previously processed data is passed. This corresponds to the generation of a vector of characteristics associated with the input data, this set of characteristics being the input of the network. Here lays the main difference in the use of Deep Learning compared to other multilayer models: a system based on Deep Learning learns such a vector or set of features automatically during its training phase. Thus, it is the model that is responsible for extracting intermediate representations of the input data so that learning can be done in a better way.

Usually, the name Deep Learning comes from the use of an architecture where a network is used, which has more than one single layer, as can be seen in Figure 2.13, generating an impression that the model is in fact, deep.

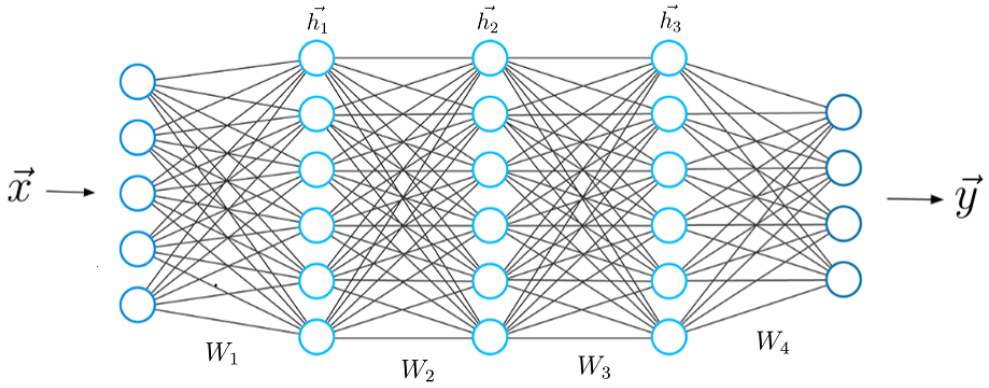


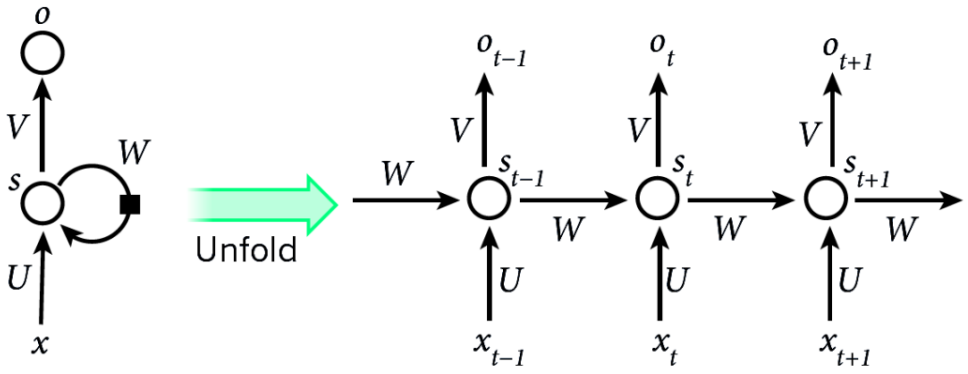
Figure 2.13: Graphic representation of a Deep Learning model corresponding to a Perceptron Multilayer with 3 hidden layers.

2.6.3. Recurrent Neural Networks

Recurrent neural networks (RNN) are an architecture associated with Deep Learning, generated for the purpose of handling inputs in successions. Thus, it is necessary to save in one way or another the state associated with the previous instant. This is done by keeping the information implicit in a node, which depending on the architecture of the model to be used, can have different forms. Examples of uses of this type of networks are the case of audio signals, or when words are being used, which tend to have a sequential order between them.

In Figure 2.14 it is possible to see the recursive structure of the nodes in a Recurrent Neural Network, which delivers implicit information itself in later moments. To the right of the figure it is possible to see how such a structure can be represented in a similar way to an architecture which has multiple nodes, receiving each of these, different inputs at the same time.

In the case of the shown figure, the neurons (for example, hidden units grouped under the s node with s_t values in instant t) get their inputs from other neurons in previous instants of time. In this way, the Recurrent Neural Networks are capable of generating a mapping of an input sequence with elements x_t to an output sequence with elements o_t. The same parameters corresponding to



.

Figure 2.14: General structure of a Recurrent Neural Network. Image obtained from [12]

the U, V, and W matrices are used for every instant of time.

The problem with the Recurrent Neural Networks as they have been presented so far (Fully Recurrent), is that they suffer from the short-term memory problem. This means that given a sufficiently long sequence, they will have problems carrying information from previous moments to later neurons.

During the training process, it is possible to perform back-propagation on the layers in order to obtain the gradient calculation, and thus train the network. Now, in the case of Fully Recurrent network architectures, they suffer from the vanishing gradient problem. This drawback is reflected in the fact that the weights of the networks are not updated, especially in the case of the first layers of the network. As these layers do not update their values, Fully Recurrent networks forget what they have seen in longer sequences, and therefore they are said to have a short memory.

That is why it is difficult for a Fully Recurrent network to store information and learn relationships between data, which are too far apart in time. Thus, a new type of architecture that would solve the aforementioned problems had to be generated.

Normally a recurring network performs concatenation between the entrance to its cell, and the hidden state of the previous cell so that when going through an activation function, the hidden state of the current cell is stored.

In the case of the Gated Recurrent Units (GRU) and Long-Short Term Memory (LSTM) architectures, these have a control flow similar to the aforementioned, processing the passage of data as it propagates forward. On the other hand, there are certain components which are different from the operations in the classic RNN. Such operations are those that allow such networks to maintain or forget information.

2.6.3.1. Long-Short Term Memory

In the case of LSTM networks, the key concept associated with these are the state of the cell, and its various gates. The state of the cell acts as the means of transport which sends the necessary information through the sequence of different cells. As the state of the cell passes between different cells, certain information is added or deleted to the state through the gates. Such gates learn during the training period what information is important and what is necessary to forget. A graphical representation of an LSTM cell can be seen in Figure 2.15.

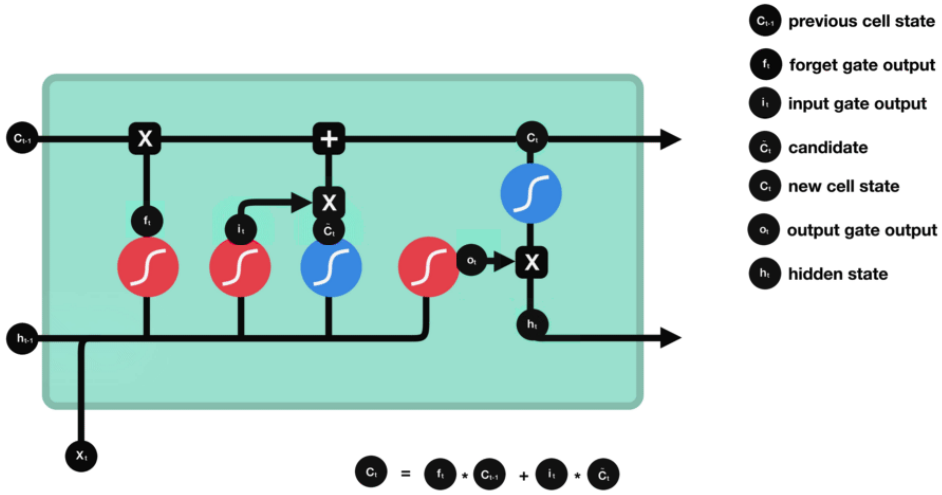


Figure 2.15: Graphical representation of an LSTM cell with the respective gates, and each of the subsections.

In the case of the forget gate, it decides what information associated with the input and the state of the previous cell should be maintained. The information of the previous hidden state of the cell and the information of the current input goes through a sigmoid activation function. Moreover, the output of the activation function is between 0 and 1. The closer the value is to 1 means that the information must be maintained, while the closer to 0 it is, means that such information must be forgotten.

Now, in order to update the state of the cell, the entry gate is used. In it, the previous state and the current input are passed through a sigmoid function, similar to the case of the forget gate. Similarly, the same concatenation between the previous state and the current input is passed through a hyperbolic tangent function, which allows the values between 0 and 1 to be regulated. The output of these two activation functions is then multiplied.

With the current knowledge, there is enough information to calculate the current state of the cell. First, a value-to-value multiplication is performed between the state of the previous cell and the output of the forget gate. The result of such multiplication is added to the candidate value, delivering the new status value of the cell.

Finally, the output gate decides what the next hidden state should be, this being the one that contains information about previous entries. To do this, multiplication is made between the output

of the output gate and the output of a hyperbolic tangent function of the current state of the cell. This completes the process of calculating the states of the cell.

2.6.3.2. Gated recurrent unit

In a similar fashion to that of LSTM networks, the Gated Recurrent Units, or GRU, eliminates the concept of the state of the cell and makes use of the hidden state to transmit information between cells. In addition, this has only two gates as can be seen in Figure 2.16.

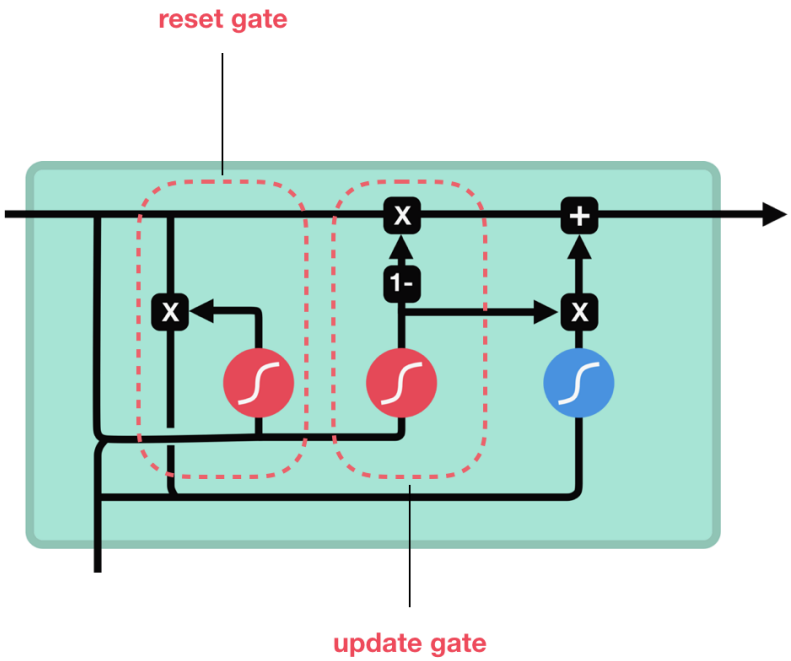


Figure 2.16: Graphical representation of a Gated Recurrent Unit, with the respective gates indicated.

In the case of the update gate, it acts in a similar way to what the forget-off gate does in the case of an LSTM, deciding what information is important, and which one should be forgotten. Similarly, the reset gate acts likewise, deciding how much information should be forgotten.

2.6.3.3. Bidirectional networks

In cases where a significant amount of available input information to the network is desired, and when it is desired that the input data is not fixed, it is useful to use a bidirectional network. As the name says, it corresponds to an architecture which separates a Recurrent Neural Network (it can be LSTM or GRU) in two directions, one for the flow of positive time, and another in the opposite direction.

Although the idea is easy to understand, there are some different ideas regarding usual RNN networks: In the first instance, when performing the back-propagation process, additional processes must be performed because updating the input and output layers cannot be done simultaneously. Thus, it passes through the positive flow cells, and then through the negative flow cells, so that the respective weights are subsequently updated.

2.6.4. Overfitting and regularization

In order to train different types of networks, it is necessary to define performance metrics so that the models are adjusted based on the result of such metrics. Examples of this are the loss functions, which vary depending on the context of the application that are used in. In the case of the regression problem, the average square error tends to be used as a measure of model performance, which is given by the Equation 2.27. In this case, Y is the vector of observed values of the predicted variable, and $\hat{Y}_i$ the vector of predicted values.

$$\text{Mean Squared Error} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.27}$$

While it is desired to have a mean square error equal to 0 (predicted values are equal to real values), this may not necessarily be a good thing. In some cases, the model, instead of learning the intrinsic characteristics behind the model (for example, a quadratic function), is learning unimportant features of the data, such as additive noise. Thus, when trying to predict with this trained model on a new data set, performance will be poor. This concept is known as overfitting.

In order to counteract the effect of overfitting, there are some techniques that are performed when training the model. Up next, different methods will be shown below to prevent this unwanted effect.

The first thing to do is to use a data validation set. This data set is intended to analyze the performance of the model, but not on the same data on which it was trained. Thus, with this data, it is possible to choose the best hyperparameters of the model (for example the architecture of a neural network), and at the same time, to be able to analyze what is the performance of this on different data.

A second method to reduce overfitting is the possibility of modifying the cost function. An example of this concept can be represented in the Equation 2.28, where J represents the loss function, adding a regularization term $R(f)$, which may correspond to a penalty on some particular parameters of the model.

$$J(f, \boldsymbol{x}, bmy) = \min_{f}\sum_{i=1}^{n} V(f(x_i), y_i) + \lambda R(f) \tag{2.28}$$

The third and final method for reducing overfitting to be mentioned, corresponds to the use of dropout, which is used in ANN networks. In this, the effects of certain neurons are randomly ignored during the training stage. This is intended to make the model more robust, since nodes within a layer are being forced to learn certain particular inputs.

It is important to mention that although there is a wide variety of regularization methods which depend on the context of use, the ones just mentioned correspond to the most usually used nowadays, especially when creating Deep Learning models.

Chapter 3

Methodology of development

In order to make a methodical work, the procedures to be carried out in the current thesis and the factors to be taken into consideration are indicated below.

3.1. Pedestrian Dead Reckoning

The Pedestrian Dead Reckoning (PDR from now on) process corresponds to a non-absolute, or relative coordinate navigation technique. As shown in Figure 3.1, from a position r_0 with an orientation θ_0, the successive changes of the position [13] are added. Thus, displacement estimates can be seen in the form of changes in the Cartesian coordinate system. Hence, for an instant of time k, it is possible to determine the new position r_{k+1} in space, from its previous position, orientation, and step length. The aforementioned is expressed in Equation 7, where the decomposition of the vectors r_k and r_{k+1} (corresponding to the step of the person in radial coordinates) into its components in the X and Y axes [14] is performed.

$$\begin{aligned} X_{k+1} &= X_k + \Delta L \cos\theta \\ Y_{K+1} &= Y_k + \Delta L \sin\theta \end{aligned} \tag{3.1}$$

In the previous equation, X_k and Y_k correspond to the projection of the vector r_k, into the X and Y axes respectively at time k. Also, ΔL_k is the length of the step of the person, and θ the angle of the given step with respect to the X axis.

In order to perform the process of PDR, it is necessary to perform three steps in parallel. Subsequently, these are integrated so as to be able to make the prediction of the position in a given moment of time. The processes to be carried out to achieve this objective are shown below.

3.1.1. Step detection

In the context of PDR, the first thing to do is to detect the moment when a person is taking a step. This is done in order to capture the time interval between subsequent strides. Having done this, the measurements of the sensors of the mobile devices are obtained during that particular lapse of time.

To perform this process using the inertial sensors of a smartphone, models have been developed which make use of classification with Machine Learning [15], probabilistic methods [16], or sys-

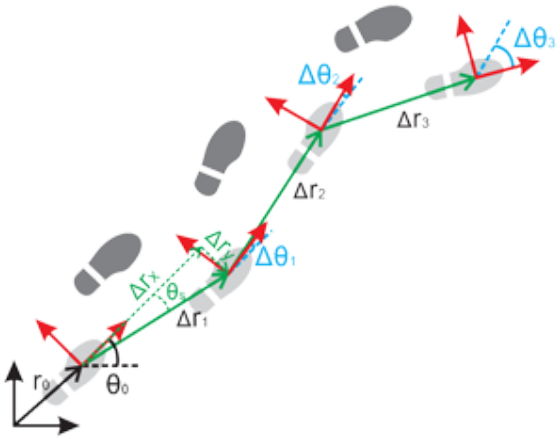


Figure 3.1: Visual representation of the process of Pedestrian Dead Reckoning

tems with analysis of inertial sensors [17].

Although the use of Machine Learning for the process of step detection of passage works with an accuracy quite close to 100 %, it is difficult to perform this process in real-time, because the sampling frequency of the data of inertial sensors tends to be high, being computationally expensive to do this. This is why we tend to use methods which, although they have some lesser accuracy with respect to those of Machine Learning, are capable of being carried out in real-time. Hence, during the work in question, the classic processes of step detection are performed, corresponding that to the use of sensor analysis.

In most cases, the smartphone's accelerometer is used to estimate the moment in which the person took the step. Since the accelerometer outputs the values of the X, Y, and Z axis, we may use the values of the accelerometer in each axis to estimate step instant. In particular, the norm of the accelerometer is calculated for this purpose, instead of using each of the axis of the accelerometer separately. This is done because the acceleration measurements are based on the local frame of the device, and not the terrestrial axes.

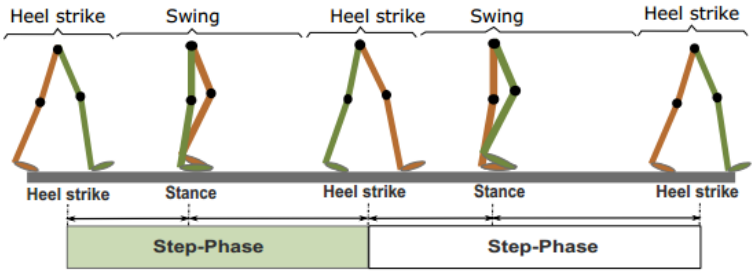


Figure 3.2: Qualitative representation of the gait during a walk.

As shown in Figure 3.2, the steps are separated into the steps of *heel strike*, and *stance*. Thus, when a person puts the heel on the floor, that is the moment when one step ends and the other begins. In a quantitative way, the same walk process can be seen in Figure 3.3, indicating the norm of the acceleration of the device as a function of time. We may notice that in Figure 3.3, the effects of gravity were removed by subtracting the value of the the gravity from the norm of the acceleration, and since the measurements were taken near sea level, the value of the gravity was $g \approx 9.8 \frac{m}{s^2}$. Also, it is possible to notice that the instants when the *heel strike* occurs, correspond to local minimums, while the instants of *stance* correspond to local maximums. It is now known that in order to detect the steps, it is necessary to generate a method which is capable of indicating the moments at which certain local minimums (heel strikes), or local maximum (associated to the stance), are generated, while at the same time being able to differentiate them with local minimums (or maximum), which are not real *heel strikes*. This differentiation process between real and fake heel strikes must be taken into consideration when formulating the respective algorithm.

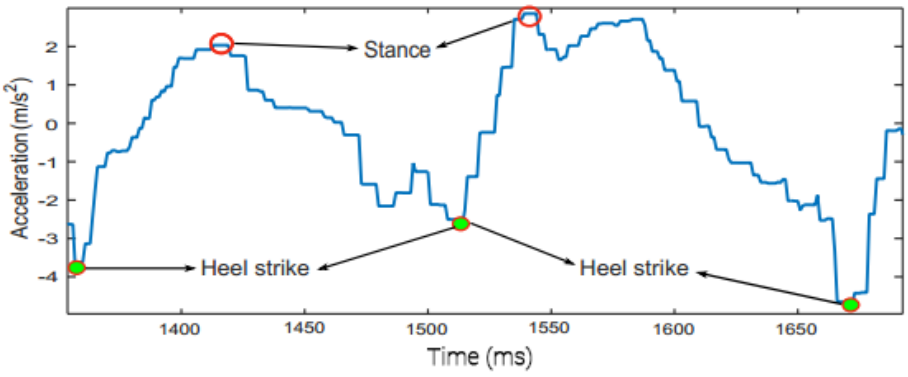


Figure 3.3: Quantitative representation of the steps of a walk, depending on the acceleration.

In order to detect such local minima, it is possible to use adaptive thresholds, whose values change depending on the type of person's walk, or even neural networks. For this process, it is necessary to generate a system robust enough that can work independently of the walks of different types of people. That's why a database is generated which covers a wide range of ages, heights, and gender.

3.1.2. Step Length Estimation

After obtaining the moment in which a step was taken, it is necessary to obtain its length, corresponding to the value of ΔL in the Equation 7. Unlike the case of the step detection, where the accelerometer of the smartphone is mainly used, the estimation of the length can make use of the gyroscope of the smartphone device.

A high number of estimators for the step length estimation process have been proposed, both for indoors, as well as outdoors. Some of these make use of basic models to make the prediction of the length of the step, taking metrics of the accelerometer values during the interval of the step [18, 19, 20]. Other works make use of Machine Learning based methods, considering Long-

Short Term Memory (LSTM) networks and deep networks, generally obtaining better performance.

