Faculdade de Engenharia da Universidade do Porto

Machine Learning Improvements to Human Motion Tracking

Pedro Manuel dos Santos Ribeiro

Dissertation

Mestrado Integrado em Bioengenharia - Engenharia Biomédica

Supervisor: Jaime dos Santos Cardoso, PhD, FEUP | INESC TEC

Supervisor: Ana Clara Matos, MSc, SWORD HEALTH

Supervisor: Pedro Henrique Santos, MSc, SWORD HEALTH

June 28, 2020

Machine Learning Improvements to Human Motion Tracking

Pedro Manuel dos Santos Ribeiro

Mestrado Integrado em Bioengenharia - Engenharia Biomédica

June 28, 2020

Resumo

O *tracking* do movimento humano consiste na captação de dados qualitativos ou quantitativos relativos ao movimento humano. Atualmente, para *tracking* de movimento, as soluções baseadas em *Inertial Measurement Units* (*IMUs*) são extremamente populares devido à sua portabilidade, ao facto de permitirem deteção contínua de movimento e à sua capacidade de providenciar *feedback* em tempo real.

Com os recentes desenvolvimentos na área dos micro-sensores, um foco especial tem sido colocado no *tracking* do movimento humano usando *IMU*s baseados em sistemas micro-eletromecânicos (*MEMS*). Este tipo de *IMU* é pequeno, leve, de baixo custo e possui um baixo consumo de energia. A maior desvantagem associada a este género de *IMU* é o facto dos seus sensores possuírem uma performance mais baixa quando comparados com sensores inerciais tradicionais de gama superior. Sendo assim, o cálculo de orientação, velocidade e posição efetuados a partir das medições inerciais destes sensores são erróneos. Melhorias no cálculo destas parâmetros traduzem-se num *tracking* de movimento humano mais preciso.

Nesta dissertação dois objetivos que visam a melhoria do *tracking* a três dimensões de *IMU*s conectados a diferentes segmentos corporais foram definidos: deteção de momentos em que a velocidade do IMU e respetivo segmento corporal é zero e correção das estimativas das translações efetuadas pelos mesmos nos momentos em que existe movimento. Técnicas de *Machine Learning* (*ML*) foram utilizadas para atingir ambos os objetivos.

A identificação de períodos estacionários foi realizada com elevada exatidão o que permitiu ajustar as estimativas de velocidade durante esses períodos e a implementação de técnicas de correção de *drift* durante períodos de movimento. O ajuste de velocidade permitiu a correção do deslocamento calculado através da dupla integração dos valores de aceleração o que possibilitou a melhoria da estimativa de translação dos diferentes segmentos corporais.

Em seguida, esses métodos foram combinados com modelos de *ML* de regressão capazes de estimar os deslocamentos durante os períodos de movimento. Estes modelos de regressão não apresentaram melhorias significativas quando comparados com a dupla integração das acelerações lineares combinada com a remoção de *drift*. Finalmente, um modelo que combinou simultaneamente a identificação de momentos estacionários e a estimativa de translações 3D entre esses momentos foi proposto com resultados bastante promissores em ambas as tarefas.

Os resultados deste trabalho mostram que as técnicas de *ML* podem ser efetivamente usadas para melhorar a estimativa de translações 3D de diferentes segmentos corporais durante processos de *tracking* de movimento humano associado a sensores inerciais.

Abstract

Human Motion Tracking is the process of collecting qualitative or quantitative data from human movement. For motion tracking, IMU-based methods are one of the most exciting methodologies, since these are portable, allow continuous motion detection and can provide real-time feedback. Due to these characteristics, the use of inertial sensors has become particularly popular for telerehabilitation.

With recent progress in microsensor technology, a particular focus has been placed in human motion tracking using MEMS-based IMUs. These are small, light, low-cost and have a low power consumption. The principal disadvantage associated with this kind of IMU is that the performance of these sensors is lower than the traditional higher-grade inertial sensors. Therefore, the acceleration and angular rate provided by the low-grade IMUs are contaminated with errors which harm the calculus of attitude, velocity and position that are estimated using the inertial measurements. These variables are essential for conducting accurate human motion tracking.

In this work, two primary objectives were set for enhancing the tracking of the 3D position of different body segments associated with IMUs: zero-velocity detection and correction of translation estimates. ML methodologies were used to achieve both goals.

The identification of stationary periods was conducted with high accuracy allowing the adjustment of velocity estimates during those periods and the implementation of drift correction techniques during the periods of movement. The velocity adjustments permitted to correct the displacement calculated by double integration of the acceleration values leading to an improved estimate of the body segments translation.

Afterwards, these techniques were combined with ML regression models capable of estimating the displacement during periods of movement. These ML regression models did not show significant improvements when compared with the more straightforward double integration of the linear acceleration combined with drift removal for translation estimate. Finally, a model that combined the simultaneous identification of zero-velocity moments and the 3D translation estimate between those moments was proposed with promising results in both tasks.

The results of this work show that ML techniques can be effectively used for improving 3D translation estimates of different body segments during human motion tracking.

Acknowledgments

A escrita desta dissertação marca também o fim de uma das etapas mais enriquecedoras da minha vida. Tal apenas pode ser concluída com o devido reconhecimento a quem contribuiu para todo este percurso.

Ao professor Jaime Cardoso, orientador da dissertação, agradecer todo o contributo científico, preocupação e disponibilidade. Aqui lhe exprimo toda a minha gratidão.

À Clara Matos e ao Pedro Santos agradecer o auxílio incomensurável. Esta tese é tão minha quanto vossa, tal foi a vossa contribuição para a mesma. Agradecer também a toda a restante equipa da SWORD Health que tão bem me recebeu.

Aos meus pais que sempre demonstraram toda a sua confiança em mim, que fizeram com que nada me faltasse e que me ensinaram todos os princípios que hão de para sempre reger a minha vida. À minha prima, que na mesma mesa na qual escrevo hoje estes agradecimentos me ajudou há anos atrás a escolher este percurso que tão boa escolha se revelou para mim. A toda a minha restante família um muito obrigado pelo suporte constante.

Aos meus companheiros de curso, principalmente aqueles que mais proximamente acompanharam o meu trajeto e que se tornaram uma segunda família para mim: Meneses, Migas, Gu, Ana, Joana, Raquel, Álvaro, Bruno, Malafaya, Leonor, Pedro e Nataliya (ordem escolhida no random.org como nós tanto gostamos). Não "foi bom", foi incrível.

Deixo também um agradecimento geral a todos os meus amigos porque a vida não é nada sem amizades. Sou um sortudo por contar com algumas das melhores que existem.

Por fim agradecer à minha namorada, a pessoa mais importante em tudo o que faço, que funciona como um pêndulo na minha vida e que está ao meu lado em todos os momentos. Obrigado pelo apoio incondicional.

Pedro Ribeiro

"Tu próprio te despedes de ti próprio
Não és o mesmo que escreveu o verso atrás
Já estás diferente neste verso e vais com ele"

"Agora Mesmo" - Manuel Alegre

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
Dense	Densely Connected Neural Network
DL	Deep Learning
DoF	Degrees of Freedom
EKF	Extended Kalman Filter
FOG	Fibre Optic Gyroscope
IMU	Inertial Measurement Unit
LSTM	Long Short-Term Memory
INS	Inertial Navigation System
LR	Linear Regression
Log-R	Logistic Regression
MEMS	Micro-Electro-Mechanical Systems
ML	Machine Learning
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit
RLG	Ring Laser Gyroscope
RNN	Recurrent Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression
ZUPT	Zero-Velocity Update

Chapter 1

Introduction

1.1 Context

Human Motion Tracking is any procedure that tries to obtain a quantitative or qualitative measure of human movement. It has seen remarkable progress in recent years and is one of the most important tasks of human motion analysis. Examples of quantitive analysis include the measurement of biomechanical variables such as the angle between joints, or parameters that describe the gait of an individual [63, 10]. The analysis of motion aims to gather data about the human movement while a task is executed. The collected data can describe, for instance, the motion of the body centre of mass or the motion of a single body segment relative to another segment [19]. This way, the major purpose of human motion tracking systems is to generate data that represents the motion and posture of the whole human body or segments of it. [124].

Human motion analysis has been a subject of interest since the 1800s, with some authors interested in quantifying human movement [112]. In 1917, on another example of motion analysis, rotoscoping was used to create 2D animated characters from human motion captured using video [4]. Due to the high number of war lesions, during the ’50s, there was also an increased interest in studying human locomotion [106, 75].

More recently, major technological improvements have allowed the development of multiple methodologies to approach the problem. Numerous fields use motion analysis systems to capture movement and posture, creating precise human kinematic models or analysing biomechanical variables of interest [75]. These fields include applications as diverse as personal fitness, sports, entertainment, intelligent environments, security or virtual reality. [63]. In the sports context, motion analysis techniques have been used to identify specific factors during dynamic tasks that may enhance the risk of injury [46, 45, 72] or to enhance performance [6]. The tracking of human motion and the recognition of gestures can also be used as a form of interaction with computers on intelligent environments [86]. In the area of security, it can be useful to identify suspicious behaviours by analysing the movement of subjects [111]. It is also possible to mention the entertainment field, in which accurate human motion tracking methodologies are essential to generate good human-like virtual characters that can have application either in videogames and movies [25]. Lately, popular implementations include motion tracking combined with virtual or augmented reality [70].

Nowadays, some of the most interesting applications of human motion tracking are medical evaluation, monitoring people, and activity recognition. The continuous recognition of the human activity is fundamental to provide healthcare and assistance services to populations that need to be continuously monitored, like the elderly or dependent subjects [9]. Furthermore, it can be used in the rehabilitation field since it allows the rehabilitation of the patient without constant monitoring by a clinician. It can also be used as a source of detailed information about the movement of the patient. [74]

To acquire adequate motion data, the choice of the capturing motion system is fundamental [14]. It can be performed using multiple underlying tracking technologies [1],[125]:

- **Mechanical tracking** – Uses mechanical elements to sense motion. These solutions are accurate and robust. However, the sensing elements used pose a limitation to the human movement.
- **Electromagnetic tracking** – Based on the change of the electric current when in the presence of a moving magnetic field. Works using an emitter antenna and a tracker. It is portable and precise, although it is limited to the range of the antenna signal and can be affected by magnetic disturbances.
- **Optical tracking** – Uses cameras to detect movement. Can use markers or be marker-free. It is a solution with high precision. Nevertheless, the area of detection is limited.
- **Inertial tracking** – Uses inertial sensors as accelerometers and gyroscopes. These are portable and relatively accurate. However, their measurements can be affected by intricate errors.

Zhou and Hu [124] separate the existing tracking systems into:

- **Robot-aided tracking systems**: included inside the class of therapeutic robots [124]. There are several examples of robots that are used to assist certain movements of a patient while using sensors to record the patient's force, speed, acceleration or resistance. The amount of help provided to the patient by the robot is determined by the limitation of the subject in performing the movement [32, 79].
- **Visual tracking systems**: can be classified as visual-marker or marker-free, depending on whether or not the technology needs trackers linked to the body. Motion capture systems based on cameras and trackers, as Vicon [2] or Qualisys [3], have high accuracy and are usually classified as the "golden standard", but they only operate in controlled environments [9, 124]. On the other side, marker-free based motion capture systems, are a kind of system that allows good monitoring of patient movement, an acquisition of precise parameters and can be used with a relatively uncontrolled setting. However, they have a limited field of observation, low performance under natural light conditions and high computational and storage costs due to the image processing necessities [9, 63].

[1]Tracking Systems in Virtual Reality - Premo Group. `https://3dcoil.grupopremo.com/blog/tracking-systems-virtual-reality-the-best-choice/` Accessed on 04-12-2019

[2]Vicon Motion Systems. `https://www.vicon.com/` Accessed on 04-12-2019

[3]Qualisys - Motion Capture Systems. `https://www.qualisys.com/` Accessed on 28-05-2020

- **Non-visual tracking systems**: use sensors attached to the body in order to collect movement information. The sensors can be mechanical, inertial, acoustic, radio, microwave or magnetic-based [124].

In recent past, non-visual sensor-based solutions have become a common practice in ambulatory motion analysis since these are portable, can detect motion continuously – even in outdoor environments – and have a low computational cost allowing a real-time feedback [63, 114]. With recent advances in microsensor technology, low-power wireless communication and wireless sensor networks, systems using inertial sensors provide an effective and low-cost solution for motion tracking [10].

This way, lately IMU-based motion tracking is used in applications as diverse as positioning, interactions between humans and robots and rehabilitation. This popularity is proven by the increasing number of companies that sell technologies related with IMUs (e.g. XSens (Xsens Technologies B.V., Enschede, The Netherlands), Invensense (Invensense, CA, USA), Trivisio (Trivisio, Trier, Germany) and Microstrain (Lord Microstrain, VT, USA) [43].

1.2 Motivation

Inertial Measurement Units based on Micro-Electro-Mechanical Systems (MEMS-based IMUs) have become a popular solution for tracking human motion. IMUs are usually constituted by a set of three accelerometers and three gyroscopes [43]. However, these types of sensors suffer from accuracy degradation after extended periods of use due to the existence of errors like biases, scale factor imperfections, drifts, misalignments or random noise [48]. Consequently, the calculation of variables as velocity, position or orientation through direct integration of the sensor's signals is an impossible task since the estimation is unreliable after only a few seconds [43]. Frequently, a three-axis magnetometer is integrated into the IMU, however, in an indoor setting, the magnetometer signal is distorted due to the soft and hard iron effects caused by the building structure and, hence requiring an elaborated model to be functional [52, 61]. Thus, currently, there are several efforts to mitigate the errors from IMUs applied to motion tracking.

The present work was developed at SWORD Health (SWORD Health, S. A) [4]. SWORD Health is a medical devices company developing solutions that allow patients to perform autonomous rehabilitation without permanent human monitoring through the use of telerehabilitation [29]. The company developed an original electronic biofeedback method for home-based rehabilitation that uses inertial motion trackers to monitor the patient. Using the data relative to the patient motion collected with the inertial motion trackers, the system offers real-time feedback concerning the exercise quality using a mobile app. This system also contains a web-based platform that enables prescription and monitoring of the rehabilitation by the clinicians without needing to be face-to-face with the patient [29]. This way, SWORD Health developed a system known as SWORD Phoenix (Figure 1.1)that uses motion tracking and a set of defined movement rules to evaluate the patient's performance and to provide real-time feedback. The system is composed by the subsequent components [29, 28, 27]:

- **Web-based Portal**: a Web-cloud platform that includes features that can be used by clinicians as the prescription of new exercises, the edition of previously established exercises

[4] SWORD Health. `https://swordhealth.com/`. Accessed on 04-12-2019

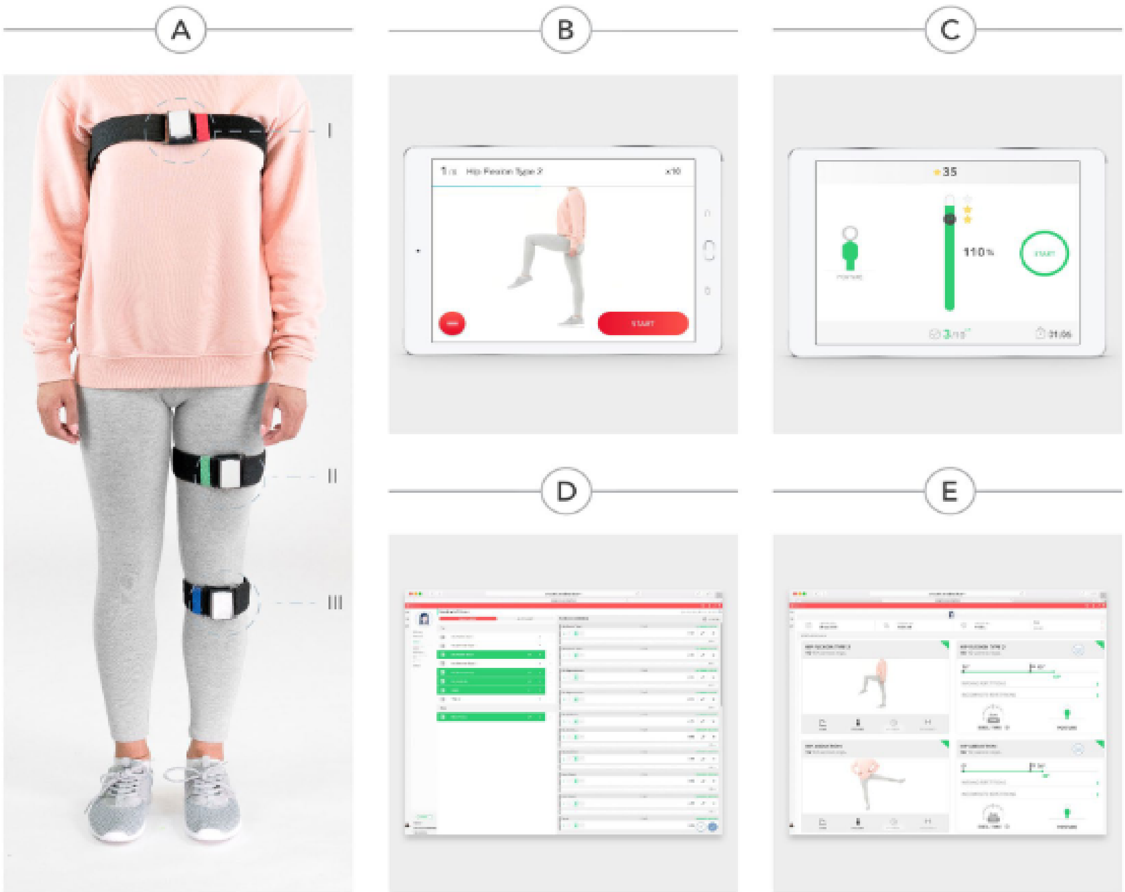


Figure 1.1: SWORD HEALTH's system components: (A)-Example of inertial motion trackers attached to the body on strategic parts (I,II,III); (B)-Mobile App: video and audio instruction's screen; (C)-Mobile App: execution screen; (D)-Web Portal: prescription screen, (E)-Web Portal: results screen, extracted from [28]

routines and the monitoring of the patient progress (Fig.1.1(D, E)). The clinician can also assess information regarding, for instance, the number of correct, wrong and incomplete repetitions as well as the angles of each repetition.

- **Mobile App**: an app that acts as an expert system using a set of physiotherapy-based rules (that evaluate the patient's motion). Besides showing instructions on how to perform rehabilitation program through video and audio commands (Fig.1.1(B), it also provides real-time feedback on the patient movement, ensuring that the exercises are correctly performed (Fig.1.1(C). The patient must complete the activity on its full amplitude, without violating any movement or posture constraint, or else the repetition is not taken into account. If any of these constraints are violated the feedback is shown to the patient.

- **Inertial motion trackers**: each tracker contains gyroscopes, accelerometers and magnetometers. Each one is positioned in strategic body parts using Velcro straps, allowing the digitization of the patient movement. The number and placement of the trackers depend on the anatomic segment that is being evaluated as well as the kind of therapy that is being performed. These trackers exchange information with a tablet, in which the mobile app is installed, via Bluetooth.

This product represents a method of digital rehabilitation that uses inertial motion trackers as a basis to human motion tracking, and that can accomplish better clinical results than conventional rehabilitation while reducing dependence on human resources [29, 28, 27].

However, as aforementioned, all the inertial motion trackers are very prone to errors and consequently can offer a wrong estimate of the patient motion. These errors can lead to a wrong estimation of the position, velocity and orientation of a certain body segment posing a threat to a correctly performed rehabilitation.

1.3 Goals

The aim of this dissertation is the development of models capable of obtaining reliable 3D translation estimates from an IMU using only its inertial data. This way, in cases in which the IMUs are firmly attached to a specific body segment, these models must be capable of accurately estimating the motion of that body segment. Such models can eventually be part of the SWORD human motion tracking process allowing an improved tracking of the patients' movement and, consequently, an enhanced rehabilitation process.

In the context of this dissertation, the produced models will only focus on the 3D displacement estimate. These models will not take into account the transformations from coordinate frames, the heading of the IMU and the gravity removal process. The data that will be used in this work is already transformed to the Earth coordinate frame, and gravity acceleration removed. Therefore, the developed models can only be applied after conducting those operations, and these models are dependent on those operations performance.

1.4 Contributes

The major contributions of this dissertation are:

- A machine learning (ML) approach for zero-velocity detection using the inertial measurement from IMUs placed all over the body
- An ML method for estimating the translation between periods in which the IMU is stationary. This method can be integrated with the zero-velocity detection approaches in inertial navigation systems allowing a robust trajectory estimate.
- The proposal of an end-to-end ML framework for estimating the trajectory of an IMU using its inertial data. This ML framework allows to simultaneously classify zero velocity moments and identify translations between those periods.
- A preparation of a paper submission is currently being developed using the methodologies presented in this work

1.5 Document Structure

This document is organised as follows: Chapter 2 presents an overview of inertial measurement units, accelerometers, gyroscopes, and their application. This overview includes the analysis of different inertial sensors types and their errors. Finally, inertial navigation systems and different attitude representations are introduced.

In Chapter 3 several techniques used to conduct 6-degree of freedom (DoF) motion estimate are examined. This analysis comprises sensor fusion, motion constraints, and end-to-end AI approaches.

Chapter 4 describes the dataset selection process from several public available ones. Furthermore, the selected dataset is presented and analysed.

Chapter 5 covers the methods proposed to perform zero-velocity detection. These methods include several supervised ML classifiers. The results of the application of these methodologies are also presented and discussed in this chapter.

In Chapter 6, the ML regressor methods used for estimating the translation between zero-velocity moments are presented. Those methods results are also provided and discussed. This chapter also includes the proposal and evaluation of a stacked neural network for simultaneous zero-velocity detection and translation estimate.

Finally, Chapter 7 presents the major conclusions that can be drawn from this dissertation and future work proposals.

Chapter 2

Inertial Measurement Units

Inertial Measurement Units (IMUs) are typically composed of accelerometers and gyroscopes. Accelerometers allow the quantification of an external specific force that acts along its sensitive axis while gyroscopes measure the angular rate. Both sensors operate without any external reference [37].

Usually, an IMU includes three orthogonal-placed accelerometers and three orthogonal-placed gyroscopes, allowing the collection of forces and angular rates regarding three dimensions [105]. These IMUs are also known as 6-Degrees of Freedom (DoF) IMUs. Others also include three mutually orthogonal magnetometers, being known as 9-DoF IMUs. The magnetometers and 9-DoF IMUs will not be presented in this chapter since that, in an indoor setting, the magnetometers signals are strongly distorted due to soft and hard iron effects from the metallic structures and electric current [52, 61].

An accelerometer allows the quantification of body acceleration. Ideally, the calculus of the body velocity and displacement would be trivial tasks using the measured acceleration. Nevertheless, accelerometers are generally associated with high errors that turn the calculation of velocity and displacement into difficult tasks. All accelerometers generally work in the same way. However, these devices can be divided into mechanical or solid-state devices, depending on the way that the acceleration is estimated.

Gyroscopes, on the other hand, can be spinning mass, optical and vibratory. Each one of these types is based on a different physical principle. The measurement of the angular velocity performed by the different types of gyroscopes is also frequently affected by errors [48].

Accordingly to the technology in which the inertial sensors are applied, its size, mass, cost and performance change and, in general, larger, more massive and more expensive sensors are associated with higher performance. Nowadays, MEMS-based sensors are becoming a popular solution for inertial sensing. These sensors are small, light, inexpensive, shock-resistant and have a low power consumption. The major drawback is that these are usually associated with low performance. However, recently, their accuracy has improved considerably. Thus, there is no ideal inertial sensor, and the choice of one is always dependent on its application [48, 57].

2.1 Accelerometers

Accelerometers allow the measurement of a specific external force that results from the sensor's acceleration and the Earth's gravity. In its simplest form, an accelerometer is composed of a proof mass connected through springs to the instrument's case. When a force is applied to the sensor, and due to the proof mass inertia, it will resist the movement. It is possible to calculate the acceleration of the sensor measuring the displacement of the proof mass relative to the sensor's case. This simple accelerometer is illustrated in Figure 2.1 [48].

When integrated into an IMU, the accelerometers also allow orientation estimation. Taking advantage of the gravity acceleration, when no other force is applied on the IMU, it is possible to estimate the attitude of the IMU relatively to the Earth using the accelerometers measurements.

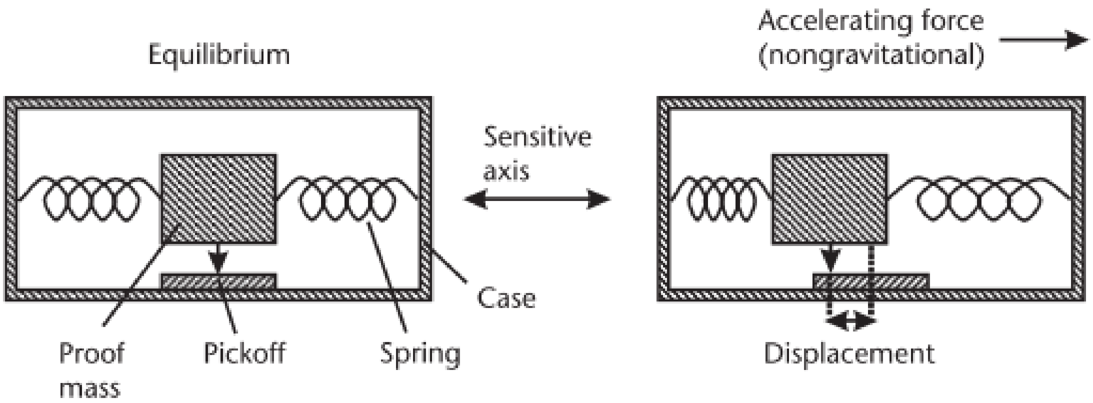


Figure 2.1: Simple accelerometer, extracted from [48]

2.1.1 Types of Accelerometers

2.1.1.1 Mechanical Accelerometers

This family of accelerometers includes the one described previously in Section 2.1. These devices have been developed through several decades and are a compact and trustworthy solution presenting high accuracy and vast dynamic range of sensing.

As explained previously, these sensors use the displacement of a pickoff to determine the acceleration of the device. The configuration of the devices can be separated in open-loop and closed-loop. An open-loop configuration is illustrated in Figure 2.1: a proof mass is restrained to a zero position using a spring. When the sensor suffers an acceleration along its sensitive axis, the displacement of the pickoff is measured. This measurement allows determining the acceleration of the sensor. Other alternatives for an open-loop configuration include, for instance, an optical pickoff.

In a closed-loop configuration, instead of a spring, an electromagnetic device is used. The proof mass is maintained in its zero position due to the effect of the electromagnetic field. When the proof mass is dislocated due to acceleration, an electric current is passed through the electromagnetic device. This current increases the electromagnetic field and avoids the displacement of the proof mass. The acceleration of the sensor is proportional to the electric current needed to keep the proof mass still [105].

An open-loop configuration possesses several problems that act as a limitation to its correct functioning. These problems include the low resolution of the pickoff and the fact that only a linear function is used to represent the action of the spring (that exhibits hysteresis and non-linearity). Therefore, the closed-loop configuration, also called force-feedback, is a more accurate and stable solution offering a more extensive dynamic range and higher linearity and, this way is the most frequently used [48].

2.1.1.2 Solid-State Accelerometers

These accelerometers are known for being small and reliable. Can be divided into other subtypes as the vibratory, silicon, quartz and surface acoustic wave.

An example of this kind of sensor is the vibratory accelerometer. It is an open-loop device usually constituted by two quartz crystal beams – that support a proof mass pendulum – which are placed symmetrically at far ends of the accelerometer. These beams are subjected to an electric current which causes them to vibrate at the same frequency. When the sensor suffers an acceleration along its sensitive axis, the quartz crystals are compressed differently. The one that is stretched vibrates at a higher rate than the original frequency. The one that is compressed vibrates at a lower one. Measuring the vibration differences it is possible to quantify the acceleration that the sensor is suffering.

Another famous example of this family of devices is the surface acoustic wave accelerometer. This sensor is composed by a piezoelectric quartz cantilever beam which is rigidly attached to the sensor case at one end. At the other end, it possesses a proof mass which is free to move. At rest, the piezoelectric quartz beam vibrates at a resonant frequency. When the sensor is submitted to acceleration along its sensitive axis, the beam bends which causes an alteration on the vibration frequency. This alteration on the surface acoustic wave which is sensed by a surface acoustic wave sensor placed on the surface of the beam. As the change in frequency is proportional to the acceleration, it is possible to estimate this one.

Solid-state accelerometers also include other options which include fibre-optic, optical, ferroelectric and solution-electrolytic sensors [105, 2].

2.2 Gyroscopes

Gyroscopes enable the measurement of the angular rate. In its most simplistic design, a gyroscope is essentially a spinning wheel/disk mounted in an axle. It includes gimbal support by one or multiple gimbals allowing the rotation of the wheel/disk in a single axis.

When the disk is spinning, if a torque is applied on the device, it will try to compensate this movement. This compensation occurs due to the conservation of the angular momentum. The application of a torque alters the angular momentum giving rise to a phenomenon known as *precession*– which is the shift of the orientation of the rotational axis of a rotating body when subjected to a torque. It is possible to determine the orientation of the device measuring the alteration of the rotational axis of the disk. More modern gyroscopes operate on different principles and are dedicated to measuring not the orientation but the angular rate [105, 48, 2].

2.2.1 Types of Gyroscopes

2.2.1.1 Spinning-Mass Gyroscopes

This type of gyroscope has the same functioning principle as the one described in Section 2.2. A spinning-mass gyroscope or a mechanical one is composed basically by the case of the instrument, a rotor, a spin motor, angle pick-offs and the gimbals or support frames. When the gyroscope operates on a closed-loop, it is also needed a torque motor that returns the rotor to its zero position. A simple two-degrees-of-freedom gyroscope is described in Figure 2.2.