In order to obtain the model which generates the most favorable results, a comparison is made between the different methods previously mentioned, using performance metrics to compare the results between them.

3.1.3. Orientation Estimation

As was seen in Equation 7, the last process that needs to be done to be able to carry out the PDR process, corresponds to the estimate of the orientation of the device in the terrestrial coordinate axis. To do this, the accelerometer, gyroscope, and magnetometer are used in parallel, being necessary to develop a system that is capable of generating real-time predictions of the orientation of the person using current measurement data, and past states.

Sensor fusion techniques are necessary in order to mitigate the integration errors generated by the gyroscope, while errors incurred by the assistive sensors are mitigated using the gyroscope output [21]. Kalman filters, and in particular, the extended Kalman filter (EKF) correspond to the most common tools used to achieve this purpose [22]. Hence, in order to get the estimate of the orientation, the Kalman filter is used as the basis of the orientation prediction system, using at the same time a complementary system for the improvement of the measurements.

During this process, external factors that may cause inconvenience to estimate orientation are taken into consideration, to minimize the final error. Such factors correspond to the conversion between local axes of the smartphone and the terrestrial axes, in addition to the external and internal magnetic fields that affect the magnetometer measurements.

3.1.4. Algorithm integration for the creation of the PDR model

Although in the Equation 7 it is implied that performing the previous three steps and integrating them is enough to generate a good PDR model, this may not be true. Depending on the case of where the device measurements are being made, there may be disturbances to the sensors of the smartphone. That is why there are methods associated with the topography of the area where measurements are made, which may correct such disturbances.

3.2. Used resources

The methodology developed in this thesis was done with a set of devices and software in particular. In case that such hardware or applications are changed, the final results do not vary much, but the speed at which the process calculations are performed does. Now, since it is desired to deliver conclusive results regarding the speed of calculation of the different algorithms used (apart from other performance metrics), it is necessary to deliver the baseline of the hardware with which the work in question was performed, so that this work may be replicable. This is shown in the Table 3.1, which indicates the specifications of the sensors of the smartphone used, in addition to the CPU, GPU, and RAM where the various models were run.

3.3. Dataset collection

In order to perform the corresponding tests for the development of a functional system, it is necessary to have a dataset representative enough of the majority of the population. Similarly, these must contain a large enough amount of data to train Machine Learning models, preventing overfitting. It is shown below how the databases used in this thesis were obtained.

3.3.1. Dataset for noise sensor analysis

To perform an analysis on the accelerometer, gyroscope, and magnetometer of the telephone mentioned in Section 3.2, it is necessary to generate a database using the same smartphone. To do this, use the GetSensorData application, available for public use [23]. In Figure 3.4, it is possible to see the main behavior of the application, being possible to mark data at used desired moments, and choosing the sampling frequency of the different sensors. It is important to note that the X, Y, and Z axis shown in the application for the accelerometer, gyroscope, and magnetometer, correspond to the local coordinates of the smartphone in the same way as shown in Figure 2.6, and are related to the earth coordinates via a quaternion which represents the rotation between both. This quaternion is estimated when doing the orientation estimation process.

In particular, in the case of gyroscopes, the fact that measurements are made over a certain temperature or another affects the bias associated with the measurements [24]. That is why 32 continuous hours of measurements are performed in a controlled environment, keeping the device at room temperature.

3.3.2. Datasets for the PDR model

Similar to the case of the dataset associated with sensor noise, the GetSensorData application is used to generate the logs associated with the walk. To do this, 40 walks were made, each of approximately 15 steps, which are done in either in a straight line or turning in corners. To be able to generalize further, tests are carried out on walks with the device in the hand, simulating the movement of writing on the phone, and with the device in the pocket.

Unlike other cases, because it is not possible to indicate with the aforementioned application

Table 3.1: Summary of the used resources.

Nokia 6 Smartphone				
Sensor Type	**Sensor name**	**Resolution**	**Maximum range**	**Output noise (RMS)**
Accelerometer	BMA2X2	0.0191 $\frac{m}{s^2}$	$\pm$156.9 $\frac{m}{s^2}$	0.05 $\frac{m}{s^2}$
Gyroscope	BMG160	0.00106 $\frac{rad}{s}$	$\pm$34.9 $\frac{rad}{s}$	$\sim$0.1 $\frac{^\circ}{seg}$
Magnetometer	BMM150	0.304 μT	$\pm$1300μT (x, y-axis) $\pm$2500μT (z-axis)	0.3 μT - 1.4 μT
Notebook HP Probook 440 G4				
GPU	Intel Core i3-7100U			
CPU	Intel HD Graphics 620			
RAM	4GB DDR4			

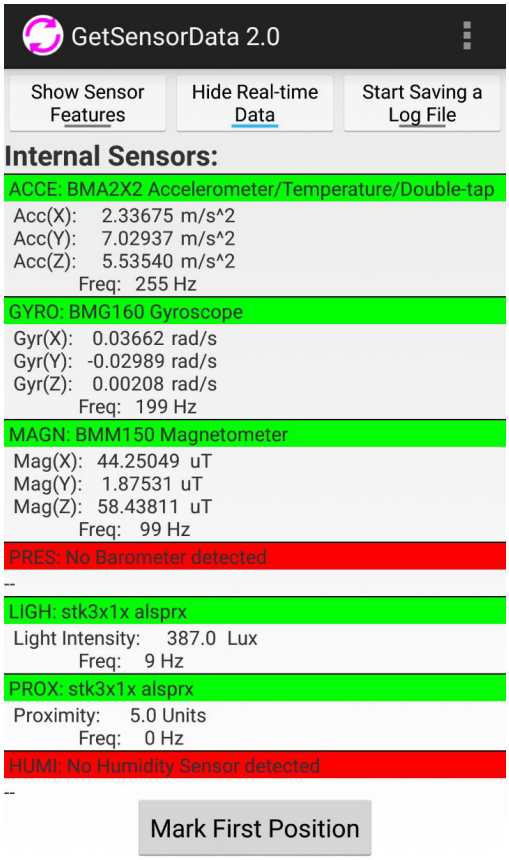


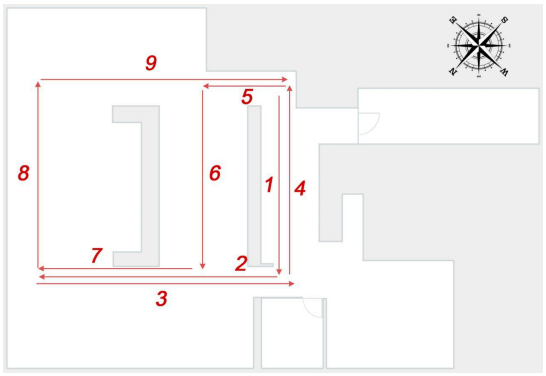
Figure 3.4: GetSensorData application, along its multiple functionalities.

the length of the step of a person, it is necessary to make use of some other dataset which contains such information. Thus, the public repository generated by the *Institute of Computing Technology* in Beijing, China, is used. This consists of about 22 km of walks, with about 10,000 steps taken, including movement by stairs, streets, shopping centers, and office spaces. Thus, great variability is achieved on the types of movements in different environments.

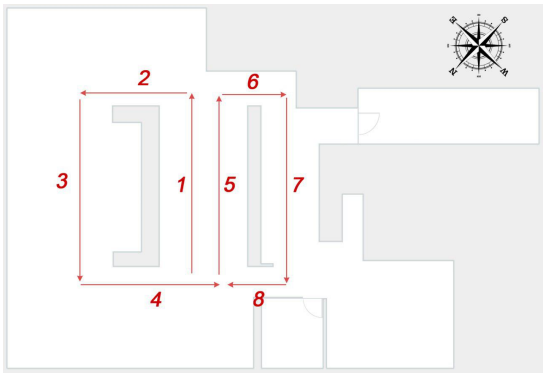
As a third step to create the Pedestrian Dead Reckoning model, we must have data associated with a person's rotations, in order to verify the behavior of obtaining their orientation. To do this, we start by performing a calibration process on the magnetometer of the smartphone, for each set of data obtained. We proceed to generate 4 different routes in an office environment of the Arara company, which take into account rotations mainly in 90 °, having a total of nearly 1 hour of different walks. The routes just mentioned can be seen in Figures 3.5, indicating the trajectories made in the Arara company office.

3.3.3. Dataset for final position estimation

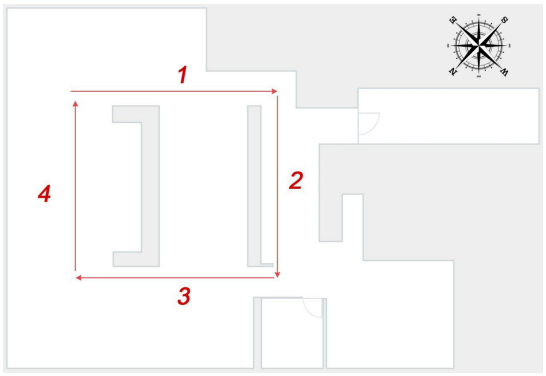
Once the three datasets that allow the analysis of the models to be used have been obtained, a final set of data must be collected, which allows the joint metric to be analyzed on the final position of the person given a certain walk. The data of the Indoor Positioning and Indoor Navigation (IPIN) competition of the year 2019 is used for this purpose. It was done in the city of Pisa, Italy, with 21 files corresponding to 15 hiking routes, including stairs and erratic movements.



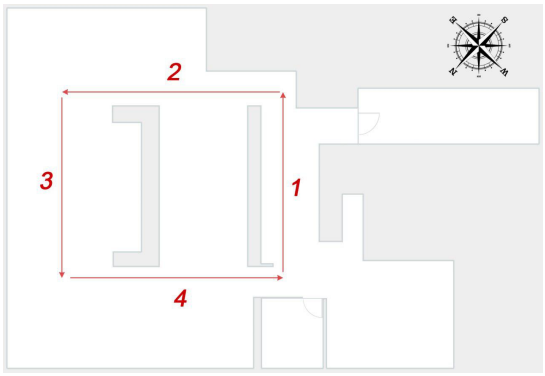
(a) Route 1.



(b) Route 2.



(c) Route 3.



(d) Route

Figure 3.5: Routes made in the offices of Arara with the purpose of verifying the orientation estimation algorithms.

In Figure 3.6, it is possible to see an example of a route made by the organizers of the IPIN competition. This generates POSI markers, which are intended to generate position indicators (or ground truths) for the training of the final models. The rest of the routes can be seen in the Appendix Section 8.2, where from Figure 8.1 up to Figure 8.15, the respective paths which were taken during the data recollection process are shown.

It is necessary to realize that since we'll be working with data corresponding to the city of Pisa in Italy, the magnetic field data associated with that city must be considered. These differ both in magnitude and in each of the components of the Cartesian axes.

3.4. Analysis and data preprocessing

Smartphone devices tend to have low-cost inertial sensors, and therefore, of low quality. These have low accuracy, in addition to being affected by external effects. Below are the methods to be applied in order to eliminate such inconveniences.

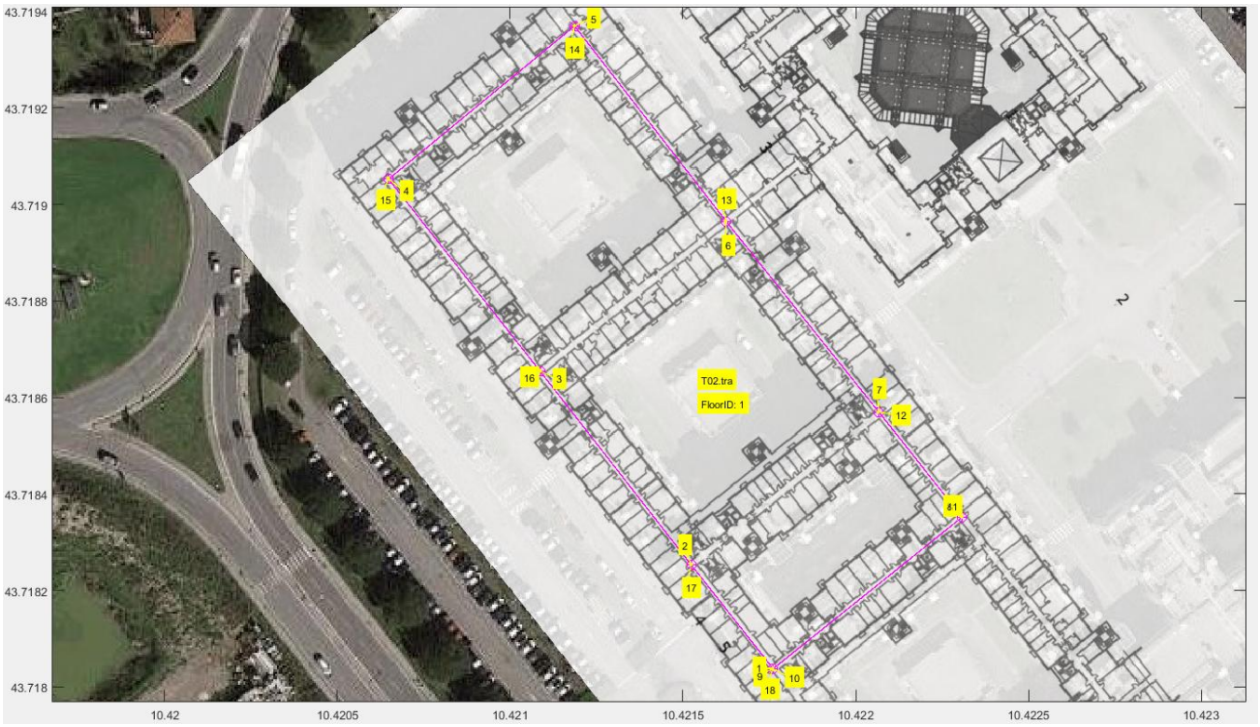


Figure 3.6: Example of a route for the final position estimation dataset.

3.4.1. Data interpolation

Obtaining data with the use of the GetSensorData application has certain inconsistencies when sampling inertial measurements. Although the same sampling frequency is chosen, the data of each of these may have differences in the sampling time. Even though these differences are in the order of nanoseconds, it is necessary to have the data in the same time intervals. This is because some processes, such as the use of Kalman Filters, require inertial synchronous measurements.

It is because of the aforementioned that interpolation is performed on the data obtained with GetSensorData, taking the inertial measurements to a common time-frame. In particular, interpolation is generated at a rate of 100Hz, these values being the point from which an increase in frequency does not generate much change in the final performance of the model [25].

3.4.2. Filter application

As shown in Section 2.5.1, the use of digital filters is necessary when you want to work with signals that are corrupted with noise. In particular, when obtaining the data from the inertial sensors, these are corrupted by different sources of noise depending on the sensor.

Interpolated signals pass through a set of low pass filters. These are designed depending on the frequency response of the filter to be designed, also wanting to have a phase response as flat as possible.

Chapter 4

Step Detection

In order to be able to make a good estimate of a user's position, the instant that the step detection is done, is a fundamental step to perform. Due to a false detection, or a detection which may be missing, substantial inconveniences can be generated at the time of making the final prediction of the walk.

4.1. Classic methods of step detection

An initial idea may come to mind to perform step detection, corresponds to analyzing the accelerometer data of the smartphone device on the Z axis [26]. This is because it is desired to analyze the effects of a person's movement in the direction that affects gravity. The drawback of using this, is that the coordinates on the XYZ axes associated with a person are not necessarily the same as that of a cell phone. In Figure 2.6,it can be noted that while a person's Z-axis corresponds to a direction vector opposite to gravity, the measurements associated with the Z-axis of a mobile device depend on their relative position in space. This is why it is not a good idea to take acceleration measurements in non-absolute coordinates.

$$\mathrm{RMS}(t) = \sqrt{a_x^2(t) + a_y^2(t) + a_z^2(t)} \tag{4.1}$$

Another classically made proposal corresponds to using the magnitude of the acceleration measurements, as indicated in Equation 4.1. Doing this, it is not necessary to know what the orientation of the smartphone device is with respect to the terrestrial axes. Hence, it is sufficient to use an acceleration threshold value to know when a step [27] was generated, so that whenever the acceleration value is over the threshold, a step is considered. This seems to be a good idea, except that each person's walks are different, so you cannot use a single global threshold.

4.2. Acceleration analysis

To be able to generate a good model, capable of predicting the instant when a step was taken, we need to analyze the walk of a person in normal conditions, with the smartphone in hand. To do this, we label the moments in which the person takes a step, using the GetSensorData application, while at the same time, accelerometer data is being saved on the mobile device. The results of this

can be seen in Figure 4.1.

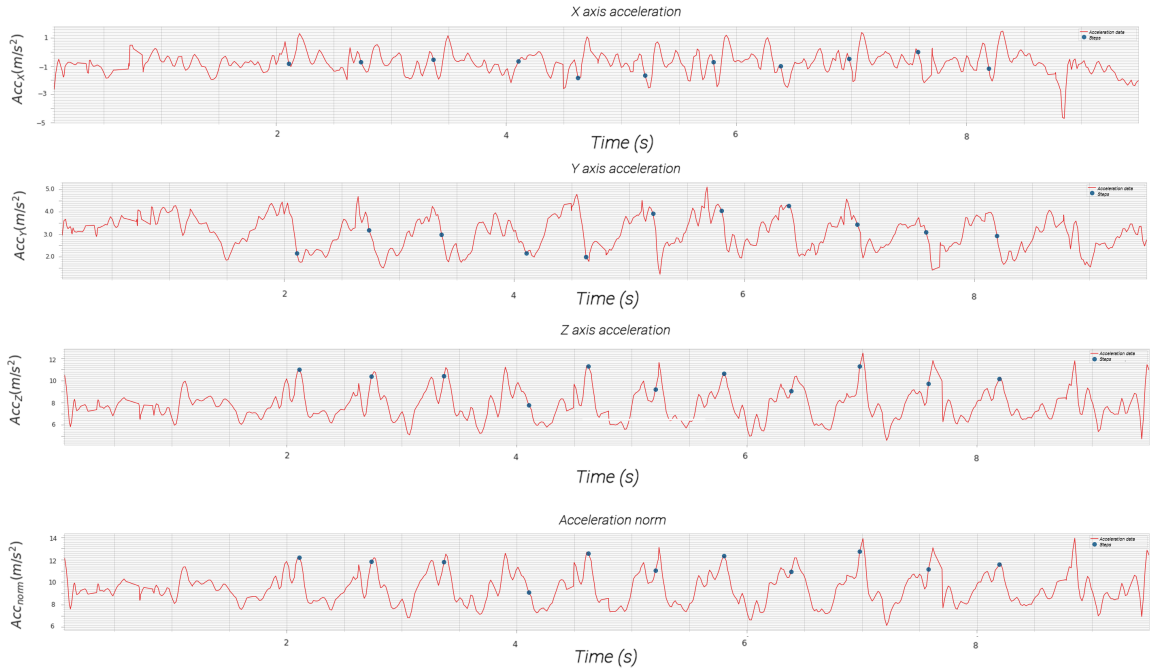


Figure 4.1: Analysis of the accelerometer data of a smartphone device, with the measurements of the steps taken.

As mentioned earlier, what matters are the measurements of the magnitude of the accelerometer data. Analyzing Figure 4.1, where the steps instants were labeled, it is possible to notice that there is a high correlation between the moments of the steps, and the local maximums of the signal. The drawback to note is that the signal has some maximum and minimum local data, which are not associated to the real step instants.

Using the previous information, the assumption that the walking processes are accompanied by certain main frequencies is made, corresponding these to the balancing processes generated by the same movement of the person. To corroborate this, a spectrogram of the acceleration norm is generated in two modes of use of the smartphone: writing a text message while the person is static, and walking with the smartphone in your hand. The results of this can be seen in Figure 4.2, where at the 300 seconds mark, the transition between both modes is made, noting a clear difference in the spectral frequency components of these. In the case of the static mode, the magnitude of the Fast Fourier Transform (FFT) associated with the spectrogram is almost flat. On the other hand, in the moments when walking with the phone in hand, frequency components appear in the order of 1-2 Hz.

While the main component of the signal of the standard of acceleration in the walk is in the range of low frequencies, the sampling of the signal is not perfect. This incurs that the signal has non-null values in the FFT associated with components of another frequency range. As mentioned in section 2.5.1, it is possible to eliminate the high-frequency components of the signal during the walking process. To do so, digital filters of particular characteristics must be generated, which allow us to fulfill the purpose in question.

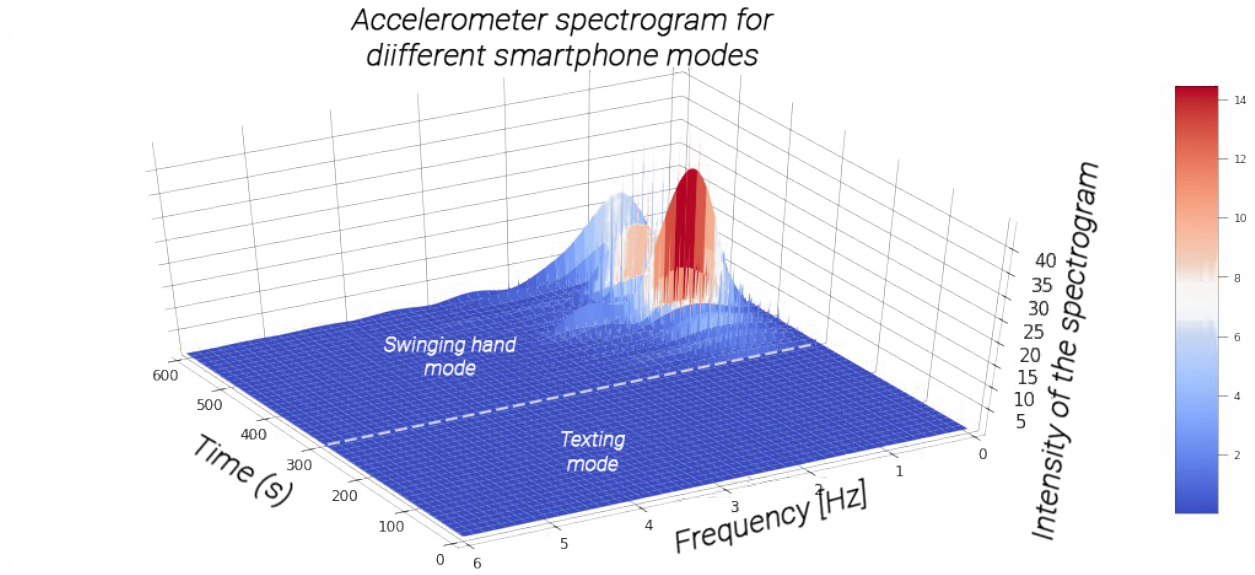


Figure 4.2: Spectrogram of the data of an accelerometer in both text mode and walk mode with a smartphone device.

4.3. Data preprocessing

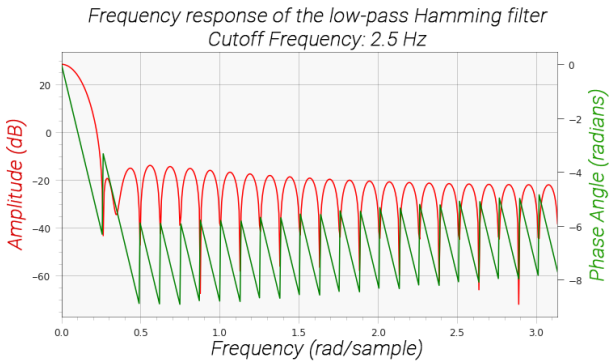
Having the idea of how to proceed, it is necessary to first solve a major problem: asynchrony in taking measurements. Since each of the mobile phone's sensors acts independently, the sampling of the data is not perfect. This can be seen in Table 4.1, where it is possible to see the difference between measurement periods for accelerometer, gyroscope, and magnetometer data. Having selected a sampling rate of 100 Hz, it should be that the average is 10 milliseconds, with a standard deviation close to 0, which is clearly not the case.

Table 4.1: Summary of the temporal difference in sampling for inertial sensors.

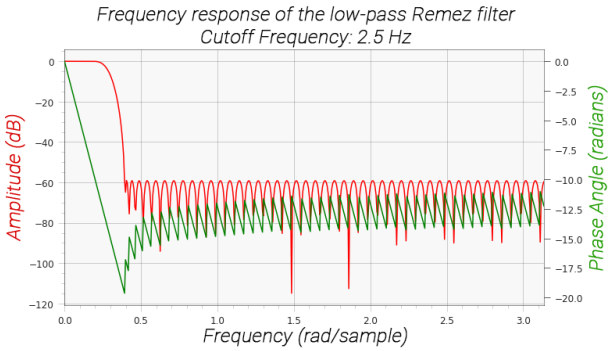
	Gyroscope	**Accelerometer**	**Magnetometer**
Average (ms)	10.52	11.16	9.92
Standard Deviation (ms)	0.31	0.67	0.52

That is why the process of interpolation of the data is carried out at a rate of 100Hz from now on, for all the methods and models performed. This value is chosen because from approximately 75Hz, higher sampling rates are not needed in order to obtain better results [25]. For this, linear interpolation is performed by splines.

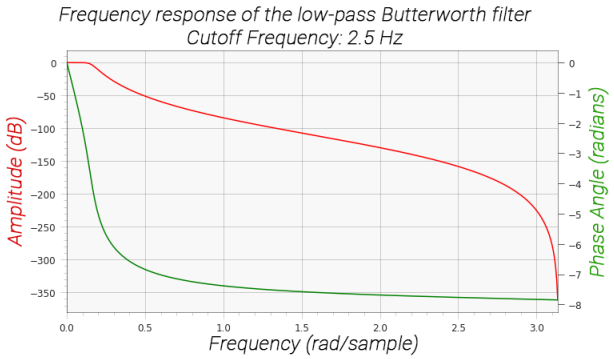
With synchronized data, it is necessary to select a filter which, while not generating a very large phase difference, is able to eliminate unwanted frequencies. For this, 4 low pass filters are tested: Chebyshev, Butterworth, Hamming, and Remez, all with a cutoff frequency of 2.5 Hz. This value was chosen because it is close to the highest value at which a person sways its body up and down when walking, being that same movement the one we want to see reflected in the filtered data. The frequency response of these can be seen in Figure 4.3.



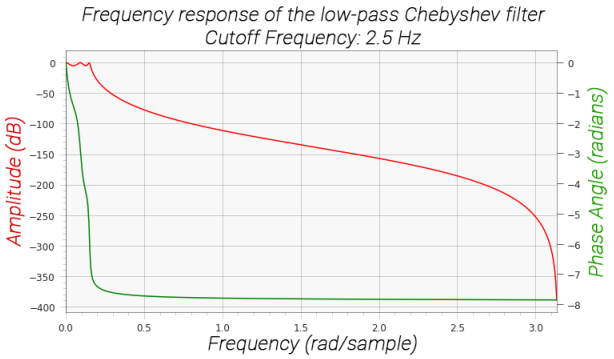
(a) Hamming Filter .



(b) Remez Filter .



(c) Butterworth Filter.



(d) Chebyshev Filter.

Figure 4.3: Frequency and phase response analysis for the low-pass filters used.

Figure 4.4 shows the output of the accelerometer signal after applying the Butterworth filter, verifying that there is a desired frequency behavior. Based on visual analysis, it is not possible to reach a conclusive result regarding which filter acts best over the rest. This is why, after verifying the main detection algorithm, the performance is verified over different filters.

4.4. Prediction algorithm generation

Having already done the data preprocessing, it is now necessary to perform the algorithm which allows the step detection. As already mentioned, the idea of using an acceleration threshold to determine if an instant of time is associated with a step, or not, seems to be suggestive. The concept that there is a metric associated with the magnitude of the acceleration to perform step detection can be emphasized from this previous idea. Other metrics associated with a person's normal walk correspond to the pace of the step, and the distance between local maximum and minimum (jerk). The intuition behind these components is that as a person's speed increases (and therefore, the ratio between consecutive steps does as well), there is a greater tendency to accentuate the steps taken. On the other hand, depending on the way of walking, a person can accentuate the movements in his Z axis more than at the moment of taking a step. This is why both the value of jerk and pace are good measures to verify if a person has taken a step or not.

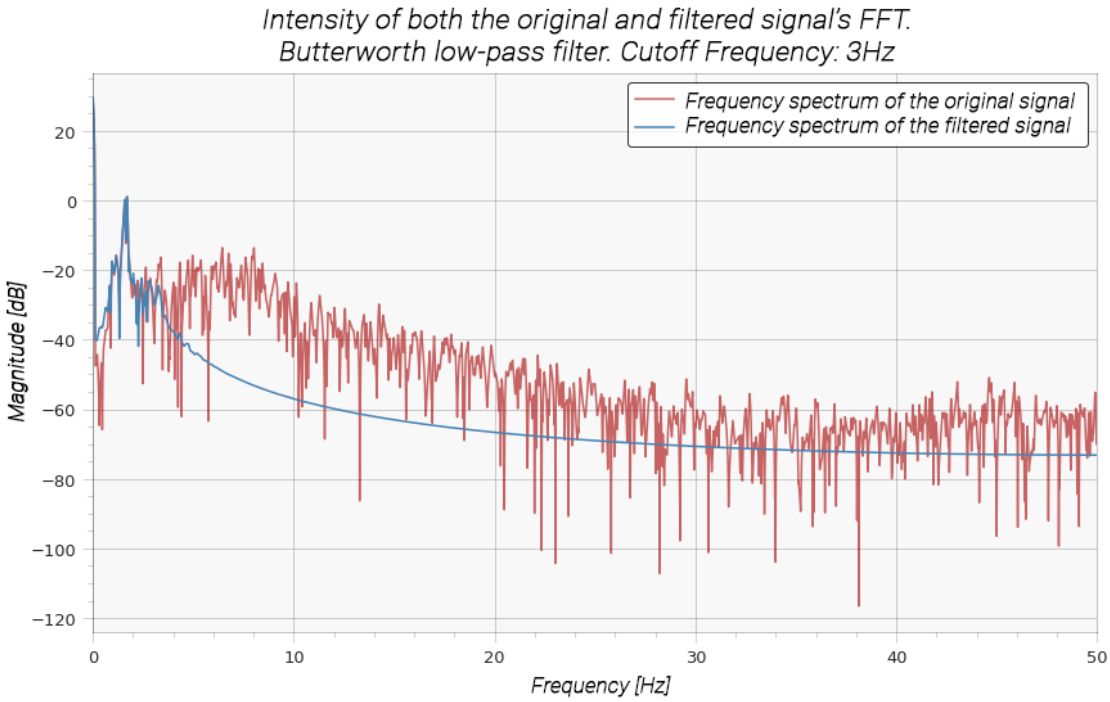


Figure 4.4: Analysis of the norm of the original acceleration signal of the device, and filtered signal with Butterworth filter.

As a last factor to consider, people's walks can vary over time, so it is a must to have an adaptive metric associated with jerk and pace, which is connected through a buffer. Hence, the name Adaptive Step Jerk Pace Buffer algorithm.

In Figures 4.5 and 4.6, it is possible to see the routines and subroutines associated with the step-by-step detection algorithm implemented in the present work.

At each interpolated and filtered accelerometer data, the algorithm analyzes if said data corresponded to an instant when a step was taken. In other words, it is verified in the main routine if each acceleration data is greater than the previous one, or not, in order to verify its slope. In case the previous acceleration data has a different slope than the current one, this may correspond to a possible local maximum or minimum. Based on a possible local minimum or possible local maximum, the subroutine in Figure4.6 is done to verify whether the data corresponds to a maximum or minimum. In case that it is a real maximum or minimum, the pace and jerk are calculated, verifying whether these values are in the average jerk and pace ranges, adjusting these values by a scale factor. In case that this is fulfilled, the algorithm returns to the main routine, adding the value of the respective minimum or local maximum to an array. Finally, it is possible to consider the instants of steps as two consecutive maximums or two consecutive minimums. In this thesis, it was decided to consider the maximums as the actual steps.

You can see the application of this in Figure 4.7, where data preprocessing is performed with the already designed low-pass Butterworth filter.

For the already presented case, a correct prediction of the 12 steps is performed, but this is not

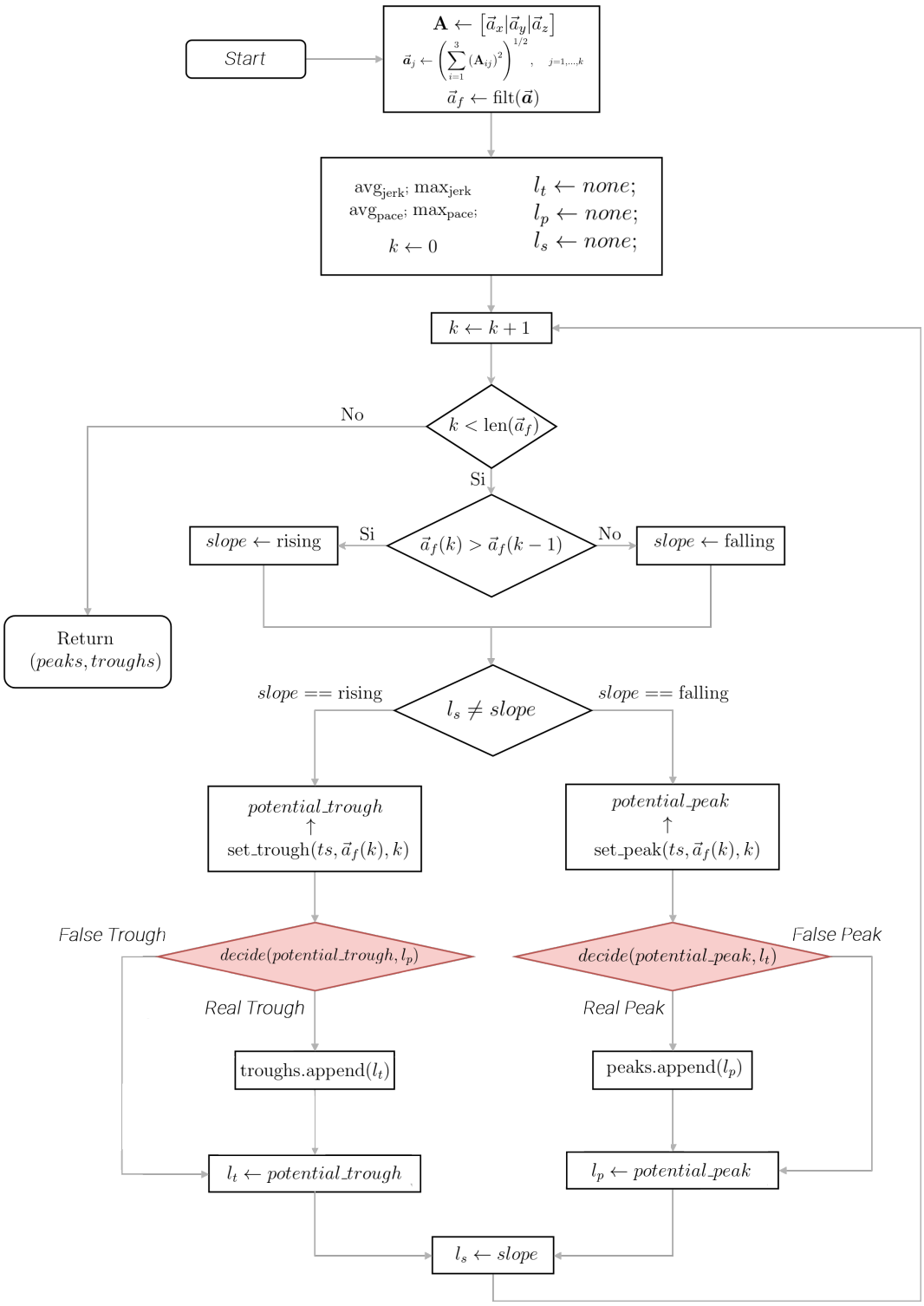


Figure 4.5: Flowchart of the Adaptive Step Jerk Pace Buffer algorithm to be used for step detection.

necessarily a given for any filter. The same set of data was applied to the step prediction routine, but having applied a Chebyshev filter with the same cutoff frequency, and a false prediction is generated, as seen in Figure 4.8.

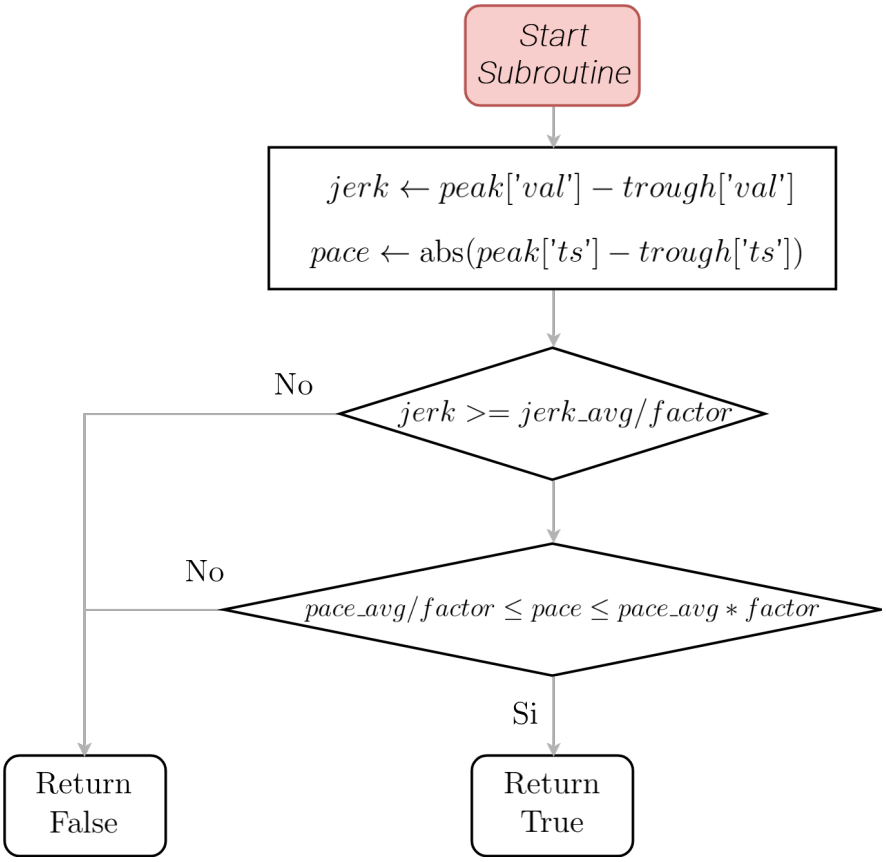


Figure 4.6: Flowchart of the subroutine to be used in the Adaptive Step Jerk Pace Buffer algorithm.

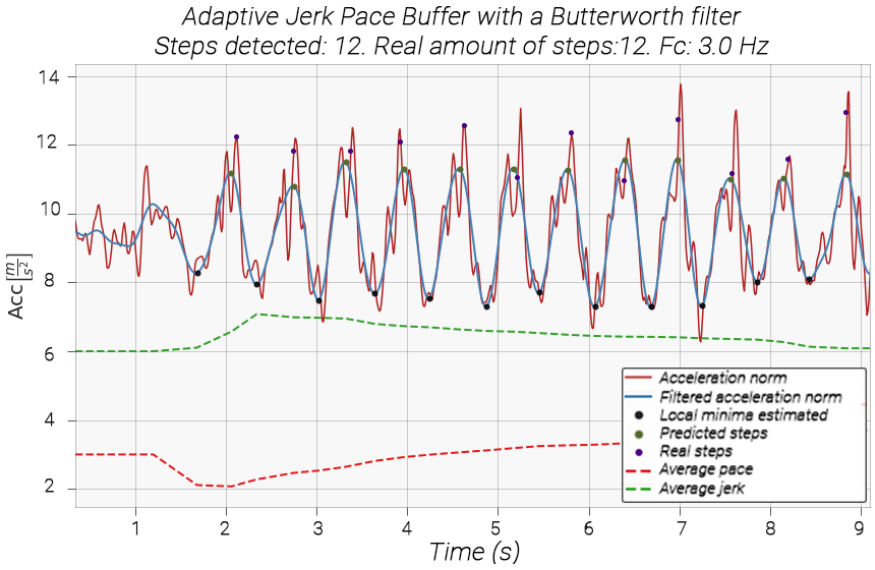


Figure 4.7: Step detection using an Adaptive Jerk Pace Buffer algorithm and Butterworth filter.