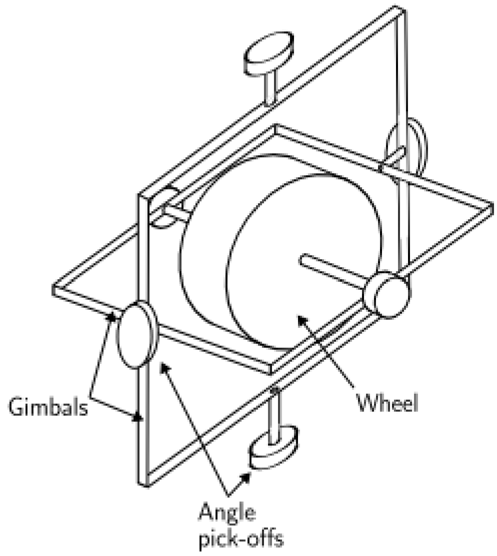


Figure 2.2: Simple two-degrees-of-freedom gyroscope, extracted from [2]

The spin motor is used to rotate the rotor which generates an angular momentum. The conservation of the angular momentum is the basis of the functioning of these gyroscopes. The configuration presented in Figure 2.2 allows the rotation of the rotor in all three axes. When the sensor is rotated the orientation of the case of the instrument can be measured in relation to the spin axis of the rotor using the angle pick-offs.

As these gyroscopes include moving parts, some friction may occur leading to drift in the output. This fact associated with the high startup time of these sensors constitutes their main disadvantages [105, 48, 2].

2.2.1.2 Optical Gyroscopes

The optical gyroscopes use the Sagnac effect as working principle. Sagnac proved in 1913 that two light beams travelling in opposite directions inside a rotating disk travel different distances. The one that is moving accordingly to the course of the rotation travels a longer distance than the one going in the opposite direction. Therefore, its travel time is larger. This phase shift can be used to determine the angular velocity [81].

The diverse types of optical gyroscopes differentiate in the way that the light is produced and in the way that the phase shift is measured. This family includes sensors as the ring laser gyroscope (RLG) and the fibre optic gyroscope (FOG).

The RLGs include a cavity filled with Helium-Neon gas. The atoms of this gas are excited by an electric field generated by a cathode and an anode. When the atoms return to their normal energy level, some energy is emitted through a photon. This photon can excite other atoms which generate other photons with the same wavelength. The photons are reflected through the lightpath with the use of mirrors. One of the extremities of the RLG permits the partial transmission of the light to a detector. Each RLG possesses two lasing modes, one in each direction. When the sensor is still, the wavelength of both lasing modes is the same. Although, when the sensor is spinning, the wavelength of the lasing mode in the direction of the rotation is higher and its frequency is lower.

The FOGs use a light source which is divided into two different beams using beam splitters. The resultant separated beams are sent in opposite directions into fibre optic coils that are placed around the FOG. When the rays reach the detector, they are combined. Due to the Sagnac effect, there is a phase shift dependent on the angular velocity of the gyroscope when it rotates on an axis that is perpendicular to its plane, which makes the rays interfere. It is possible to calculate the angular velocity measuring the intensity of the combined beams.

Optical gyroscopes are usually more accurate than the spinning-mass gyroscopes presented in Section 2.2.1.1 since it does not contain moving parts, possesses an instant startup and a wide dynamic range [105, 48, 2].

2.2.1.3 Vibratory Gyroscopes

This type of gyroscope is based on the Coriolis effect. The sensor's main component is a vibratory element (usually an element with piezoelectric properties) which is forced to vibrate in a single axis. When the sensor is rotated in an axis perpendicular to the vibratory axis, a Coriolis acceleration is generated which is orthogonal to the direction of the initial vibration. It is possible to calculate the angular rate of the device measuring this secondary vibration. The way of detection of this vibration depends on the architecture of the gyroscope which can be single-beam, double-beam or tuning-fork [58].

The main advantages of this type of gyroscope are its low price, low weight and the fact that it can be easily integrated with MEMS technology. However, frequently these gyroscopes are associated with a low-performance [48].

2.3 MEMS-Based Inertial Sensors

Miniaturized electromechanical elements technology has seen remarkable progress in recent years due to advances in microfabrication techniques. Therefore, nowadays, the use of MEMS-based inertial sensors is becoming common. It allows a significant reduction in cost, weight, size and power consumption. These type of sensors are used, for instance, in almost all smartphones, in wearables, toys, gaming systems and in inertial navigation systems (INS) due to their properties [95].

MEMS accelerometers are divided into two types. This division depends on the process of sensing the acceleration to which the device is being subjected. The first type detects the displacement of a mass and is similar to the mechanical accelerometers described in Section 2.1.1.1

but using silicon components. The second type evaluates the changing of frequency of a vibrating element same way as the vibratory accelerometers explained in Section 2.1.1.2 and can use silicon or quartz components [105].

MEMS gyroscopes take advantage of the Coriolis effect and work similarly to the vibratory gyroscopes described in Section 2.2.1.3. This way, the angular rotation is sensed by sensing the Coriolis force which occurs in an axis perpendicular to the axis of vibration [2].

Usually, MEMS inertial sensors are associated with lower performances than the sensors manufactured using traditional techniques. Although, lately, this technology continues to evolve to reduce the size of the sensors, reduce their power consumption and to increase their performance [16].

For human motion tracking, it is essential to use light and small-dimension inertial trackers. Due to their properties, MEMS-based inertial sensors are an ideal solution to collect data about the human movement. It is fundamental to develop methodologies that can improve the performance of this kind of inertial sensors to enhance the quality of the data acquired.

2.4 Error Characteristics

It is possible to divide the inertial sensor errors into two sections: deterministic or systematic errors and stochastic or random inaccuracies. The systematic errors include bias and scale factor errors which can be calibrated in the laboratory. Bias is the offset between the real output and the ideal output while the scale factor is the ratio between the output and the input. An illustration of these errors is present in Figure 2.3. Bias and scale factor errors, as well as the inaccuracies from temperature-dependent variations, can be corrected by the IMU processor using specific laboratory calibration data [38, 24, 7].

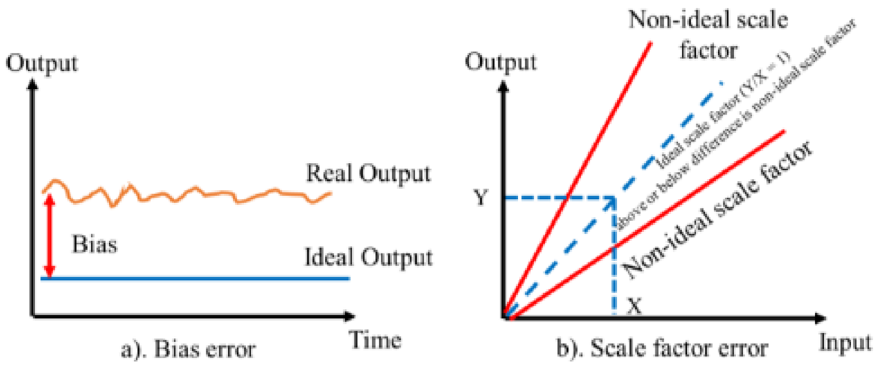


Figure 2.3: Deterministic errors - a) Bias Error; b) Scale Factor Error, extracted from [24]

On the other hand, the stochastic errors include bias drifts and scale factors drifts affected by noise whose rate change over time. For that reason, these can not be modelled using deterministic processes and have to be modelled using stochastic modelling. These errors can also include white noise (uncorrelated noise generated by power sources or semiconductors). Figure 2.4 represents the random errors that affect IMUs. The main problem is that these errors accumulate over time and, if not modelled correctly, associated, for instance, with the double integration that has to be made to determine the displacement of a sensor from the accelerometer data, can lead to high uncertainties in motion tracking [38, 24, 7].

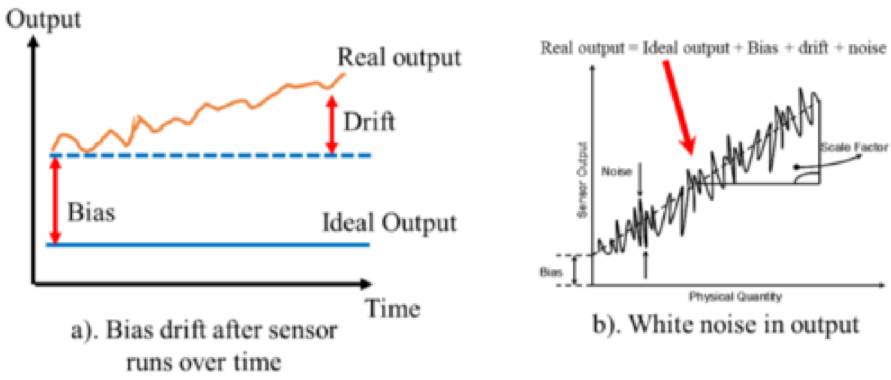


Figure 2.4: Random errors - a) Bias changing over time (bias drift); b) White noise represented in the output, extracted from [24]

Biases

As described previously, the bias is the offset between the real output and the ideal output (Figure 2.3 -a). It is possible to separate the biases into run-to-run bias and in-run bias. The first one characterizes the bias presented each time the accelerometer is turned on, and can also be named static bias component. It also includes the remaining error from the calibration. The second one characterizes the changes that the bias suffers during the time that the sensor is on, and can also be named dynamic bias component [48].

Scale Factor and Cross-Coupling Errors

The scale factor is the ratio between the output and the input (Figure 2.3 -b). Another critical factor is the non-linearity of the scale factor. If it is non-linear, then a change in the input will lead to a different shift in the output.

The cross-coupling errors represent the errors that occur on IMUs due to the misalignment of the sensors. This non-orthogonality between the three gyroscopes and the three accelerometers, as pictured in Figure 2.5, generates additional scale factor errors [48].

Temperature Effects

The device heating and the changes in the temperature of the environment lead to a drift in the bias. This bias drift is often non-linear. However, the majority of the IMUs are associated with a temperature sensor which allows the correction of the error [2].

Random Noise

The random noise includes drifts – a change of the output with time when the input is not altered – as described in Figure 2.4-a), and white noise, as described in Figure 2.4-b). Electrical

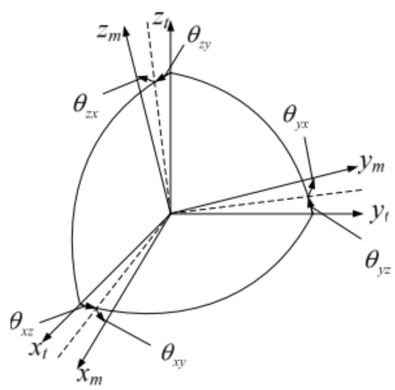


Figure 2.5: Misalignment error which represents the non-orthogonality of the IMU sensors, extracted from [110]

sources are a particularly important source of this kind of noise and affect particularly MEMS IMUs.

The integration of the output of the accelerometer associated with a random noise error leads to an error known as the random-walk error of the velocity whilst the integration of the output of the gyroscope leads to an attitude random-walk error [48].

2.5 Error Model

Taking into consideration the errors mentioned in Section 2.4, it is possible to develop a unified model that represents the errors associated with both gyroscopes and accelerometers.

This way, a general model for an inertial sensor can be represented by [110]:

$$q_m = K\ q_t + b(t) \tag{2.1}$$

In which q_m represents the measured output quantity, K the scale factor, q_t the real output quantity and b(t) the total bias component present in the output which is dependent of time. Important to mention that, although Equation 2.1 does not consider it, in truth, the scale factor K may be nonlinear and might change with time. This way a more appropriate representation would be $K\,(q_t, t)$. For this simplified version of the error model, K is assumed to be linear and time-invariant.

The total bias present in the output is composed of multiple components:

$$b(t) = b_0 + b_r(t) + b_w(t) \tag{2.2}$$

In which b_0 represents the constant null shift, $b_r(t)$ the bias drift and $b_w(t)$ the white noise bias which is random. Both $b_r(t)$ and $b_w(t)$ are unstable over time with $b_r(t)$ being a slow varying component that can be portrayed through a stochastic time-series.

For the three accelerometers existent in a IMU, this error can be characterised as:

$$\begin{bmatrix} a_{mx} \\ a_{my} \\ a_{mz} \end{bmatrix} = \begin{bmatrix} K_{mx} & 0 & 0 \\ 0 & K_{my} & 0 \\ 0 & 0 & K_{mz} \end{bmatrix} R(\theta_{ij}) \begin{bmatrix} a_{tx} \\ a_{ty} \\ a_{tz} \end{bmatrix} + \begin{bmatrix} b_{ax} \\ b_{ay} \\ b_{az} \end{bmatrix} \tag{2.3}$$

In which $a_{mi}(i = x, y, z)$ represents the value measured by the accelerometer, $K_{mi}(i = x, y, z)$ the scale factor of the three MEMS accelerometers, $R(\theta_{ij})$ the rotation from the misaligned coordinate frame (m) to a real orthogonal coordinate frame (t) – with $\theta_{ij}(i, j = x, y, z)$ as represented in Figure 2.5. $a_{ti}(i = x, y, z)$ corresponds to the true amounts measured by the accelerometers and $b_{ti}(i = x, y, z)$ to the bias associated to each accelerometer.

For the three gyroscopes present in a IMU this error can be defined as:

$$\begin{bmatrix} \omega_{mx} \\ \omega_{my} \\ \omega_{mz} \end{bmatrix} = \begin{bmatrix} K_{mx} & 0 & 0 \\ 0 & K_{my} & 0 \\ 0 & 0 & K_{mz} \end{bmatrix} R(\theta_{ij}) \begin{bmatrix} \omega_{tx} \\ \omega_{ty} \\ \omega_{tz} \end{bmatrix} + \begin{bmatrix} b_{ax} \\ b_{ay} \\ b_{az} \end{bmatrix} + \begin{bmatrix} d_{xx} & d_{xy} & d_{xz} \\ d_{yx} & d_{yy} & d_{yz} \\ d_{zx} & d_{zy} & d_{zz} \end{bmatrix} \begin{bmatrix} a_{mx} \\ a_{my} \\ a_{mz} \end{bmatrix} \tag{2.4}$$

In which $\omega_{mi}(i = x, y, z)$ represents the value measured by the gyroscope, $K_{mi}(i = x, y, z)$ the scale factor of the three MEMS gyroscopes, $R(\theta_{ij})$ the rotation from the misaligned coordinate frame (m) to a real orthogonal coordinate frame (t). $\omega_{ti}(i = x, y, z)$ corresponds to the true amounts measured by the gyroscope and $b_{ti}(i = x, y, z)$ to the bias associated to each gyroscope. Finally an extra element is added which represents the parameter of the gyroscope drift that are associated to the acceleration of the IMU $d_{ij}(i, j = x, y, z)$ [110].

2.6 Strapdown Inertial Navigation Systems

From the measurements provided by the accelerometers and gyroscopes of an IMU it is possible to estimate the values relative to the velocity, displacement, and attitude of the IMU using an inertial navigation system (INS) as it is represented in Figure 2.6.

To fully understand an INS the simplest way is to start by a 1D example: picturing a body b moving along an axis perpendicular to the gravity direction relatively to an Earth coordinate frame e and imagining that the body cannot rotate and can only move forward and backwards in the mentioned axis. Due to the mentioned restrictions, a single accelerometer aligned with the movement axis can be used to measure the acceleration of the body. Therefore, it is possible to calculate the current velocity knowing the initial velocity ($v_{eb}(t_0)$) and the acceleration (a_{eb}) [49]:

$$v_{eb}(t) = v_{eb}(t_0) + \int_{t_0}^{t} a_{eb}(t')dt' \tag{2.5}$$

Likewise the position (r_{eb}) of the body can also be calculated through the integration the velocity:

$$r_{eb}(t) = r_{eb}(t_0) + \int_{t_0}^{t} v_{eb}(t')dt' \tag{2.6}$$

Imagining now a 2D example in which b can move in the plane perpendicular to the gravity direction, i.e the x and y axes of the Earth coordinate frame p. Considering also that b can take

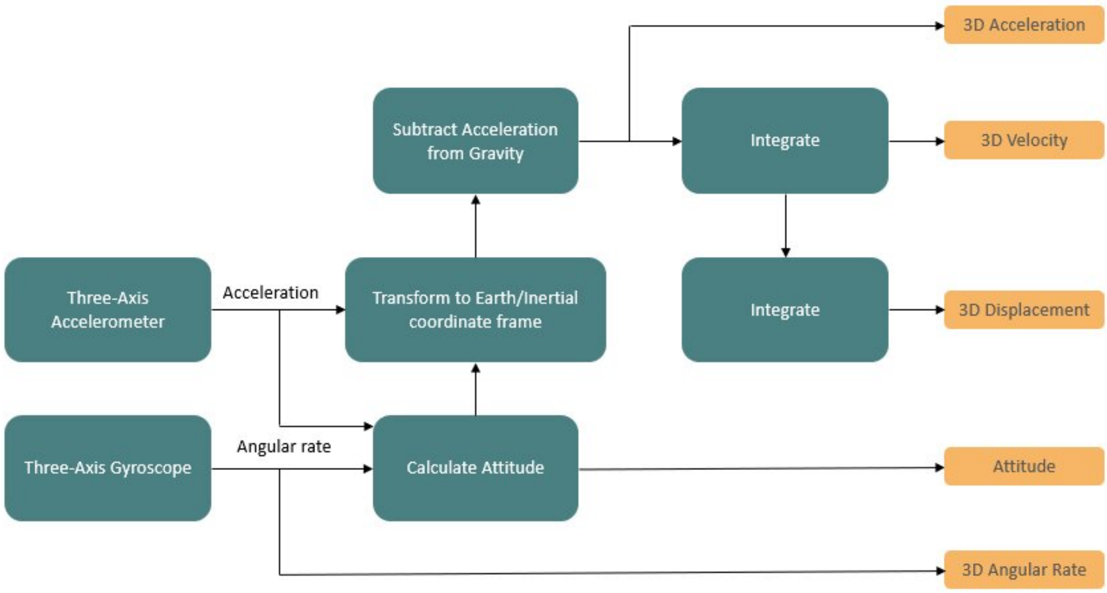


Figure 2.6: Inertial Navigation System, adapted from [85]

any orientation in this plane, then b has three degree of freedom (DoF) (one angular and two linear). This way, two accelerometers orthogonally placed along the body axes it is possible to calculate the velocity and the position of b:

$$\begin{pmatrix} v_{eb,\,x}(t) \\ v_{eb,\,y}(t) \end{pmatrix} = \begin{pmatrix} v_{eb,\,x}(t_0) \\ v_{eb,\,y}(t_0) \end{pmatrix} + \int_{t_0}^{t} \begin{pmatrix} a_{eb,\,x}(t') \\ a_{eb,\,y}(t') \end{pmatrix} dt' \tag{2.7}$$

$$\begin{pmatrix} x_{eb}(t) \\ y_{eb}(t) \end{pmatrix} = \begin{pmatrix} x_{eb}(t_0) \\ y_{eb}(t_0) \end{pmatrix} + \int_{t_0}^{t} \begin{pmatrix} v_{eb,\,x}(t') \\ v_{eb,\,y}(t') \end{pmatrix} dt' \tag{2.8}$$

Considering that the acceleration of the body is measured by two orthogonal accelerometers placed along the body axes it is necessary to rotate the accelerations measured in the body frame to the earth frame. This way, one needs to determine the attitude of b relatively to the Earth frame, ψ_{eb}, to perform the adequate rotation R:

$$\begin{pmatrix} a^b_{eb,\,x}(t') \\ a^b_{eb,\,y}(t) \end{pmatrix} = \begin{pmatrix} R(\psi_{eb}(t)) \\ R(\psi_{eb}(t)) \end{pmatrix} + \int_{t_0}^{t} \begin{pmatrix} a^b_{eb,\,x}(t') \\ a^b_{eb,\,y}(t') \end{pmatrix} dt' \tag{2.9}$$

The current attitude of the body can be measured by a gyroscope perpendicular to the plane of movement knowing the initial attitude ($\psi_{eb}(t_0)$). Therefore, using the angular rate measured by the gyroscope ($\omega^b_{eb,z}$) it is possible to calculate the current attitude:

$$\psi_{eb}(t) = \psi_{eb}(t_0) + \int_{t_0}^{t} \omega^b_{eb,\,z}(t')dt' \tag{2.10}$$

In the 3D space the body possesses 6-DoF (three angular and three linear). Consequently, six orthogonal inertial sensors are needed to estimate the body position and the attitude. Essentially four steps are needed to track the velocity, position and orientation of a body using these inertial sensors [49]:

1. Update the attitude
2. Transform the acceleration measurements to the Earth/Inertial coordinate frame in which the position and velocity will be resolved
3. Subtract the acceleration caused by the gravity from the acceleration readings
4. Calculate the velocity and the position

2.7 Attitude Representation

The accelerometers and the gyroscopes from an IMU provide readings in the body coordinate frame. However, this reference frame is changing with the motion experienced by the IMU. This way, it is necessary to transform the data collected in the body coordinate reference frame to an inertial/ Earth reference frame. After conducting this transformation of coordinates, it is possible to remove the gravity acceleration and adequately process the signals.

In theory, knowing the initial attitude of the IMU and integrating the angular velocity provided by the gyroscope, it should be possible to obtain a reasonable estimate of the IMU attitude. However, due to the presence of potentially significant biases that can also change with time (bias drift) in the measurements supplied by the gyroscope, which lead to cumulative errors over time, these are not reliable on the long-term.

To estimate the attitude of the IMU coordinate frame relatively to the Earth frame the information provided by the 3D linear acceleration measured by the IMU accelerometers can be used. When the only acceleration measured by the accelerometer is due to the gravity force, then it is possible to estimate the attitude relatively to the Earth using the relation between the measurements from the three accelerometers. This way, the accelerometers measurements can be used as vertical reference [59]. The main problem is that accelerometer orientation estimation is affected in specific periods by external forces and vibrations, which cause other accelerations. This way, one can say that the attitude estimation using accelerometers signals is untrustworthy on short periods but trustworthy on long periods because the average movement-induced acceleration throughout the entire movement will be zero in every axis (except for the accelerometer biases) [107, 50].

The position of an IMU in space is normally represented using a Cartesian frame. However, for the attitude representation several solutions are used. These include rotation matrices, Euler angles and quaternions [43]. Throughout this chapter, a focus is placed in those representations. It is essential to understand how attitude or orientation of a body is represented in a particular coordinate frame and how can a transformation to another coordinate frame be expressed considering the context of this work.

2.7.1 Coordinate Reference Frames

A coordinate reference frame is essentially a perspective from which the measurements are made and, therefore, a performed measurement is at all times dependent on its reference frame. A frame is always defined by a set of three mutually orthogonal unit vectors. To understand the value presented by the IMUs, either the ones related to the accelerometer either the values related to the gyroscope, it is essential to comprehend the different coordinate frames [57]:

- **Body Coordinate Frame (b)**- The reference frame from the IMU, which is fixed to the IMU and rotates with it. The measurements from the sensors are relative to this frame.
- **Earth Coordinate Frame (e)**- A frame which rotates since it is associated with the earth. Its Z-axis is aligned with the gravity.
- **Inertial Coordinate Frame (i)**- Its Z-axis is also aligned with the gravity but, unlike the earth frame, this is a fixed reference and does not rotate. When performing inertial motion tracking, this is the most used since it is possible to disregard the earth movement.

When using IMUs, it is fundamental to rotate from the Body coordinate frame to the Inertial/Earth coordinate frame (taking into account the type of motions that are being measured when performing human motion tracking it is possible to ignore the Earth rotation) to be able to identify the IMU acceleration due to gravity and also to represent the orientation of the IMU relatively to the Inertial/Earth coordinate frame.

In the next subsections, different ways of representing rotations are presented.

2.7.2 Rotation Matrix

The rotation matrix allows relating the coordinates of a body in a certain coordinate system with its coordinates in another coordinate system. Giving a point p in a coordinate system B (p^B), the point in a coordinate system A (p^A) can be represented using a rotation matrix (R_B^A) which relates the coordinate frame B with the coordinate frame A as expressed in Equation 2.11 [101].

$$p^A = R_B^A \ p^B \tag{2.11}$$

This rotation matrix can be represented by:

$$R_B^A = \begin{bmatrix} x_B.x_A & y_B.x_A & z_B.x_A \\ x_B.y_A & y_B.x_A & z_B.y_A \\ x_B.z_A & y_B.x_A & z_B.z_A \end{bmatrix} \tag{2.12}$$

In which the dot product represents the projection of the unit vector of each of the axis of the frame $o_B x_B y_B y_B$ onto the coordinate axes of the frame $o_A x_A y_A y_A$.

Since the axes of each coordinate frame are mutually orthogonal, the rotation matrix R_B^A is orthogonal and so:

$$(R_B^A)^T = (R_B^A)^{-1} \tag{2.13}$$

2.7.3 Euler Angles

Euler Angles allow the representation of a rotation from a reference frame to any target orientation using only three parameters. These three independent parameters are represented by three different angles - ϕ, θ and ψ.

Different conventions are used to represent a rotation with Euler Angles. These are dependent on the sets of rotation axes used. It is possible to separate these conventions into Proper Euler angles and Tait–Bryan angles. Proper Euler angles define a rotation using just two axes for rotation – like (z,y,z) or (y-x-y). Tait–Bryan angles define a rotation using three axes – like (x,y,z) or (z-y-x). If the rotations are established around the principal coordinate frames axis (x,y,z) the angles of the rotation are defined as roll, pitch and yaw respectively.

It is possible to define the rotation by intrinsic and by extrinsic rotations using either of the conventions presented. Intrinsic rotations are rotations that occur around the body coordinate frame while extrinsic rotations are rotations that take place around a fixed axis like the inertial reference frame.

Figure 2.7 represents the rotation of a moving frame that initially is placed over a fixed reference frame (xyz). It is simple to define a rotation using intrinsic rotations looking at this figure. The first one to occur is a rotation of ϕ around the z-axis. Subsequently, a rotation of θ occurs around the ξ'-axis, and finally a rotation of ψ around the z'-axis.

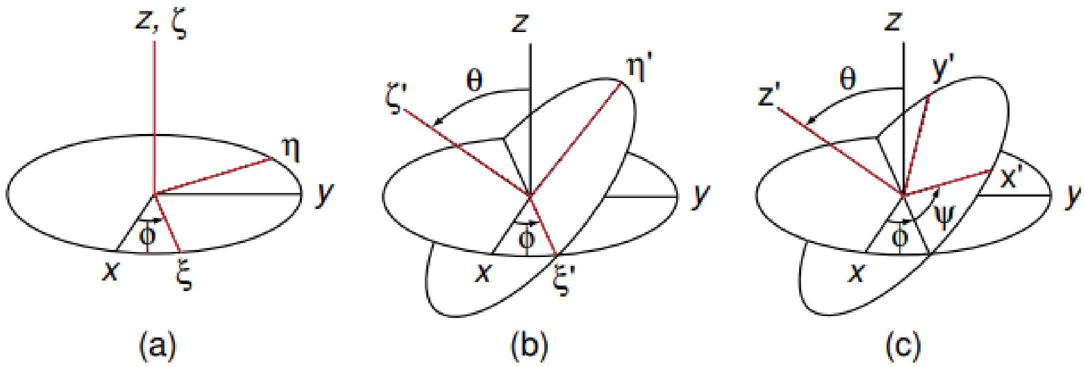


Figure 2.7: Rotation established using proper Euler angles defined using a set of intrinsic rotations, extracted from [92]

A rotation from a coordinate frame B to a coordinate frame A can also be defined using extrinsic rotations as described in Equation 2.14. In this case, it is represented as a sequence of rotations z-y-z. The first rotation is done with an angle ϕ about the z-axis. Subsequently, a rotation with an angle θ is performed about the y-axis. Finally, a rotation with an angle ψ is conducted about the z-axis. This way, the rotation matrix R_B^A can be obtained as:

$$R_B^A = R_{z,\psi}\ R_{y,\theta}\ R_{z,\phi} \tag{2.14}$$

In which $R_{z,\phi}$, $R_{y,\theta}$ and $R_{z,\psi}$ represent respectively:

$$R_{z,\phi} = \begin{bmatrix} cos_\phi & \text{-}sin_\phi & 0 \\ sin_\phi & cos_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_{y,\theta} = \begin{bmatrix} cos_\theta & 0 & sin_\theta \\ 0 & 1 & 0 \\ \text{-}sin_\theta & 0 & cos_\theta \end{bmatrix} \quad R_{z,\psi} = \begin{bmatrix} cos_\psi & \text{-}sin_\psi & 0 \\ sin_\psi & cos_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

As long as the convention, type and order of rotations are set and consistent, any of the representations can be used.

The main disadvantage of these types of representations is the gimbal lock, which occurs when two of the axis get aligned. Since this method uses sequential rotations, the later rotations of the sequence will affect the earlier rotation axis and, this way, the alignment of the axis change with the Euler rotations. As a result, two axes may get aligned, leading to a degree of freedom lost. On the rotational order presented in Equation 2.14 if θ is set to 90º then the y-axis and the x-axis will get aligned, resulting in loss of a degree of freedom [36].

2.7.4 Axis-Angle

With this representation, initially, two values are defined: a unit vector and a rotation angle. The vector symbolizes the axis to rotate around and the angle the quantity of rotation. Given any two orientations on the 3D space, there is always an axis to which it is possible to rotate around to get between orientations. Single-axis rotations are always the shortest direct path between two orientations.

This way, defining a unit vector $\mathbf{k} = (k_x, k_y, k_z)^T$ expressed in the original reference frame coordinates and an angle θ as the angle to rotate around $\mathbf{k}$ it is possible to rotate between two orientations.

This method overcomes the gimbal lock problem. However, the quaternions method presented in the next section has some interesting mathematical properties which make the method more interesting considering the application.

2.7.5 Quaternions

Quaternions are an interesting solution to represent rotations once these can be represented as a 4D complex number:

$$q = q_0 + \mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{2.16}$$

In which q_0 represents the scalar part and $\mathbf{q}$ the vector part.