To compare the different models, we use as a comparison metric the error percentage given by Equation 4.2. Table 4.2 shows the results of different detection algorithms using the Butterworth

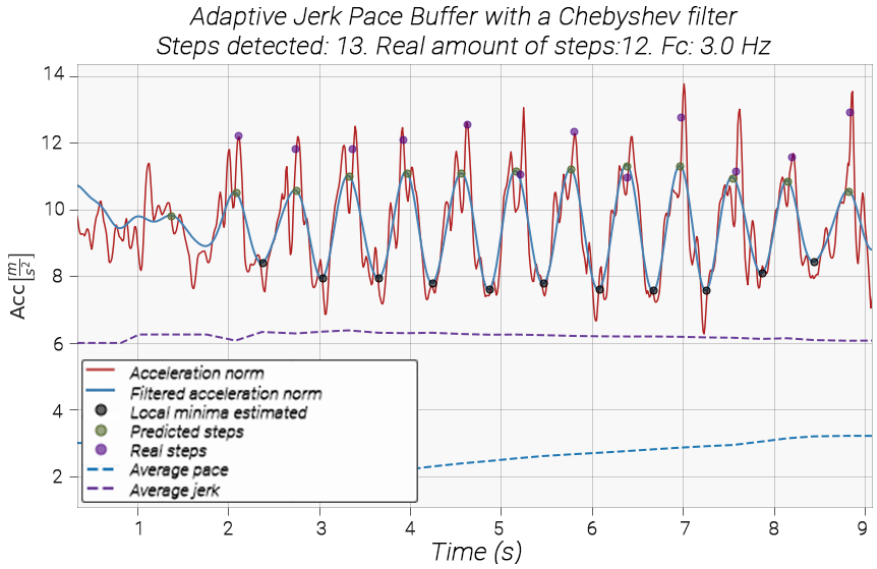


Figure 4.8: Step detection using an Adaptive Jerk Pace Buffer algorithm and Chebyshev filter.

low-pass filter, which has the best average performance among the 4 designed filters. It is possible to note that the Adaptive Step Jerk Pace Buffer performs better in the three analyzed categories, even over the Android black-box algorithm.

$$\text{Error percentage} = \frac{|\text{Real ammount of steps} - \text{Predicted ammount of steps}|}{\text{Real ammount of steps}} \cdot 100 \tag{4.2}$$

In the cases where there was a wrong step detection, it usually happened during stair walks, being necessary to perform a subclassification in order to improve these results. Similarly, the prediction of steps with the phone in the person's pocket is the most complicated, due to the random movements that occur within it.

The present results are within the desired values, being possible to advance to the next process associated with PDR.

Table 4.2: Results of the step detection algorithms.

	Maximum acceleration		**Android**		**Adaptive Jerk Pace Buffer**	
Modes of movement	**Estimated Average**	**Error Percentage**	**Estimated Average**	**Error Percentage**	**Estimated Average**	**Error Percentage**
Swinging in hand	576	4.00%	585	2.50%	591	1.50%
Texting mode	570	5.00&	579	3.50%	592	1.33%
In pocket	559	6.83%	566	5.66%	577	3.83%
Total mean	568.3	5.27%	576.6	3.88%	586.6	2.22%

Chapter 5

Step Length Estimation

Nowadays, developing a Step Length Estimation model with the use of electromechanical systems included in the telephone, is a complicated task due to the following reasons [28]:

1. The length of a person's stride differs with height, age, weight, and gender.

2. It is difficult to keep the stride of a person constant, even for the same person in a single environment.

3. Even for a single person, the length of the stride is dependent on the environment and the pattern of walking: fast walk, normal walk, slow walk, running, or jumping.

Step Length Estimation methods tend to be separated into two categories: A direct method based on the integration of the acceleration; the other method corresponds to an indirect one, which assumes a certain model, extracting characteristics of the acceleration signal or angular velocity to calculate the length of the step.

While the method of double integration of acceleration along the axis of the walk seems, in theory, the best method to calculate the length of the step, the problem of doing this is that the acceleration signal does not necessarily correspond to the acceleration of the person who owns the smartphone. First, as it was seen in the case of the detection of steps, the axes of the smartphone do not necessarily correspond to the same axes of the person, so that the acceleration in the axis of the movement cannot be obtained directly. On the other hand, even if you had the transformation matrix that represents the rotation between axes, the obtained signals tend to be corrupted by noise from different sources, which end up generating an error of order $\mathcal{O}(t^2)$ in the position.

Because of the aforementioned, different indirect methods for the calculation of the step length are compared. Among these, a modification of the method generated by Qu Wang [29], based on an LSTM network, is postulated.

5.1. Classic methods

As a way to compare between different models, we use some classic methods of step length estimation to contrast them with the proposed model. In each one of these classic methods, there is a conversion factor (K_1, K_2, K_3, or K_4) which is used to convert the units of the value by which it is

being multiplied by, to distance.

One of the best-known estimators for PDR-based applications used in smartphones was proposed by Weinberg [18]. He proved empirically that the vertical acceleration applied to the device is directly correlated to the length of a particular step. Equation 5.1 explains this in a quantitative way, where A_{max} and A_{min} correspond respectively to the maximum and minimum accelerations of the device projected onto the earth's axis associated to the gravity, measured for a particular step.

$$L_1 = K_1 \cdot \sqrt[4]{A_{max} - A_{min}} \tag{5.1}$$

Subsequently, Kim [19] developed an empirical method based on the dependence of the average acceleration with each step during the walk. Equation 5.2 indicates the previously mentioned relationship, where A_i is the acceleration measured for a sample i in a single step, M is the number of samples corresponding to the acceleration measurements of said step, and K_2 corresponds to a conversion factor.

$$L_2 = K_2 \cdot \sqrt[3]{\frac{\sum_{i=1}^{M} |A_i|}{M}} \tag{5.2}$$

While not as used as the last two, Tian [20] developed a method for estimating the length of steps based on the person's height h and the frequency f_s of the step taken. Equation 5.3 shows the correspondence between the length of step L_3, a conversion factor K_3, the height of the person h, and the frequency of step f_s.

$$L_3 = K_3 \cdot h \cdot \sqrt{f_s} \tag{5.3}$$

As the last classic method to consider, the model proposed by Scarlett [30] solves the problem of step length variation for the same individual generated by Weinberg. The step length is calculated as seen in the Equation 5.4.

$$L_4 = K_4 \frac{\frac{\sum_{i=1}^{N} |A_i|}{N} - A_{min}}{A_{max} - A_{min}} \tag{5.4}$$

5.2. Proposed model

Deep Learning has the ability to learn features automatically at a high level of abstraction, using various nonlinear transformations. As it was seen in the case of the step detection, it is possible to verify that there is an implicit relationship between the acceleration data, and the instants in which these are performed. In the case of the step length, this relationship is usually more hidden, and it is not possible to perform an analysis in the frequency spectrum in order to reach conclusive results.

Because there is a temporary component associated with the accelerometer and gyroscope data, it is a good idea to use Deep Learning in the form of an LSTM network to perform the desired

estimate. Thus, in the same way that Deep Learning has allowed the improvement of the results of systems based on time series (such as the case of natural language processing and voice recognition), the improvement of the results of step length estimation models are expected to be better than the classic methods mentioned previously.

The model in question is separated into the following sections, these must be done sequentially:

1. Only a LSTM network is generated, making predictions based on the acceleration inputs, gyroscope, and high-level features. The purpose of this part is to train the LSTM layers, so as to have a relatively optimal distribution of the weights associated with it.
2. Using the LSTM network, a Denoising-Autoencoder model is added, allowing us to obtain a data representation which is not affected by noise.
3. The final regression is constructed from the output of the Encoder, adding 3 Fully Connected layers, and fine tuning the parameters of these.

In Figure 5.1 it is possible to see the architecture just indicated, splting each one of the modules, depending on the function of each of these.

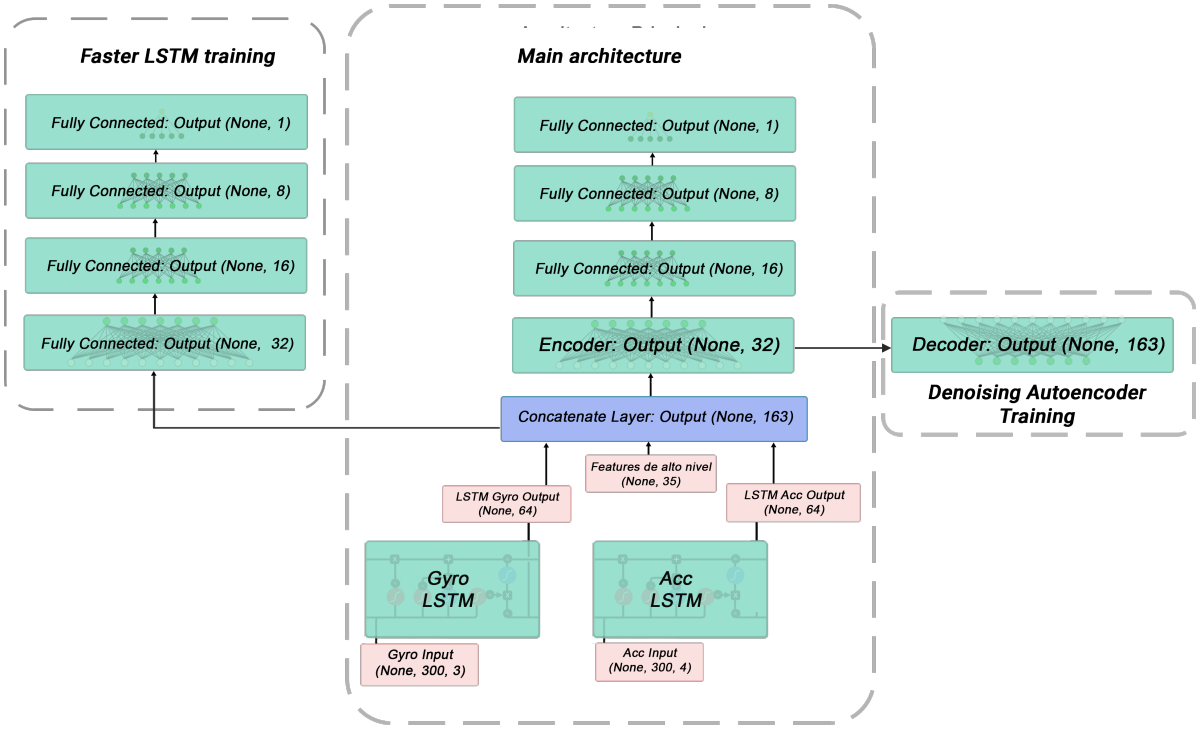


Figure 5.1: Machine Learning architecture for the estimation of the final step length.

5.2.1. Extraction of temporary features based on LSTM

For the first part of the network, it is necessary to start by defining the input data. Because different sensors are available for each of the mobile devices, it is necessary to take those values to a common measurement representation. This is why normalization is performed over the gyroscope

and acceleration data, using 25 % of the total data for the purpose of getting the mean and standard deviation values.

With the normalized values of acceleration and angular velocity, and the high-level characteristics extracted from the latter two, the required input data is available. The accelerometer and gyroscope data can be represented matrixally as postulated in Equations 5.5 and 5.6 respectively, with K being a predefined fixed length. For the present work, $K = 300$ is considered. This means that for a sampling rate of 100Hz, the data set corresponds to 3 seconds long. While most of the steps have a much shorter duration than this, zero-padding of the data is performed in case that the time period of the step is, in fact, shorter.

High-level characteristics correspond to the following acceleration and gyroscope metrics: mean, standard deviation, skew, kurtosis, signal energy, actual signal length, zero crossing rate, cross-axis cross correlation, and energy in different bands of the FFT.

$$\mathbf{Acc}_i = \begin{bmatrix} a_{x1} & a_{x2} & & a_{xK} \\ a_{y1} & a_{y2} & \dots & a_{yK} \\ a_{z1} & a_{z2} & & a_{zK} \end{bmatrix} \tag{5.5}$$

$$\mathbf{Gyr}_i = \begin{bmatrix} g_{x1} & g_{x2} & & g_{xK} \\ g_{y1} & g_{y2} & \dots & g_{yK} \\ g_{z1} & g_{z2} & & g_{zK} \end{bmatrix} \tag{5.6}$$

Gyroscope data, acceleration, and high-level features are concatenated, being this the input to the regression network. In this network, the hidden layers of the regression section possess a ReLU activation function.

5.2.2. Noise sanitization using Denoising Autoencoders

Because the phone's sensors inherently possess noise, it is necessary to generate a filtering process. Unlike the classic signal filtering techniques, such as the use of digital lowpass filters, the Denoising Autoencoder indicated in Section 2.6.2.2 is used. These are able to efficiently learn the characteristics of the signal, and recognize which components are noise, and which are the actual signal.

Let $\mathrm{H}_{\mathrm{acc}}$ be the function associated with the LSTM network of acceleration, and $\mathrm{H}_{\mathrm{gyr}}$ the mapping of the LSTM network of the Gyroscope, the DAE section has three fundamental components: Dropout, Encoder, and Decoder. The dropout process is intended to avoid overfitting, performing a more robust Encode-Decode process.

$$\mathbf{h} = \begin{bmatrix} \mathrm{H}_{\mathrm{acc}}(\mathbf{Acc}) \\ \mathrm{H}_{\mathrm{gyr}}(\mathbf{Gyr}) \\ \mathbf{Feats} \end{bmatrix} \tag{5.7}$$

$$\hat{\mathbf{h}} = \mathrm{Decode}(\mathrm{Encode}(\mathrm{Dropout}(\mathbf{h}))) \tag{5.8}$$

Given a feature $\mathbf{h}$, the DAE trains the Encoder and Decoder in order to minimize the reconstruction error, corresponding to the minimization of the following objective function:

$$J_{\mathrm{DAE}}(\mathbf{h},\hat{\mathbf{h}}) = \frac{1}{2N}\sum_{i=1}^{N}(h_i - \hat{h}_i)^2 \tag{5.9}$$

where h_i and $\hat{h}_i$ correspond to each of the values in the vectors $\mathbf{h}$ and $\hat{\mathbf{h}}$ respectively, which were previously mentioned in the Equations 5.7 and 5.8

5.2.3. Step length estimation regression using a LSTM-DAE network

Once the Denoising Autoencoder has been built and trained, the actual step length is used to be able to monitor the training of the final network. If the function associated with the final Fully Connected layers is G, it is desired to minimize the mean square error between the actual labels and the predicted data:

$$J(\mathbf{y},G) = \frac{1}{2M}\sum_{i=1}^{M}(y_i - \hat{y}_i) \tag{5.10}$$

In the previous equation, y_i the real value of the step length, and $\hat{y}_i$ the step length estimate from the output of G.

In Figure 5.2, it is possible to see the final network, where the step length estimation is made.

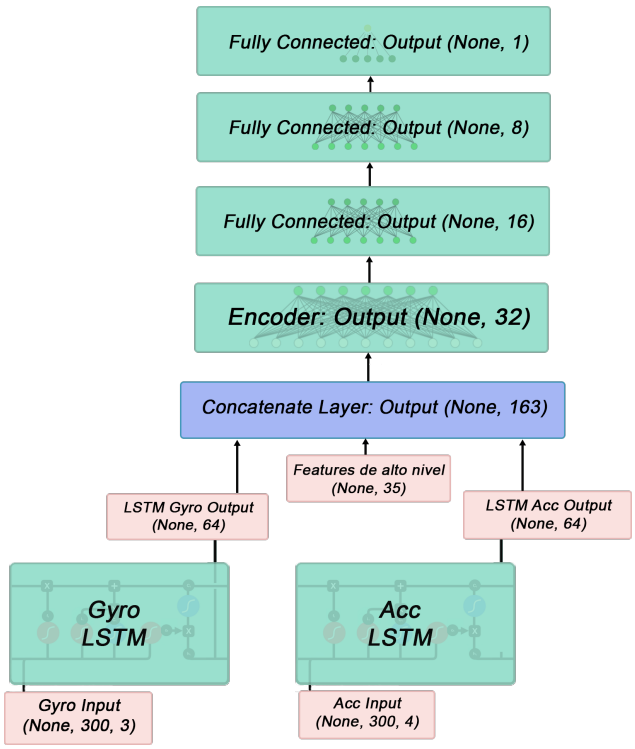


Figure 5.2: Machine Learning architecture for the estimation of the final step length.

5.3. Hyperparametric selection and training analysis

Having used the Pytorch library for the generation of the models, hyperparametric optimization was used by optimizing the learning rate of each of the models, the optimizer, and the loss function. Thus, the best performance hyperparameters are reached in the test data set, which can be seen in the Table 5.1.

Table 5.1: Summary of the selection of hyperparameters obtained for the network.

Parameter	LSTM	DAE	LSTM-DAE
MiniBatch Size	128	128	128
Hidden Layers	32-16-8-1	32-163	32-16-8-1
Activation Function	ReLU	Sigmoidea	ReLU
Optimizer	RMSprop	RMSprop	RMSprop
Learning Rate	0.001	0.01	0.001
Maximum Epochs	500	50	500
Loss Function	MSE	MSE	MSE

As a method to avoid overfitting, early-stopping is performed, which stops the training of a particular model once it has spent n times without reduction in the loss function. Figure 5.3

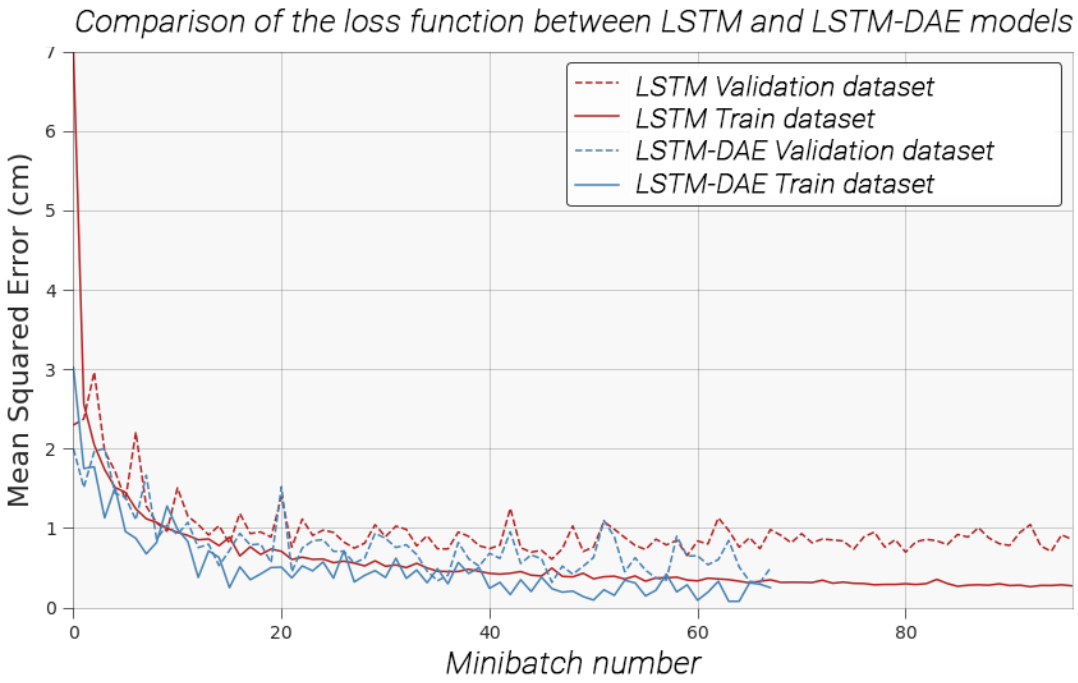


Figure 5.3: Cost function of trained models. Comparison between training and validation sets.

Having trained the models, an evaluation metric is defined to compare between each one of these. Let L_e^i be the estimated long step and L_t^i the actual step length, the relative error is defined by the following equation:

$$E_r = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|L_e^i - L_t^i|}{L_t^i} \times 100\%\right) \tag{5.11}$$

We will be referring to this metric as the error rate.

5.4. Classic methods comparison

A comparison between the LSTM-DAE model with the classical methods is made. The cumulative distribution function of each one of these in the test set can be seen in Figure 5.4.

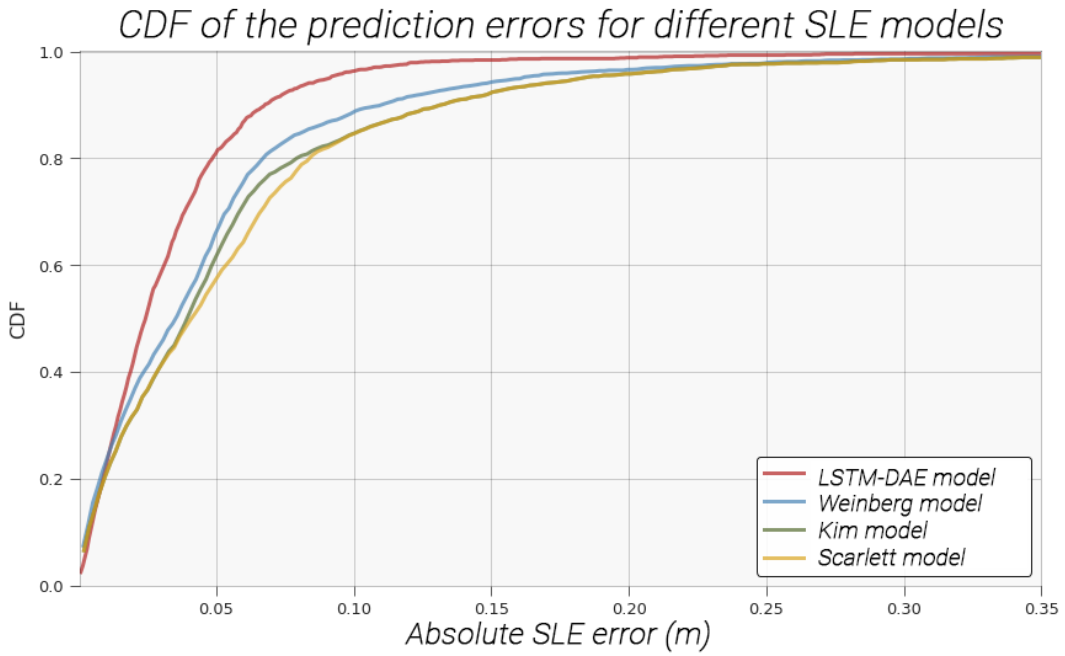


Figure 5.4: CDF of the step length estimation models.

It is possible to note that the LSTM-DAE model has a much better performance compared to those of the classical methods, independently of the length for which they are compared.

5.5. Comparison between proposed models

In order to analyze how much was improved using Denoising Autoencoder in conjunction with LSTM, versus the LSTM network alone, some metrics are compared using estimation errors. These results are seen in the Table 5.2.

One of the biggest drawbacks that can lead to errors in the step length estimation, corresponds to the false detection of the step, or the lack of this detection. The advantage of the current work is that the step prediction model has good results, so this error does not spread to a large extent.

With an average error rate of 3.78 %, corresponding to 4.9 centimeters, the results of this model are generally good. Cases where there is a high error tends to occur in situations with stairs, where the length of the step is limited to the length of the stair itself.

Table 5.2: Comparison of step length estimation methods using LSTM and LSTM-DAE.

Attributes	**LSTM**		**DAE**	
	Error [m]	**Error rate**	**Error [m]**	**Error rate**
Average	0.062	4.12%	0.049	3.78%
Standard deviation	0.041	-	0.040	-
25%	0.027	1.89%	0.021	1.32%
50%	0.052	3.26%	0.039	2.55%
75%	0.065	4.77%	0.058	4.26%
Minimum	3.52×10^{-4}	~ 0	6.32×10^{-5}	~ 0
Maximum	0.520	38.2%	0.367	26.9%

Chapter 6

Orientation Estimation

Being considered usually the most complicated step, the process of orientation estimation of a person using the inertial sensors of the mobile phone is a critical task. Since 1965, multiple solutions have been developed against this problem, such as TRIAD [31], QUaternion ESTimator (QUEST) [32], Kalman Filters [33, 34], Filters Extended Kalman [35, 36], Kalman Filters Unscented [37], and Nonlinear Filters[38]. Within these, the use of the extended Kalman filter has consistently delivered good results, being used nowadays in state of the art algorithms. Thus, the main focus of this section is the use of this particular filter.

6.1. Orientation estimation difficulties

6.1.1. Sensor noise

As it was seen in the Step Length Estimation process, and in the Step Detection, the use of low-cost sensors, tends to induce unwanted noise in inertial measurements. As mentioned in Section 2.2, in order to verify the behavior of certain sensors, it is possible to make use of Allan's analysis by variance for each sensor. Figures 6.1 and 6.2 shows the overlapping Allan variance in the gyroscope and magnetometer sensors. To do this, the recordings had to be done in a controlled environment at constant temperature, in order to keep the internal parameters associated with the gyroscope constant.

From Figures 2.5, it is possible to notice that for each of the sensors, there is a factor associated with quantization noise, in addition to a random walk. This is checked using Python tools, verifying that the correspondence to the other noises is close to 0.

6.1.2. Gimbal Lock

When creating the model-solving problem, a parametrization of Euler's angles is done, in order to estimate the rotation matrix between the local reference system (mobile device), and the global one (Earth). Algorithms using these angles describe three-dimensional rotation through three successive rotations on the three local coordinate axes of the device: *yaw* (Ψ), *pitch* (θ) , and *roll* (ϕ). With these, it is possible to calculate the *direct cosine matrix* (DCM), or rotation matrix, which allows the transformation between both reference systems.

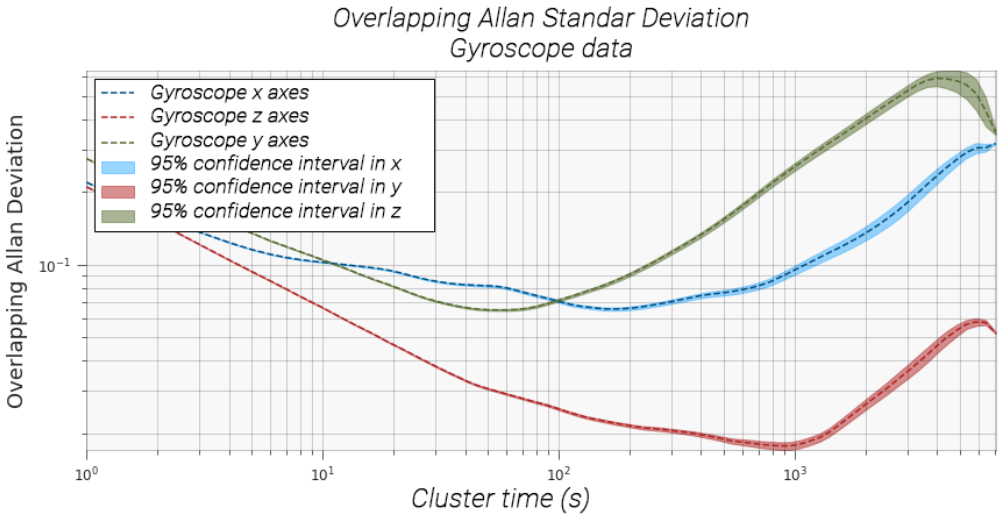


Figure 6.1: Allan variance data for the gyroscope.

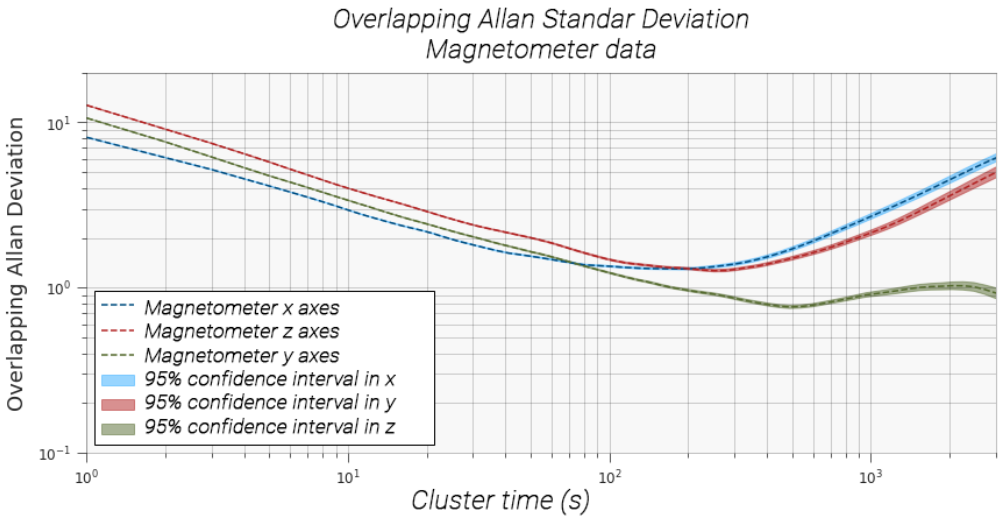


Figure 6.2: Allan variance data for the magnetometer

Unfortunately, using the parametrization with the DCM shows singularities at certain angles, losing one degree of freedom in the system, this is commonly known as the Gimbal Lock effect. For example, in the case that $\Psi = \pi/2$, several pairs of yaw and roll angles allow that same rotation, losing a degree of freedom. That is why it is desired to use quaternions to the greatest extent possible, which are also less expensive computationally, using only Euler's angles, when wanting to show graphic results.

6.1.3. Earth orientation

When talking about the orientation of a person, one tends to think about where a compass is pointing to, if the individual holds it in front of it. The angle of the compass is an indicator of how rotated this person is with respect to the magnetic north. In order to generalize this abstraction, the concept of Local Tangent Plane (LTP) is introduced, which corresponds to a coordinate system based on the local vertical direction, and the axis of rotation of the Earth. Thus, there are three

components, or coordinates to define the LTP's reference frame: An axes in the gravity direction, one in the direction of the magnetic north, and one orthogonal to both these axes . In Figure 6.3 you can see the graphical representation of the above, using the tangent plane defined by East-North-UP (ENU), and the combination of North-East-Down (NED) can exist. Although it seems a trivial issue, the choice of the tangent plane affects both the way in which the angles of rotation are represented, as the signs of components associated with gravity, and local magnetic data. During the present work, the ENU representation is used to model such factors.

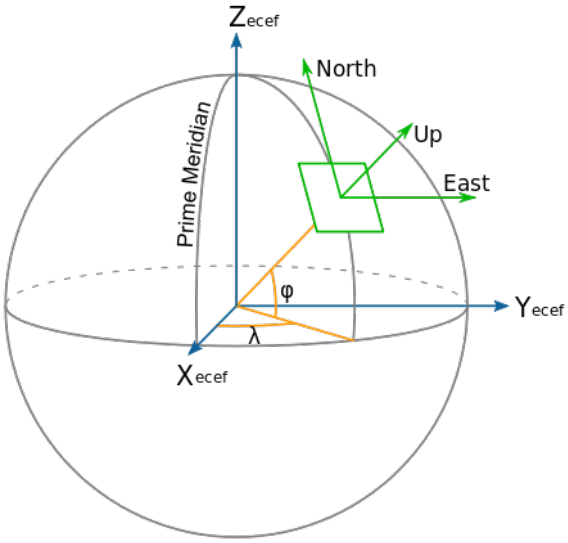
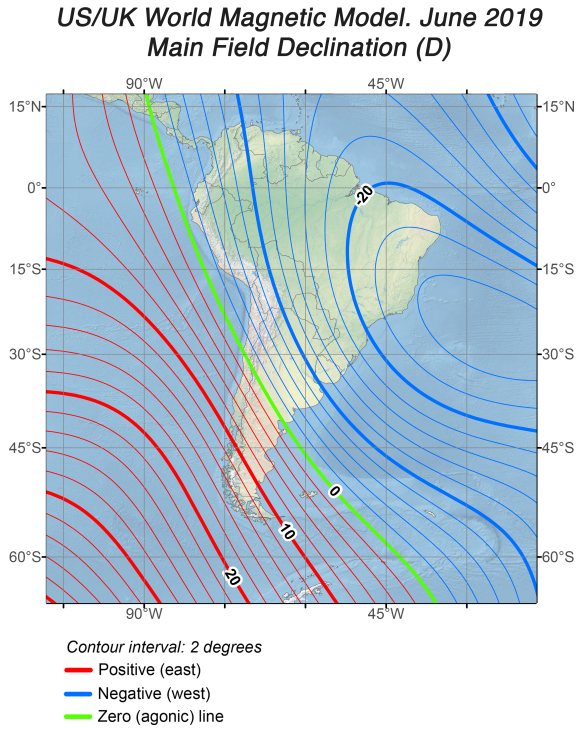


Figure 6.3: Local tangent plane using an East-North-Up frame

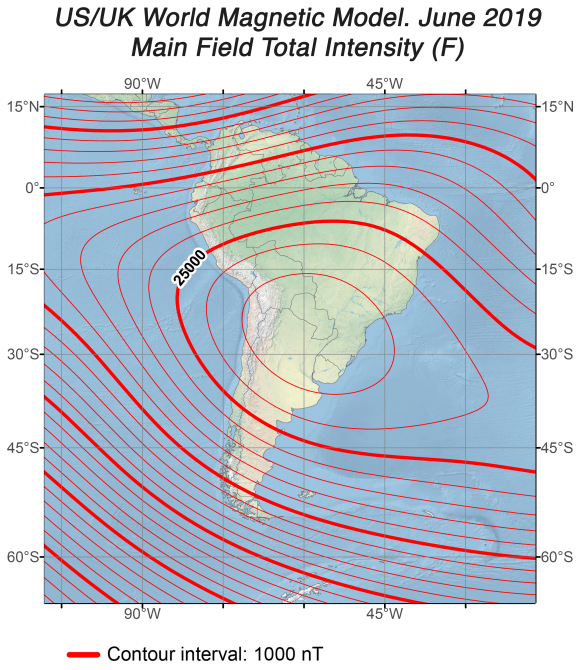
While it has been repeatedly mentioned that inertial sensors do not reflect perfectly the measurement data, these are not the only sources of uncertainty which are added to the process. If the magnetometer measures data without uncertainty, it should be enough to compare the values obtained with the Earth's magnetic field in the location where it is, to know our orientation, but in most indoor cases, that may not be entirely true.

The Earth's magnetic field is believed to come from the movements of a large ocean of liquid iron in the Earth's outer core. Acting in the same way as a driver in the dynamo of a bicycle, it generates currents which constantly change the electromagnetic field of the Earth in the troposphere. Changes in the acceleration of the Earth's magnetic field intensity are highly correlated with changes in how this liquid iron flows in the outer layer. The problem lies in the fact that the iron flow does not generate a constant magnetic intensity field (changing only the degree of rotation). Figure 6.4 shows how the intensity and the angle declination (difference between geographic north and magnetic north) vary depending on the position.

The Earth's magnetic field not only changes depending on the position, but there is also an associated temporal factor. Figure 6.5 shows how the magnitude of the field changes depending on the year in Santiago, Chile. That is why, it is necessary to update the values every couple of years to make sure that the system is working as expected.



(a) Declination.



(b) Magnitude of the Earth's magnetic field.

Figure 6.4: Map of the magnetic field strength and the declination according to the real north, in South America. Measurements of the year 2019.

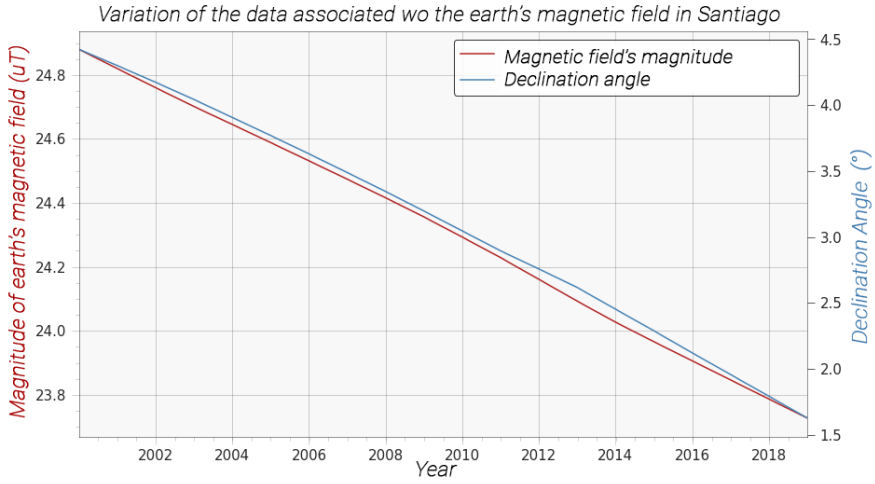


Figure 6.5: Temporal variation of the Earth's magnetic field in Santiago, Chile

Hence, we have the real reference of the magnetic value $m_E(\mathbf{x},t)$ in the Earth's reference system E, depending on the position $\mathbf{x}$, and time factor t. Theoretically, without external magnetic disturbances, the smartphone magnetometer data should correspond to these reference values, rotated according to the quaternion of the device. Thus, in the algorithm to be developed, the reference is

compared with the magnetometer data, achieving better results overall.

6.1.4. Magnetic distortions

Let m_E be the Earth's magnetic field at a certain fixed location, with $\mathbf{C}_n^b(q)$ the rotation matrix between the reference system n corresponding to the Earth, and the body b corresponding to the smartphone device, the magnetic measurements of the smartphone can be written as follows:

$$h(t) = K^m C_n^b(\mathbf{q}) m_e + b^m + \delta_h(t) \tag{6.1}$$

The value δ_h corresponds to non modeled noise, while the values of K^m and b^m correspond to the soft iron distortions and hard iron distortions respectively . The hard iron distortions correspond to a bias introduced product of the magnetic fields adjacent to the device, either by induced electromagnetic fields (produced by elements such as microwaves), or metal structures. On the other hand, distortions due to soft iron occur due to the small electronic components within the same phone. Ideally, it is desired that K^m be as close to the identity matrix, while b^m is as close to 0 as possible.

Depending on the case, hard iron distortions can greatly affect the measurements. Figure 6.6 shows the measurements of the magnetic field norm inside the Arara company office in Providencia. In circumstances without magnetic field distortion, these should oscillate in the order of 24 μT, however, this value may vary plenty depending on the position within the enclosure.

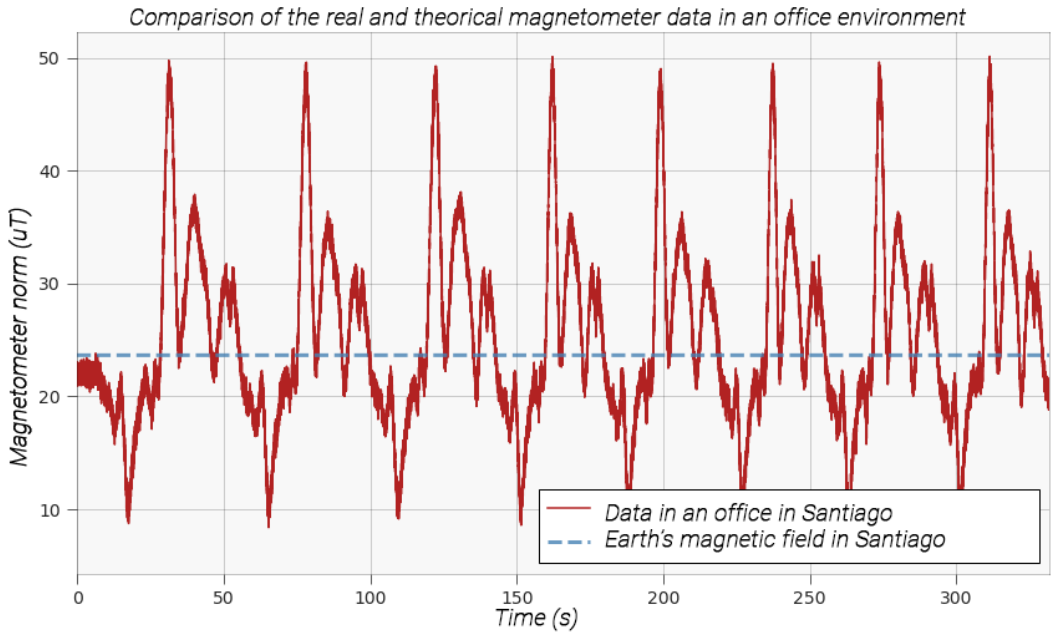


Figure 6.6: Measurement of the magnetic field strength inside an office, for a walk on Route 3 inside the Arara office.

That is why a good practice before using magnetic sensors is to perform a calibration process. The purpose of this action, is to adjust the values obtained by the sensor, so that the measurement output is as close as possible to the real output. Although there are manual calibration techniques

[39, 40], we make use of the black-box calibration process done by Android, which is performed automatically every few moments.

6.2. Kalman Filter design

6.2.1. Orientation cinematics

As already mentioned, the transformation between the representations of a vector $mathbfx$ in reference systems n, and b, can be described as:

$$\mathbf{x}^b(t) = \mathbf{C}_n^b(\mathbf{q}(t))\mathbf{x}^n(t) \tag{6.2}$$

From now on, in order to simplify the notation, the temporal dependence on each of the components of the equations is implicitly assumed. Also, any multiplication between quaternions is assumed to be dictated by the operations associated with them.

6.2.1.1. Quaternionic representation of the orientation

The quaternion $\mathbf{q}(t)$, which represents the rotation of the smartphone with respect to the terrestrial absolute reference system, can be broken down into its real part q_0 and its vector part $\mathbf{e} = [q_1, q_2, q_3]$, so that $\mathbf{q} = [q_0, \mathbf{e}]^T$. This quaternion is the state to estimate.