Most of the operations performed with quaternions require its previous normalization [66]. A normalized quaternion is called a unit quaternion. This way, any orientation in the 3D space from a coordinate frame A concerning a coordinate frame B can be represented by a unit quaternion, representing a rotation θ around a rotation axis $\mathbf{r}$:

$$^{B}_{A}q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix} = \begin{bmatrix} cos\frac{\theta}{2} & r_x\, sin\frac{\theta}{2} & r_y\, sin\frac{\theta}{2} & r_z\, sin\frac{\theta}{2} \end{bmatrix} \tag{2.17}$$

Quaternions allow a simple, direct, predictable and consistent interpolation between rotations and are not affected by gimbal lock nor changing axis. These also permit calculus simplification, simplify the process of finding the lower angle or distance between two orientations and, for instance, the process of finding the opposite of a 3D rotation [108].

It is simple to exchange between reference frames using the quaternion conjugate, $q_A^{B\,*}$:

$$q_A^{B\,*} = q_B^A = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{2.18}$$

Additionally, a compounded orientation can be easily expressed by the multiplication of quaternions:

$$q_C^A = q_C^B \otimes q_B^A \tag{2.19}$$

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \end{bmatrix} \otimes \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix} = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 - p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 - p_2q_0 - p_3q_1 \\ p_0q_3 - p_1q_2 - p_2q_1 - p_3q_0 \end{bmatrix} \tag{2.20}$$

Quaternions are also useful to change the representation of a vector from a coordinate frame A to a coordinate frame B:

$$v_B = q_A^B \otimes v_A \otimes q_A^{B*} \tag{2.21}$$

Another interesting property is the differentiation of quaternions, that can be used to update the attitude using the angular rates $(\omega_x(t), \omega_y(t), \omega_z(t))$ and the time interval:

$$\begin{pmatrix} \frac{d}{dt}q_w(t) \\ \frac{d}{dt}q_x(t) \\ \frac{d}{dt}q_y(t) \\ \frac{d}{dt}q_z(t) \end{pmatrix} = \frac{1}{2} \begin{bmatrix} -q_x(t) & -q_y(t) & -q_z(t) \\ q_w(t) & q_z(t) & -q_y(t) \\ -q_z(t) & q_w(t) & q_x(t) \\ q_y(t) & -q_x(t) & q_w(t) \end{bmatrix} \begin{pmatrix} \omega_x(t) \\ \omega_y(t) \\ \omega_z(t) \end{pmatrix} \tag{2.22}$$

2.8 Summary

A typical IMU is composed of three mutually orthogonal accelerometers and three mutually orthogonal gyroscopes. Recently, a focus has been placed in MEMS-IMUs due to advances in microfabrication techniques. MEMS-based IMUs – as the ones used by SWORD Health in their inertial motion trackers – are low-cost, lightweight and portable. These properties make them an adequate option for conducting human motion tracking. The main problem associated with this kind of device is its low-performance. IMUs are affected by deterministic and random errors. These include, for instance, biases drifts, scale factor uncertainties, misalignment of sensors and white noise. These errors are particularly exacerbated when using MEMS-based IMUs.

Using an IMU composed of three orthogonal accelerometers and three orthogonal gyroscopes, it is possible to obtain the motion in 6-DoF. To do so, it is necessary to convert the measurements from the IMU coordinate frame to the Earth/Inertial coordinate frame and therefore, identifying the attitude of the device. Applications like human motion tracking with IMUs need an attitude representation that permits efficient monitoring in all orientations. As the IMU sensors measure data in the body coordinate frame, it is necessary a rotation into the Earth/Inertial Reference Frame so that the signals can be efficiently processed and that the signal from the gravity acceleration can be removed. In inertial human motion tracking, the orientation of a body segment is often represented using Euler angles due to their intuitive physical meaning. Problems like the gimbal lock of Euler Angles or the lack of efficiency presented by the rotation matrix often make these methods unsuitable for motion tracking. Quaternions allow to overcome these limitations, avoid the use of trigonometric functions, and also have interesting mathematical properties constituting an effective method for human motion tracking being one of the most used options [11].

Chapter 3

6-DoF Trajectory Estimate using Inertial Data

The determination of 6 degrees of freedom (DoF) motion includes both the calculation of a 3D position and a 3D orientation. This calculus is essential in a Human Motion Tracking process to allow the proper tracking of different body segments. This topic is also extremely explored in areas such as robotics and automation. In these fields this procedure is frequently related to the odometry process, which consists in collecting data from sensors placed, for instance, in a robot or in a self-driven car to collect information regarding its motion [96].

The use of inertial sensors to collect this information is interesting, since that, in theory, and as presented in Chapter 2, an Inertial Navigation System (INS) based on a single IMU is capable of achieving a robust 6-DoF pose estimate. Through the merging of data from 3 mutually orthogonal gyroscopes and 3 mutually orthogonal accelerometers this system continuously calculates the position, the velocity, and the orientation of the IMU as described in Figure 2.6. The main problems of an INS system are related to errors like noises, biases and drifts, present both in the accelerometers and in the gyroscopes. This way, and due to the sequential nature of the estimations, the error quickly accumulates and the estimates diverge.

These errors can be reduced by using several position or velocity assisting measurements that can be continuously integrated in the systems or that can aid the initial alignment process. In outdoor systems an INS is often integrated with information from a GPS system to improve the position estimate. Indoor solutions include the use of radio signals, cameras, and lasers. Zero-velocity and angular updates, as well as other motion constraints, can also be used to assist the measurements provided by an IMU. For attitude aiding measurements, the magnetometer is a popular solution, although in indoor setting its measurements can be severely affected by distortions caused by magnetic and ferrous metal objects known as soft-iron and hard-iron distortions [49, 100, 90].

Traditionally, sensor fusion algorithms, which combine data from accelerometers and gyroscopes have been used in order to obtain more reliable pose estimates. These algorithms can integrate information from other sources and include algorithms as the Kalman Filter, the complementary filter and the gradient descent algorithm which are further explained throughout this chapter [1, 66].

The tracking of translational motion is substantially more difficult than the orientation tracking. Since double integration is needed to calculate the displacement of the device, errors rapidly accumulate. A major problem of this procedure is related to gravity 'leaking'. A bad estimate of the IMU orientation leads to an incorrect removal of the acceleration caused by the gravity which corrupts the calculation of the velocity and position of the IMU [96, 99].

Recently some AI-based systems have been presented to estimate 6-DoF motion problems using only inertial measurements. Some implement an end-to-end approach while others use AI to provide information that can be applied using a sensor fusion algorithm. Throughout this chapter, some of those systems are presented.

3.1 Sensor Fusion

Sensor Fusion algorithms combine data from multiple sources (in this case from accelerometers and gyroscopes) in order to obtain more reliable attitude estimates. The fusion of data from different sources allows the creation of a more accurate model.

3.1.1 Kalman Filter

The Kalman filter was already used in such diverse applications as spacecraft navigation, demographic modelling, navigation or nuclear power plant instrumentation. It is a state-space method which can be used to estimate position or orientation [63].

A Kalman filter is an estimation algorithm used to predict a variable of interest. It allows extracting information about a variable which cannot be measured directly, and it allows to merge data from different sources that might be corrupted with noise. That is why it is called a sensor fusion algorithm. It can be faced as a type of state observer, but that is designed for stochastic systems [97].

When applied to inertial sensing, a Kalman Filter uses data from accelerometers and from gyroscopes – acceleration and angular rate – to determine the position and the orientation of the IMU. The signals obtained are noisy, and so the Kalman filter tries to estimate the state of the system. For this estimation, the filter uses the present and the previous states which makes it suitable for real-time applications [97].

Mathematically we can formulate the problem that the Kalman filter tries to solve as a linear stochastic difference equation to estimate the current state x_k [113, 76]:

$$x_k = A\, x_{k-1} + B\, u_k + w_k \tag{3.1}$$

This way is it possible to understand that any x_k is a linear combination of the previous state x_{k-1}, a control signal u_k and a process noise represented by a stochastic term w_k.

With a measurement z:

$$z_k = H\, x_k + v_k \tag{3.2}$$

The current measure is a function of the current state and a noise represented by a stochastic term (v_k). A, B and H are form matrices.

w_k represents the process noise and v_k the measurement noise. Both are considered to have a Gaussian distribution and are independent of each other:

$$w_k \sim N(0, Q) \tag{3.3}$$

$$v_k \sim N(0, R) \tag{3.4}$$

Q and R are the covariance matrixes that change with time.

Simplistically, the Kalman Filter can be represented by two steps: prediction and filtering/updating. For the prediction phase, the algorithm uses the output from the previous state and the process model to generate a new prediction. In the update phase, the prediction for the current state is updated using information collected using the sensors and the measurement model.

The process starts with an estimate of x and P at present. P_k symbolizes the estimation of the covariance of the state, with P_k^- being the estimate without taking into account the measurement z_k at the current time. A, B and H are known from the process model. Subsequently, the two phases mentioned previously happen iteratively.

The prediction phase can be defined as:

$$\hat{x}_k^- = A\,\hat{x}_{k-1} + B\,u_k \tag{3.5}$$

$$P_k^- = A\,P_{k-1}\,A^t + Q \tag{3.6}$$

And the update/filtering phase:

$$K_k = P_k^-\,H^T\,(H\,P_k^-\,H^T + R)^{-1} \tag{3.7}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\,(z_k - H\,\hat{x}_k) \tag{3.8}$$

$$P_k^- = (I - K_k\,H)\,P_k^- \tag{3.9}$$

In which actor K_k represents the Kalman Gain.

In the particular case of orientation estimation using quaternions, in the prediction phase, the quaternion q_k that represents the current orientation is estimated using the gyroscope measurements:

$$\hat{q}_k = \hat{q}_{k-1} + \frac{d}{dt}q(t) * \Delta\,t \tag{3.10}$$

In which q_{k-1} represents the previous orientation estimation, $\frac{d}{dt}q(t)$ the quaternion derivative (calculated using the gyroscope measurements as described in Equation 2.22) and $\Delta\,t$ the time elapsed between estimations. Subsequently, the update phase consists in correcting the verticality errors using the accelerations measured by the accelerometer. This way, a corrected state is obtained, taking into consideration both the process and the measurement steps. The model could also include, for instance, the prediction of the gyroscope bias. Considering this bias in the model could be useful to improve the orientation estimation.

The Kalman filter equations can be derived from different manners, and there are several derivations of this filter developed through the years like the Extended Kalman Filter, which is used when the system model is non-linear. Other solutions include Unscented Kalman Filters (UKF) that provide a better estimate of probability density functions (PDF) under nonlinear conditions [43].

Since that the Kalman filter is a recursive process, it integrates information from all the previous steps in some parameters being extremely advantageous in terms of computational time [76].

The traditional Kalman filter is linear and, therefore, it has performance difficulties when applied to the often classified as non-linear MEMS IMU signals. It also simplifies the models assuming that the noise of the process and measurement can be represented by a Gaussian distribution, which in some cases is not true [54]. Approaches based on Kalman filter applied to IMU signal enhancement can be found in [123, 88, 34].

3.1.2 Complementary Filter

A complementary filter (Figure 3.1) acts as a low-pass filter for accelerometer data (removing accelerations from vibrations or movements) and as a high-pass filter for gyroscope data (excluding the trend in data that comes from the bias drift). This way, unlike Kalman Filter, a Complementary Filter consists only in a plain analysis in the frequency domain and does not consider any model for the noises that contaminate the measurements.

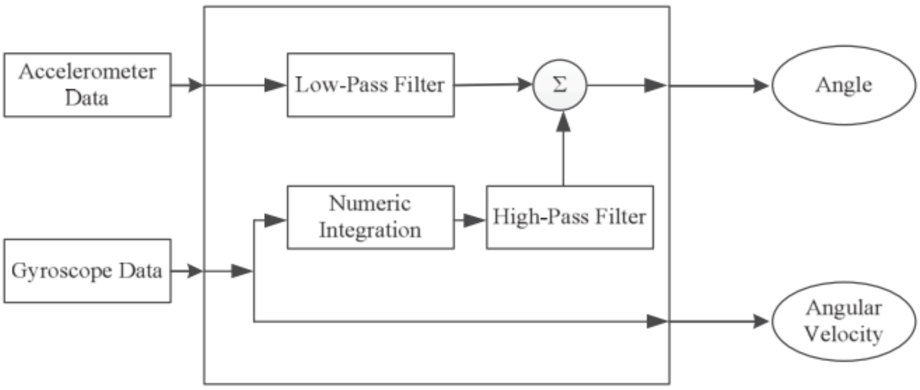


Figure 3.1: Complementary Filter, extracted from [50]

The low-pass function allows smoothing the measurements from the accelerometer. This way, the short-term fluctuations are removed, and the long-term ones are maintained. The opposite occurs when applying a high pass filter to the gyroscope measurements. The long-term variation that comes from the bias drift is removed while the short-term ones are maintained [50].

A Complementary Filter can be modelled as:

$$\theta = \alpha * (\theta + \omega_{gyro} * dt) + (1 - \alpha) * a_{acc} \tag{3.11}$$

α can be regarded as a smoothing factor. The higher its value, the greater is the confidence in the gyroscope measurements. A lower α is associated with higher confidence in the accelerometer readings.

It is a filter easy to implement, simpler than Kalman Filter and with a lower computational cost. However, depending on its application, often it is not sufficiently robust [50].

3.1.3 Gradient Descent Algorithm

The gradient descent algorithm, proposed by Madgwick et al. in [66, 67], uses a quaternion representation of orientation. The main goal is to find the orientation of the sensor in the Earth frame (assuming that for the movements of interest the earth rotation can be ignored) concerning the body frame (${}^B_E q$). This estimation is found, in an IMU, by fusing information measured with the gyroscopes and with the accelerometers. The problem is faced as an optimization problem solved using gradient descent. The goal is to minimize the difference of the z-axis direction estimation and the estimated direction of the gravity vector. The problem can be faced as:

$$\min_{{}^B_E\hat{q} \in \mathbb{R}^4} f({}^B_E\hat{q}\,,\, {}^E\hat{g}\,,\, {}^B\hat{a}) \tag{3.12}$$

$$f({}^B_E\hat{q}\,,\, {}^E\hat{g}\,,\, {}^B\hat{a}) = {}^B_E\hat{q}^* \otimes {}^E\hat{g} \otimes {}^B_E\hat{q} - {}^B\hat{a} \tag{3.13}$$

In which ${}^E\hat{g}$ represents the normalized gravity vector (${}^E\hat{g} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$), ${}^B\hat{a}$ the normalized acceleration measurements and ${}^B_E\hat{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}$.

In the first step of this algorithm the gyroscope (${}^B\omega_t$) and the accelerometer (${}^B a_t$) measurements are collected. The orientation is estimated using the previous estimated orientation and the angular velocity. Subsequently, the orientation estimation is incremented with the objective function gradient, ∇f using the objective function, f, and the Jacobian, J, based on the acceleration measurements:

$$\nabla f({}^B_E\hat{q}_{est,t}, {}^E\hat{g}, {}^B\hat{a}_{t+1} = J^T({}^B_E\hat{q}_{est,t}, {}^E\hat{g}) f({}^B_E\hat{q}_{est,t}, {}^E\hat{g}, {}^B\hat{a}_{t+1}) \tag{3.14}$$

$$f({}^B_E\hat{q}_{est,t}, {}^E\hat{g}, {}^B\hat{a}_{t+1}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 - q_3q_4) - a_y \\ 2(\frac{1}{2} - q_2^2 - q_3^2) - a_z \end{bmatrix} \tag{3.15}$$

$$J({}^B_E\hat{q}_{est,t}, {}^E\hat{g}) = \begin{bmatrix} -2q_3 & 2q_3 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{3.16}$$

With the updated attitude term calculated as:

$${}^B_E q_{\nabla,t+1} = {}^B_E q_{est,t} - \mu \frac{\nabla f({}^B_E\hat{q}_{est,t}, {}^E\hat{g}, {}^B\hat{a}_{t+1})}{||\nabla f({}^B_E\hat{q}_{est,t}, {}^E\hat{g}, {}^B\hat{a}_{t+1}||} \tag{3.17}$$

The term μ allows to control the convergence rate of the estimated orientation, and its value must be lower than the physical orientation rate.

The orientation estimation is also incremented using the gyroscope measurements:

$$ {}^{B}_{E}q_{\omega,t+1} = \frac{1}{2}\hat{q}_{est,t} \otimes \left[0, {}^{B}\omega_{t+1}\right]^{T} \Delta t \tag{3.18} $$

In which Δt represents the elapsed time between t and t+1.

Finally, the attitude is estimated fusing the attitudes estimates from the gyroscope and from the accelerometer such that high-frequency errors from the accelerometer and the gyroscope drift are compensated:

$$ {}^{B}_{E}q_{est,t+1} = \gamma_t \, {}^{B}_{E}q_{\nabla,t+1} + (1-\gamma_t) \, {}^{B}_{E}\dot{q}_{\omega,t+1} \quad , \; 0 \leq \gamma_t \leq 1 \tag{3.19} $$

This filter allows the alignment of the z-axis estimate with the earth's gravity with high accuracy and is very efficient in term of computation. Unlike other applications – in which gradient descent is used interactively until the desired solution is reached– in this case an adequate accuracy is achieved in a single step.

3.1.4 Kalman Filters with other information sources

Recently, some approaches have been proposed that combine sensor fusion using Kalman Filters and constrains from other information sources.

In [100] Solin *et al.* (2018) combined inertial and video data through the use of an Extended Kalman Filter (EKF). A major focus was placed in the use of inertial data to guarantee that the systems still works in occlusion conditions.

The same authours presented an interesting approach in [99] (2018) in which the authors use a method based on an EKF applied to inertial data from MEMS IMU sensors of a smartphone that is placed in several positions. This EKF faces the IMU signals as control signals and their measurement noises as process noise. The authors use position fixes, loop closures, zero-velocity updates, velocity pseudo-measurement updates, and also barometer readings and fuse them with the model to improve the estimates for position, velocity, and orientation performed by the EKF.

Based on that method, Cortés *et al.* (2018) in [30] proposes a deep learning-based model using a CNN for estimating the velocity of an IMU using data from a smartphone again. This way, the authors use a CNN to regress the scalar speed of the device on a moving window. That value is used as a constraint on an EKF instead of the pseudo-measurements presented by Solin *et al.* in [99], with an increase in performance.

3.1.5 Zero Updates

Zero Updates and motion constraints can be useful to bound the sensors errors. Applying constraints means to use information regarding the motion of the IMU carrier (that can be a subject, a vehicle or a robot) to limit and correct the errors present in the inertial measurements. These techniques are frequently used in land vehicles movements and pedestrian dead reckoning (PDR). In PDR by identifying periods in which the foot is fixed on the ground - and therefore has an angular velocity and a horizontal acceleration close to zero with a vertical acceleration that corresponds to the gravity acceleration - it is then possible to compensate for errors which result in non-zero acceleration or angular velocity. Although zero update algorithms are mainly used in pedestrian dead reckoning and periodic motions, these can be applied to other types of actions [8].

3.1.5.1 Zero-Velocity Update (ZUPT)

As the accelerometers measurements are noisy, when these signals are integrated to estimate velocity, this velocity presents a drift. This way, the calculus of a displacement (which occurs by the integration of the velocity signal) is incorrect. Since cumulative errors contaminate it, the error will grow with time (cubic-error growth) [98]. By determining the periods in which the IMU is still, it is possible to adjust the velocity estimation eliminating the drift. This adjustment is usually made with a curving fitting method or with a Kalman Filter [119].

The identification of the zero velocity periods is traditionally conducted using threshold-based methods [115, 60]. The problem with these algorithms is that they are dependent on the application and IMU performance [119]. Some implementations try to identify the motion to adjust the used thresholds [68] while others use information from other sensors (like pressure sensors [64] or additional accelerometers [121]) to assist these periods recognition. Popular algorithms for the detection of these time instants include the "acceleration-moving variance detector", the "acceleration-magnitude detector" (which use accelerometer signals) and the "angular rate energy detector" (which analyses the gyroscope signals) [98]. For instance in [99], the authors use an iterative Dickey-Fuller stationarity test [35] using a moving window of accelerometer data taking also into account the standard deviation of the signal.

Other approaches like the one presented by Akeila, Salcic and Swain (2014) [8], also use certain assumptions to identify zero velocity periods in indoor inertial navigation systems. This way, when the measured acceleration is zero or is constant during a specific period, then it is assumed that the velocity is zero instead of constant or linearly increasing. This kind of assumption is also interesting for human motion tracking applications due to the characteristics of the human movement.

Park et al (2016) [80] introduced a methodology for stationary periods identification that uses two SVMs. The first one was used to identify the type of motion while the second was used to recognize the stationary periods. This technique was applied using foot-mounted sensors and to recognize zero velocity periods during gait with promising results.

More recently Wagstaff and Kelly (2018) [109] proposed a methodology that tries to overcome the limitation of the threshold-based methods by implementing a LSTM to recognize stationary periods in inertial measurements of a foot mounted IMU and adjusting the velocity with an Extended Kalman Filter. The results show that the LSTM-based method to reset the velocity was able to improve the position recognition by 34% in a foot travelled distance of 7.5 km when compared to the traditional methods.

3.1.5.2 Zero-Angular Rate Update (ZARU)

ZARU can be performed performed together with ZUPT or separately. The goal of this update is to compensate the the gyroscope noisy measurements by identifying periods in which the angular rate is zero. By detecting these periods it is possible to compensate the drift of the gyroscope [51].

For PDR problems, the application of ZARUs must be performed with caution. Although often the angular rate of turn is considered to be zero during the stance phase of the human gate, the residual angular motion can exist and affect the application of this algorithm [49].

3.2 End-to-End AI Aproaches

Recently some methodologies have proposed to look at the trajectory estimate problem using an end-to-end technique or similar techniques. These include approaches that, using the inertial data, try to estimate both position and orientation of an IMU (end-to-end). Others try to use the IMU data to guess its velocity and correct the acceleration readings or integrate it to produce a displacement estimation. These techniques have consistently obtained better results than simple INS or pedestrian dead reckoning based approaches. Throughout this section, some of those methods are presented.

Some of the approaches fused video and inertial data using neural networks to perform the 6-DOF motion estimation. Some examples of this kind of methodologies are presented in [26, 21]. However, these examples require a higher computational cost as well as a specific set of conditions for an accurate estimate, like the fact that the camera needs to be unobstructed.

Taking those problems into consideration, other authors use uniquely inertial data to solve the problem. For instance, Yan *et al.* in [118] presented *RIDI* (2018), an approach which regresses the velocity vector using inertial data and uses that velocity to correct the accelerometer readings. The corrected linear accelerations are then integrated twice to provide a trajectory estimate. Data is acquired from a smartphone positioned in several placements. A Support Vector Machine (SVM) is trained to classify the placement of the smartphone, and two placement-specific Support Vector Regressors (SVR) are used to estimate the 2D velocity. Instead of performing simple integration of the regressed speed, the authors model a low-frequency bias to correct the linear acceleration. Finally, they double integrate the corrected accelerations producing a position estimate. The results are similar to the ones achieved using inertial navigation associated with video and, therefore, quite promising.

In [20] Chen *et al.* proposed *IONet* (2018), an AI approach that uses fixe-sized windows of inertial data to predict the trajectory of the IMU. This way, the input of their neural network is a vector of inertial data, and the output is a polar vector. The authors presented different architectures for the deep neural network: a Recurrent Neural Network (RNN), a Convolutional Neural Network (CNN), a 1-Layer Long-Short-Term Memory (LSTM), a 2-Layer LSTM, and a 2-Layer Bidirectional LSTM (BLSTM). The results suggest that these methodologies surpass traditional inertial navigation systems and pedestrian dead reckoning approaches.

The same authors that proposed *IONet* [20] recently presented *L-IONet* (2020) in [23]. This is a similar technique but that replaces the used LSTMs by WaveNet [78] (popular for its applications in audio generation). This model allows to continue to take time dependencies into consideration while improving the training time and reducing the computational cost.

Feigl *et al.* (2019) in [42] used a windows-based method involving the signal magnitude vector of the linear acceleration and the angular rate. Several ML-based and deep learning (DL) models were used to regress the signal magnitude vector of human velocity. These include a Classification and Regression Tree (CART), a SVR and Gaussian Processes. A DL method was also used applying a CNN combined with a BLSTM. The results show that the CNN-BLSTM yield the best results in instantaneous velocity estimate and travelled distance error. The model also showed the ability to generalize to different rates of motion.

More recently, Lima *et al.* (2019) in [96] proposed and end-to-end approach that uses data from a low-cost IMU. The authors used a neural network that combines both CNN and BLSTM layers to estimate the trajectory of the IMU. The loss function of the network was adapted to consider both the translational error and the orientation error. The results indicate a higher accuracy when compared with inertial odometry techniques considered state of the art.

RoNIN (2019) was introduced by Yan *et al.* in [117]. Again, inertial data from a smartphone IMU is used to estimate position and orientation. Three different architectures were tested: a 1D version of a Resnet [53], a LSTM, and a Temporal Convolutional Network (TCN) [12]. For the position estimate, the ResNet was trained to predict a translation. LSTM and TCN architectures, on the other hand, were used to regress instantaneous velocities which were integrated. To estimate the IMU orientation, the LSTM architecture was used to regress a 2D vector which represents the sin and the cos values of the heading angle.

Esfahani *et al.* (2019) in [39] presented *AbolDeepIO* which used DL to estimate the movement of a robot. For that purpose, the authors used several LSTM layers to learn features present in the accelerometer and gyroscopes signals separately. The time elapsed between two measurements is also passed through the same number of LSTM layers. The output is therefore concatenated using slow fusion and passed trough another combination of LSTM layers. The output is the position and orientation change.

Afterwards, the same authors in [40] presented *OriNet* (2019). This framework uses gyroscope readings and a deep learning methodology to improve the orientation quaternion estimate. The network consists of two sections. The first one is composed by four LSTMs which take respectively as input the angular rate from the current timestep, the angular rate from the previous timestep, the sampling period and the previous quaternion. The outputs of each LSTM are concatenated and passed trough another LSTM followed by a Fully Connected (FC) layer. The final output is the displacement quaternion. The second part consists of using the Genetic Algorithm (GA) to perform bias calibration. This is used to calibrate the angular velocity measurement that is inserted in the previously mentioned LSTM. The results show improvements relative to solutions that use cameras to perform orientation estimation.

3.3 Summary

Sensor fusion algorithms based on IMU data, combine information from accelerometers and gyroscopes, allowing the improvement of translation and attitude estimations. Some popular algorithms with these characteristics include the Kalman filter, the Complementary Filter and the Gradient Descent Algorithm. Recently some interesting techniques have been proposed that produce motion contraints that can be integrated in those sensor fusion algorithms. These constraints include, for instance the detection of zero-velocity.

ZUPT algorithms allow to detect stationary periods and subsequently correct the velocity estimated. The most commonly used algorithms are threshold-based. However, these algorithms have difficulties with the existence of an ideal threshold for different types of motions. More recently, some AI-based techniques have also been proposed as solutions for these problems with promising results. Detecting zero velocity with high accuracy is challenging and can be extremely useful in displacement reconstruction.

As a way of replacing sensor fusion algorithms, lately, diverse methodologies that use AI to estimate the trajectory of an IMU have been presented and consistently outperformed the traditional approaches as inertial navigation systems methods or pedestrian dead reckoning. These AI techniques include end-to-end strategies (which use inertial data to guess straight position and orientation) and more limited approaches (which only try to estimate the IMU velocity or orientation, for instance).

Chapter 4

Dataset Selection

4.1 Problem Definition

The tracking of translational motion is a challenging task in a human motion tracking process based on inertial data. Theoretically, it should be possible to calculate the displacement of an IMU in a 3D space using only the acceleration values provided by its accelerometers, computing a simple double integration. However, due to errors like noises, biases, and drifts present on the acceleration values, this calculation is not reliable after only a short period. Besides, a wrong estimation of the orientation of the device causes gravity "leakage". This gravity "leakage" means that the acceleration caused by gravity is wrongly understood as an acceleration caused by movement and so the calculation of velocity and position is corrupted [96, 99].

This problem is particularly relevant when conducting human motion tracking and tracing different body segments during a rehabilitation session in which, commonly, actions are short and repetitive. So it is pertinent to obtain accurate translation measurements.

Taking into account a human motion tracking process, two challenges were identified. The solution of these tasks can lead to improvements in the estimate of translations performed by several body segments when these are associated with an IMU. It is possible to define these problems as:

1. The first tackled problem can be defined as a classification problem. In this problem, the main goal is the identification of periods in which the IMU is still. The identification of stationary moments can be used in zero-velocity update algorithms allowing a better estimate of position by permitting the correction of the velocity estimates.

2. The second addressed challenge can be defined as a regression problem. After identifying the moments in which the IMU is moving, it is essential to understand the translation performed by the IMU during those periods. Again, relying solely on the double integration of the acceleration data can lead to wrong estimations of displacements. This way, a better estimate of movements, especially in small displacements like the ones performed often during a rehabilitation session can be crucial in a human motion tracking process.

4.2 Analysed Datasets

As in most ML projects, the selection of an appropriate dataset is a vital step of the process. In this project, particularly, the need was for a dataset which contained inertial data from several body segments collected during the execution of movements similar to the ones performed during a typical rehabilitation session. The dataset needed to provide inertial motion from human actions since these are the main focus of this work and due to their singularities when compared with robots or vehicle motion. Additionally, the dataset needed to possess data collected from various movements that could approach the diversity of actions performed during a rehabilitation session. The presence of data from several body sections was also essential to allow the development of algorithms that could generalize to several body sections.

Also crucial in the same dataset was the presence of a ground truth labelling which indicates the correct positioning of the IMU throughout time so that supervised machine learning algorithms can be developed.

The first considered option was to build a dataset using the inertial motion trackers from SWORD since these provide the ability to collect inertial data from several body segments. However, the unavailability of a system that could deliver the accurate position of each motion tracker throughout time (like an optical motion capture system) made this option non-viable.

Bearing these features in mind, a search for a public available appropriate dataset was conducted. The analyzed datasets are presented in this section and are summarized in Table 4.1.

KITTI Dataset [47]

This dataset includes IMU accelerations from a combined GPS/IMU inertial navigation system, GPS measures, and video recordings as well as laser scans. The data was collected using a setup placed on top of a moving car. Therefore, the dataset is mainly directed to applications related to autonomous driving and computer vision.

This way, although the dataset possesses IMU data and GPS data that can be eventually be used as ground truth, this dataset does not contain human movements and, consequently is not relevant to this project.

KIT Whole-Body Human Motion Database [69]

This dataset contains several human motions as walking at different paces, over obstacles, climbing stairs, running, turning, etc. The movements were captured using a Vicon motion capture system and video recordings. All the data is labelled according to the motion that is being performed by the subjects. Data from a total of 43 different participants was collected.

From the point of view of motion diversity, this is an interesting dataset. However, the lack of inertial data, which is the main focus of this project makes this dataset inadequate.

EuRoC MAV Datasets [18]

These datasets comprise data collected using micro aerial vehicles (MAV). They all contain image data and synchronized inertial data. On some of them, the ground truth labelling is performed using a laser tracking system, while on others, it is conducted with a Vicon motion capture system.

Again this dataset is suitable for research related to vehicle movement but is not appropriate for studying human motion.

Oxford RobotCar Dataset [65]

This is another example of a dataset focused on autonomous driving problems. It contains video data from several cameras, GPS data, inertial data, and LIDAR (Light Detection And Ranging) data from a total of over 1000 km.

The sensors are rigidly attached to a vehicle, and therefore the dataset is not suitable for human motion studies.

TUM VI Dataset [91]

This dataset's primary goal is to evaluate visual-inertial odometry. This way, data is collected using cameras and IMU measurements using a device that is held by a subject in front of him. However, for the majority of the acquisitions, accurate groud truth is only provided at the beginning and at the end of the collection with an optical motion capture system (OptiTrack Flex13) placed in a room in which the trials begin and end.

Thus, although this dataset represents some human motion, the absence of a continuous accurate ground truth makes it unsuitable for this project.

ADVIO Dataset [31]

In this dataset, the authors created a pseudo ground truth using a handcrafted inertial odometry algorithm that uses data from an IMU and position fixes determined with an external reference camera. The data is collected using an Apple iPhone 6s, a Google Tango, and a Google Pixel. The devices are handheld while the subjects walk. Camera images and raw data from the devices are collected during the entire sequence.

The lack of accurate ground truth and the insufficient diversity of motions are the main issues of this dataset.

OxIOD Dataset [22]

This dataset contains inertial and magnetic data from mobile devices positioned in several placements as the hand, pocket, handbag, and a trolley. Several trials were conducted using different subjects and different motion types, which include halting, walking slowly, normally, and running. In the majority of the data acquisition sequences, the ground truth is obtained through the use of a Vicon (Optical Motion Capturing System) which allows an accuracy of 0,5 mm. It contains large amounts of data (a total of 42.587 km) which makes this dataset suitable for the use in data-driven applications.

The main disadvantage of this dataset is the fact that most of the data is recorded with a walking/ running subject. This way, the motion is repetitive and lacks diversity as the movements performed during a rehabilitation session.

CMU Graphics Lab Motion Capture Database [1]

This dataset contains a large variety of motions which can be inserted in categories as: human interactions (shake hands, football - throw and catch, etc.), interaction with the environment (grip bar, swing body, etc.), locomotion (walking, jumping, running), physical activities (golf, boxing, etc.) and situations and scenarios (basketball signals, hand signals, etc.). A large amount of data and the diverse types of existent exercises make this dataset captivating for human motion studies.

However, the data gathered only includes information collected simultaneously using a Vicon motion capture system and video recordings. This way, the lack of inertial data makes this dataset inappropriate for the project goals.

Human Movement and Ergonomics: an Industry-Oriented Dataset [71]

This dataset was designed to simulate human motion in working conditions in the industry. It contains inertial data collected using a Xsens MVN Link system (Xsens, Enschede, Netherlands) which comprises 17 IMUs placed in several body segments. Various trials were conducted with a total of 13 subjects performing activities as screwing, untying knots, or carrying loads across a room for a total of 5 hours. The participant had to raise the arms, bent the torso, or crouch to perform the mentioned activities. The ground truth was collected with a Qualisys Optical Motion System (Qualisys, Göteborg, Sweden). Video and data from an E-glove are also provided.

The main disadvantage of this dataset is the absence of accurate ground truth for the device's orientations. On the other hand, the high quantity of data present makes it suitable for data-driven applications. Also important is the fact that it contains various movements with data collected from several body segments.

Taking into consideration the needs of the projects and the aforementioned datasets (which are summarized in Table 4.1) it was considered that the most appropriate dataset to face the problems defined previously was the "Human Movement and Ergonomics: an Industry-Oriented Dataset" [71]. Although the movements collected are not from a rehabilitation session or physical exercise, the diversity of actions existent as well as the fact that motion is captured from several body segments make this dataset interesting for the goals of this project. Furthermore, this dataset possesses also inertial information and data obtained using an optical system that can be used as ground truth.

[1]CMU Graphics Lab Motion Capture Database `http://mocap.cs.cmu.edu/motcat.php` Accessed on 19-03-2020

Table 4.1: Summary of the analysed datasets

Dataset	**Year**	**Carrier**	**Inertial Data**	**Ground Truth**	**Ground Truth Accuracy**	**Movement Description**
KITTI [47]	2013	Car	✓	Higher quality IMU	10cm	Moving car
KIT Whole-Body [69]	2015	Several human body placements	X	Vicon Optical System	Sub-millimeter	Walking, Running, Turning, etc
EuRoC MAV [18]	2016	MAV	✓	Laser Tracker/ Vicon Optical System	1mm	Moving Micro Aerial Vehicles
Oxford RobotCar [65]	2016	Car	✓	GPS + IMU	-	Moving car
TUM VI [91]	2018	Handheld	✓	OptiTrack Optical System (Only at the beggining and at the end)	1mm	Walking
ADVIO [91]	2018	Handheld	✓	Handcrafted inertial odometry algorithm	-	Walking
OxIOD [22]	2018	Handheld, Handbag, Pocket	✓	Vicon Optical System	Sub-millimeter	Halting, walking slowly, normally and running.
CMU Motion Capture 4.2	2019	Several human body placements	X	Vicon Optical System	Sub-millimeter	Human interactions, physical activity, locomotion, etc
Industry-Oriented [71]	2019	Several human body placements	✓	Qualisys Optical System	Sub-millimeter	Industry-like activities such as: screwing, carrying loads and untying knots

4.3 Dataset Description

In this section, a full description of the selected dataset -"Human Movement and Ergonomics: an Industry-Oriented Dataset" [71] - is presented. For more details regarding this dataset, [71] should be consulted.

As noted in the earlier section, the dataset contains data collected during the execution of different industry-like activities. Data was collected from a total of 13 subjects using a Xsens MVN Link System, a Qualisys Optical Motion Capture System, an E-glove, and video recording. From the participants, nine were male, and the other four were female. Their average age was 25.7 ± 5 years, their average height was 175.4 ± 7.9 cm, and their average weight 72.3 ± 14.4 kg. Each subject carried out 15 trials leading to a total of 195 trials (about 90s each) which corresponds to approximately 5h of recording.

Experimental Set-Up

The activities performed were inspired by the ones performed typically in the car manufacturing industry. Six different activities were performed by the participants (Figure 4.1) in different sequences. Maurice *et al.*[71] described the performed activities as:

- Screw High (SH): "Take a screw and a bolt on a 75 cm-high table, walk to the shelf, screw at the height of 175 cm (performed with bare hands, i.e., no screwdriver)"
- Screw Middle (SM): "Take a screw and a bolt on a 75 cm-high table, walk to the shelf, screw at the height of 115 cm"
- Screw Low (SL): "Take a screw and a bolt on a 75 cm- high table, walk to the shelf, screw at the height of 25 cm (6 participants) or 60 cm (7 participants)"
- Untie Knot (UK): "Untie a knot placed on a 45 cm-high table"
- Carry 5kg (C5): "Take a 5 kg load on a 55 cm-high table, walk to the shelf, put the load on a 20 cm-high shelf"
- Carry 10kg (C10): "Take a 10 kg load on a 55 cm-high table, walk to the shelf, put the load on a 110 cm-high shelf".

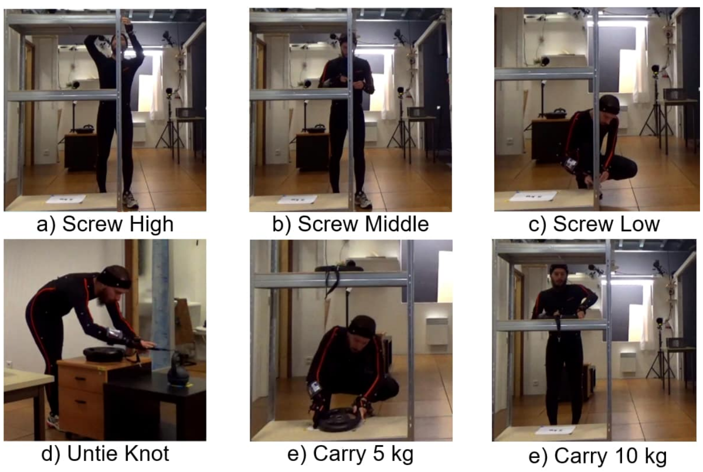


Figure 4.1: Activities performed during the data collection, from [71]

Each trial contained the six activities performed in a specific order. This way six different movement sequences were created (Seq. 1: SL-SM-SH-UK-C10-C5; Seq. 2: SH-SM-SL-UK-C10-C5; Seq. 3: SL-SM-UK-C10-SH-C5; Seq. 4: SL-UK-C10-SM-C5-SH; Seq. 5: UK-C10-SH-SM-C5-SL; Seq. 6: UK-C10-C5-SH-SM-SL). From the six possible sequences, three were attributed to a given subject that executed five trials of each. To add variability to the collections, the room was also organized in two different setups.

Xsens MVN Link System Data

This dataset contains data collected from 17 IMUs from a Xsens MVN Link system placed in different body parts. Accordingly to the authors of [71] the IMUs are placed: "1 sensor on the head, 1 on the sternum, 1 on the pelvis, 1 on each scapula, 1 on each upper arm, 1 on each forearm, 1 on each hand, 1 on each thigh, 1 on each shank and 1 on each foot". The placement of those IMUs is represented in Figure 4.2-a). This way, the magnetic field, 3D linear accelerations, and orientation were collected from each sensor.

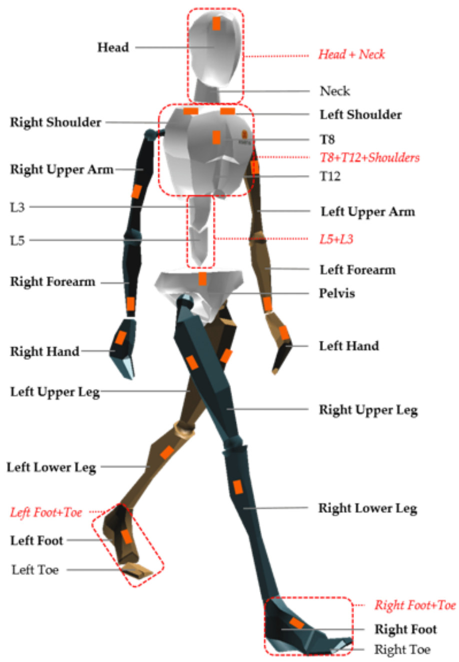


Figure 4.2: a) Placement of the 17 IMUs used in the Xsens MVN Link system, extracted from [87]; b) Xsens Avatar and representation of the body segments, extracted from [55]

Merging this data with a biomechanical model (that is adapted to each subject body) and using sensor fusion algorithms, data from 23 body segments is estimated – Figure 4.2-b) [87]. This data includes:

- 3D position, orientation (quaternion), linear and angular velocity, and linear and angular acceleration of the origin of the 23 body segments of the Xsens avatar.
- Angles of the 22 joints (3 DoFs each) of the Xsens avatar.
- 3D position of the Xsens avatar's centre of mass.

This data was captured using a sampling rate of 240Hz. At the beginning of each trial, the system was calibrated, and the Xsens world frame was reinitialized to match the Qualisys world frame.

Qualisys Optical Motion Capture System Data

This data is constituted by the 3D position of each marker, and consequently of each body segment to which the marker is attached. To collect this information, 43 markers of diameter 12.5mm were placed on the body of each subject, and 12 Oqus cameras were used to track the placement of the markers. 39 markers were placed accordingly to Vicon Plug-In Gait Marker Placement recommendations [2], as represented in Figure 4.3. The marker that, according to these recommendations should be placed on the sacrum was not utilized. The four extra markers were placed on the feet: two on each first metatarsal head and two on each fifth metatarsal head.

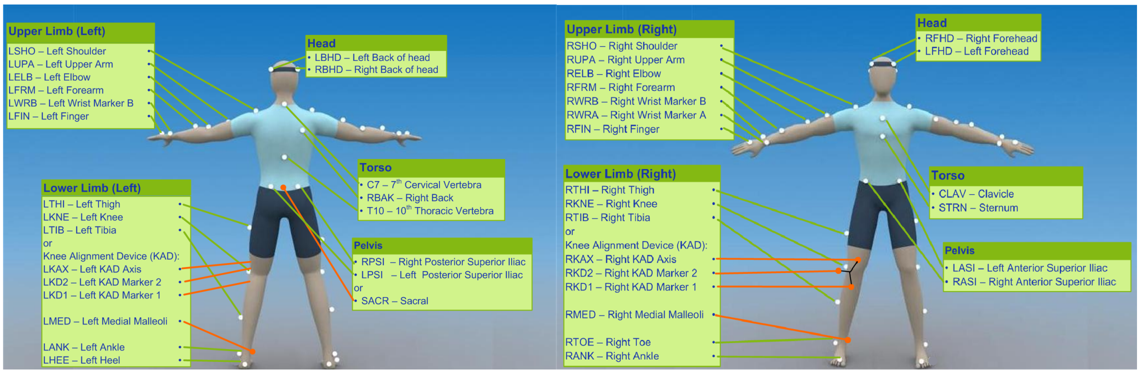


Figure 4.3: Vicon Plug-In Gait Marker Placement recommendations 2

This data was collected using a sampling rate of 120Hz and the system recalibrated at the beginning of each trial.

Sometimes, the markers were occluded from the Oqus cameras, and therefore data is not available from those instants. However, for 35 out of the 43 markers, the median of the percentage of the frames in which the markers are visible is above or equal to 95%. This way, it is possible to conclude that this lack of data does not harm the primary project goals.

Additional Data

Additional data was also collected, and although it was not used during this project, it is interesting to mention it:

- **E-glove** - Using an E-glove from Emphasis Telematics (Emphasis Telematics, Athens, Greece) placed on a single hand, data regarding flexion and pressure was collected.
- **Video Camera** - All trials were recorded using 2 video cameras.
- **Data Annotation** - The collected data was manually annotated by three independent annotators. This annotation included three different levels: general posture, specific posture, and current action.

[2]Vicon Documentation - Plug-in Gait full body modeling `https://docs.vicon.com/display/Nexus25/Plug-in+Gait+models+and+templates` Accessed on 19-05-2020

All the collected data (except the video recordings) was timestamped and, therefore, it is possible to synchronize both Xsens MVN Link system data and Qualisys Optical Motion Capture system data.

4.4 Data Pre-processing

Since that the inertial data from the Xsens system was captured at a rate of 240 Hz and that the data from the Qualisys system was captured at a rate of 120 Hz, the first step of the preprocessing was the downsampling of the Xsens data to 120 Hz. Then, the Unix timestamps - that were placed both on the inertial data and on the optical motion data - were used to interpolate the Xsens data linearly and that way synchronize both sources of information.

4.5 Dataset Analysis

Placements of markers and IMUs

Analyzing the placements of the IMUs and the placements of the Qualisys markers, it is possible to draw some conclusions. As previously referred, there are 17 IMU placements and 43 Qualisys marker placements, as represented in Table 4.3. The authors provide no information regarding the exact placement of both.

Table 4.3: Placements of the IMUs and markers of the motion capture systems (R\L- Right and Left)

Motion Capture System	Placements
Xsens MVN Link System	Head, T8, Shoulder(R\L), Upper Arm(R\L), Forearm(R\L), Hand(R\L), Upper Leg(R\L), Leg(R\L), Foot(R\L), Pelvis
Qualisys Optical System	Forehead(R\L), Back Head(R\L), Clavicle, Sternum, Right Back(R\L), Shoulder(R\L), Upper Arm(R\L), Posterior Superior Iliac(R\L), Anterior Superior Iliac(R\L), Elbow(R\L), Forearm(R\L), Wrist(A,B)(R\L), Finger(R\L), Thigh(R\L), Knee(R\L), Tibia(R\L), Ankle(R\L), Heel(R\L), Toe(R\L), Toe(A,B)(R\L), C7, T10

Although there are not IMUs placed in Right\Left Toe, in L5, in L3, in T12, and in the Neck, the model generated by the Xsens MVN System still provides inertial measurements concerning those body segments.

Analyzing both motion capture systems data, it is possible to conclude that there are six common placement names: Shoulder (R\L), Upper Arm (R\L), and Toe (R\L). This way, even though there is no information regarding on how close the Qualisys markers and the Xsens MVN IMUs of those placements are, a possible assumption is to presume that these are close enough so that position information provided by the Qualisys markers detection can be used as ground truth of the IMUs position throughout the time.

To support this hypothesis, an analysis was conducted considering the 3D position of the Qualisys markers and the estimate of the 3D position of the same body segments provided by the Xsens MVN model. The results of this analysis are presented in Table 4.4.

Table 4.4: Comparison of the position obtained by the Qualisys Optical Motion System and the position estimated by the Xsens MVN Link System (RSHO- Right Shoulder, LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

	X axis	**Y axis**	**Z axis**
RMSE RSHO	0.1052±0.0360 m	0.2369±0.1141 m	0.0537 ±0.0185 m
RMSE LSHO	0.1238±0.0422 m	0.2326±0.0649 m	0.0501±0.0180 m
RMSE RUPA	0.1117±0.0401 m	0.2243±0.0978 m	0.0678±0.0301 m
RMSE LUPA	0.1102±0.0394 m	0.2132±0.0905 m	0.0659±0.0264 m
RMSE RTOE	0.1073±0.0401 m	0.1984±0.0911 m	0.0540±0.0309 m
RMSE LTOE	0.1056±0.0411 m	0.1882±0.0906 m	0.0470±0.0310 m

Looking at those values, it is possible to conclude that all the axis present error and that for every placement, the error is similar in a given axis.

First of all, it is relevant to remember that the position data provided by the Xsens MVN system is an estimate and should not be faced as a highly accurate measurement. For instance, a previous study conducted by Damgrav *et al.* [33] identified a horizontal deviation of 2.860 cm per each meter of movement or a deflection of 2.666 cm per second when performing human motion tracking using the Xsens system. The authors concluded that longer and faster actions, with less contact with the ground, generate more drift.

Also interesting is to analyze the RMSE on the Z-axis, which is lower than the one presented in the X and in the Y-axis. Provided that the IMUs are not always under high linear acceleration (which is true for human movement), then these are very accurate in terms of vertical measurements. In contrast, IMUs fail in horizontal measurements because there is no way to compensate the gyro drift around global Z (without proper magnetometer corrections). To remove gravity from the accelerometer readings, one needs to know the orientation of the sensor relative to Earth. Since both accelerometers and gyroscopes are affected by errors, the estimation of that orientation is never totally accurate. This inaccurate orientation estimation leads to errors during the gravity removal process.

An additional analysis can be made considering, for instance, the example represented in Figure 4.4 (in which e represents the Earth coordinate frame, and b represents the IMU coordinate Frame). Due to the presence of an error of α, the orientation of the IMU is wrongly estimated as being the one represented by $'b$. This leads to an erroneous gravity removal: to the linear acceleration measured in the IMU x-axis, it will be removed a value of $g \times cos(\alpha)$. In contrast, the correct value to be removed was g (since the IMU x-axis is collinear with the Earth z-axis). To the linear acceleration measured in the IMU z-axis, no value due to gravity should be subtracted, however, due to the orientation estimation error, a value of $g \times sin(\alpha)$ will be removed. For a

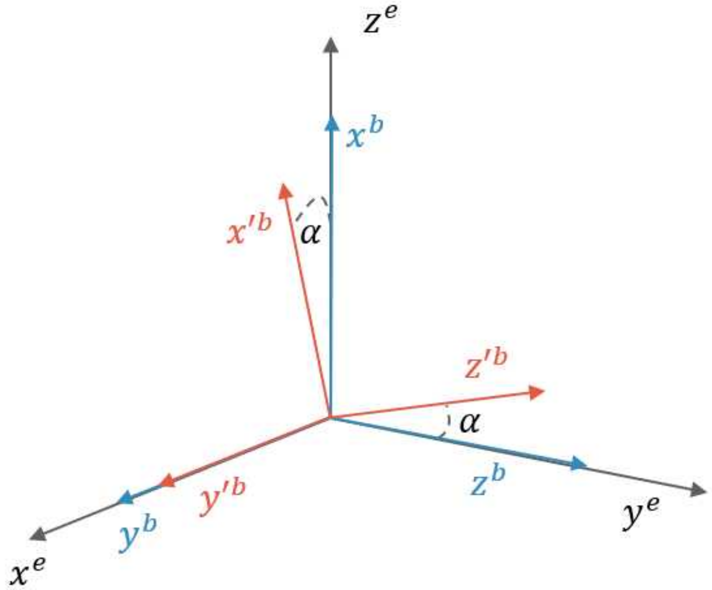


Figure 4.4: Incorrect orientation estimation of the IMU coordinate frame (b) in relation to the Earth coordinate frame (e)

small α, $\sin(\alpha)$ will be further away from 0 than $\cos(x)$ is away from 1 due to the derivative of the function at those two points. Therefore, a greater error is present on the non-vertical axis than on the vertical axis.

Another important fact is that the Xsens MVN link system integrates information using a biomechanical model of each subject (that takes into account each subject body measurements). This way, it should be easier for the system to estimate the height of a specific body segment, given that it can limit that estimate using body measurements. These body measurements can also help the model to constraint the motion on the horizontal plane but not so effectively.

Finally, it is also important to consider that even that the Qualisys markers are not placed on top of the IMUs with the same placement labelling, most likely these are placed close enough so that the translation performed by the IMUs is very similar to the translation performed by the markers.

This way, and considering the errors of the Xsens MVN position relative to the position provided by the Qualisys, one can state that it is acceptable to use the position of the markers as ground truth to the position of the IMUs with the same placement names.

Xsens and Qualisys data

To better understand the data existent in the dataset a visual representation of linear acceleration, angular velocity, and 3D ground truth position is illustrated in Figure 4.5.

Analysing this figure it is possible to conclude that the acceleration caused by gravity is already removed from the acceleration provided by the Xsens system. Also relevant is to see that that the axes in which more movement occurs are the X-axis and the Y-axis. The lack of movement in the Z-axis can also explain the smaller error in that axis between position estimated

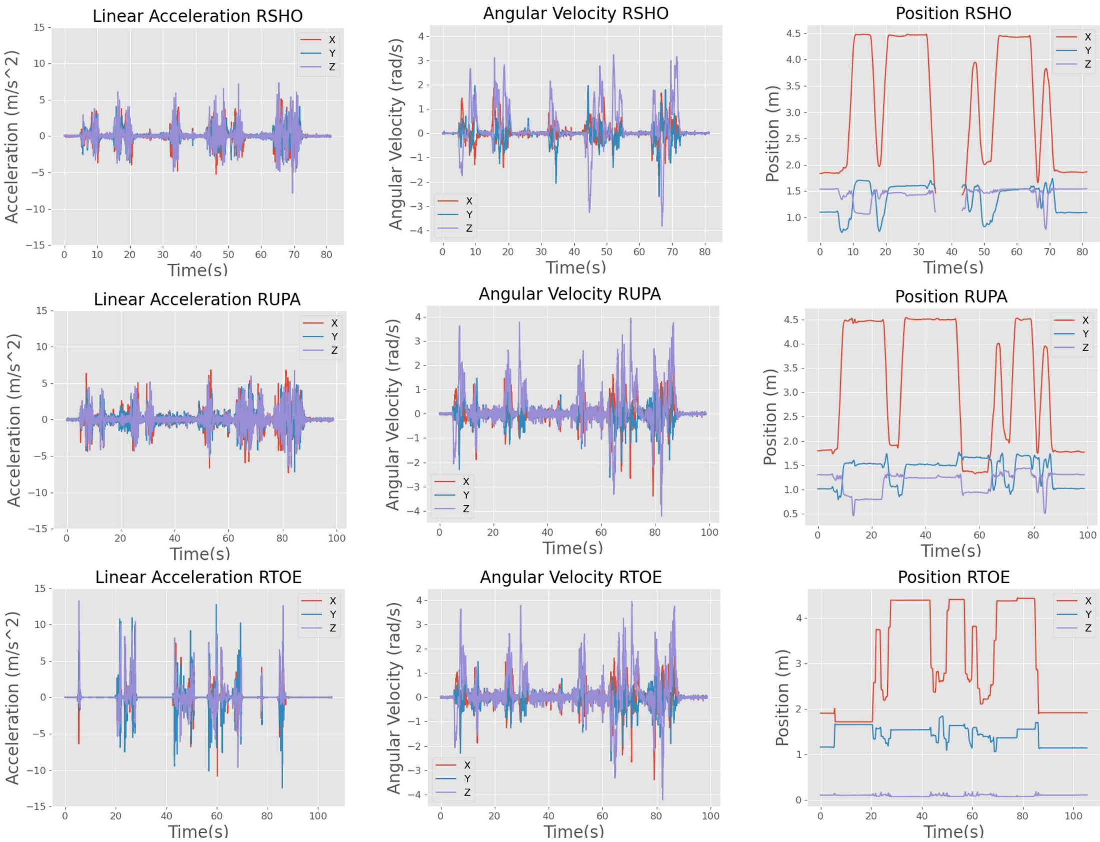


Figure 4.5: Linear acceleration, and angular velocity from the Xsens motion capture system and and 3D ground truth position from the Qualisys optical system. Different trials are represented for each acquisition placement.(RSHO- Right Shoulder, RUPA- Right Upper Arm, RTOE- Right Toe)

by the Xsens model and ground truth position since that a smaller distance will generate a smaller error.

Looking at the linear accelerations of the right toe it is perceivable that the range of values is different from the one existent in the upper arm and shoulder data. This higher range shows that the toe movements occur frequently with a higher acceleration. Since that the range of position is very similar between all signals this can lead to the conclusion that the toe movements are more short and quick. This way, it is expected that the use of inertial data from the toes placements can introduce more generalization ability to the developed models allowing their application to data from other IMUs and to other movements.

Additionaly, in Figure 4.5, particularly in the plot from the right shoulder ground truth position, a period in which the Qualisys markers were occluded is visible (represented by the lack of data on that period of time). Those periods of occlusion must be taken into account when using the Qualisys data.

4.6 Summary

The tracking of translations motions during a human motion tracking process based on inertial data is the main challenge of this project. This challenge was separated into two different problems: 1. zero-velocity detection, that can be faced as a classification problem; 2. translation estimate between zero velocity moments, that can be faced as a regression problem.

Taking these problems into account a search for an appropriate dataset was conducted and several datasets were analysed. The selected one was the "Human Movement and Ergonomics: an Industry-Oriented Dataset" [71]. This dataset provides a series of human motions performed typically in the car manufacturing industry and includes inertial data from several IMUs as well as position data from a highly accurate optical motion system. The diversity of movements and the existence of data from several body placements make this dataset suitable for the mentioned problems. An analysis of this dataset is presented throughout this chapter.

Chapter 5

Zero-Velocity Detection - a Classification Problem

As described previously, the first problem tackled in this dissertation can be regarded as a classification problem. The goal was to identify the periods in which the IMU and respective associated body part are stationary and the periods in which these are moving using just the inertial data. The identification of periods of non-movement can then be used to adjust the velocity estimation allowing to eliminate some of the drift that occurs when a noisy acceleration signal is integrated. This way, it is possible to obtain a more accurate estimate of displacement.

Regarding this binary classification task, the first step consisted in using the 3D position provided by the Qualisys motion system (ground truth position) to label the time periods into stationary or moving periods. Afterwards, those labels were used as output to train supervised ML algorithms. The inertial data of the IMUs was used as input of the same algorithms.

This way, in the next step, different ML classification algorithms were tested: a Logistic Regression (LR), a model which models the likelihood of an instance belonging to a class using a logistic function; a Support Vector Machine (SVM), a ML model that uses the available data to find support vectors which can guarantee an optimal separation between classes; a Random Forest (RF), a parallel ensemble of Decision Trees; a Densely Connected neural network (Dense), a feed-forward Deep Learning (DL) model; and three different models based on Long Term Short Memory (LSTM) neural networks, a type of recurrent neural network (RNN).

In theory, the LSTM neural networks should be the more suitable model to face the problem of detecting stopped or movement periods in human motion tracking due to their recurrent architecture which provides them the ability of capturing temporal/ sequential information from the data. Additionally, when compared with simpler RNNs, the LSTMs are capable of capturing longer time dependencies without being as much affected by vanishing or exploding gradient problems [77]. However, as other DL models, LSTMs are models difficult to interpret and are more complex than other ML models requiring higher sample sizes to avoid overfitting problems. Therefore, to guarantee that LSTMs are an adequate model to solve this problem, more traditional ML methods were also experimented to act as baseline for the LSTMs' performance.

Firstly the models were trained and evaluated using a combination of data from several body segments. Later, to understand how well a model trained using data from a single body segment could generalize to other body segments, these different models were also trained using only data from one body segment at a time and tested in the data from the other body segments.