Since this quaternion represents the absolute orientation of the device, the variation in time of the smartphone depends on its angular velocity. Thus, using the angular velocity in the body reference system b, ω, the angular movement of the body is governed by the following differential equation [41]:

$$\frac{d}{dq}\mathbf{q} = \Omega[\omega]\mathbf{q} \tag{6.3}$$

Also, using the definition of the cross product matrix $Cmatrix(\mathbf{p})$ of the Section 2.3, the following relationship is satisfied:

$$\Omega = \frac{1}{2}\begin{bmatrix}[C(\omega)] & \omega \\ -\omega^T & 0\end{bmatrix} \tag{6.4}$$

Since this corresponds to the continuous-time model, it is necessary to convert it into discrete-time. To do this, the exponential matrix is used, having:

$$\begin{aligned}\mathbf{q}_{k+1} &= \exp(\Omega_k t_s)\mathbf{q}_k \\ &= \left(\mathbf{I}_4 \cdot \cos(||\omega(t_k)||t_s/2) + \blacksquare(\omega(t_k)) \cdot \frac{\sin(||\omega(t_k)||t_s/2)}{||\omega(t_k)||t_s/2}\right)\mathbf{q}_k \\ &= \Phi_k \mathbf{q}_k\end{aligned} \tag{6.5}$$

The value of t_s corresponds to the sampling time of the data, this being 1/100 (s) using the interpolated data. On the other hand, the plausibility of the Equation 6.5 depends on the assumption that $\dot{\omega} = 0$ during the interval [k, k + 1].

6.2.1.2. Sensor modelling

In the analysis of the Allan variance of the sensors, it can be seen that they suffer from imperfections which make the measurements noisy. When modeling these, the main noise components seen in the Allan analysis are considered.

Defining the real angular velocity ω_{true}, the total acceleration given by gravity $\mathbf{g}$ and the acceleration of the body $\mathbf{a}_{body}$, the real earth magnetic field $\mathbf{m}_E$, the scale factors $\mathbf{K}^a$, $\mathbf{K}^h$, and $\mathbf{K}^g$, and the respective biases $\mathbf{b}^a$, $\mathbf{b}^h$, $\mathbf{b}^g$, the output of the sensors is modeled as follows:

$$\begin{aligned} \boldsymbol{\omega} &= \mathbf{K}^g \boldsymbol{\omega}_{true} + \mathbf{b}^g + \mathbf{v}^g \\ \mathbf{a} &= \mathbf{K}^a \mathbf{C}_n^b (\mathbf{g} + \mathbf{a}_{body}) + \mathbf{b}^a + \mathbf{v}^a \\ \mathbf{h} &= \mathbf{K}^h \mathbf{C}_n^b \mathbf{m} + \mathbf{b}^h + \mathbf{v}^h \end{aligned} \tag{6.6}$$

$\mathbf{v}^g$,$\mathbf{v}^h$, y $\mathbf{v}^a$ are vectors of Gaussian noise, with average 0, and covariance matrices $\boldsymbol{\Sigma}_g$, $\boldsymbol{\Sigma}_h$, y $\boldsymbol{\Sigma}_a$.

In this case, it is assumed that all measurements and tests are carried out at a constant temperature, so that the values of the scale factors of each of the sensors are kept constant [42].

6.2.2. Filter design

With the modeling already carried out, the remaining components of the Kalman filter in question have to be determined. In order not to make excessive use of notation, the same letters are used as those used in Section 2.4. Thus, we have the state vector $\mathbf{x}_k$, state transition matrix $\mathbf{F}_k$, observation matrix $\mathbf{H}_k$, covariance of process noise $\mathbf{Q}_k$, and observation noise covariance $\mathbf{R}_k$.

The state vector is defined as the increased state between the components of the rotation quaternion, and the magnetometer and gyro bias. Equation 6.7 shows the transition of the state vector.

$$\begin{aligned} \mathbf{x}(k+1|k) &= \begin{bmatrix} \mathbf{q}_{k+1} \\ \mathbf{b}^g_{k+1} \\ \mathbf{b}^h_{k+1} \end{bmatrix} \\ &= \mathbf{F}_k \begin{bmatrix} \mathbf{q}_k \\ \mathbf{b}^g_k \\ \mathbf{b}^h_k \end{bmatrix} + \begin{bmatrix} \mathbf{w}^q_k \\ \mathbf{w}^g_k \\ \mathbf{w}^h_k \end{bmatrix} \end{aligned} \tag{6.7}$$

As seen in the previous equation, the biases are modelled as a random walk [21], which is consistent with that obtained from Allan's variance. The following relationships are also satisfied:

$$\mathbf{F}_k = \begin{bmatrix} \boldsymbol{\Phi}_k & \mathbf{0}_{4x3} & \mathbf{0}_{4x3} \\ \mathbf{0}_{3x4} & \mathbf{I}_3 & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x4} & \mathbf{0}_{3x3} & \mathbf{I}_3 \end{bmatrix} \tag{6.8}$$

$$\mathbf{Q}_k = \begin{bmatrix} \left(\frac{t_s}{2}\right)^2 \Xi_k \Sigma_g \Xi_k^t & \mathbf{0}_{4x3} & \mathbf{0}_{4x3} \\ \mathbf{0}_{3x4} & \Sigma_a & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x4} & \mathbf{0}_{3x3} & \Sigma_m \end{bmatrix} \tag{6.9}$$

Now we need to obtain bot the observation model, and the covariance matrix associated with it. From the equations 6.6, it is possible to rewrite these as follows:

$$\begin{bmatrix} \mathbf{a}_{m,k} \\ \mathbf{h}_{m,k} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_n^b(\mathbf{q}_k) & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{C}_n^b(\mathbf{q}_k) \end{bmatrix} \begin{bmatrix} -\mathbf{g} \\ \mathbf{b}^e \end{bmatrix} + \begin{bmatrix} \mathbf{v}^a \\ \mathbf{v}^h \end{bmatrix} \tag{6.10}$$

Because these equations are nonlinear, it is necessary to perform the calculation of the Jacobian matrix H, with the partial derivative calculations in [43]:

$$\mathbf{H}_k = \begin{bmatrix} \frac{\partial}{\partial \mathbf{q}} \mathbf{C}_n^b(\mathbf{q})\mathbf{g} & \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ \frac{\partial}{\partial \mathbf{q}} \mathbf{C}_n^b(\mathbf{q})\mathbf{b}^e & \mathbf{C}_n^b(\mathbf{q}) & \mathbf{0}_{3x3} \end{bmatrix} \tag{6.11}$$

Finally, the covariance of the observation noise can be seen below:

$$\mathbf{R} = \begin{bmatrix} \Sigma_a & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \Sigma_h \end{bmatrix} \tag{6.12}$$

6.2.3. Data reprocessing

The effects of magnetic disturbances do not tend to be taken too much into consideration in the literature. However, in the context of PDR, smartphones are regularly exposed to ferromagnetic objects, which tend to deliver bad results [44].

Among those authors who do consider this factor, some adjust the value of the covariance matrix associated with magnetic measurements to infinity when the following relationship is true:

$$\text{abs}(||\text{mag}_S|| - ||\text{mag}_E||) \leq \gamma_{mag} \tag{6.13}$$

This means that at the time that there is a difference greater than γ_{mag} between the norm of the Earth's magnetic field, and the norm of the magnetic measurements of the device, the covariance matrix is adjusted.

Using the aforementioned information as a basis, in [25], it is stated that once the magnetic difference is greater than γ_{mag} the magnetic measurements are already too noisy, and therefore, the data obtained from a past time is incorrect, so it is necessary to correct them. Below is a modification of the algorithm proposed by Fourati et. al, in order to verify its behavior in conjunction with the Kalman Filter already developed.

Algoritmo 1: Limiting the impact of magnetic disturbances

Data: **f(gyr,acc,mag,dT,mag_update)** is an Extended Kalman Filter, where mag_update is a boolean value which indicates whether the magnetic measurements are used

Data: **vec_states_and_values** is a vector which keeps the record of the state of the filter on a sliding window.

Data: **last_mag_pert** is the time that has passed since a magnetic disturbance was detected.

Detección de perturbaciones magnéticas

$\text{mag\_update}_k = \text{abs}(||\text{mag}_S|| - ||\text{mag}_E||) \leq \gamma_{mag}$

Minimum durations

if mag_update_k **then**
    last_mag_pert = last_mag_pert + dT
    if $last\_mag\_pert < t_{mag,\ no\ pert}$ **then**
        mag_update_k = false
    end
else
    last_mag_pert=0
end

Reprocessing the latest data without magnetic data

if $!mag\_update_{k-1}$ **then**
    if mag_update_k **then**
        f.setState(vec_states_and_value.first)
        foreach *element e in vec_states_and_values* **do**
            f(e.gyr, e.acc, e.mag, e.dT, false)
        end
    end
end

This algorithm has two main components: The first part associated with meeting the minimum durations, aims to ensure that there has been a certain minimum amount of time ($t_{\text{mag, not pert}}$) in which there have been no magnetic disturbances. On the other hand, the following part is responsible for reprocessing the last N data, but without using the magnetic component data.

6.3. Results and model comparison

When talking about the orientation of an object, saying that it is 370 ° with respect to the North, is the same as saying that it is 10 ° with respect to the North. This is because the rotations implicitly have the $mod 2\pi rad$ operator associated. Therefore, the use of circular graphs is a must, where the temporal component is represented on the radial axis, while the orientation component is represented on the angular axis.

From now on, the noise values of sensors are adjusted to those indicated by the their datasheets, but multiplied by 2.5 [43], product of the possible deterioration in time.

The routes associated with the test of orientation models already mentioned in section 3.3.2 are used. Afterward, the extended Kalman filter is applied in the same way as explained in Section 2.4. In particular, walks on route 1, and 3 are shown, these being the most complicated, and the simplest, respectively.

Just so we can check how the state transition model works, the estimates are made setting the Kalman gain to 0. Also, using a quaternion initialization with the TRIAD algorithm presented in cite triad , the results of this can be seen in Figures 6.7 and 6.8.

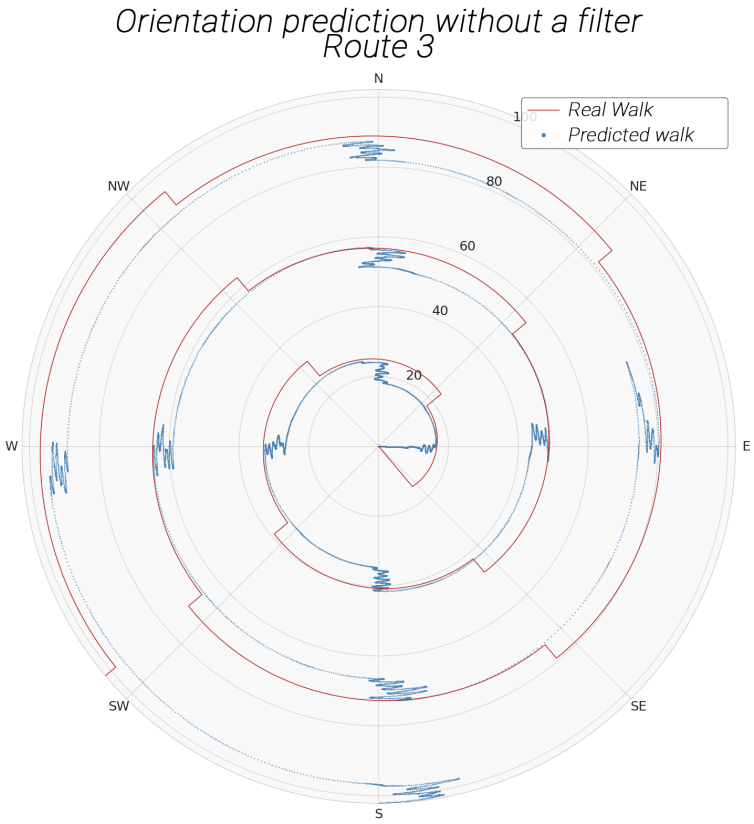


Figure 6.7: Result of orientation estimate for a walk on Route 3, using only the prediction model

When using only the prediction model, only the gyroscope data is used. Doing this, it is impossible to have a reference point with respect to the orientation, which was previously given by the magnetometer data. This leads to obtaining results which represent the rotations of the individual, and not the absolute orientation. This is reflected in the figures, where although it is noted that the rotation effects are being captured correctly through the gyroscope, it is necessary to have a rotation of the predicted orientation to get to the real orientation.

Using the Kalman filter already developed, the results of walks for routes 1 and 3 are shown in

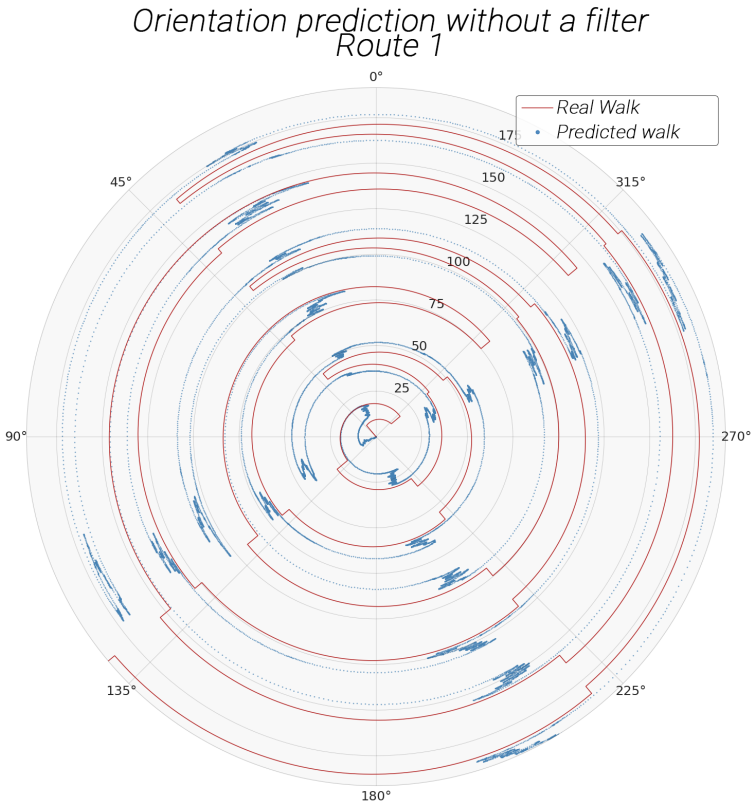


Figure 6.8: Result of orientation estimate for a walk on Route 1, using only the prediction model

Figures 6.10 and 6.9.

Starting with a difference of 90°, the orientation estimate in some cases can be good, but in others, it has errors close to 40°. While this may seem odd because the extended Kalman filter tries to solve the problems associated with sensor noises, there is a very important point to consider: measurements are being made in an environment with high variance in the magnitude of the adjacent magnetic field, as shown in Figure 6.6. This means that the covariance matrix of measurements varies depending on the position of the person, not being a factor considered in the common Kalman filter.

The problem that the noise covariance matrix is time and space-variant should be solved with the use of data reprocessing and the extended Kalman filter. Figures 6.11 and 6.12 show the results of the respective estimates.

In cases where there is data reprocessing, the vector of estimated data has a vector of Boolean values associated. These have the purpose of indicating whether or not data reprocessing was performed at a certain point.

In the graphs of Figures 6.11 and 6.12, it is possible to notice that as time passes, the reprocessing of data allows more and more reliable estimates to be generated. This is due to the fact that this algorithm starts once the magnetometer and accelerometer data are reliable enough to be used to update the prediction.

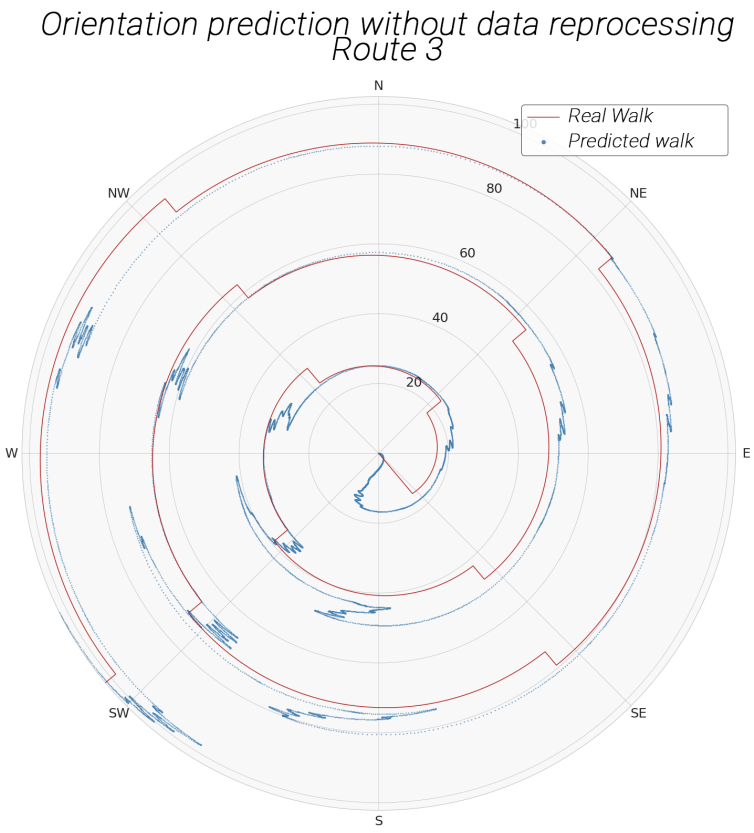


Figure 6.9: Result of the orientation estimate for a walk on Route 3. Use of the Common Extended Kalman filter.

Table 6.1: Summary of results for orientation estimation.

	EKF		EKF reprocessed	
	Difference mean (°)	**Standard deviation (°)**	**Difference mean (°)**	**Standard deviation (°)**
Route 1	12.3	0.36	8.1	0.31
Route 2	14.2	0.54	7.8	0.32
Route 3	10.6	0.40	7.5	0.19
Route 4	10.8	0.32	7.6	0.26

As seen in Table 6.1, the use of data reprocessing improves the results of the Kalman filter substantially. This shows that maintaining constant sensor noise covariance values is not a good idea. On the other hand, adjusting these to very high values when there are magnetic disturbances allows us to use only the gyroscope data, mitigating the estimation errors.

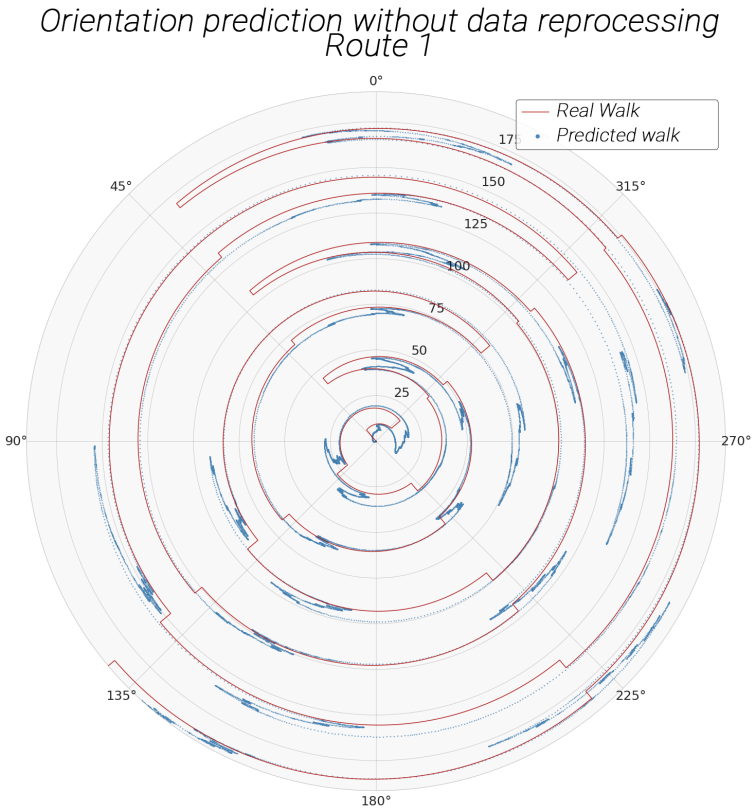


Figure 6.10: Result of the orientation estimate for a walk on Route 1. Use of the Common Extended Kalman filter.