Finally, the classification results were used to improve the translation estimates allowing to set the velocity of the IMU to zero when the period is classified as stationary.

All the experiments were conducted in Python, using packages such as scikit-learn [83] (ML models and several interesting functions), Keras [1] with Tensorflow [2] backend (Neural Networks training) and HyperOpt [3] (for Bayesian Hyperparameter Optimizattion).

5.1 Methods

5.1.1 Automatic labelling of Qualisys Data

As aforementioned, to solve this binary classification problem, the first step was to adapt the available labels to binary labels. This labelling means that the ground truth 3D position provided by the Qualisys optical system was converted to a binary label that indicates either if the body segment is moving or if it is stopped. This method was implemented dividing the Qualisys 3D signal in several windows and evaluating the translation that occurred inside each window. Given a 3D position window of N frames, the label of the window was placed in the frame N/2. This process is represented in Figure 5.1.

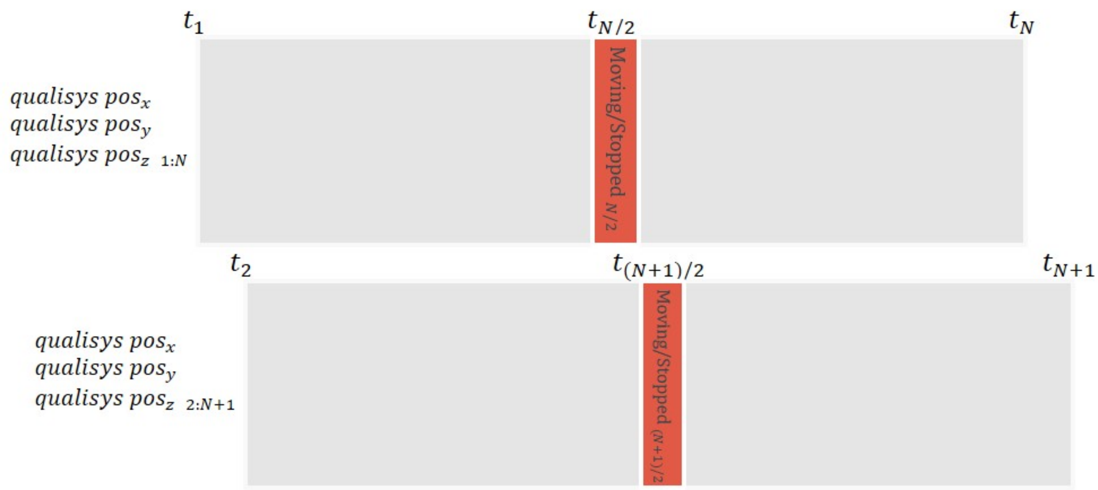


Figure 5.1: Automatic labeling using Qualisys Data

The algorithm used to create and label windows is described in Algorithm 1.

The first step of the algorithm is defining a threshold *thresh*, a "hysteresis term"*T*, and a window size *N*. If the total movement inside the evaluated window is above the specified limit, then it is considered that the respective body segment is moving. Else, it is admitted that the associated body segment is stopped. To introduce hysteresis in the algorithm, it was considered that if the body segment was stopped, then the movement needed to be T times above the defined threshold to change its state.

[1]Keras - https://keras.io/ Accessed on 02-06-2020
[2]Tensorflow - https://www.tensorflow.org/ Accessed on 02-06-2020
[3]HyperOpt - http://hyperopt.github.io/hyperopt/ Accessed on 02-06-2020

Algorithm 1 Labelling Qualisys Data

$thresh \leftarrow$ threshold
$N \leftarrow$ window size
$T \leftarrow$ hysteresis term
$state \leftarrow$ 'stopped'
$L \leftarrow$ *signal lenght*
for `n = 0 to L-N` **do**
    $mov \leftarrow \sum_{k=n}^{n+N-1} \sqrt{(pos_x[k+1] - pos_x[k])^2 + (pos_y[k+1] - pos_y[k])^2 + (pos_z[k+1] - pos_z)^2}$
    if $state =$ 'stopped' **then**
        if $mov > T \times thresh$ **then**
            $state \leftarrow$ 'moving'
        end if
    else
        if $mov < thresh$ **then**
            $state \leftarrow$ 'stopped'
        end if
    end if
end for

To evaluate if this algorithm is capable of classifying the Qualisys position data as desired, a randomly selected subset of the dataset was extracted and labelled manually. To perform the manual labelling, the magnitude of the position vector was visually inspected and marked. Out of the 13 subjects that conducted the trials, the manual labelling was performed on one trial for six patients. For the selected trial the labelling was made in each of the six selected placements (Shoulder (right and left), Upper Arm (right and left), and Toe (right and left)). This means that a total of 36 acquisitions were manually labelled. Finally, the manual labelling was compared with automatic labelling.

The threshold was defined as 1cm, and the window size as 30 samples. This means that in each window of 30 samples – which corresponds to a window 0.25s considering the 120 Hz sampling rate – if the movement is above 1cm then it is considered that the body segment is moving. In other words, if the average velocity in that window is above 0.04m/s, then it is considered that the body segment is moving.

To introduce hysteresis the value T was defined as 3. This implies that if the body segment was previously at rest, then the threshold is elevated to 3cm. The results of the comparison of the automatic labeling with the hand labeling are presented in Table 5.1. The positive class was considered to be the "*stopped*" class.

The accuracy values of this labelling are all close to 0.90 as well as the precision and recall values. The automatic labelling of the toes data is the one that presents a lower performance.

Overall, the results suggest that the developed algorithm is an adequate method to perform the labelling of the 3D placements and that the generated labels can, therefore, be used as ground truth for the subsequently developed supervised machine learning models.

Table 5.1: Results of the automatic labeling when compared with the manual labeling (RSHO- Right Shoulder, LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Placement	Accuracy	Precision	Recall
RSHO	0.9564 ± 0.0036	0.9722 ± 0.0126	0.9471 ± 0.0126
LSHO	0.9375 ± 0.0187	0.9499 ± 0.0264	0.9345 ± 0.0264
RUPA	0.9134 ± 0.0590	0.9290 ± 0.0333	0.9107 ± 0.0333
LUPA	0.9469 ± 0.0151	0.9691 ± 0.0166	0.9359 ± 0.0166
RTOE	0.8907 ± 0.0270	0.9487 ± 0.0221	0.9126 ± 0.0221
LTOE	0.8881 ± 0.0358	0.9306 ± 0.0365	0.9268 ± 0.0365
Total	0.9222 ± 0.0415	0.9499 ± 0.0376	0.9279 ± 0.0290

5.1.2 Zero-Velocity Detection - Multi-placement

As referred in the previous chapter, the data in which the work focused was data from the shoulder (right and left), upper arm (right and left) and toe (right and left). This way, after creating a ground truth which indicated if the respective body segment is moving or not moving, the next step consisted in evaluating how well would different models perform after trained using data from all the selected placements. To perform this task, only a representative portion of the whole dataset was used due to the high computational cost that using the entire dataset would carry. The generation of the train and test set is described in Figure 5.2.

The data was split in such a way to guarantee the presence of data from several body segments in both sets. This can increase the generalization ability of the developed models allowing their appliance to a higher number of body placements. Additionaly, data from a particular participant is only present in the train set or in the test set. This way, it is possible to ensure that the models did not learn any particularities from the participant trials. These particularities can be, for instance, specific errors that can happen during the initialization of the Xsens system

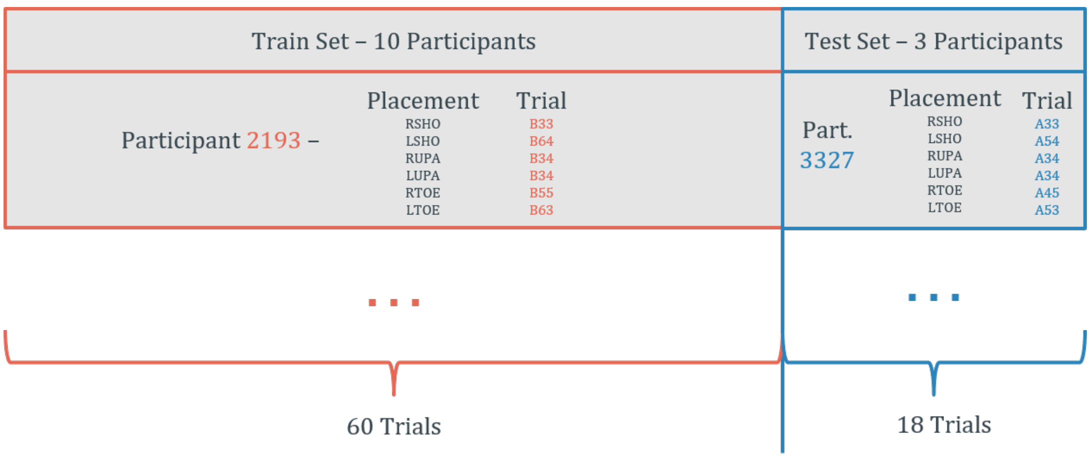


Figure 5.2: Train and test split

that can affect the correct gravity removal or particular placements of the IMUs relative to the Qualisys markers. This split also secures that data from several trials is present in both sets.

A total of 60 trials was used in the train set and 18 were used in the test set. The total duration of the train set is 6304.72 s, and the length of the test set is 2057.31 s. A total of 3.55% of the time of the train set corresponds to moments in which the Qualisys markers were occluded and therefore no ground truth information related with the position is available (making these samples unavailable for the models development and evaluation). In the test set, this value is 2.94%. The mean duration of each trial in the train set is 105.08 $\pm$ 15.52 s and in the test set 114.29 $\pm$ 17.97 s. The imbalance of the dataset in terms of movement/rest moments was evaluated to guarantee that both train and test datasets are balanced. In the training set, about 59.50% of the moments correspond to moving moments. In the test set, this value is 62.32%.

All those statistics support that the proposed split allows the correct development and evaluation of ML models.

5.1.2.1 Machine Learning Models

Three different ML models were evaluated: a Logistic Regression (Log-R), a Support Vector Machine (SVM) and a Random Forest (RF).

Logistic-Regression

A Log-R works by fitting a logistic function to a specific training set. The logistic function returns the probability of a particular class to occur. In this case, it returns the likelihood that a specific IMU and associated body segment are stopped. This probability is used to classify the state of the body segment in stopped or moving. Further details regarding this model are presented in Appendix A.1.1.

Support Vector Machine

An SVM, in a binary classification problem as this one, tries to identify the maximum margin hyperplane between classes. This hyperplane acts as decision border guaranteeing the maximum separability between classes [73]. When the model is trained, the separation margin between classes is defined using examples from both categories that act as support vectors. More information regarding SVMs is presented in Appendix A.1.2.

Random Forest

An RF is a parallel homogeneous ensemble that uses decision trees as base models. Each decision tree is created using a different bootstrap. Additionally, for each of the nodes, only a randomly selected set of the available features is used. In a classification task like the current problem, the class is selected by majority voting. Additional information about RFs can be found in Appendix A.1.3

Common Experimental Methodology - ML

The generation of features was performed following the rationale presented in the automatic labelling of the Qualisys data section (Section 5.1.1). This way, features were generated using windows equivalent to the ones used for the automatic labelling of the Qualisys data (Figure 5.1). For every window, a set of features was created for each component of accelerometer and angular velocity signals. The produced features are mostly statistical: mean, maximum, minimum, sum, standard deviation, quartile 0.25, quartile 0.75, median, variance, energy, and maximum discrete difference along the time axis. After extracting these features, pairs of highly correlated features were identified (Pearson's $r > 0.9$) in the train set. This way, it should be feasible to remove some of those features maintaining the performance of the algorithms while reducing the computational cost by reducing the dimension of the feature space. Therefore, a backward feature selection method was applied to guarantee that the performance of the models (evaluated during the cross-validation) would not be affected by the removal of correlated features. Afterwards, the removed features were also removed from the test set.

For every used classifier, a Group-3-fold cross-validation was performed to optimize hyperparameters keeping inter-user independence in the validation procedure. This group cross-validation means that, for each iteration, each patient's data was only present or on the training set or the validation set. The optimization of the hyperparameters of each model was conducted using Bayesian optimization. The parameters optimized for each model are described in Table 5.2.

For the Log-R and SVM models, the features present on the training set were standardized by removing the mean and scaling to unit variance. The test set features were standardized using the same parameters. The same schema was used during validation: the validation set in each fold was standardized using the parameters used to standardize the training set in that fold. Since that the RF model is tree-based, standardization was not applied.

5.1.2.2 Deep Learning Models

Four different DL models were evaluated: a Densely Connected Neural Network (Dense); a Long-Short-Term-Memory (LSTM) Neural Network; a 6-Layer LSTM used in [109] with the same goal; and an adaptation of the Neural Network that combines Convolutional Layers and LSTM Layers used in [96] but in that case for an end-to-end approach.

For these models, instead of generating features, the raw signal of each component of the accelerometer and angular velocity signals were used as input. Again, windows equivalent to the ones used for the automatic labelling of the Qualisys Data (Section 5.1.1) were used. The output is likewise the state in the middle of the window. This is represented in Figure 5.3.

Due to the computational cost of training neural networks, no Group-K-Fold was implemented. Instead, for validation purposes the train set was divided in train set (composed of 7 randomly selected participants) and in validation set (3 participants). The six-channel inertial data in the train set was standardized, transforming the values to a distribution with mean 0 and standard deviation 1. The same parameters were used to rescale the test set. During validation the same transformation was applied to the validation set using the parameters used to standardize the train set.

Table 5.2: Hyperparameters used to perform Bayesian hyperparameter search in each model

Model	Hyperparameter	Definition	Values
Log-R	C	Inverse of the regularization strength. A higher value of C indicates a lower regularization	[0.05, 3]
	max iter	Maximum number of iterations for the solver to converge	{1000, 2500, 5000}
	tol	Tolerance for the stopping criteria	[0.00001, 0.0001]
SVM	C	Inverse of the regularization strength. A higher value of C indicates a lower regularization	[0.1, 100]
	gamma	Kernel coefficient	[0.001, 1]
	kernel	Type of kernel used	{'rbf'; 'linear'}
RF	min samples leaf	Minimum number of samples required for that node to be consided a leaf node	[1, 2,..., 20]
	max features	Number of features used in each leave	[1, 2,..., N]
	n estimators	Number of trees used	[100, 101,..., 500]
	min samples split	Minimum number of samples required to split using an internal node	[1, 2,..., 20]

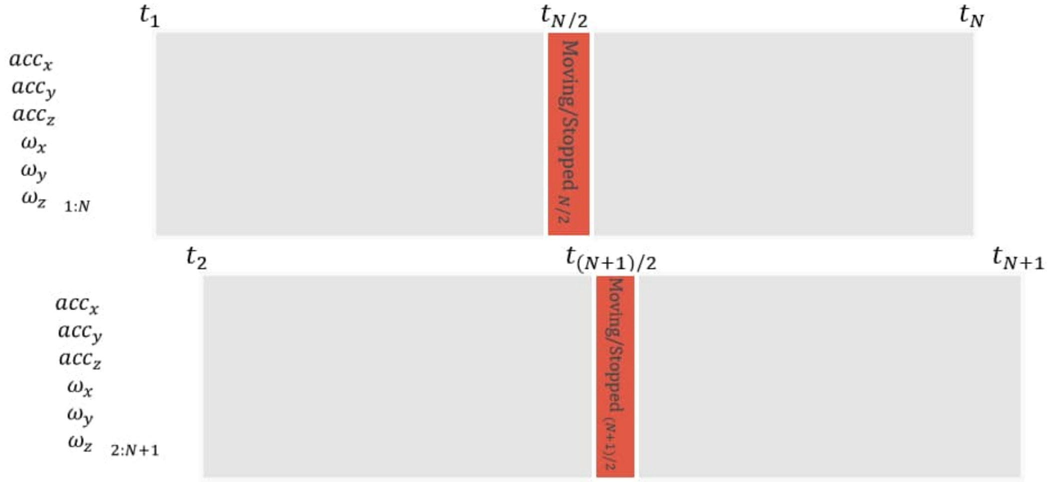


Figure 5.3: Input (six-channel inertial data) and output (moving/stopped in the middle position of each window) of the Deep Learning Models

Densely Connected Neural Network (Dense)

A Dense network is a set of neurons classifiers organized in a feed-forward multi-layer architecture. These networks have an input layer, one or several hidden layers and an output layer composed by one neuron for each predicted class. The connections between neurons are associated with a specific weight. Each neuron possesses an activation function which is applied to the sum of its inputs. These inputs result from the multiplication of the output of each neuron of the previous layer and respective associated weights. More details regarding neural networks can be found in Appendix A.1.4.

The proposed Dense network contains a first layer which flattens the input of N x 6 into (6N x 1). After, a densely connected layer of 30 units with rectified linear unit (ReLU) activation is added. The final layer is a densely connected layer with only one unit and sigmoid activation to reduce the ouput to one dimension.

Long Short-Term Memory Neural Network - LSTM

An LSTM is a type of recurrent neural network. This kind of models take into account the time dependencies present in the data and so these are often used in time-series modelling. These models also have an input layer, an output layer and one or more hidden layers. The time dependency is inserted through the use of a cell state which travels between neurons. This cell state is controlled through the use of a forget gate, an input gate, and an output gate. More details about LSTMs can be found in Section 6.1.1 and in Appendix A.1.4.1.

The proposed LSTM architecture contains a single LSTM layer with 30 units. This layer includes recurrent dropout of 20%, dropping neurons directly in the recurrent connections as suggested in [94] to reduce overfitting. Finally, a densely-connected layer with only one unit and sigmoid activation is added.

6-Layer LSTM Neural Network [109]

The 6-Layer LSTM proposed by Wagstaff and Kelly in [109] contains 6 LSTM layers with 80 units per layer. After that, a single densely-connected layer was added to reduce the output to a 2D problem. Considering the way our output was defined (binary output), this last layer is substituted by a densely-connected layer with only one unit and sigmoid activation. Besides, as proposed by the authors, a dropout layer with a rate of 20% was added before the last fully-connected layer to avoid overfitting.

Convolutional + LSTM Neural Network [96]

Although, the goal of Silva *et al.* in [96] was to develop an end-to-end model that estimates both position and orientation, this neural network was adapted from the one used by them to our goal. This adaptation consisted of passing the input through two 1D convolutional layers of 128 features and kernel of size 11. The next layer is a max-pooling layer of size 3.

Then, a bidirectional LSTM with 128 units is added. A bidirectional LSTM is composed of two LSTMs in which the inputs are passed in both ways. This means that the data is passed through a "forward LSTM" and a "backwards LSTM". The "backwards LSTM" does the same as a typical LSTM but takes the input inverted in time. This allows to obtain information about the past and the future. The hidden states of both LSTMs are then concatenated.

After, a dropout layer with a dropout of 25% is added. The next step is to pass the output of this layer into another bidirectional layer, followed by a dropout layer of 25%. Finally, instead of using a fully connected layer with seven units (like the authors did to estimate the 3D position the orientation quaternion), it was used a densely-connected layer with 1 unit and sigmoid activation.

The main idea behind the model is that convolutional layers can extract features from the raw inertial signals and use them as the input of the bidirectional LSTM layers.

Common Experimental Methodology - DL

All the previously described models were trained using an Adam optimizer [56] and binary cross-entropy loss. Similarly to the method implemented by Wagstaff and Kelly [109], a learning rate of $5x10^{-3}$, and weight decay of $1x10^{-5}$ were used. A batch size of 32 was used in an attempt to balance both the memory requirements and the accurate estimation of the error gradient. Early stopping was implemented to interrupt the training when the validation loss was no longer decreasing over the course of 20 epochs to reduce the risk of overfitting. Finally, gradient clipping was applied in all the methods that used LSTM layers to avoid exploding gradients.

To avoid situations in which a miss-classification would generate a small window of movement inside a stop phase or vice-versa (which can be regarded as noise) the effect of applying a median filter was also evaluated. The median filter works as a low-pass filter and, therefore, in this case, can remove isolated and quick transitions of state from moving to stopped or from stopped to moving. The median filter evaluated was a filter of size 61, meaning that it is a filter that removes transitions lower than 0.51s.

5.1.3 Zero-Velocity Detection using inertial data - One placement to all

Also interesting is to understand how well could a model trained using just data collected from one body placement perform when classifying the movement of other body segments. This allows to understand if it is possible to develop a model capable of generalize to several IMU placements using just data from one body placement.

Additionally, the analysis of this question could be attractive from a data collection point of view for a future project. If a model trained using only data from one body placement could generalize well to other body segments then, for instance the data collecting of a future project could be performed focusing on a single body placement alternatively to collecting data from several body placements.

This examination was conducted using a similar split like the one described in Figure 5.2. Data from the same patients was used to build the train set. However, instead of using data from all the placements, only data from each placement was used at each time. This means that six different models were trained, one for every placement. The performance of each model was evaluated individually for each placement of the test set. All the models mentioned in the previous section were evaluated using this data. This means that the models were separately trained using a single acquisition from one placement for each trial. Therefore the model was trained at each time using data from 10 trials and evaluated independently using 3 trials for each placement.

5.1.4 Translation estimates using the classification of zero-velocity moments

The output of the classification algorithms introduced in Section 5.1.2 (i.e. using data from all the placements to train) were then used to improve the displacement estimates. This way, it was assumed that when a moment is classified as stopped, then the IMU velocity is zero and no change occurs on its position. To calculate the displacements between zero-velocity moments, double integration was performed.

Additionally, some drift removal was executed. The accumulated drift was calculated as the difference between the velocity estimation at the end of a moving period and at the start of that period. That accumulated drift during each moving period was assumed to be accumulated linearly. That way, the respective drift was removed from each sample. To calculate the corrected position, the drift fixed velocity was integrated.

The velocity correction can be expressed by:

$$\text{drift rate} = \frac{v_e - v_b}{e - b} \tag{5.1}$$

$$v_{corrected,\ t_i} = v_{t_i} - \text{drift rate} \times (t_i - t_b),\ for\ t_i = t_b, ...t_e \tag{5.2}$$

In which v_e represents the velocity at the end of a movement, v_b the velocity at the beginning of a movement. t_b and te represent the time of the last and first sample of the movement respectively. The corrected velocity for each movement - that occurs between b and e - is then calculated subtracting the drift rate at the respective time.

Again, the performance of the translation estimates was only evaluated in the test set using the split referred in Section 5.1.2

5.2 Results and Discussion

5.2.1 Zero-Velocity Update Detection using inertial data - Multi-placement

Accuracy was selected as the performance measure to evaluate each model. This choice is justified by the distribution of the classes (in the test set the classes are relatively balanced with the positive *"stopped"* class corresponding to 37.68% of the values and the negative *"moving"* class corresponding to 62.32%), and by the fact that true positives and true negatives have equal importance in our model. The accuracy of the different used classifiers is presented in Figure 5.4.

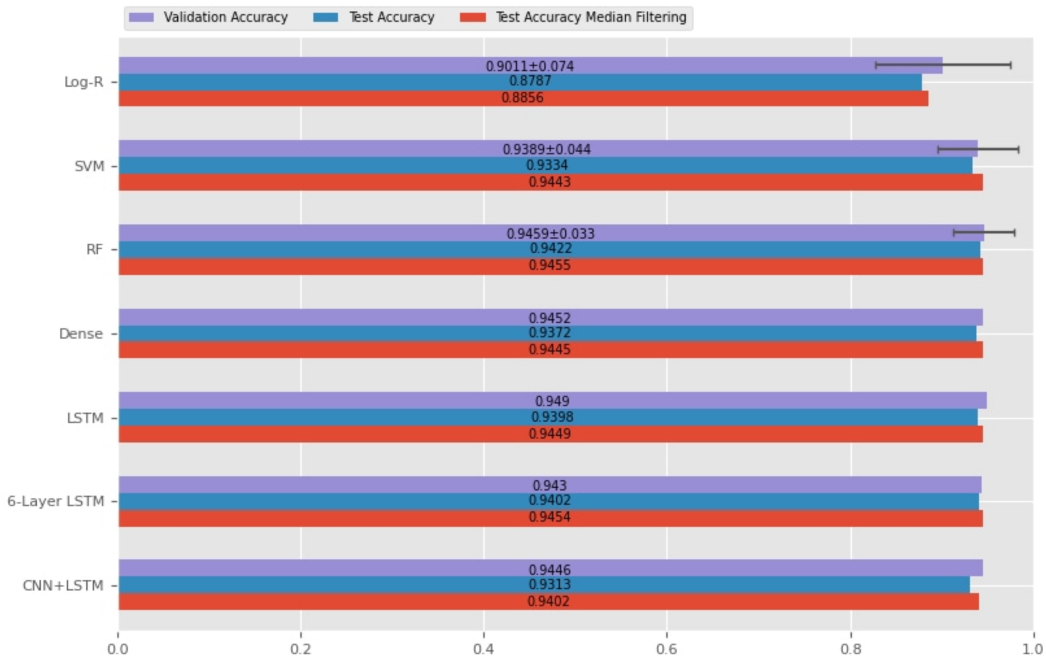


Figure 5.4: Accuracy of the different used classifiers

Relatively to the Bayesian hyperparameter optimization, the parameters that produced a better accuracy during the Group-3 fold were: for the Logistic Regression (C= 2.96, max iter=5000, tol= 6,65$\times$10^{-5}); for the SVM (C= 67.74, gamma= 0.24, kernel= rbf); and for the Random Forest (min samples leaf= 3, max features= 9, n estimators= 500, min samples split= 3). The results of the application of these parameters during the group-3-fold are represented in Figure 5.4. The validation accuracy of the ML models is represented with the respective associated standard deviations due the Group-3 fold process.

Analyzing the results it is possible to conclude that all the chosen models can be successful in differentiating moments of movement from stationary moments. Applying a median filtering also slightly improves the accuracy of all the models. Analyzing the test set results only the Logistic Regression model has a accuracy lower than 0.94. The rest of the models have a very similar performance. The model with a better accuracy in the test set was the Random Forest model with median filtering.

An example of a trial classification performed using the Random Forest classifier is presented in Figure 5.5. Analyzing this figure it is possible to confirm that the majority of the classifications are correct. The median filtering allows to remove uncertainties in the transitions (as the ones in

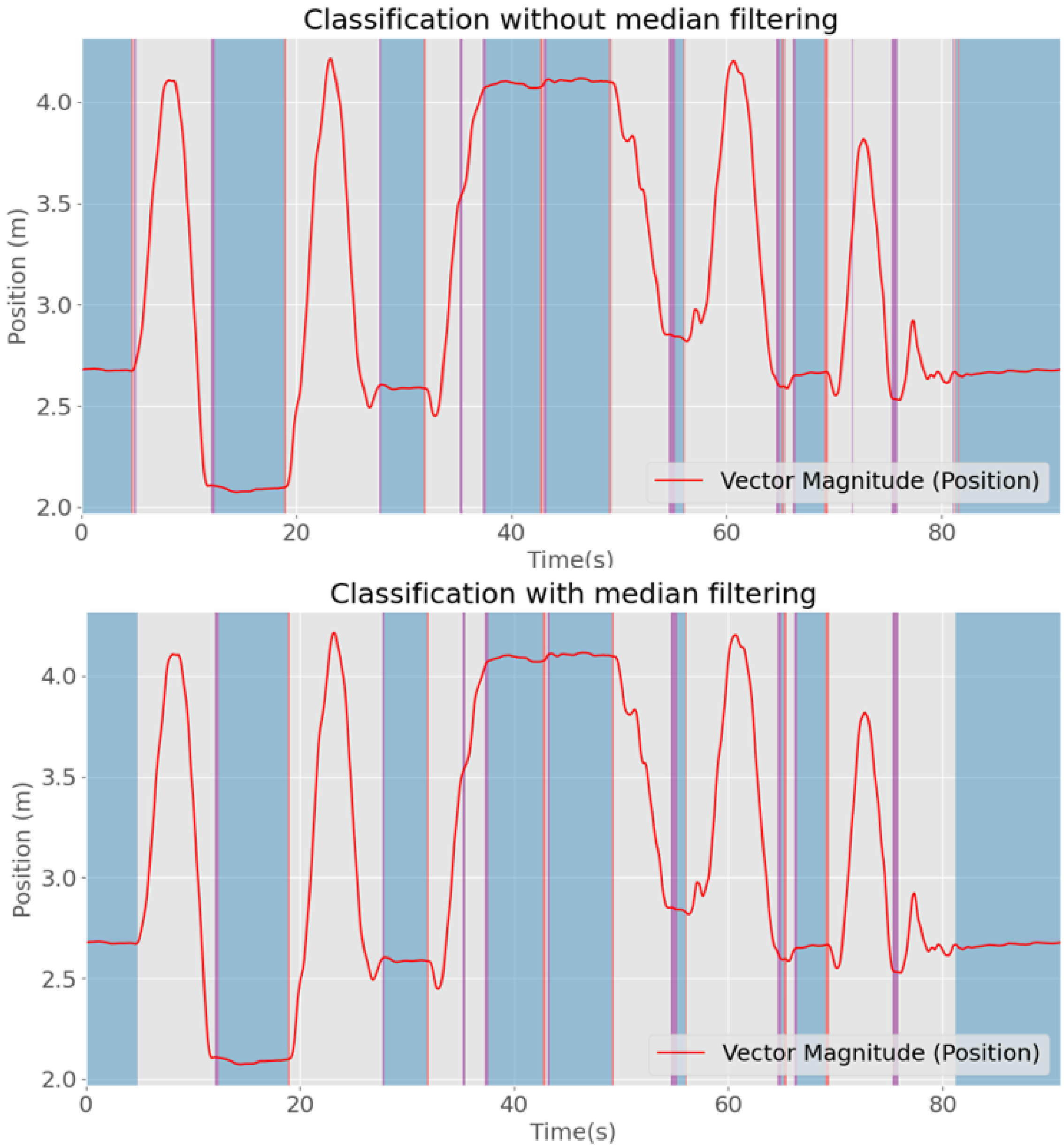


Figure 5.5: Example of a trial classification using the Random Forest classifier without and with median filtering. (Blue background - True positives (Correctly identified stopped periods), Gray Background - True Negatives (Correctly identified periods of movement), Purple Background - False positives (Wrongly labelled as stopped periods); Red Background - False negatives (Wrongly labelled as moving periods))

the start of the first movement and in the end of the last movement), and also allows to remove quick isolated transitions (as the one that occurred in the classification around the 70s mark).

Also interesting is to analyse the feature importance in the same Random Forest classifier presented in Figure 5.6. The examination of this importance allows to better understand how much did each feature contributed to decreasing the weighted impurity on average in each tree of the model. This decrease in inpurity is determined by the number of splits performed using that feature and by the importance of those splits. If a low number of samples reaches a

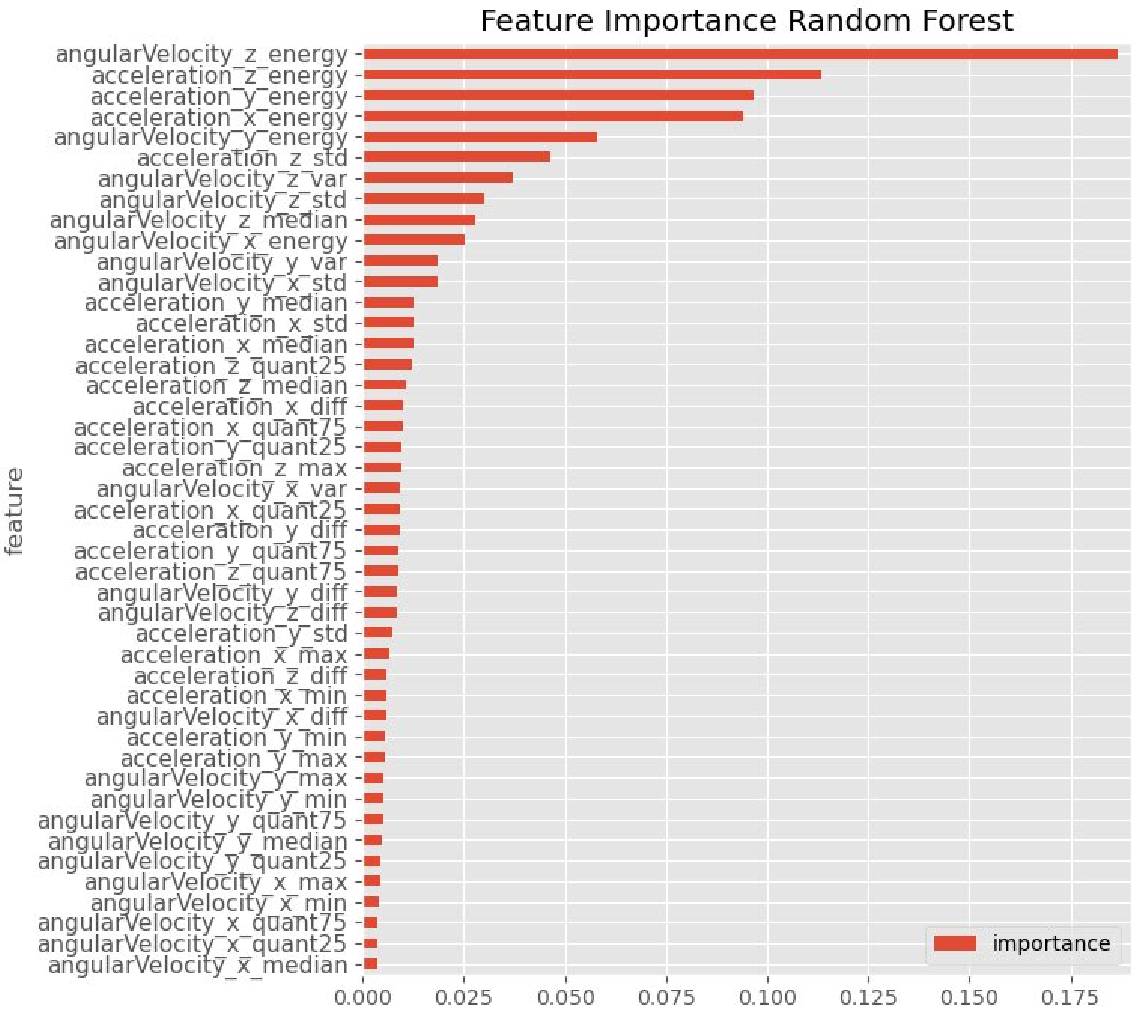


Figure 5.6: Feature importance - Random Forest classifier

particular split, then that split is not as important as one that occurs previously in that tree.

The results show that the features with the highest importance were mainly energy features. This was already expected since that periods in which motion occurs are related with higher signal energies than stationary periods.

5.2.2 Zero-Velocity Update Detection using inertial data - One placement to all

Relatively to the classification of data from several body segments using data from a single body segment, the results of the model that performed better on average for all the placements - Random Forest Classifier- are presented in Table 5.3. The results of the other models are presented in Appendix B.

Analyzing these results, it is possible to understand that the most problematic placements are the right and left toe. When trained with data from these placements, the generalization ability of the models is reduced as represented by the low accuracy values. However, when the models are trained with data either from the right or left shoulder or from the right or left upper

Table 5.3: Accuracy performance of the Random Forest Classifier trained with data from a single placement and evaluated in several placements (RSHO- Right Shoulder,LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Train \ Test	RSHO	LSHO	RUPA	LUPA	RTOE	LTOE	Mean
RSHO	0.9498	0.9467	0.8476	0.8737	0.8681	0.9109	0.8995
LSHO	0.9516	0.9533	0.8554	0.8758	0.8724	0.9123	0.9035
RUPA	0.9463	0.9512	0.9513	0.9438	0.9044	0.9316	0.9381
LUPA	0.9438	0.9506	0.9390	0.9432	0.9024	0.9269	0.9343
RTOE	0.7108	0.7233	0.7615	0.7902	0.9388	0.9549	0.8133
LTOE	0.6688	0.6927	0.7424	0.7773	0.9426	0.9630	0.7978
Mean	0.8618	0.8696	0.8495	0.8673	0.90478	0.9333	0.8811

arm, these are capable of performing well when classifying the right and left toe data. Paying attention to the inertial measurements of the toes presented in Figure 4.5 it is possible to see that the linear acceleration values are distributed to a higher range of values than the data from the shoulder and from the upper arm. This can justify the lack of generalization ability of the models trained with toe data.

Looking at the rest of the placements, one can also conclude that when the model uses data from the upper arms to train, it also performs better classifying the shoulder data then when the opposite occurs (i.e. using the shoulder data to train and classifying the upper arm data).

Another interesting conclusion is that all the models obtained a good performance when applied to data from the other side of the body. This means that a model trained with data from a certain placement on the right side of the body placement it was able to obtain a good predictive performance when evaluated on the same placement but on the left side of the body.

In conclusion, using data from all the upper arms can guarantee a greater generalization ability of the model than using data from the shoulders and from the toes.

5.2.3 Translation estimates using the classification of zero-velocity moments

To assess how the zero-velocity update performed by each model influences the trajectory estimate, the three classifiers with the best classification performance in the test set were used: Random Forest, Single Layer LSTM model, and 6-Layer LSTM model. All of them were combined with the median filtering post-processing.

This way, the root mean squared error (RMSE) of the position estimated in each axis was evaluated:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(p_i - \hat{p}_i)^2} \tag{5.3}$$

The RMSE is a representation of the average prediction error of each model that gives higher weight to more substantial errors (since these are squared and only after averaged). To allow a more straightforward interpretation of the results the mean absolute error (MAE) of the translation in each axis was also evaluated:

$$MAE = \frac{1}{N} \sum_{i=0}^{N-1} |(p_i - \hat{p}_i)| \tag{5.4}$$

In both equations, p_i represents the true position in a given axis in the timestep i, and $\hat{p}_i$ the estimated position. Since that, at the beginning of each trial, the IMUs coordinate frame was reinitialized to match the Qualisys world frame, then, to compare p_i and $\hat{p}_i$ no rotation needs to be applied. The RMSE and the MAE were calculated for the x, y, and z-axis. It was considered that the starting point of all the trials was the point of coordinates [0,0,0].

Besides, it is also important to understand the average prediction error of each model in the 3D trajectory. With that purpose the absolute trajectory error (ATE) of the position was also assessed. The absolute trajectory error of the position can be regarded as a 3D RMSE. This way, ATE is defined as [122]:

$$ATE = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} \|\Delta \mathbf{p}_i\|^2} \tag{5.5}$$

In which $\|\Delta \mathbf{p}_i\|$ represents the Euclidean distance between the estimate and the ground truth:

$$\|\Delta \mathbf{p}_i\| = d(\mathbf{p}_i, \hat{\mathbf{p}}_i) = \sqrt{(p_{(x,i)} - \hat{p}_{(x,i)})^2 + (p_{(y,i)} - \hat{p}_{(y,i)})^2 + (p_{(z,i)} - \hat{p}_{(z,i)})^2} \tag{5.6}$$

Additionally a 3D MAE was also calculated :

$$MAE(\Delta \mathbf{p}_i) = \frac{1}{N} \sum_{i=0}^{N-1} \|\Delta \mathbf{p}_i\| \tag{5.7}$$

The calculation of the RMSE and MAE provides a single, easy to compare value. However, it is essential to mention that these metrics are time-sensitive: if a position estimate mistake happens at the beginning of a trial, then the error tends to be higher then if the same error takes place as the end of the trial [122]. Besides, it is also important to keep in mind that the RMSE gives higher weight to more significant errors [102].

All the metrics were evaluated individually for each trial and, therefore, the average values between the test trials are presented with the respective standard deviations.

The results of the zero-velocity update using the different supervised classifiers are presented in Table 5.4. These results show that there is a significant improvement in applying the zero-velocity update using the different classifiers compared with the simple double integration of the signals without any correction. The use of drift correction also improves the RMSE and the MAE of all the trajectory estimates with a zero-velocity update.

The model which stopped moments identification leads to a trajectory estimate with a lower error is the 6-Layer LSTM model. In this case, the drift correction also improves this estimate in all the axes. However, all the models in which this correction was applied possess a similar performance between them before and after the drift correction. This is consonant with the similar classification performances presented in Figure 5.4

Another interesting point is to note that the translation estimates possess a lower error in the Z-axis. Again, this may be due to the less amount of movement that occurs in the Z-axis (the

Table 5.4: MAE and RMSE of the translation estimate in each axis of the full trial 3D translation estimate using the different zero-velocity detectors (Uncorrected - 3D position estimate obtained by the double integration of the linear accelerations without any correction; RF Zero-Velocity Update - Random Forest identification of zero-velocity moments; LSTM Zero-Velocity Update - LSTM identification of zero-velocity moments; 6-Layer LSTM Zero-Velocity Update - 6-Layer LSTM identification of zero-velocity moments).

Correction	Error (m)	p_x	p_y	p_z	$\lVert\mathbf{p}\rVert$
Uncorrected	RMSE	28.71±24.55	22.75±22.03	6.60±5.61	23.82±16.35
	MAE	22.45±19.98	17.30±17.25	5.25±4.40	32.41±22.80
RF Zero-Velocity Update	RMSE	1.41±1.23	0.81±0.62	0.69±0.37	1.99±1.10
	MAE	1.25±1.18	0.65±0.45	0.57±0.33	1.77±1.05
RF Zero-Velocity Update + Drift Correction	RMSE	0.60±0.39	0.56±0.49	0.37±0.23	1.02±0.45
	MAE	0.51±0.37	0.47±0.44	0.31±0.20	0.92±0.43
LSTM Zero-Velocity Update	RMSE	1.48±1.13	0.94±0.75	0.79±0.53	1.27±0.55
	MAE	1.30±1.07	0.74±0.60	0.65±0.47	1.94±0.93
LSTM Zero-Velocity Update + Drift Correction	RMSE	0.54±0.34	0.59±0.48	0.49±0.32	1.06±0.45
	MAE	0.46±0.31	0.49±0.42	0.40±0.28	0.93±0.40
6-LSTM Zero-Velocity Update	RMSE	1.25±1.00	0.94±0.78	0.68±0.40	1.90±1.04
	MAE	1.08±0.95	0.75±0.65	0.57±0.34	1.67±0.98
6-LSTM Zero-Velocity Update + Drift Correction	RMSE	0.50±0.31	0.51±0.49	0.42±0.21	0.93±0.44
	MAE	0.42±0.28	0.43±0.43	0.35±0.18	0.83±0.39

subjects major movements occur in the horizontal plane). This way, the removal of the gravity may be optimized in axes in which less motion occurs. More discussion regarding the smaller error for vertical translations estimates using IMUs was already examined and can be found in Section 4.5.

Additionally, and to better understand what the influence of applying these zero-velocity update classifiers in an entire trial is, the mean average percentage error (MAPE) of the total translation estimated in each trial was calculated. The complete translation occurred in a trial can be calculated as:

$$trans = \sum_{i=0}^{N-1} d(\mathbf{p}_{i+1}, \mathbf{p}_i) \tag{5.8}$$

Table 5.5: Evaluation of the estimation of the total translation ($\widehat{trans}$) performed during all the test trials (Uncorrected - 3D position estimate obtained by the double integration of the linear accelerations without any correction; RF Zero-Velocity Update - Random Forest identification of zero-velocity moments; LSTM Zero-Velocity Update - LSTM identification of zero-velocity moments; 6-Layer LSTM Zero-Velocity Update - 6-Layer LSTM identification of zero-velocity moments)

Correction	$trans$	$\widehat{trans}$	MAPE
Uncorrected	30.34±3.49 m	107.99±48.03 m	276.51%
RF Zero-Velocity Update	30.34±3.49 m	28.58±3.86 m	6.20%
RF Zero-Velocity Update + Drift Correction	30.34±3.49 m	27.71±3.43 m	8.71%
LSTM Zero-Velocity Update	30.34±3.49 m	28.38±3.45 m	6.74%
LSTM Zero-Velocity Update + Drift Correction	30.34±3.49 m	27.43±3.34 m	9.62%
6-LSTM Zero-Velocity Update	30.34±3.49 m	28.24±3.65 m	7.42%
6-LSTM Zero-Velocity Update + Drift Correction	30.34±3.49 m	27.34±3.56 m	9.84%

And this way the MAPE across all trials can be calculated as:

$$MAPE(trans) = \frac{1}{M} \sum_{j=0}^{M-1} \left| \frac{trans - \widehat{trans}}{trans} \right| \tag{5.9}$$

In which M represents the number of trials, *trans* the ground truth total translation, and $\widehat{trans}$ the total estimated translation using each zero-velocity classifier. The results of the evaluated MAPE are presented in Table 5.5.

Analyzing this table, it is possible to understand that the total translation performed during each trial is close to 30 m. The full translation estimate performed using just simple double integration leads to a significant error which is represented by its MAPE. On the contrary, all the zero-velocity update classifiers allow reducing the MAPE value to a value below than 10%. The model that achieves a better total translation estimate is the Random Forest classifier without drift correction.

An example of the correction performed by the stopped moments' identification performed using the 6-Layer LSTM classifier with and without drift correction is represented in Figure 5.7. The double integration of the linear acceleration signals quickly leads to an erroneous classification (with errors as significant as 50m in the y-axis). The correction performed by the zero-velocity update significantly improves the trajectory estimate. The effect of the drift correction is visible in the Y-axis signal allowing to control the error that otherwise would accumulate.

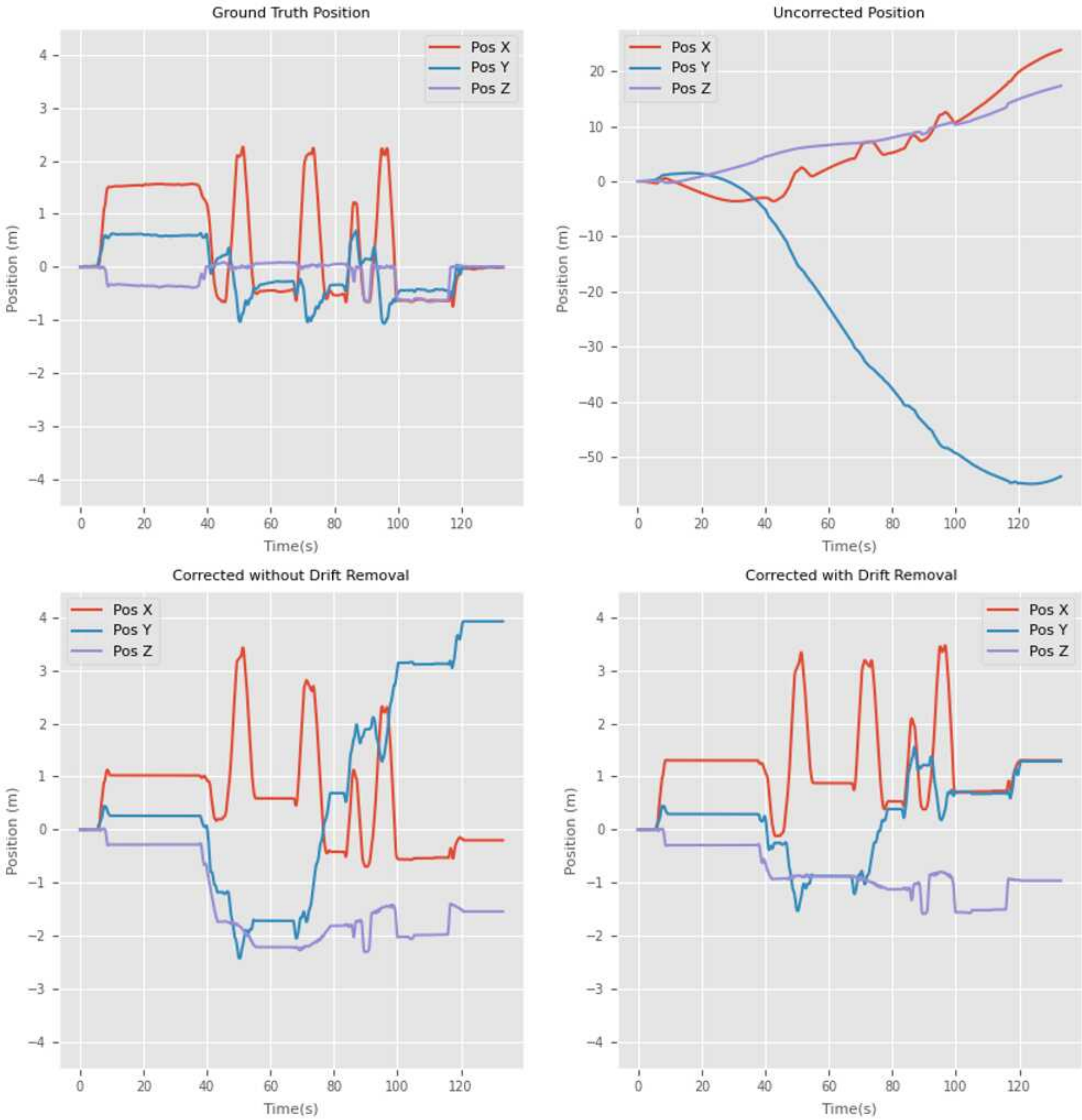


Figure 5.7: Full trial translation estimate in all the axes with drift and without drift correction (Ground Truth - 3D position provided by the Qualisys optical motion system; Uncorrected position - 3D position estimate obtained by the double integration; Corrected without Drift - 3D position estimate obtained by the double integration with zero-velocity update performed using the results of the 6-Layer LSTM classifier; Corrected without Drift - 3D position estimate obtained by the double integration with zero-velocity update performed using the results of the 6-Layer LSTM classifier and drift correction)

5.3 Summary

The goal of this chapter was the correct identification of zero-velocity periods. This identification allows to adjust the velocity estimation calculated with the inertial data, and this way, to reduce its drift.

This objective was treated as a classification problem. Therefore, several supervised ML classifiers were used to identify those periods using inertial data as input and Qualisys labelled position data as output. The results obtained by the several classifiers allow concluding that the detection of zero-velocity periods is performed with high accuracy (with the majority of the used classifiers reaching an accuracy of 94%).

Analysing the models trained with data from each placement individually it was possible to conclude that some placements present more generalization ability than others.

Relatively to the application of zero-velocity detection to correct the velocity estimation during the integration of the accelerometer data, the results show that this correction allowed to reduce the ATE from a mean value of 24m to mean values of 1m. This way, it is possible to conclude that the use of the zero-velocity classifiers and subsequent velocity correction provided a much better tracking of the position of an IMU and of associated body placement throughout time.

Chapter 6

Rectify Translation Estimates – a Regression Problem

Using the predictions elaborated by the models developed in the previous chapter, it is possible to estimate the moments in which a certain IMU and respectively associated body segment are still. These estimations allow stating that during a specific period, the velocity of the IMU was null and, therefore, no change in position occurred. This way, it is possible to adjust the velocity estimation and mitigate the drift error in the position estimate generated by the naive double integration of the noisy accelerometer readings whose errors would, otherwise, accumulate and rapidly diverge. Therefore, some correction is introduced in the displacement estimation, as demonstrated in Section 5.2.3.

However, in the periods in which motion occurs, the estimate of the translation provided by the double integration of the accelerometer noisy readings is still erroneous, and it is not being corrected. This way, the challenge tackled throughout this chapter is a regression problem. Using the three-axis IMU measures, the challenge is to provide an improved estimate of the displacements that occur between zero-velocity moments.

Taking this regression task into account, different DL models were used. All of them are LSTM based. The use of LSTMs is useful since these neural networks take into consideration the sequential nature of the inertial measurements signals and allow inputs with variable length like the inertial data of the different sized moving periods. The goal of the developed models was to achieve a "smart" integration that could compensate for the noises, bias and drifts present in the accelerometer signal.

Bearing this "smart" integration in mind, different LSTM initializations were tried. The goal of these initializations was to initialize the neural networks in such a way that these would approximate a double integration without any training. This way, during the training process, the neural network would only need to learn how to compensate for the bias and drifts to reach an ideal displacement prediction.

The translation estimates of the models were then compared with the ground truth provided by the Qualisys system to evaluate their performance. Finally, the effect of using these regression models, together with the classification models of the previous chapter, was also assessed.

6.1 Methods

6.1.1 Translation estimate of moving moments using regression

Different LSTM-based methods were tested in order to estimate the displacement that took place in a period of movement using as input the inertial data from that period. The main idea behind each model is to assemble a 2-Layer LSTM in which each layer performs an integration that also accounts for the errors present in the linear acceleration signal, and, therefore can compensate them reaching a better displacement estimate. A representation of this idea is presented in Figure 6.1.

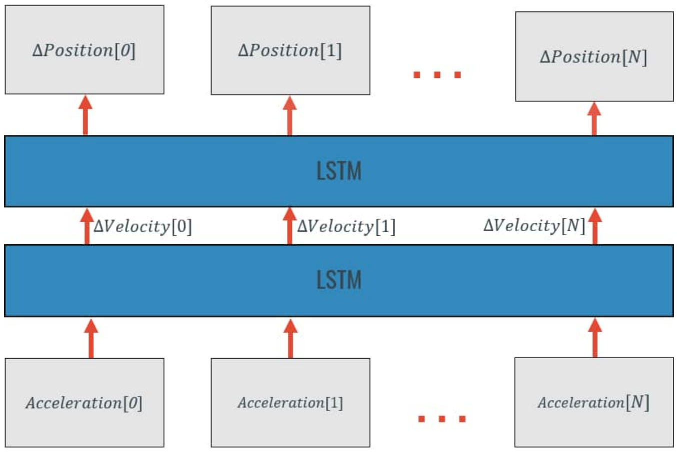


Figure 6.1: "Smart" double integration by a 2-Layer LSTM

Since some methods described in the next subsections consist of altering some of the LSTM equations, in this section, a more in-depth analysis of LSTMs is presented. An LSTM is a type of Recurrent Neural Network (RNN) capable of modelling long and short time dependencies in the data. The principal constituents of an LSTM are the cell state - which provides the ability to keep information in the memory from LSTM block to LSTM block - and its gates. An LSTM has three gates: a forget gate, an input gate, and an output gate. The forget gate is responsible for deciding what information to keep from the previous cell state. The input gate is responsible for determining what new information will be added to the cell state. The output gate is responsible for deciding what is the data from the cell state that is going to be output. Each gate possesses a sigmoid activation function. This means that the output of each one is a value between 0 and 1. If the value is 0, the gate blocks everything; if this value is 1 then the gate lets everything pass. The gate equations can be represented by:

$$\mathbf{f}_t = \sigma(\mathbf{w}_f[\mathbf{h}_{t-1}, \mathbf{x}_t]^T + \mathbf{b}_f) \tag{6.1}$$

$$\mathbf{i}_t = \sigma(\mathbf{w}i[\mathbf{h}_{t-1}, \mathbf{x}_t]^T + \mathbf{b}_i) \tag{6.2}$$

$$\mathbf{o}_t = \sigma(\mathbf{w}_o[\mathbf{h}_{t-1}, \mathbf{x}_t]^T + \mathbf{b}_o) \tag{6.3}$$

In which $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ represent respectively, the forget gate, the input gate, and the output gate. In the same way, $\mathbf{w}_f$, $\mathbf{b}_f$, $\mathbf{w}_i$, $\mathbf{b}_i$ and $\mathbf{w}_o$, $\mathbf{b}_o$ represent respectively, the forget gate weights and biases, the input gate weights and biases, and the output gate weights and biases. $\mathbf{x}_t$ is the input for the current timestep, and $\mathbf{h}_{t-1}$ the output from the previous LSTM block. σ is the sigmoid activation function.

To calculate the cell state and the output the following calculus are performed:

$$\widetilde{\mathbf{c}}_t = tanh(\mathbf{w}_c[\mathbf{h}_{t-1}, \mathbf{x}_t]^T + \mathbf{b}_c) \tag{6.4}$$

$$\mathbf{c}_t = \mathbf{f}_t \times \mathbf{c}_{t-1} + \mathbf{i}_t \times \widetilde{\mathbf{c}}_t \tag{6.5}$$

$$\mathbf{h}_t = \mathbf{o}_t \times tanh(\mathbf{c}_t) \tag{6.6}$$

In which $\widetilde{\mathbf{c}}_t$ simbolizes the candidate for the cell state at the current timestep, $\mathbf{c}_t$ the cell state at the current timestep and $\mathbf{h}_t$ the current output.

A visual representation of an LSTM is presented in Figure 6.2. The number of blocks is the same as the number of timesteps present in the input. The number of blocks in an LSTM should not be confused with its number of units. The number of units represents the dimension of the output, and it is not related to the number of blocks of an LSTM. More information regarding LSTMs can be found in Appendix A.1.4.1.

LSTMs were chosen due to their memory ability, and their capacity of dealing with problems in which the order of the data must be taken into account in the modelling process like the one faced in this project. An important feature is also the fact that LSTMs as all the RNNs, and unlike feed-forward neural networks, can deal with input and output sequences of different length [77]. In this case, the periods in which movement occurs are all of different length, and therefore the inertial data inputs and the position data outputs between different sequences are of different

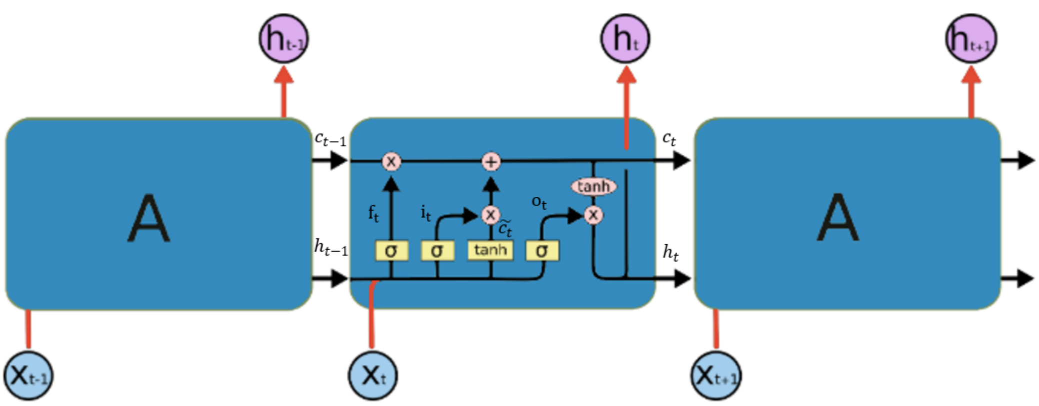


Figure 6.2: LSTM repeating blocks, from [1]

[1]Understanding LSTMs. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ Accessed on 27-01-2020

size. Additionally, when compared with simpler RNNs, LSTMs can deal better with sequential data since these can capture long-term dependencies.

6.1.1.1 Warm-Start with "Integrative Weights"

The first applied method consisted of a 2-layer LSTM combined with an "integrative initialization". An overview of the architecture of this approach is represented in Figure 6.3. This "integrative initialization" means that the neural network weights were initialized in such a way that, without any training, the network would simulate a double integration. Such an initialization makes intuitive sense in the way that the network would start its training already knowing the mathematical operation that it should conduct. This way, during the training procedure, the model would only learn how to account for the errors present in the acceleration data that affect the estimate produced by the double integration operation. The performance of a deep neural model is much reliant on the effective initialization of its parameters, and a smart initialization strategy could improve the performance of the model [17].