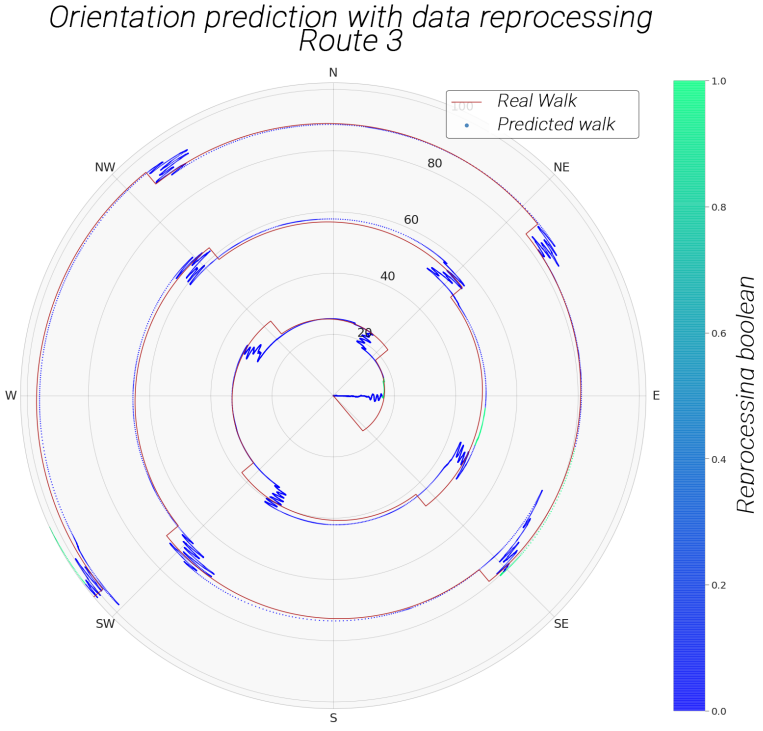


Figure 6.11: Result of the orientation estimate for a walk on Route 3. Use of Extended Kalman filter with data reprocessing.

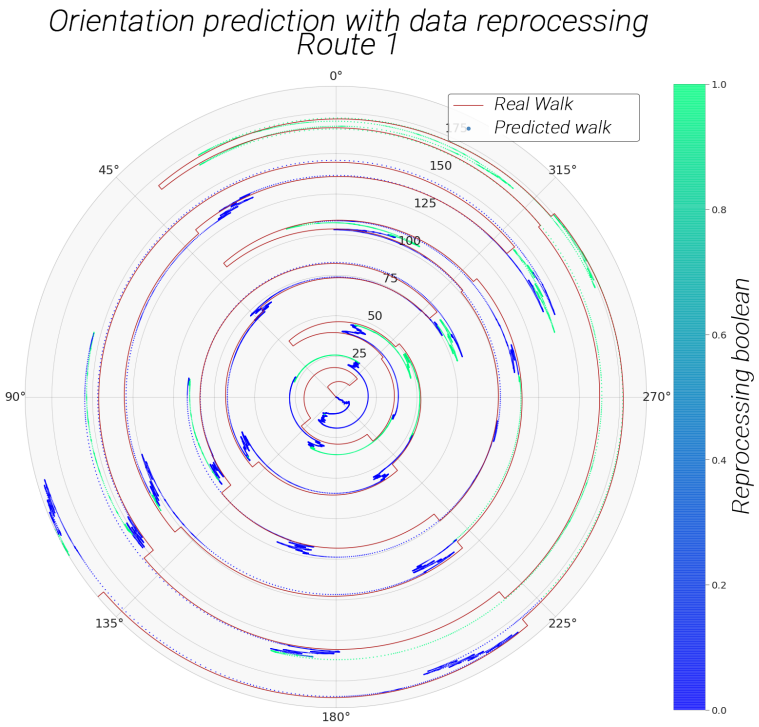


Figure 6.12: Result of the orientation estimate for a walk on Route 3. Use of Extended Kalman filter with data reprocessing.

Chapter 7

Pedestrian Dead Reckoning model

Having each of the parts associated with the PDR system, it is possible to put these together to create the complete model. Using the Equation to update the position of the person at each detection of a step, we get the results shown in Figures 7.1, 7.2, and 7.3.

It should be remembered that a system whose new state depends solely on the previous one plus a specific value is being used. Regardless of how good the orientation estimators are, it always ends up integrating an error. In particular, for the generated model, the gyro drift factor is being integrated.

While it is impossible to eliminate this source of noise completely, it is possible to make plausible assumptions, which allow reducing the integrated error to a greater extent. The first of these assumptions is that people's walks are mostly straight. The second assumption is that depending on the position of the person in an enclosure, most of the walks are dependant on the layout of the enclosure. In other words, people tend to walk in the same direction as the aisles, not in the direction perpendicular to them, unless it is for the purpose of crossing doors.

Hence, a system of *Zero Angular Rate Update* (ZARU) is added to the present model, which assumes that there must be at least a minimum variation in the angular velocity data, or else, the orientation change is set to zero. As a second factor, the number of possible orientations that a person can walk is limited depending on the specifications and the map used. Particularly for this case, it is considered that the person in question can walk along 8 different orientations, which are equally-spaced, and fit the aisles of the facilities where the measurements were taken. The results of these two aggregated factors may be seen in Figures 7.4, 7.5, and 7.6.

In the case of the model without reprocessing, an average of 23,521 seconds of data is predicted for every second that passes, with a standard deviation of 1.09 [s]. In the case with data reprocessing, about 15.88 s are predicted for each sec that passes, with a standard deviation of 3.40 s, which is because the processing time is dependent on whether there is data to reprocess, or not. This demonstrates that the present system is possible to implement in an online system context.

Figure 7.1: PDR system result for Route 1 in CNR, Pisa. No orientation cache.

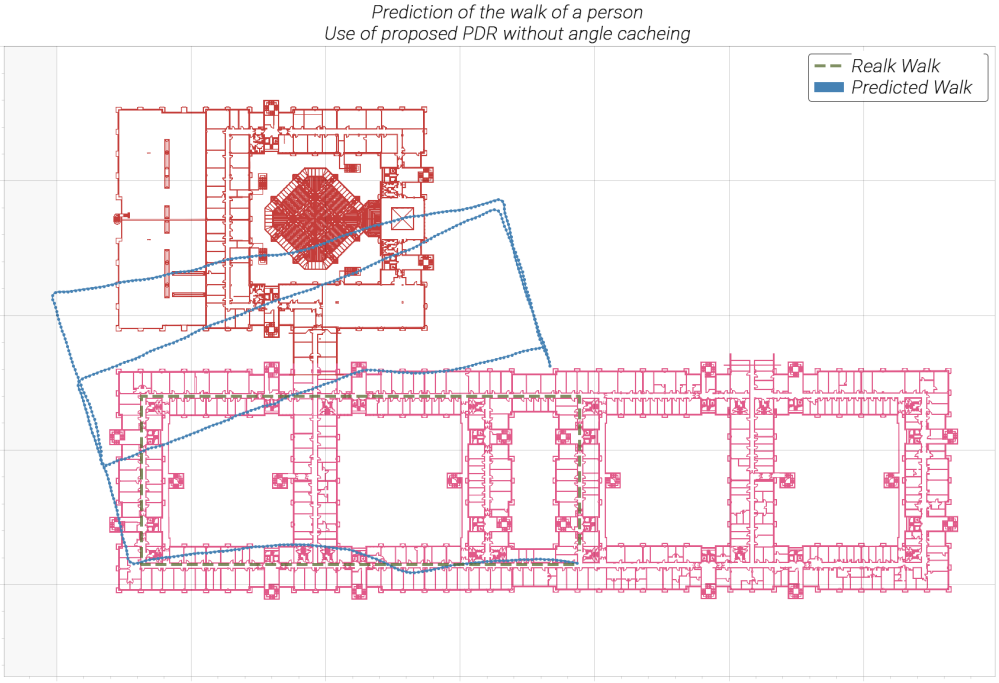


Figure 7.2: PDR system result for Route 2 in CNR, Pisa. No orientation cache.

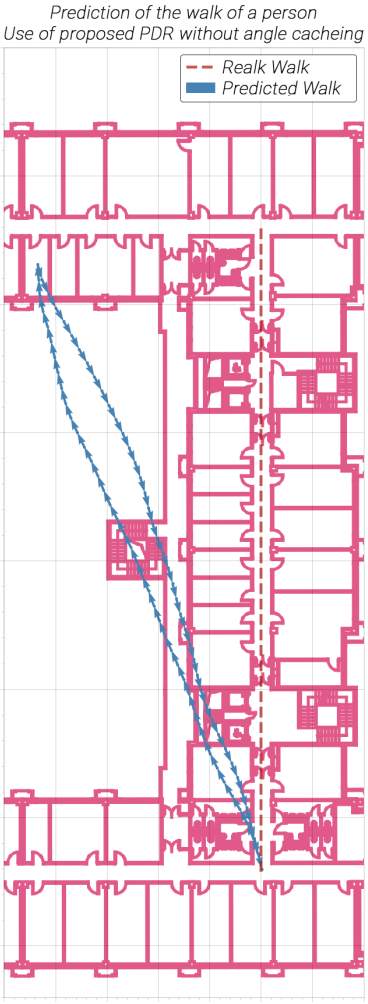


Figure 7.3: PDR system result for Route 3 in CNR, Pisa. No targeting cache.

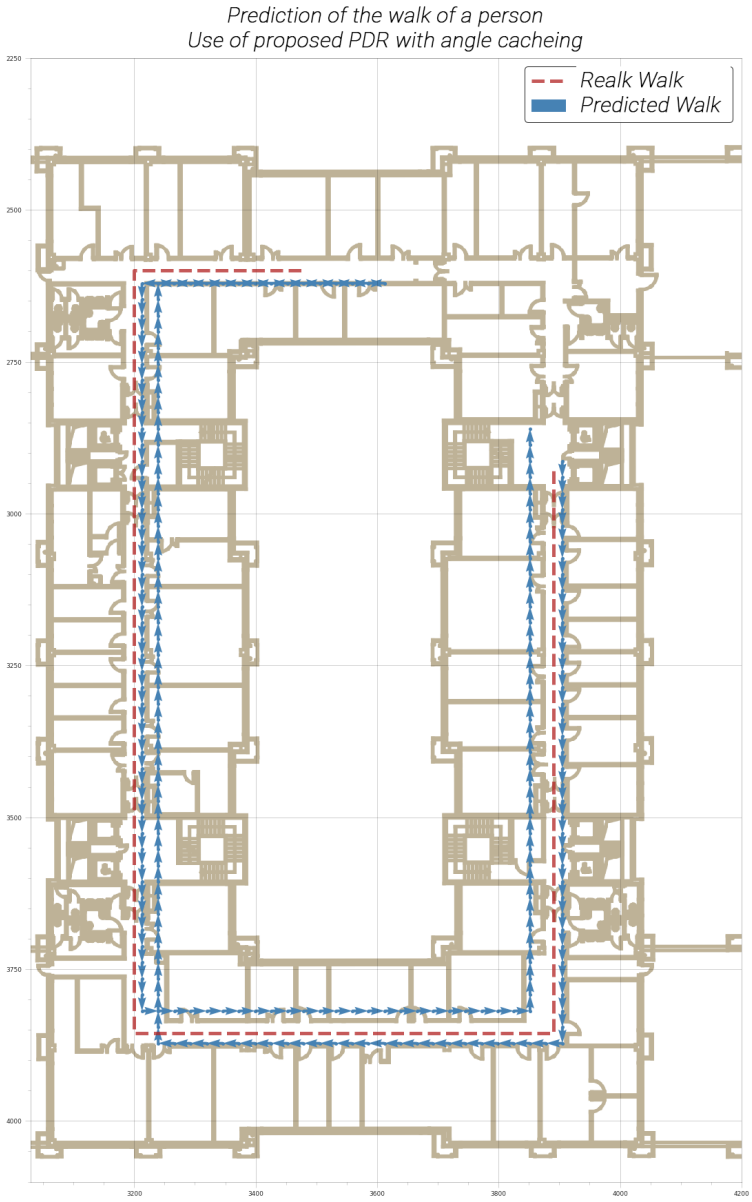


Figure 7.4: PDR system result for Route 1 in CNR, Pisa. Orientation cache used.

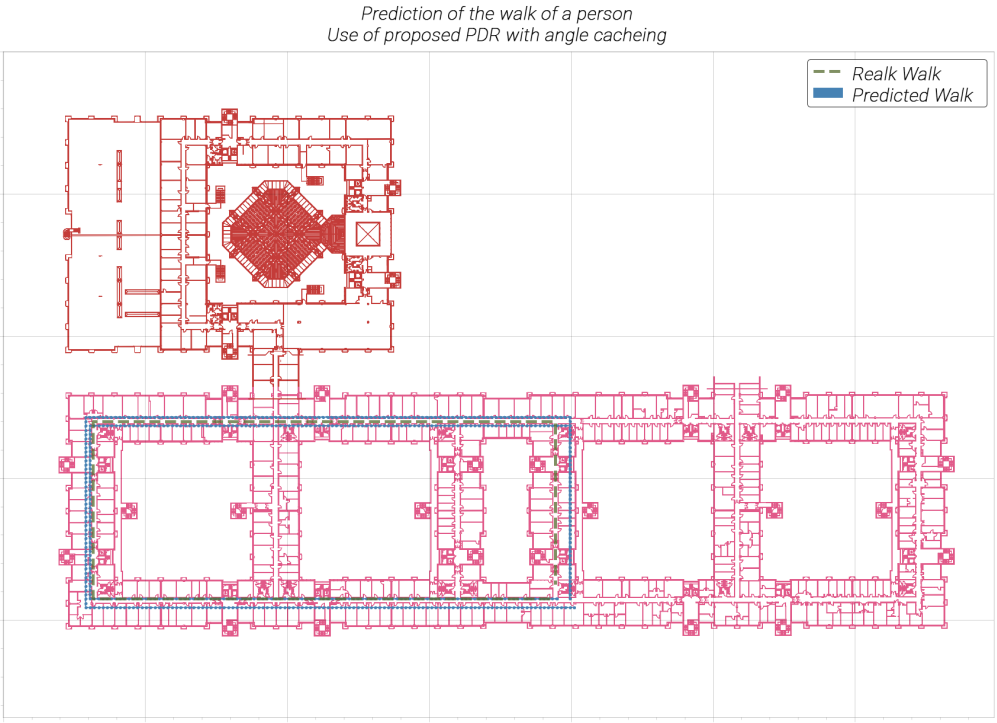


Figure 7.5: PDR system result for Route 1 in CNR, Pisa. Orientation cache used.

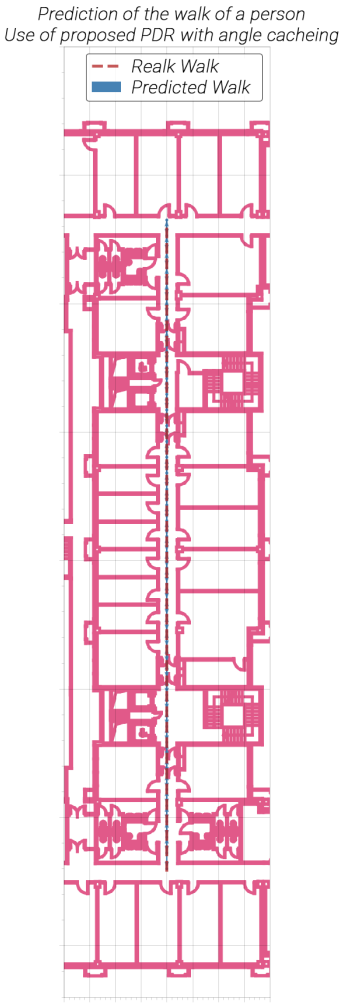


Figure 7.6: PDR system result for Route 1 in CNR, Pisa. Orientation cache used.

Chapter 8

Final Observations

8.1. Conclusions

The work done includes both the formulation and the complete implementation of a Pedestrian Dead Reckoning based system using the accelerometer, gyroscope, and magnetometer sensors of a mobile device. To do this, the results of each of the sections associated with this process were analyzed: step detection, step length estimation, and orientation estimation.

The greatest difficulty of this work is the high uncertainty that is generated around the output data of the sensors. Throughout the thesis, it was seen that the use of low-cost sensors, as in the case of smartphones, introduces a large amount of noise, so it has to be processed from different points of view.

In the first stage of step detection, classical methods of signal analysis and processing were used to achieve that goal. On the other hand, the fact that they are classical methods does not mean that they are simple. It could be seen that for a bad filter selection, the results of the detections can vary considerably. In addition, the use of an Adaptive Jerk Pace Buffer system allowed us to generate results at levels close to those of the state-of-art.

State-of-the-art methods were used to achieve step length estimation. In particular, the use of the LSTM-DAE network made it possible to exceed the classic estimation models greatly. This model, along with hyperparametric optimization, made it possible to generate a system which, being robust enough against different types of walks, allowed us to obtain results with a high level of reliability.

The last step that had to be done, corresponding to the estimation of the orientation, is constantly considered the most difficult of them all. This is due to the difficulty of mathematical modeling, sources of uncertainty such as magnetic distortions, and complications with reference systems. When using the Extended Kalman Filter on its own, no desirable results are obtained in cases with high magnetic disturbance, which is why the use of a disturbance detector is a fundamental point to be fulfilled. This is even more true in places where orientation estimation tests were performed (Santiago, Chile), where buildings have more metal structures than normal, due to the constant earthquakes in the country.

Considering the results as a whole, these are at the level of being able to be used for online

positioning systems. In addition, we were able to see that the use of the orientation cache based on the map grid used, generating substantial improvements in the positioning results.

It is important to remember that PDR corresponds to a system which, independently of the used model, ends up integrating errors. That is why, at present, most of the complete indoor positioning systems make use of both PDR and a system of bounded results. Within the latter, Wi-Fi tends to be used with fingerprinting, trilateration, or multilateration with the use of Bluetooth beacons.

The presented model was used in the previously mentioned IPIN 2019 competition held in Pisa, Italy. Integrating the proposed model in this thesis with a probabilistic fingerprinting estimator which uses Wi-Fi, the joint model was able to get fourth place out of twelve teams, all of them using state of the art proposals. This shows that the present work is able to be compete with other state of the art models.

8.2. Future work

Although the good behavior of the designed PDR model was demonstrated, it is possible to improve it in certain aspects, both by using the geometry of the map, and the model itself.

We should remember that the geometry of the enclosure where measurements are taken, is used at the time of caching over a maximum limit of orientations, these being the main aisles of the indoor site. Although tests were carried out in corridors, stairs, and outside, the constant entry and exit of rooms was not tested. This case may be common, depending on the person that uses the model. It is at this point where some inconveniences can be generated when considering the walls of the enclosure, since it should not be possible to make a transition between the node on one side of the wall to another one on the other side of the same wall. Figure 8.1 shows two cases of the aforementioned, indicating the possibility of collision with a wall, without any other possibility, and collision with one of these, being able to reprocess to reach a more probable result [45]. A possible future change may be done taking this factor into consideration.

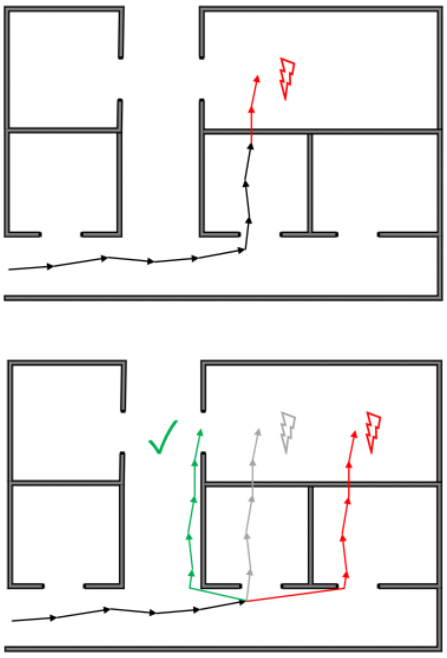


Figure 8.1: Reprocessing the trajectories as a product of collisions with walls.

Acronyms

ANN Artificial Neural Network.

DL Deep Learning.

DNA Denoising Autoencoder.

EKF Extended Kalman Filter.

FFT Fast Fourier Transform.

GRU Gated Recurrent Unit.

IPIN Indoor Positioning and Indoor Navigation.

KF Kalman Filter.

LSTM Long-Short Term Memory.

MEMS Micro-Electromechanical System.

MSE Mean Squared Error.

PDR Pedestrian Dead Reckoning.

RNN Recurrent Neural Network.

SLE Step Length Estimation.

ZARU Zero Angular Rate Update.

ZUPT Zero-Update Velocity.

Bibliography

[1] M. Weiser, "The computer for the 21st century," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, pp. 3–11, July 1999.

[2] F. J. Bruwer and M. J. Booysen, "Comparison of gps and mems support for smartphone-based driver behavior monitoring," in *2015 IEEE Symposium Series on Computational Intelligence*, (Cape Town, South Africa), pp. 434–441, December 2015.

[3] M. Del Rosario, S. Redmond, and N. Lovell, "Tracking the evolution of smartphone sensing for monitoring human movement," *Sensors (Basel)*, vol. 15, pp. 18901–18933, August 2015.

[4] V. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella, "Gyroscope technology and applications: A review in the industrial perspective," *Sensors*, vol. 17, pp. 2284–2306, October 2017.

[5] Y. Cai, Y. Zhao, X. Ding, and J. Fennelly, "Magnetometer basics for mobile phone applications," *Electron. Prod. (Garden City, New York)*, vol. 54, February 2012.

[6] H. Hou, "Modeling inertial sensors errors using allan variance," Master's thesis, University of Calgary, Calgary, Alberta, 9 2004.