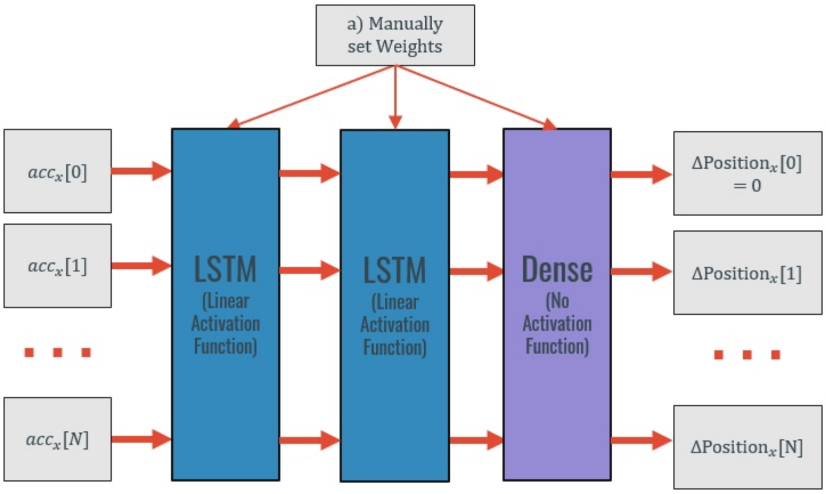


Figure 6.3: Representation of the warm-Start with "integrative" weights methodology – Example for the X-axis

To initialize the weights as desired, the equations used to determine the candidate cell state and the output - Equations 6.4, and 6.6 - were firstly modified. By considering only the example for the first layer, the idea is to use the acceleration as input and to use the cell state to save the velocity in each timestep. In the second layer, the velocity works as input, and the cell state saves the previous position. This way, and assuming that the initial velocity and positions are zero, the output of the first layer is the velocity in each timestep and the output of the second layer is the position in each timestep.

To do so, the hyperbolic tangent activation functions of each LSTM block were transformed in linear functions and the weights of the forget, input and output gates were manually initialized to a value close to zero. It is crucial to guarantee symmetry breaking when initializing these weights. If all the weights were initialized to 0 or to the same values, symmetry would exist, and the weight updates during backpropagation could be not effective enough, leading to a poor learning process [82]. This way, the weights of the gates were randomly initialized to different values between -10^{-6} and 10^{-6}. The biases of these gates were initialized to positive values that would guarantee that all the information would pass through the gates setting the sigmoid function to a value close to 1.

Relatively to the weights and biases related to the cell state, the $\mathbf{w}_c$ was defined as a matrix containing the sampling period (that multiplies by the current input and is represented by dt) and a value close to zero (that multiplies by the previous output). $\mathbf{b}_c$ is also initialized to with values between -10^{-6} and 10^{-6}.

This way, the transformations applied to the Equations 6.4, 6.5 and 6.6 can be approximated as:

$$\widetilde{\mathbf{c}}_t = [0, dt] \times [\mathbf{h}_{t-1}, x_t]^T + \mathbf{b}_c = a_t \times dt \tag{6.7}$$

$$\mathbf{c}_t = \mathbf{f}_t \times \mathbf{c}_{t-1} + \mathbf{i}_t \times \widetilde{\mathbf{c}}_t = 1 \times v_{t-1} + 1 \times a_t \times dt = v_{t-1} + a_t \times dt \tag{6.8}$$

$$\mathbf{h}_t = \mathbf{o}_t \times \mathbf{c}_t = 1 \times v_{t-1} + at \times dt = v_t \tag{6.9}$$

In which a_t represents the acceleration input of the current timestep, and v_{t-1} the velocity of the previous timestep.

Looking and the Equations 6.8 and 6.9 it is possible to understand that the cell state (c_t) and the output (h_t) both represent the velocity in the current timestep. The same weights were used in the second LSTM layer that is responsible for integrating the velocity into position.

The models were trained using the accelerations of a moving period in each axis as input and the displacement in the respective axis as output. This way, for instance, the acceleration of a certain moving period in the y-axis was used as input, and the displacement on the same period in the y-axis was used as output. Information from all the axes was used to train a single model.

Different models were tested using different number of units, and using different forget, input, and output gates biases.

After the 2-Layer LSTM, a densely connected layer with no activation was added to reduce the dimensionality of the output to 1D (in the cases in which the number of units of the LSTM was higher than one). Again, the weights of this dense layer were initialized, maintaining the purpose of the warm start of the previous layer simulating a double mathematical integration.

6.1.1.2 Warm-Start with "Pre-Trained Weights"

The second approach is very similar to the previous one as it is described in Figure 6.4. The goal was also to perform a warm-start to a 2-layer LSTM model in such a manner that the network, without any training, would resemble a double mathematical integration.

However, in this approach, no modifications were introduced in the LSTM equations, and the weights initialization was not performed manually. In the present case, two different ML models were trained: firstly, a 1-layer LSTM followed by a densely connected layer with no activation was trained to learn how to perform a simple integration. This means that the input of the model was the acceleration in a single axis, and its output was the velocity estimated by simple integration. Inertial data from the three-axis was used to train this model. This way, it was expected that this first model would learn how to perform a simple mathematical integration.

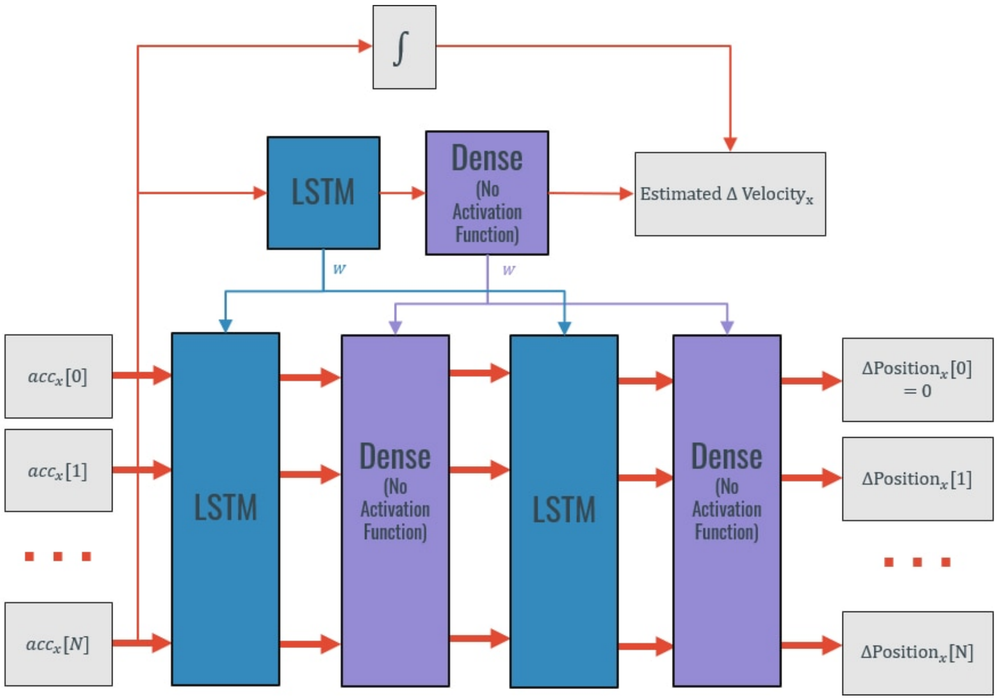


Figure 6.4: Representation of the warm-Start with "pre-trained" weights methodology – Example for the X-axis

Afterwards, the second ML model, a 2-layer LSTM model (with each LSTM layer followed a densely connected layer with no activation was initialized with a warm start using the previously trained weights. This means that each LSTM layer weights were initialized with the final weights of the firstly trained 1-layer LSTM model. Again, it was expected that this warm start would provide a starting point to the neural network in such a manner that the network would start its training already knowing how to perform a double mathematical integration.

Once more, the model was trained using accelerations of a moving period in each axis as input and the displacement in the same axis as output. Information from all the axes was used in the same model.

Another version of this approach was also tested. It included the training of an equal model (1-Layer LSTM + densely connected layer with no activation) with the single-axis linear acceleration as input. However, instead of the velocity calculated through a simple integration, the output used for the first model was the drift corrected velocity (calculated with simple integration and corrected using the drift correction described in Section 5.1.4). The weights from this model were then used to initialize the first layer (LSTM + densely connected layer) of the second ML model. In theory, this should provide to the neural network a starting point in which it already knows how to perform an integration from acceleration to velocity with the benefit of removing the drift. The weights used in the second layer of this version were the weights used in both layers of the previous version. The "drift removed weights" were not used in the second layer since that particular correction (drift removal) is specific of the integration from acceleration to velocity.

6.1.1.3 No customized initialization

The third approach used no warm-start. The neural network structure is the same as the used in the previously presented methodologies: a 2-layer LSTM followed by a densely connected layer as represented in Figure 6.5.

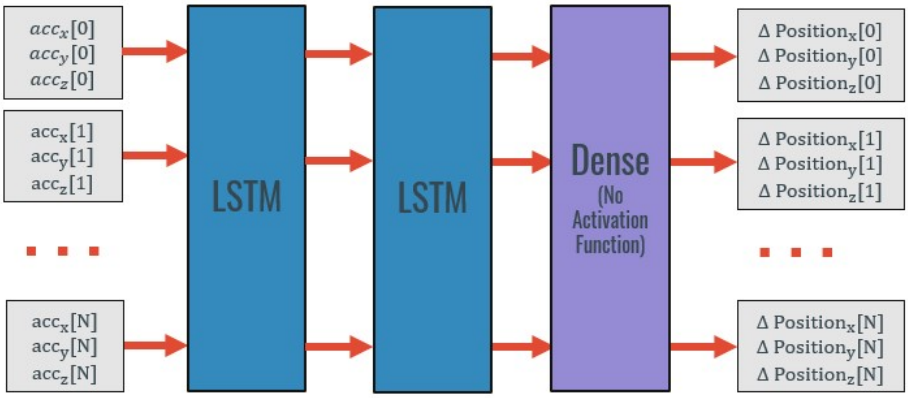


Figure 6.5: Neural network architecture with no type of warm start for 3-axis translation estimate

However, in this approach, different inputs and outputs from the ones used in the previous approaches were experimented. The inputs utilized were different combinations of the 3D linear accelerations, 3D angular velocity and 3D position estimated by double integration. This means that the three different models were trained: the first one using the linear acceleration in the three axes as input, the second using the 3D linear acceleration and the 3D angular velocity as input, and the third one using the 3D linear acceleration, the 3D angular velocity, and the 3D estimated position (through double integration) as inputs. All these models were trained to predict the 3D displacement.

Common Experimental Methodology

All the earlier explained models were trained using an Adam optimizer [56] and mean squared error (MSE) loss. As the methods used in the previous chapter, a learning rate of $5x10^{-3}$, and weight decay of $1x10^{-5}$ were applied. In these experiments, a batch size of 1 was used due to the different sized inputs.

The same split as described in Figure 5.2 was applied. The train set was then split with the data from seven patients for training and the data from the other three patients for validation. Early stopping was also carried out to interrupt the training when the validation loss was no longer decreasing over the course of 20 epochs to reduce the risk of overfitting. Finally, gradient clipping was applied to avoid exploding gradients.

The selection of the moving periods was conducted using the reference from the classification models. This data selection means that only the data from moving periods determined by the ground truth was used to train and evaluate the models mentioned above. Consequently, the output was the translation that occurred during that period. This way, imagining a moving period of 200 samples, the input, depending on the model, is composed of the inertial data during that period. The output also contains 200 samples. In the output, each timestep represents the translation that happened since the beginning of that moving period.

6.1.2 Combining Methodologies - Stacked Neural Network

After identifying stationary periods and estimating translation for the periods of movement, it is possible to combine both approaches to obtain a displacement estimate during an entire trial. This way, as previously presented, one can use the classification models developed in Chapter 5 to understand in which periods motion occurs. Afterwards, using the regression models previously presented in this chapter, one can obtain a translation estimate in those periods. Thus, it is possible to get a translation estimate in a full trial that can be compared with the ground truth position provided by the Qualisys optical motion system.

The focus of this section is the development of a model that integrates both zero-velocity detection and translation estimates for periods of movement simultaneously. This way, an LSTM-based stacked neural network is proposed. This neural network is presented in Figure 6.6.

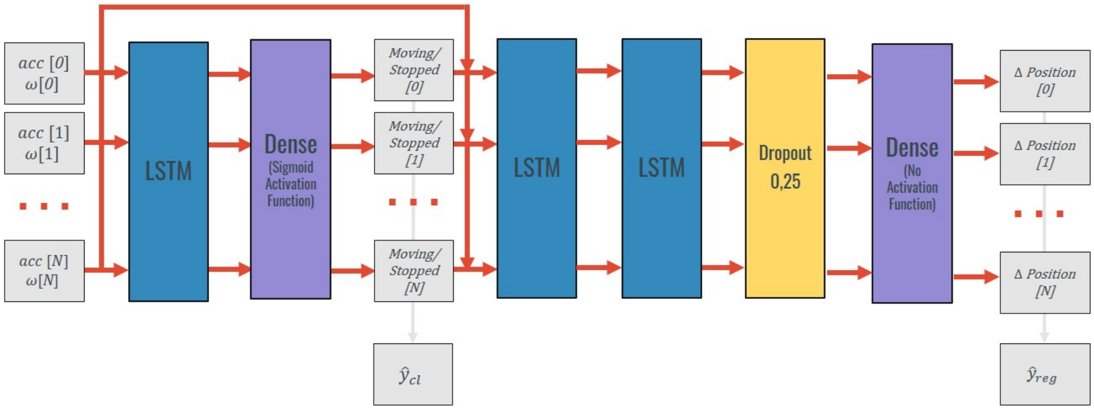


Figure 6.6: Stacked Neural Network for simultaneous classification and regression

The first part of the network is composed of an LSTM layer with 30 units and a densely connected layer with the intention of identifying the timesteps with zero-velocity. This forces the model to extract this information from the data which can be regarded as a feature. After, the output of that layer is concatenated with the inertial data input. This way, the input of the following layers is composed of inertial data of each timestep and its labelling. The model is composed of other two extra LSTM with 30 units each. Afterwards, a dropout layer with a dropout rate of 0.25 was used. Since this model is more complex than the ones previously mentioned in this chapter, this layer was added to avoid overfitting. The final output (3D position) is calculated with a densely connected layer that reduces the output to 3D.

In the methodologies previously presented in this chapter, the inputs of the models were the inertial measurements from periods classified as periods of motion. In this method, distinctively from the last ones, the inputs used are composed by the inertial data of an entire trial. This input contains inertial data from periods of movement and of zero-velocity. The outputs include the translation in each axis from the beginning of the trial. This translation is calculated by subtracting the 3D ground truth position at the beginning of the trial from the 3D ground truth position in each timestep.

This stacked neural network was built to identify moving/stopped moments and to estimate the displacement that occurs during a data collection trial. To achieve this result, the loss function of the neural network was customized so that, during the training process, it takes into consideration, simultaneously, the classification error and the regression error.

This way, the loss function (L) for a trial with N samples can be defined as:

$$L(y_{cl}, \hat{y}_{cl}, y_{reg}, \hat{y}_{reg}) = (1-\alpha) \times L_{Binary}(y_{cl}, \hat{y}_{cl}) + \alpha \times L_{MSE}(y_{reg}, \hat{y}_{reg}) =$$
$$(1-\alpha) \times \left(-\frac{1}{N} \sum_{i=0}^{N-1} y_{cl} \times log(p(\hat{y}_{cl})) + (1-y_{cl}) \times log(1-p(\hat{y}_{cl})) \right) + \tag{6.10}$$
$$+\alpha \times \left(\frac{1}{N} \sum_{i=0}^{N-1} (y_{reg} - \hat{y}_{reg})^2 \right)$$

In which y_{cl} represents the ground truth zero-velocity labelling and $\hat{y}_{cl}$ the respective model estimate. $p(\hat{y}_{cl})$ represents the probability of the IMU having velocity zero in that timestep. y_{reg} symbolizes the translation's ground truth and $\hat{y}_{reg}$ the model's translation estimate. α represents a manually set weight: an α larger than 0.5 means that the MSE of the regression has a higher weight in the loss function; an α smaller than 0.5 means that the binary cross-entropy loss of the classification has a higher weight in the loss function.

Again, this model was trained using an Adam optimizer [56] with a learning rate of $5x10^{-3}$, weight decay of $1x10^{-5}$, and batch size 1. The same split as described in Figure 5.2 was applied. From the ten patients present in the train set, data from seven of them was used for training, and the data from the other three was used for validation. Early stopping was conducted to stop the training when the validation loss was not decreasing over the course of 20 epochs.

6.2 Results and Discussion

6.2.1 Translation estimate of moving moments using regression

To evaluate the performance of the models in regressing the translation in periods of movement, the absolute trajectory error (ATE - Equation 5.7), similarly to what was done in Section 5.2.3, was used. Additionally, to allow a more straightforward interpretation of the results, the mean absolute error (MAE) of the 3D translation estimate was also evaluated.

First of all, it is essential to remind that differently from what was done in Section 5.2.3 – in which the position throughout the whole trial was evaluated – in this section only the displacement in a moving period is being evaluated. This means that an error occurred in the position estimate in the same trial but in a previous movement period does not affect the estimates being assessed in this section since that every moving period is being evaluated independently. A comparison with Section 5.2.3 is presented in the next section.

The results of the different regression methods are presented in Table 6.1. These show the performance of the developed models in comparison with a naive double integration when applied on the validation and test set. The first thing to note is that the simple double integration in the validation set leads to an average prediction error higher than in the test set. This way, it is expected that the models can achieve better performance also in the test set comparatively with the validation set.

Starting with the models in which the warm start was conducted using manually set weights, it is possible to observe that none of them achieved better results when compared with the double integration methods. Relatively to these models, different forget, input and output biases were tested. These biases are responsible for altering the sigmoid activation function values of the

Table 6.1: MAE and ATE of the regression models on the validation and test sets for the moving periods. Integrative - Warm-Start with "Integrative Weights" (u- units; b- biases of the forget, input and output gates); Pre-Trained - Warm-Start with "Pre-Trained Weights" (without drift removal- both layers' weights initialized with weights from a model that simulates a simple integration; with drift removal- first layer weights initialized with weights that simulate a simple integration with drift removal and second layer weights initialized with weights from a model that simulates a simple integration); No initialization- No warm start (features used: Acc-3D acceleration; Ang Vel- 3D angular velocity; Double Int Pos- 3D displacement obtained through acceleration double integration).

Model	Validation		Test	
	MAE	ATE	MAE	ATE
Double Integration	0.9859	1.8811	0.5376	0.8336
Double Integration + Drift Removal	0.6098	1.0275	0.3305	0.4597
Integrative (u=1; b=4.595)	0.9962	1.3601	0.7973	1.1122
Integrative (u=10; b=4.595)	0.9898	1.3503	0.7843	1.0867
Integrative (u=1; b=9)	1.1064	2.1514	0.6353	1.0373
Integrative (u=10; b=9)	1.0136	2.0746	0.5468	0.8719
Integrative (u=1; b=2.20)	1.3483	1.4153	1.0616	1.1139
Integrative (u=10; b=2.20)	1.3480	1.4152	1.0613	1.1138
Pre-trained (without drift removal; u= 10)	0.8978	1.5011	0.5455	0.8360
Pre-trained (with drift removal; u= 10)	0.7252	1.1640	0.4725	0.6971
No initialization (Acc; u=10)	0.5443	0.7740	0.5872	0.7833
No initialization (Acc + Ang Vel; u=10)	0.5872	0.8465	0.7272	1.0237
No initialization (Acc + Ang Vel + Double Int Pos; u=10)	0.6383	0.9722	0.5294	0.7537

gates. As already mentioned, a value close to 1 means that the gate lets everything pass and a value close to 0 means that the gate blocks all the information.

Analyzing the Equations 6.1, 6.2, and 6.3, and considering that all the gates weights were set to a value close to zero (a random value between -10^{-6} and 10^{-6}), then the biases were the main component responsible for controlling the information that went through each gate. Considering the sigmoid activation function of each gate, then a bias value of 2.20 leads to a gate value of approximately 0.9. A bias value of 4.595 leads to a gate value of approximately 0.99, and a bias of 9 leads to a gate value of 0.999. The closer to 1 the values of the gate are, the more similar is the initial state of the network to a double mathematical integration (as it is proved by Equations 6.8, and 6.9).

This way, ideally, the gate values would be as high as possible. However, it is necessary to take into account the process of updating the weights and biases of the network - through backpropagation. Considering the "S" shaped form of the sigmoid function, then the slope of the curve on very high positive and negative values is almost inexistent. This means that when calculating the gradients during the process of backpropagation, practically no changes in the values of the gates will occur, and these will continue to let all the information pass.

Considering also that the hyperbolic tangents represented in Equations 6.4 and 6.6 were replaced by linear functions, then the model is converted to an almost linear model (the only non-linearity is present in the sigmoid activation functions of the gates).

The results support this analysis: for any of the manually set biases setting the units to 1 or to 10 led to very similar results which can make us conclude that, even though the initial biases values were set to relatively low values, the models are approximating a linear model (since that a combination of linear models is equivalent to a single linear model). Either way, comparing these models, it is possible to conclude that the one that yields better results was the one with 10 units and a bias of 9.

Relatively to the models in which "pre-trained" weights were used to initialize the model, when compared with the naive double integration process, both achieved better performance. However, when compared with the double integration method with drift removal, their performance is worse. In these models, contrarily to what was done in the models in which the weights were set manually, the LSTM original equations were maintained. Therefore, it is expected that these models can deal better with non-linearities as suggested by the performance improvement.

Comparing the performance of the 'pre-trained' models, the model with drift removal (i.e. the model in which the first layer weights were initialized with weights that simulate a simple integration with drift removal) achieved a better performance than the model without drift removal (i.e. the model in which both layers' weights were initialized with weights from a model that simulates a simple integration). This shows that accounting for the drift when initializing the weights can improve the results.

Finally, looking at the models in which no customized initialization was performed, it can be seen that the model that used as inputs the 3D linear acceleration achieves the better performance in the validation set. Important to recall that all the models without customized initialization were trained using as input a combination of the 3D data and, as output, the 3D displacement. This way, the models may be capable of learning interdependences existent between data of different axis differently from what is happening in the integrative and pre-trained models in which data from a single axis at a time is provided to each model during training.

The "no initialization" methods present additional interesting behaviours. The one that uses only the 3D acceleration as input, contrarily to what happens in the rest of the models presents a similar performance both in the validation and in the test set. The one that uses both 3D acceleration and 3D angular velocity as input shows an even better performance in the validation set than on the test set contrarily to what occurs in the other models. This could mean that this model adapted to well to the validation set since that although it presents some of the lowest average errors in the validation set, the same does not happen in the test set.

In conclusion, one can state that for the developed models, in the test set – a set in which the double integration position error is relatively low – the models could not improve the performance of the double integration followed by drift removal. In the validation set – a set in which the double integration error is higher – the "pre-trained" models and the models without warm-start were capable of reducing the average error of the translation predicted by double integration. However, one should recall that during the development of these models, there was data leakage from the validation set since early stopping was implemented to track the validation loss during the training procedure.

6.2.2 Translation estimates using Zero-Velocity Detection Updates and Regression of Motion Periods

The next step consisted of combining the classification models presented in the last chapter with the best regression models presented in this chapter and evaluate their combined performance in the test trials translations estimates. The best model of each of the three presented methodologies was selected using the test set performance: Integrative (u=10; b=9); Pre-trained (with drift removal; u= 10); No initialization (Acc + Ang Vel + Double Int Pos, u=10)

The best result obtained through the combination of a classification method with double integration in the Section 5.2.3 was achieved by the 6-Layer LSTM classifier with drift correction as represented in Table 5.4. The results of the application of this classifier to perform zero velocity update together with double integration with drift correction for displacement estimate are also represented in Table 6.2 for comparison purposes.

To achieve this comparison, the same 6-LSTM classifier was used to conduct zero-velocity detection. Afterwards, the mentioned regression models were used to estimate the translation between zero-velocity periods. The results of the application of these methods in the test set are presented in Table 6.2.

Table 6.2: MAE and RMSE of the translation estimate in each axis of the full trial 3D translation estimate using the different regression models. (6-Layer LSTM Zero Velocity Update - 6-Layer LSTM identification of zero velocity moments; Integrative- Warm-Start with "Integrative Weights" (u- units; b- biases of the forget, input and outputgates); Pre-trained (with drift removal- first layer weights initialized with weights that simulate a simple integration with drift removal and second layer weights initialized with weights from a model that simulates a simple integration); No initialization- No warm start (features used: Acc-3D acceleration; Ang Vel-3D angular velocity; Double Int Pos- 3D displacement obtained through acceleration double integration)

Zero-Velocity Classifier	Regressor	Error (m)	p_x	p_y	p_z	$\|\mathbf{p}\|$
6-LSTM	Double Integration + Drift Correction	RMSE	0.50±0.31	0.51±0.49	0.42±0.21	0.93±0.44
		MAE	0.42±0.28	0.43±0.43	0.35±0.18	0.83±0.39
6-LSTM	Integrative (u=10; b=9)	RMSE	1.63±1.64	1.58±2.37	1.01±0.89	2.68±2.85
		MAE	1.44±1.54	1.31±2.20	0.85±0.80	2.38±2.65
6-LSTM	Pre-trained (with drift removal; u= 10)	RMSE	0.87±0.59	0.95±0.84	0.31±0.29	1.43±0.91
		MAE	0.71±0.55	0.78±0.72	0.24±0.25	1.80±1.29
6-LSTM	No initialization (Acc + Ang Vel + Double Int Pos; u=10)	RMSE	1.29±0.75	0.52±0.2	0.35±0.16	1.47±0.72
		MAE	1.09±0.66	0.42±0.19	0.29±0.14	1.30±0.65

Starting by comparing the double integration with drift correction results with the ML regressors, one can conclude that these could not improve the overall translation estimate during an entire trial. These results were already expected since the regression methods were not able to improve the double integration with drift removal translation estimate for moving periods in the test set (as shown in Table 6.1).

By comparing the different regressors with each other, it is possible to observe that the 'pre-trained' model achieves a lower average error in the x-axis while the 'no initialization' model achieves a better performance in the y-axis. In the z-axis and in the 3D translation estimate, the performance of these models is similar.

To allow a visual examination, the results of a trial prediction using these models are presented in Figure 6.7. The presence of drift is visible in all the models' translation estimated when compared with the ground truth position. This drift is particularly more visible in the x-axis (the axis in which more movement occurs).

Another interesting detail is the fact that the translation estimate provided by the "no initialization" model is noisier than the other models' estimates (more visible in the y and z-axis predictions). This noisier forecasts could be eventually corrected using a low-pass filter to remove that high-frequency noise. The presence of this noise is noticeable when evaluating the MAPE of the estimate of total translation performing during an entire trial as represented in Table 6.3 (this table can be compared to the one presented in Table 5.5). The presence of noise generates an increased estimate of total translation which results in a higher MAPE. In the "integrative" and "pre-trained" models this noise is not so present (as it is visible in Figure 6.7) and, therefore, the errors in the total translation estimate are lower.

Table 6.3: Evaluation of the estimation of the total translation ($\widehat{trans}$) performed during all the test trials using the different regression models (Uncorrected - 3D position estimate obtained by the double integration of the linear accelerations without any correction; RF Zero Velocity Update - Random Forest identification of zero velocity moments; LSTM Zero Velocity Update - LSTM identification of zero velocity moments; 6-Layer LSTM Zero Velocity Update - 6-Layer LSTM identification of zero velocity moments)

Zero-Velocity Classifier	Regressor	$trans$	$\widehat{trans}$	MAPE
6-LSTM	Double Integration + Drift Correction	30.34±3.49 m	27.34±3.56 m	9.84%
6-LSTM	Integrative (u=10; b=9)	30.34±3.49 m	28.78±3.87 m	10.78%
6-LSTM	Pre-trained (with drift removal; u=10)	30.34±3.49 m	27.55±3.11 m	9.43%
6-LSTM	No initialization(Acc + Ang Vel +Double Int Pos; u=10)	30.34±3.49 m	50.15±11.64 m	70.11%

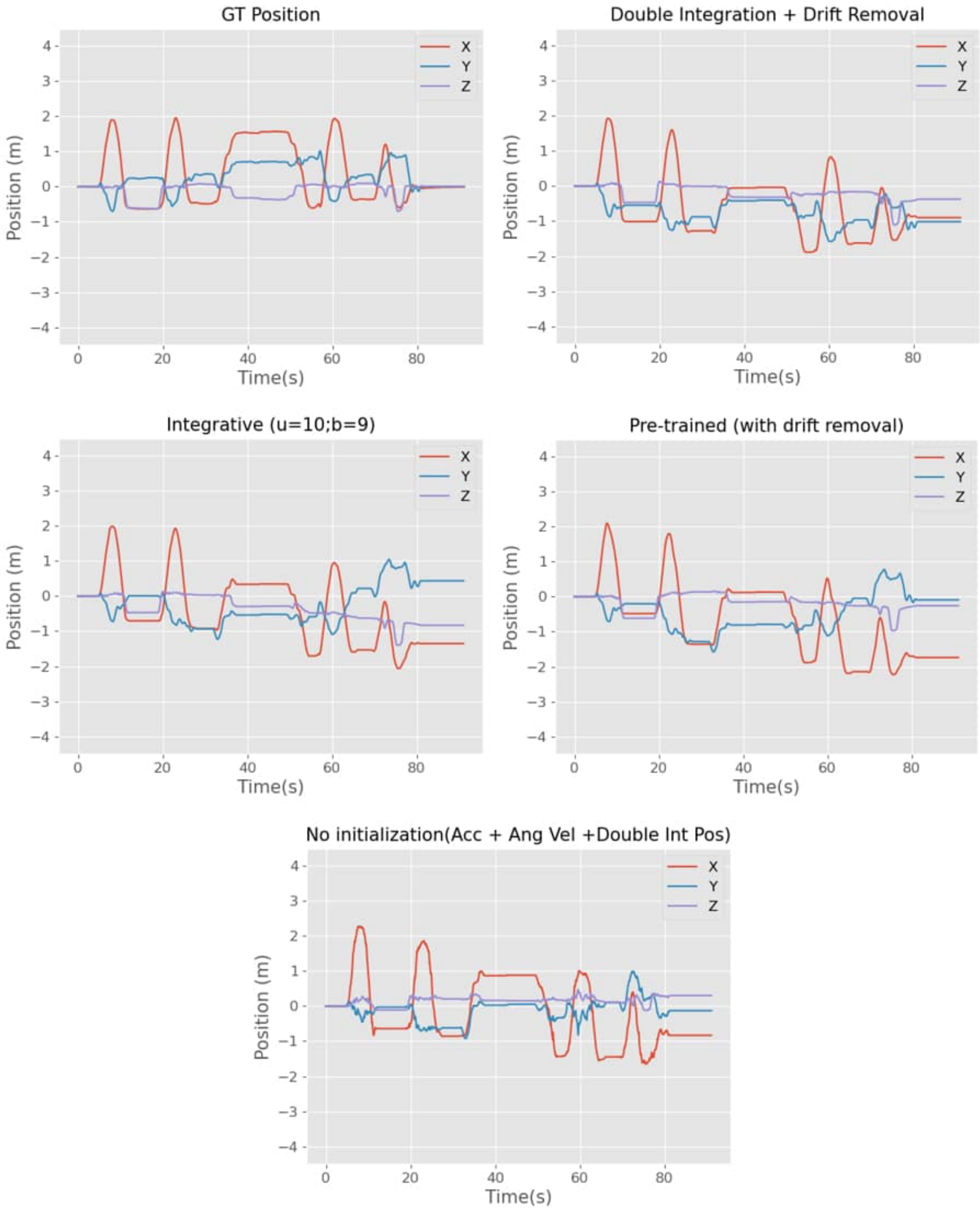


Figure 6.7: Full trial translation estimate applying the different developed regression models with 6-LSTM for zero velocity detection

6.2.3 Zero-Velocity Detection and Translation Estimates using a Stacked Neural Network

The results of the stacked neural network application – which simultaneously performs both the detection of zero-velocity and the estimate of the translation in each axis in moving periods – are presented in Table 6.4. These results are compared with the methodology that exhibited higher performance in the previous sections (6-LSTM ZUPT with median filtering + double integration + drift correction). Additionally, the neural network described in Figure 6.5 (architecture from the "no initialization" approach from the previous section) was also trained for predicting the translations in the whole trial using the mean squared error as the loss function.