[7] "IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros," *IEEE Std 952-1997*, pp. 1–84, Feb 1998.

[8] L. B. Pupo, "Characterization of errors and noises in mems inertial sensors using allan variance method," Master's thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2016.

[9] H. Hou and N. El-Sheimy, "Inertial sensors errors modeling using allan variance," *Proc. ION GPS/GNSS*, pp. 2860–2867, September 2003.

[10] "Allan variance: Noise analysis for gyroscopes," *Freescale Semiconductor, Inc.*, vol. Document Number: AN5087, 2 2015.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2nd ed., 1998.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, Jan 2015.

[13] J. Kuang, X. Niu, and X. Chen, "Robust Pedestrian Dead Reckoning Based on MEMS-IMU for Smartphones," *Sensors (Basel)*, vol. 18, pp. 1391–1398, May 2018.

[14] S. Beauregard and H. Haas, "Pedestrian dead reckoning: A basis for personal positioning," pp. 27–36, January 2006.

[15] S. Y. Park, S. J. Heo, and C. G. Park, "Accelerometer-based smartphone step detection using machine learning technique," in *2017 International Electrical Engineering Congress*

(iEECON), pp. 1–4, March 2017.

[16] A. Martinelli, H. Gao, P. D. Groves, and S. Morosi, "Probabilistic context-aware step length estimation for pedestrian dead reckoning," *IEEE Sensors Journal*, vol. 18, pp. 1600–1611, Feb 2018.

[17] N. Ho, P. Truong, and G.-M. Jeong, "Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone," *Sensors*, vol. 16, p. 1423, September 2016.

[18] H. Weinberg, "Using the adxl202 in pedometer and personal navigation applications," *Analog Devices*, 2002.

[19] J. Won Kim, H. Jin Jang, D. Hwang, and C. Park, "A step, stride and heading determination for the pedestrian navigation system," *Journal of Global Positioning Systems*, vol. 3, pp. 273–279, 12 2004.

[20] Q. Tian, Z. Salcic, K. I. Wang, and Y. Pan, "A multi-mode dead reckoning system for pedestrian tracking using smartphones," *IEEE Sensors Journal*, vol. 16, pp. 2079–2093, April 2016.

[21] A. Sabatini, *Inertial Sensing in Biomechanics: A Survey of Computational Techniques Bridging Motion Analysis and Personal Navigation*, pp. 70–100. R. Begg, M. Palaniswami (Eds.), January 2008.

[22] D. Roetenberg, H. Luinge, C. Baten, and P. Veltink, "Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 13, pp. 395 – 405, October 2005.

[23] A. R. Jiménez, F. Seco, and J. Torres-Sospedra, "Tools for smartphone multi-sensor data registration and gt mapping for positioning applications," in *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, Sep. 2019.

[24] M. Cui, Y. Huang, W. Wang, and H. Cao, "Mems gyroscope temperature compensation based on drive mode vibration characteristic control," *Micromachines*, vol. 10, p. 248, April 2019.

[25] T. Michel, P. Genevès, H. Fourati, and N. Layaïda, "On attitude estimation with smartphones," in *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 267–275, March 2017.

[26] S. H. Shin, M. S. Lee, C. G. Park, and H. S. Hong, "Pedestrian dead reckoning system with phone location awareness algorithm," in *IEEE/ION Position, Location and Navigation Symposium*, pp. 97–101, May 2010.

[27] S. Shin and C. Park, "Adaptive step length estimation algorithm using optimal parameters and movement status awareness," *Medical engineering physics*, vol. 33, pp. 1064–71, May 2011.

[28] L. E. Díez, A. Bahillo, J. Otegui, and T. Otim, "Step length estimation methods based on inertial sensors: A review," *IEEE Sensors Journal*, vol. 18, pp. 6908–6926, Sep. 2018.

[29] Q. Wang, L. Ye, H. Luo, A. Men, F. Zhao, and Y. Huang, "Pedestrian stride-length estimation based on lstm and denoising autoencoders," *Sensors*, vol. 19, p. 840, February 2019.

[30] J. Scarlett, "Enhancing the performance of pedometers using a single accelerometer," *Application Note, Analog Devices*, vol. 41, pp. 1–16, March 2007.

[31] H. BLACK, "A passive system for determining the attitude of a satellite," *AIAA Journal*,

vol. 2, p. 14, August 1963.

[32] M. D. SHUSTER and S. D. OH, "Three-axis attitude determination from vector observations," *Journal of Guidance and Control*, vol. 4, no. 1, pp. 70–77, 1981.

[33] T. Harada, H. Uchino, T. Mori, and T. Sato, "Portable absolute orientation estimation device with wireless network under accelerated situation," vol. 2, pp. 1412 – 1417 Vol.2, January 2004.

[34] H. Rehbinder and X. Hu, "Drift-free attitude estimation for accelerated rigid bodies," vol. 40, pp. 4244 – 4249 vol.4, February 2001.

[35] J. Marins, X. Yun, E. Bachmann, R. Mcghee, and M. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," *IEEE International Conference on Intelligent Robots and Systems*, vol. 4, January 2002.

[36] A. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *Biomedical Engineering, IEEE Transactions on*, vol. 53, pp. 1346 – 1356, August 2006.

[37] J. Crassidis, "Unscented filtering for spacecraft attitude estimation," *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, vol. 26, July 2003.

[38] H. Fourati, N. Manamanni, L. Afilal, and Y. Handrich, "A nonlinear filtering approach for the attitude and dynamic body acceleration estimation based on inertial and magnetic sensors: Bio-logging application," vol. 11, February 2011.

[39] P. Bartz, "Magnetic compass calibration, US Patent," October 1969.

[40] I. Frosio, F. Pedersini, and N. A. Borghese, "Autocalibration of mems accelerometers," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, pp. 2034–2041, June 2009.

[41] D. M. Henderson, "Euler angles, quaternions, and transformation matrices," tech. rep., NASA (National Aeronautics and Space Administration), July 1977.

[42] R. Smith, A. Frost, and P. Probert, "Aspects of heading determination via fusion of inclinometer and magnetometer data," in *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97*, pp. 739–744, July 1997.

[43] A. Sabatini, "Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation," *Sensors (Basel, Switzerland)*, vol. 11, pp. 9182–206, December 2011.

[44] M. H. Afzal, V. Renaudin, and G. Lachapelle, "Magnetic field based heading estimation for pedestrian navigation environments," in *2011 International Conference on Indoor Positioning and Indoor Navigation*, pp. 1–10, 2011.

[45] F. Hölzke, J. Wolff, and C. Haubelt, "Improving pedestrian dead reckoning using likely paths and backtracking for mobile devices," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 273–278, March 2019.

Appendix

Routes used for the final PDR model

The following figures correspond to the routes used during the IPIN 2019 competition held in Italy, Pisa. These were used to test the final model proposed in the current thesis.

Figure 8.1: Route number 1 used in the IPIN 2019 competition, used to test the final PDR Model

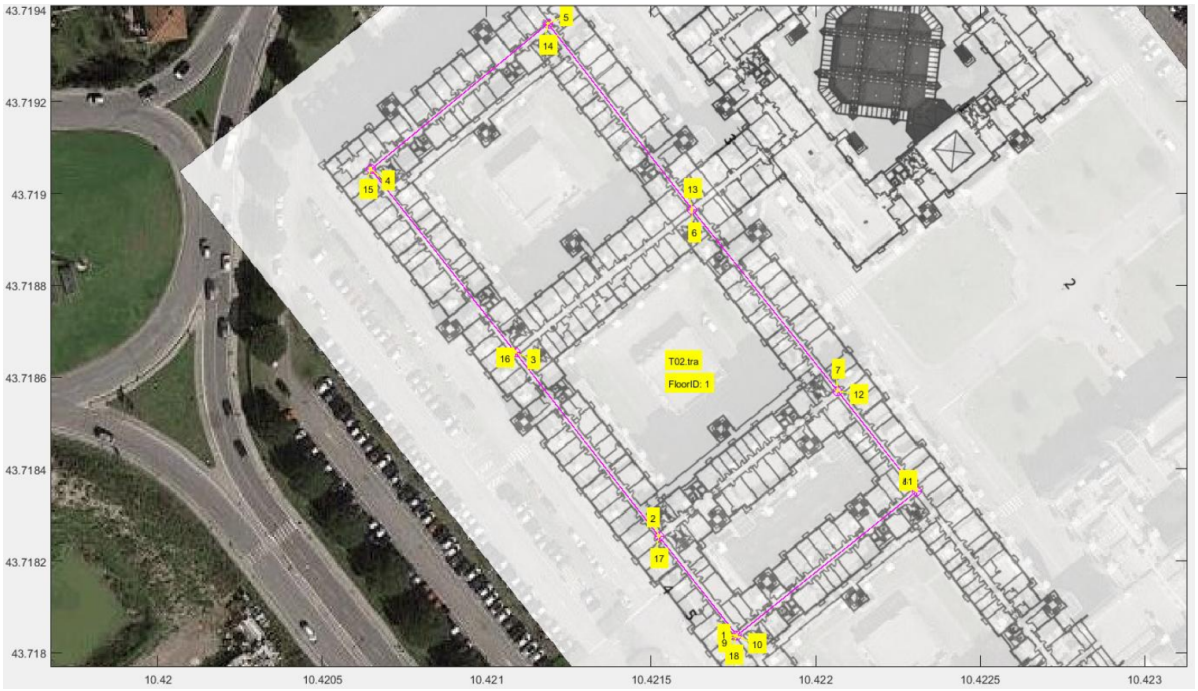


Figure 8.2: Route number 2 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.3: Route number 3 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.4: Route number 4 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.5: Route number 5 used in the IPIN 2019 competition, used to test the final PDR Model

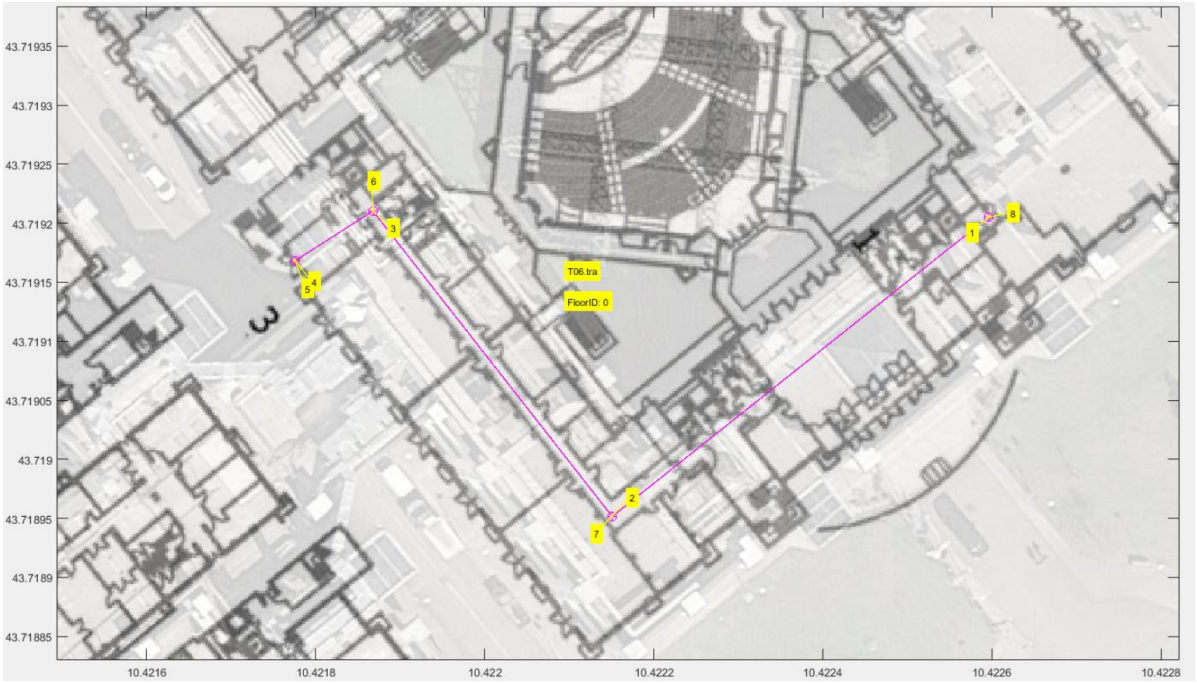


Figure 8.6: Route number 6 used in the IPIN 2019 competition, used to test the final PDR Model

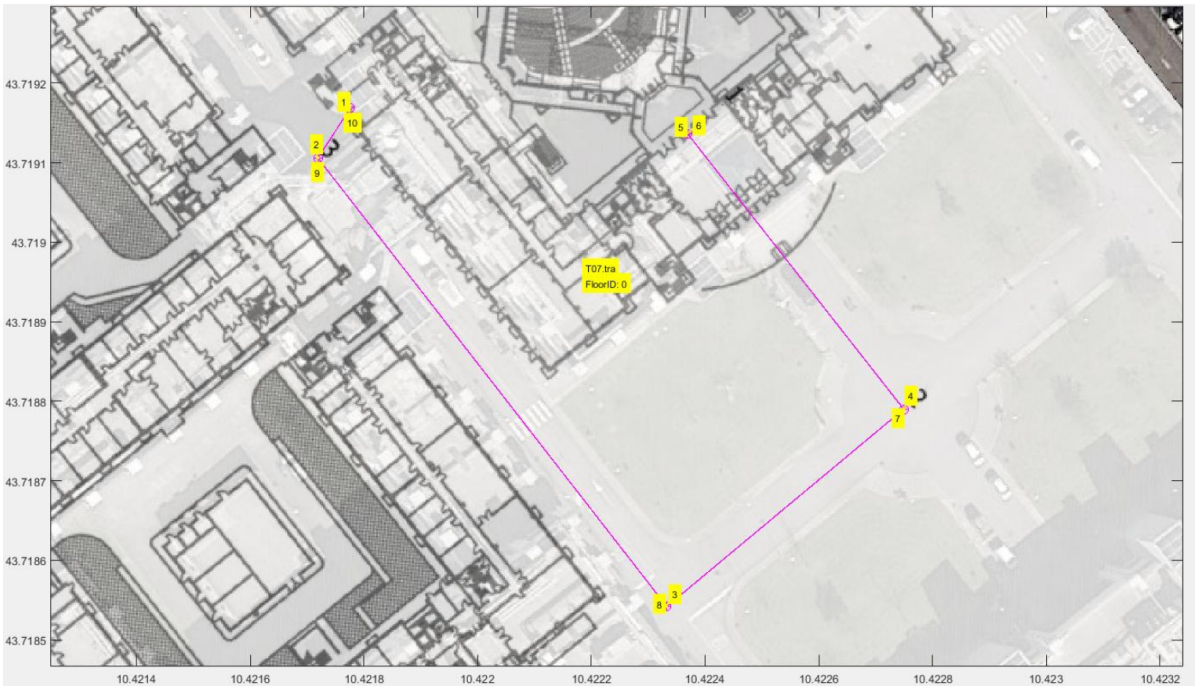


Figure 8.7: Route number 7 used in the IPIN 2019 competition, used to test the final PDR Model

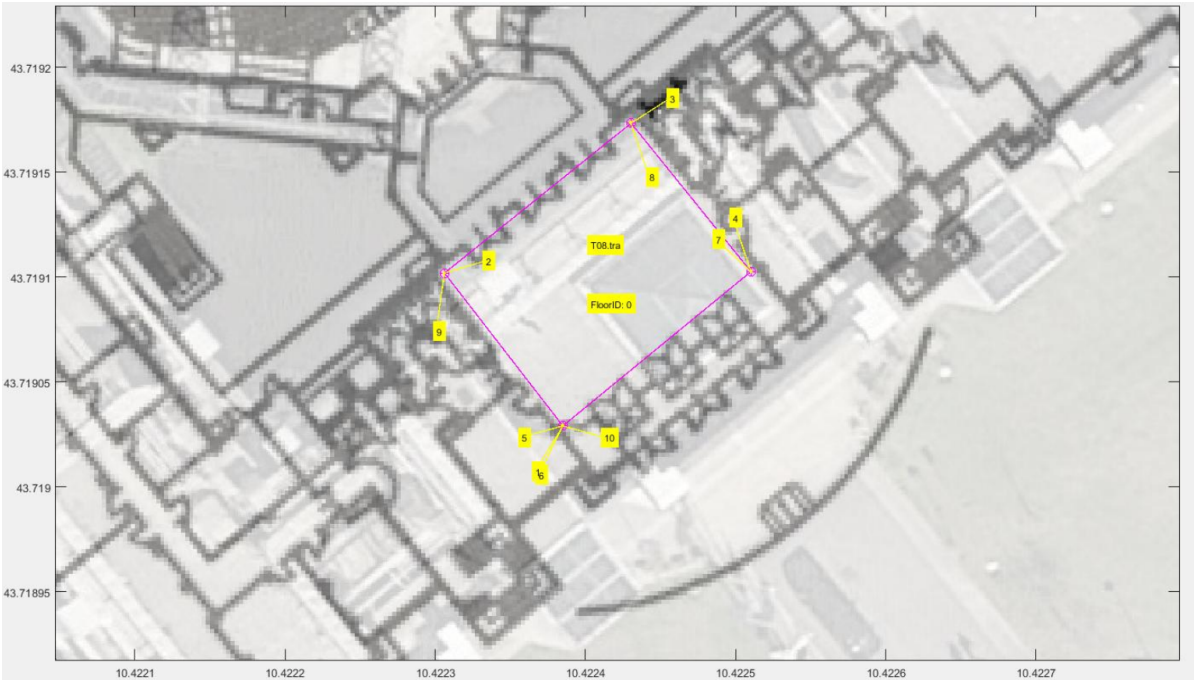


Figure 8.8: Route number 8 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.9: Route number 9 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.10: Route number 10 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.11: Route number 11 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.12: Route number 12 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.13: Route number 13 used in the IPIN 2019 competition, used to test the final PDR Model

Figure 8.14: Route number 14 used in the IPIN 2019 competition, used to test the final PDR Model

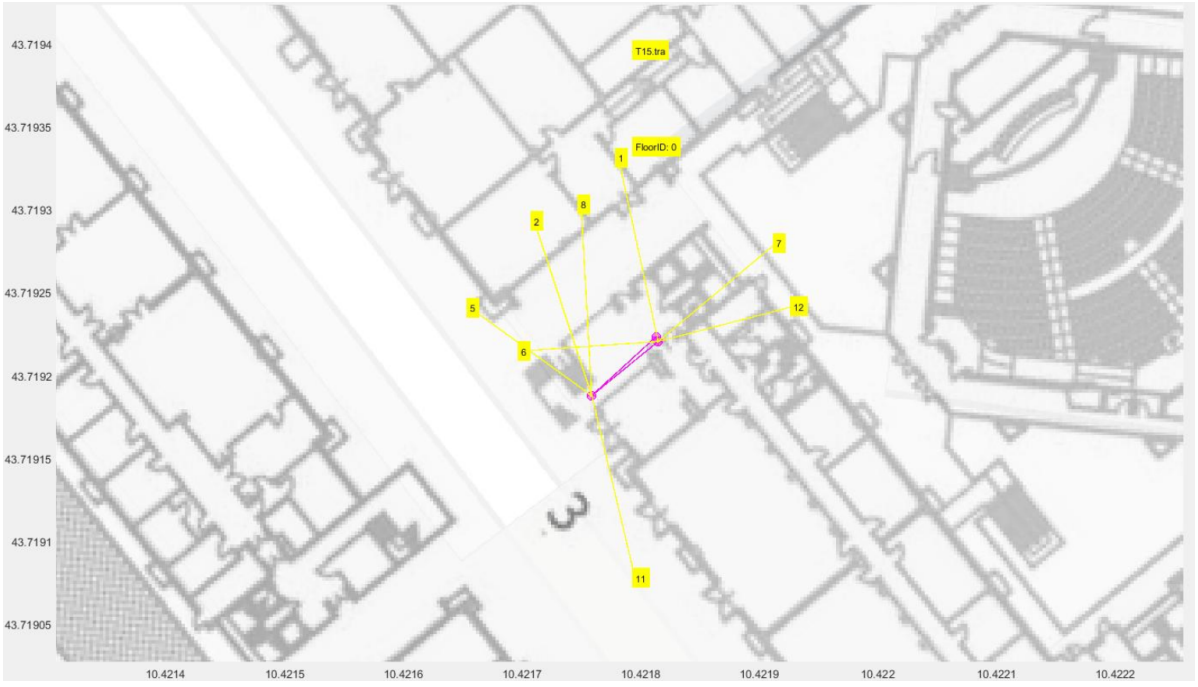


Figure 8.15: Route number 15 used in the IPIN 2019 competition, used to test the final PDR Model