Table 6.4: MAE and RMSE of the translation estimate in each axis and of the 3D translation estimate applying different models.

Model	Accuracy	Error (m)	p_x	p_y	p_z	$\|\mathbf{p}\|$
6-LSTM ZUPT (Median Filtering) + Double Integration + Drift Correction	0.9454	RMSE	0.50±0.31	0.51±0.49	0.42±0.21	0.93±0.44
		MAE	0.42±0.28	0.43±0.43	0.35±0.18	0.83±0.39
No Initialization Architecture (Section 6.1.1.3)	-	RMSE	0.63±0.23	0.43±0.07	0.20±0.08	0.81±0.18
		MAE	0.51±0.16	0.35±0.07	0.15±0.04	0.70±0.14
Stacked ($\alpha = 0.5$)	0.9502	RMSE	0.81±0.16	0.44±0.06	0.19±0.09	0.95±0.15
		MAE	0.62±0.13	0.37±0.07	0.15±0.06	0.78±0.14
Stacked ($\alpha = 0.8$)	0.9509	RMSE	0.78±0.21	0.44±0.06	0.19±0.08	0.93±0.21
		MAE	0.59±0.13	0.36±0.06	0.15±0.05	0.78±0.14
Stacked ($\alpha = 1$)	-	RMSE	0.54±0.18	0.42±0.07	0.20±0.08	0.73±0.17
		MAE	0.44±0.14	0.35±0.07	0.15±0.05	0.64±0.14

The stacked neural network methodology was experimented using three different α values for the loss function (Equation 6.10) - 0.5, 0.8 and 1. A α value of 0.5 gives similar weight to the classification loss and to the regression loss. A α value of 0.8 gives higher weight to the regression loss. A α value of 1 means that the loss function is equivalent to the mean squared error loss function. This way, setting the α to 1 removes the ability of the loss function consider the error existent in the identification of stationary periods. Therefore, in the case in which α is

1, the zero-velocity detection accuracy was not evaluated since the network was not trained for that.

Relatively to the zero velocity detection accuracy, the stacked neural networks were able to obtain a higher accuracy than the 6-LSTM zero velocity classifier with post-processing median filtering.

Looking at the average translation errors, the average error of the translation estimate performed by double integration with drift correction after detecting zero-velocity with the 6-LSTM classifier presents a lower error for the x-axis position (the axis with greater movement) when compared with the networks trained using an α of 0.5 and 0.8, and a similar one when compared with the one trained with an α of 1 and with the "no initialization" architecture. For the y- and z-axis, all the stacked neural networks and the "no initialization architecture" show a better performance. The best average 3D position error is achieved by the neural network with a loss function α of 1.

By comparing the stacked neural networks with different α in the loss function, it is possible to see that the results are similar for the three methods. The network in which α was set to 1 achieves better results in the X-axis average error than the other ones, which results in a lower error in the 3D translation estimate.

Comparing the no initialization architecture with the stacked neural network architecture, the stacked with α equal to 1 achieved better performance. Considering that the loss function of both architectures is the same (mean squared error), it is possible to conclude that the increased complexity of the stacked model allows to increases the performance.

The results presented in Table 6.2 can be complemented by the analysis of Figure 6.8. First of all, it is visible that the stacked neural networks and the no initialization architecture translation estimates possess a large quantity of high-frequency noise. This could be eventually solved with the use of a low-pass filter. The presence of this high noise quantity is proved by the MAPE presented by these approaches in Table 6.5, since the presence of noise results in higher total translation estimates.

Additionally, the stacked neural networks and the "no initialization" architecture show difficulties in dealing with negative position values (as it is noticeable in the period between the 40s and 130s). Finally, the networks also seem to add a bias in the final periods (also present in other of the analyzed examples). These results show that further improvements must be achieved in these architectures to solve these problems.

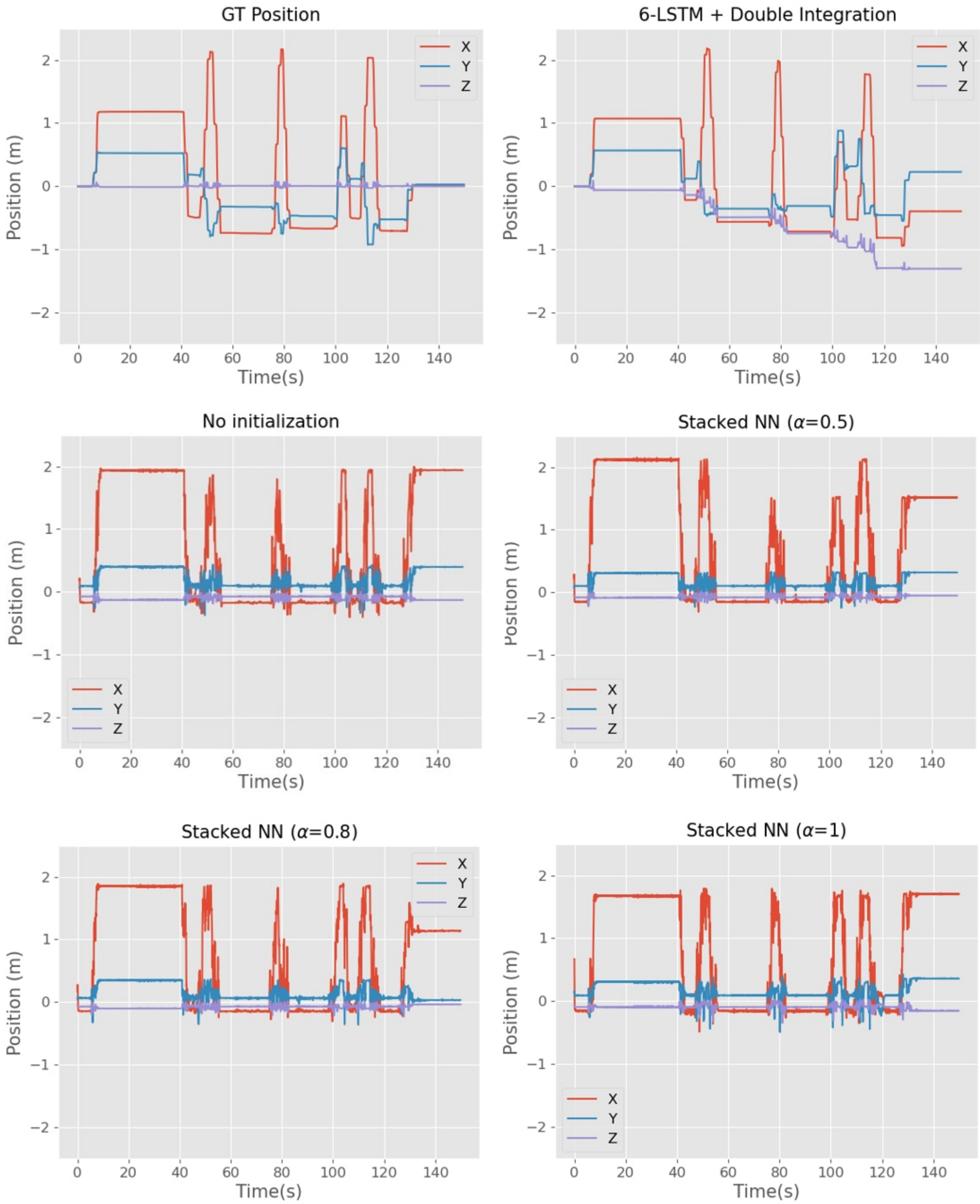


Figure 6.8: Full trial translation estimate applying the stacked neural network methodology (6-Layer LSTM Zero Velocity Update - 6-Layer LSTM identification of zero velocity moments; Stacked - Stacked Neural Network; α- Stacked loss function term)

Table 6.5: Evaluation of the estimation of the total translation ($\widehat{trans}$) performed during all the test trials (6-Layer LSTM ZUPT - 6-Layer LSTM identification of zero velocity moments; Stacked - Stacked Neural Network; α - Stacked loss function term)

Model	$trans$	$\widehat{trans}$	MAPE
6-LSTM ZUPT (Median Filtering) + Double Integration + Drift Correction	30.34±3.49 m	27.34±3.56 m	9.84%
No Initialization Architecture (Section 6.1.1.3)	30.34±3.49 m	123.78±22.96 m	315.57%
Stacked ($\alpha = 0.5$)	30.34±3.49 m	115.67±18.40 m	287.81%
Stacked ($\alpha = 0.8$)	30.34±3.49 m	110.37±19.04 m	269.88%
Stacked ($\alpha = 1$)	30.34±3.49 m	147.60±33.15 m	398.51%

6.3 Summary

In this chapter, different methodologies were introduced for regressing the displacement on the 3D space of IMUs and respective associated body segments.

The first part of the chapter consisted of using different techniques for estimating the translation of the IMU during periods of motion. Distinct LSTM-based models were experimented as well as different LSTMs initializations. The idea behind the various initializations was to provide a warm-start to the models. This way, these networks would start their training process already knowing how to perform a double mathematical integration of the linear acceleration data and, during the training procedure, these would only need to learn how to account for the different errors existent in the inertial data. These methods results suggest that no significant improvements were achieved with the developed models when compared with a double integration with drift removal (as performed in the previous chapter).

The next part of the chapter consisted of the proposal of a stacked neural network. By customizing the model loss function, this model was capable of simultaneously detect zero-velocity periods and estimate the translation of the IMUs throughout time. The results of these methods are promising but present some inconsistencies suggesting that future work must be focused on this type of methodology.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Human motion tracking using IMUs has become a popular solution due to the portability of the method, its capability of collecting motion continuously, low power consumption, and low cost. However, low-cost IMUs suffer from accuracy degradation in long periods of use. This accuracy degradation is due to the several errors that affect the sensors that constitute an IMU - three mutually orthogonal accelerometers and gyroscopes. These errors include biases, scale factor errors and random noise. These errors are often non-linear and change over time. Additionally, other errors arise from the non-orthogonality of these sensors when integrated into IMUs. The presence of these errors affects the estimate of the position and orientation of different body segments during the monitoring of the subject motion considerably. Due to the sequential nature of the estimations, the error quickly accumulates, and the estimates diverge. Particularly during a rehabilitation session, it is essential to accurately determine the location and orientation of the different body segments in order to provide an adequate evaluation of the patient's motion.

This work focused on providing an improvement in the position estimate of different body segments (to which an IMU is attached) when a subject is performing several movements using inertial data. For this purpose two objectives were defined: 1. zero-velocity detection for identifying periods in which the IMU is stationary; 2. regression of translations that took place in periods of movement.

The achieve the purposed goals, this work incorporated the analysis of several publicly available datasets that include inertial measurements collected in different setups. The pros and cons of each dataset were weighed, and one was selected. The selected dataset provided IMU data from several body parts collected during the execution of several industry-like movements. The ground truth position of each IMU was determined using an optical motion capture system. The fact that this dataset contained a large variety of actions performed with different velocities and with several stationary periods made this dataset attractive since these actions resemble the movements performed during a rehabilitation sesssion.

Zero-velocity detection and translation estimate during periods of motion were implemented using the selected dataset and applying ML techniques. Lastly, an ML model that can accomplish both tasks (zero-velocity detection and translation estimate) was also evaluated, showing promising results.

The developed zero-velocity detectors provided accurate detection of IMU stationary periods which allowed to constraint the error that accumulates during the integration of the noisy inertial measurements. The application of these detectors to correct the velocity estimation allowed an enhanced tracking when compared with the simple double integration of the accelerometer data. The removal of the estimated drift between zero-velocity moments provided an additional improvement.

Relatively to the regression models, used to estimate translations between zero-velocity moments, no significant improvements in comparison with more straightforward methods - as the double integration with drift removal - were achieved.

The approach that combined both zero-velocity detection and translations estimate showed promising results with high accuracy in detecting stationary periods and low error in estimating the translation performed by the IMU during more extended periods of time. However, the results obtained with this method still present some difficulties specially dealing with negative outputs.

The majority of the work existent in this area is dedicated to different variants of the problem tackled in this project. Some authors use inertial data to track subjects walking and estimate the trajectory of their path or their velocity. Regarding zero-velocity detection, the existent work concentrates into zero-velocity detection during gait using IMUs placed on the foot, lacking both movement diversity and data from several body segments. Others concentrate on evaluating 6-DoF pose estimate for tracking in areas as robotics and automation and, therefore, do not focus on the more complex task of tracking human motion. This absence of work in the area difficulted the comparison of the developed methods with prior techniques.

7.2 Future Work

This work provides several research opportunities. First of all, further optimization must be performed in all the developed models. Zero-velocity detection was performed with high accuracy, although it can be improved with more exploration and optimization. The regression estimate presented results with lower performances and, this way, future work can be focused on this area. Some examples of work that can pottentially be performed on this part, following the methods implemented in this project include: initializing the weights of an LSTM-based model using simultaneously data from the 3-axis to provide a warm start to the network and the ability to capture information at the same time from inertial data from different axes; conducting warm start for the stacked neural network methodology- particularly on the first layer responsible for detecting stationary periods using the weights optimized during in the zero-velocity detection LSTM classifier. The work should also be focused in improving and solving the capacity to adapt to negative values presented by the approach that combined simultaneous zero-velocity detection and translations estimate as well as filtering the results obtained in that section.

Other future work includes the validation of the developed methods in different datasets. Although the used dataset presents wide motion variety and data from several body segments, it is vital to understand if the developed models can generalize to other parts of the body as well as to different IMUs. The final goal should be to achieve a model that can improve the human motion tracking being applied to data from any IMU that is placed in any body segment.

Following this line of work, the developed models should be evaluated when applied on the SWORD inertial motion trackers, again to understand if these methodologies can generalize to data from those trackers.

Additionally, it is also essential to extend these models to orientation estimation using the inertial data to perform an end-to-end 6-DoF trajectory estimate. The models developed in this work rely on an effective orientation estimation and subsequent gravity removal. A model that could perform both tasks could lead to a more robust tracking.

Finally, future works suggestions include the implementation of a system based on the developed models that could work on real-time. This system could consist of a zero-velocity detector model - always classifying periods as periods of movement or stationary periods - and a translation estimate regressor model - that would rectify the displacement estimates between the detections of zero-velocity periods. This implementation could allow to improve the live feedback given to the patients during a rehabilitation session like the one conducted with the SWORD system.

References

[1] F. Abyarjoo, A. Barreto, J. Cofino, and F. R. Ortega. Implementing a sensor fusion algorithm for 3d orientation detection with inertial/magnetic sensors. In *Innovations and advances in computing, informatics, systems sciences, networking and engineering*, pages 305–310. Springer, 2015. Cited on page 23.

[2] D. E. Adams. An Introduction to Inertial Navigation. *Journal of Navigation*, 9(3):249–259, 2007. Cited on pages 9, 10, 11, 12, and 13.

[3] R. Adhikari and R. K. Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013. Cited on pages 95 and 96.

[4] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM SIGGRAPH 2004 Papers, SIGGRAPH 2004*, 1(212):584–591, 2004. Cited on page 1.

[5] C. C. Aggarwal. *Data mining: the textbook*. Springer, 2015. Cited on pages 94, 95, 96, and 97.

[6] J. K. Aggarwal and S. Park. Human motion: Modeling and recognition of actions and interactions. *Proceedings - 2nd International Symposium on 3D Data Processing, Visualization, and Transmission. 3DPVT 2004*, pages 640–647, 2004. Cited on page 1.

[7] P. Aggarwal, Z. Syed, X. Niu, and N. El-Sheimy. A standard testing and calibration procedure for low cost MEMS inertial sensors and units. *Journal of Navigation*, 61(2):323–336, 2008. Cited on pages 12 and 13.

[8] E. Akeila, Z. Salcic, and A. Swain. Reducing low-cost ins error accumulation in distance estimation using self-resetting. *IEEE Transactions on Instrumentation and Measurement*, 63(1):177–184, 2013. Cited on page 29.

[9] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15:31314–31338, 12 2015. Cited on page 2.

[10] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Activity Recognition Using Inertial Sensing for Healthcare, Wellbeing and Sports Applications: A Survey. *Proc. of the 23th Int. Conf. on Architecture of Computing Systems*, pages 167–176, 2010. Cited on pages 1 and 3.

[11] E. R. Bachmann, I. Duman, U. Usta, R. B. McGhee, X. Yun, and M. Zyda. Orientation tracking for humans and robots using inertial sensors. In *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No. 99EX375)*, pages 187–194. IEEE, 1999. Cited on page 21.

[12] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. Cited on page 31.

[13] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. Cited on page 98.

[14] T. Beth, I. Boesnach, M. Haimerl, J. Moldenhauer, K. Bös, and V. Wank. Characteristics in human motion – From acquisition to analysis. *International Conference on Humanoid Robots (Humanoids), Karlsruhe/M\\"unchen, Germany*, 2003. Cited on page 2.

[15] D. Bhatt, P. Aggarwal, P. Bhattacharya, and V. Devabhaktuni. An enhanced MEMS error modeling approach based on Nu-Support vector regression. *Sensors (Switzerland)*, 12(7):9448–9466, 2012. Cited on page 97.

[16] R. Bogue. Recent developments in MEMS sensors: A review of applications, markets and technologies. *Sensor Review*, 33(4):300–304, 2013. Cited on page 12.

[17] N. Buduma and N. Locascio. *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.", 2017. Cited on page 68.

[18] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016. Cited on pages 34 and 37.

[19] A. Cappozzo, U. Della Croce, A. Leardini, and L. Chiari. *Gait and Posture*, (2):186–196 title = Human movement analysis using stereophotogrammetry. Part 1: Theoretical background, volume = 21, year = 2005. Cited on page 1.

[20] C. Chen, X. Lu, A. Markham, and N. Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. Cited on page 30.

[21] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham, and N. Trigoni. Selective sensor fusion for neural visual-inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10542–10551, 2019. Cited on page 30.

[22] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni. Oxiod: The dataset for deep inertial odometry. *arXiv preprint arXiv:1809.07491*, 2018. Cited on pages 35 and 37.

[23] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni. Deep learning based pedestrian inertial navigation: Methods, dataset and on-device inference. *arXiv preprint arXiv:2001.04061*, 2020. Cited on page 30.

[24] H. Chen, P. Aggarwal, T. M. Taha, and V. P. Chodavarapu. Improving Inertial Sensor by Reducing Errors using Deep Learning Methodology. *Proceedings of the IEEE National Aerospace Electronics Conference, NAECON*, 2018-July:197–202, 2018. Cited on pages 12 and 13.

[25] K. M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time part II: Applications to human modeling and markerless motion tracking. *International Journal of Computer Vision*, 63(3):225–245, 2005. Cited on page 1.

[26] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. Cited on page 30.

[27] F. D. Correia, A. Nogueira, I. Magalhães, J. Guimarães, M. Moreira, I. Barradas, M. Molinos, L. Teixeira, J. Pires, R. Seabra, J. Lains, and V. Bento. Digital versus conventional rehabilitation after total hip arthroplasty: A single-center, parallel-group pilot study. *Journal of Medical Internet Research*, 21(6), 2019. Cited on pages 3 and 5.

[28] F. D. Correia, A. Nogueira, I. Magalhães, J. Guimarães, M. Moreira, I. Barradas, M. Molinos, L. Teixeira, J. Tulha, R. Seabra, J. Lains, and V. Bento. Medium-term outcomes of digital versus conventional home-based rehabilitation after total knee arthroplasty: Prospective, parallel-group feasibility study. *Journal of Medical Internet Research*, 21(2), 2019. Cited on pages 3, 4, and 5.

[29] F. D. Correia, A. Nogueira, I. Magalhães, J. Guimarães, M. Moreira, I. Barradas, L. Teixeira, J. Tulha, R. Seabra, J. Lains, and V. Bento. Home-based Rehabilitation With A Novel Digital Biofeedback System versus Conventional In-person Rehabilitation after Total Knee Replacement: a feasibility study. *Scientific Reports*, 8(1):1–12, 2018. Cited on pages 3 and 5.

[30] S. Cortés, A. Solin, and J. Kannala. Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2018. Cited on page 28.

[31] S. Cortés, A. Solin, E. Rahtu, and J. Kannala. Advio: An authentic dataset for visual-inertial odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 419–434, 2018. Cited on page 35.

[32] J. A. Cozens. Robotic assistance of an active upper limb exercise in neurologically impaired patients. *IEEE Transactions on Rehabilitation Engineering*, 7(2):254–256, June 1999. Cited on page 2.

[33] R. G. J. Damgrave and D. Lutters. The drift of the xsens moven motion capturing suit during common movements in a working environment. In *Proceedings of the 19th CIRP Design Conference–Competitive Design*. Cranfield University Press, 2009. Cited on page 42.

[34] D. Das and R. R. Mary. Real time noise filtering for low cost imu sensors. *Asian Journal of Information Technology*, 16(6):536–543, 2017. Cited on page 26.

[35] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979. Cited on page 29.

[36] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006. Cited on page 20.

[37] Y. Dong. Mems inertial navigation systems for aircraft. In *MEMS for Automotive and Aerospace Applications*, pages 177–219. Elsevier, 2013. Cited on page 7.

[38] M. El-Diasty, A. El-Rabbany, and S. Pagiatakis. Temperature variation effects on stochastic characteristics for low-cost MEMS-based inertial sensor error. *Measurement Science and Technology*, 18(11):3321–3328, 2007. Cited on pages 12 and 13.

[39] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan. Aboldeepio: A novel deep inertial odometry network for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2019. Cited on page 31.

[40] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan. Orinet: Robust 3-d orientation estimation with a single particular imu. *IEEE Robotics and Automation Letters*, 5(2):399–406, 2019. Cited on page 31.

[41] Y. Fan, P. Li, and Z. Song. Dynamic least squares support vector machine. In *2006 6Th World Congress on Intelligent Control and Automation*, volume 1, pages 4886–4889. IEEE, 2006. Cited on page 96.

[42] T. Feigl, S. Kram, P. Woller, R. H. Siddiqui, M. Philippsen, and C. Mutschler. A bidirectional lstm for estimating dynamic human velocities from a single imu. In *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2019. Cited on page 30.

[43] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi, and D. Stricker. Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion. *Sensors (Switzerland)*, 17(6):1–40, 2017. Cited on pages 3, 17, and 26.

[44] T. Fischer and C. Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018. Cited on page 98.

[45] K. R. Ford, G. D. Myer, and T. E. Hewett. Valgus knee motion during landing in high school female and male basketball players. *Medicine and Science in Sports and Exercise*, 35(10):1745–1750, 2003. Cited on page 1.

[46] K. R. Ford, G. D. Myer, and T. E. Hewett. Reliability of landing 3D motion analysis: Implications for longitudinal analyses. *Medicine and Science in Sports and Exercise*, 39(11):2021–2028, 2007. Cited on page 1.

[47] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. Cited on pages 34 and 37.

[48] P. D. Groves. *Principles of GNSS , Inertial , and Multisensor Integrated Navigation Systems*. 2008. Cited on pages 3, 7, 8, 9, 10, 11, 13, and 14.

[49] P. D. Groves. Navigation using inertial sensors [tutorial]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):42–69, 2015. Cited on pages 15, 16, 23, and 30.

[50] P. Gui, L. Tang, and S. Mukhopadhyay. Mems based imu for tilting measurement: Comparison of complementary and kalman filter based data fusion. In *2015 IEEE 10th conference on Industrial Electronics and Applications (ICIEA)*, pages 2004–2009. IEEE, 2015. Cited on pages 17, 26, and 27.

[51] H. Guo, M. Uradzinski, H. Yin, and M. Yu. Indoor positioning based on foot-mounted imu. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 63(3):629–634, 2015. Cited on page 29.

[52] P. Guo, H. Qiu, Y. Yang, and Z. Ren. The soft iron and hard iron calibration method using extended kalman filter for attitude and heading reference system. *Record - IEEE PLANS, Position Location and Navigation Symposium*, (20070851011):1167–1174, 2008. Cited on pages 3 and 7.

[53] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. Cited on page 31.

[54] S. Hosseinyalamdary. Deep kalman filter: Simultaneous multi-sensor integration and modelling; a gnss/imu case study. *Sensors*, 18(5):1316, 2018. Cited on page 26.

[55] A. Karatsidis, G. Bellusci, H. M. Schepers, M. De Zee, M. S. Andersen, and P. H. Veltink. Estimation of ground reaction forces and moments during gait using only inertial motion capture. *Sensors*, 17(1):75, 2017. Cited on page 39.

[56] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Cited on pages 54, 71, and 73.

[57] M. Kok, J. D. Hol, and T. B. Schön. Using inertial sensors for position and orientation estimation. *Foundations and Trends in Signal Processing*, 11(1-2):1–153, 2017. Cited on pages 7 and 17.

[58] N. V. Lavrik and P. G. Datskos. Optically read Coriolis vibratory gyroscope based on a silicon tuning fork. *Microsystems and Nanoengineering*, 5(1), 2019. Cited on page 11.

[59] J. K. Lee and E. J. Park. A fast quaternion-based orientation optimizer via virtual rotation for human motion tracking. *IEEE Transactions on Biomedical Engineering*, 56(5):1574–1582, 2008. Cited on page 17.

[60] L. Li, Y. Pan, J.-K. Lee, C. Ren, Y. Liu, D. A. Grejner-Brzezinska, and C. K. Toth. Cart-mounted geolocation system for unexploded ordnance with adaptive zupt assistance. *IEEE Transactions on Instrumentation and Measurement*, 61(4):974–979, 2012. Cited on page 29.

[61] G. Ligorio and A. M. Sabatini. Dealing with magnetic disturbances in human motion capture: A survey of techniques. *Micromachines*, 7(3), 2016. Cited on pages 3 and 7.

[62] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016. Cited on page 97.

[63] I. H. Lopez-Nava and M. M. Angelica. Wearable Inertial Sensors for Human Motion Analysis: A review. *IEEE Sensors Journal*, PP(99):7821–7834, 2016. Cited on pages 1, 2, 3, and 24.

[64] M. Ma, Q. Song, Y. Gu, Y. Li, and Z. Zhou. An adaptive zero velocity detection algorithm based on multi-sensor fusion for a pedestrian navigation system. *Sensors*, 18(10):3261, 2018. Cited on page 29.

[65] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017. Cited on pages 35 and 37.

[66] S. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 25:113–118, 2010. Cited on pages 20, 23, and 27.

[67] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE international conference on rehabilitation robotics*, pages 1–7. IEEE, 2011. Cited on page 27.

[68] M. Mäkelä, J. Rantanen, M. Kirkko-Jaakkola, and L. Ruotsalainen. Context recognition in infrastructure-free pedestrian navigation—toward adaptive filtering algorithm. *IEEE Sensors Journal*, 18(17):7253–7264, 2018. Cited on page 29.

[69] C. Mandery, Ö. Terlemez, M. Do, N. Vahrenkamp, and T. Asfour. The kit whole-body human motion database. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 329–336. IEEE, 2015. Cited on pages 34 and 37.

[70] J. L. Maples-keller, B. E. Bunnell, S.-j. Kim, O. Barbara, and B. Sciences. The use of virtual reality technology in the treatment of anxiety and other psychiatric disorders. 25(3):103–113, 2017. Cited on page 1.

[71] P. Maurice, A. Malaisé, C. Amiot, N. Paris, G.-J. Richard, O. Rochel, and S. Ivaldi. Human movement and ergonomics: An industry-oriented dataset for collaborative robotics. *The International Journal of Robotics Research*, 38(14):1529–1537, 2019. Cited on pages 36, 37, 38, 39, and 45.

[72] S. G. McLean, S. W. Lipfert, and A. J. Van Den Bogert. Effect of gender and defensive opponent on the biomechanics of sidestep cutting. *Medicine and Science in Sports and Exercise*, 36(6):1008–1016, 2004. Cited on page 1.

[73] J. M. Moreira, A. C. P. L. F. de Carvalho, and T. Horváth. *A General Introduction to Data Analytics*. 2018. Cited on pages 50, 95, 96, and 97.

[74] P. A. Morreale. Wireless sensor network applications in urban telehealth. *Proceedings - 21st International Conference on Advanced Information Networking and Applications Workshops/Symposia, AINAW'07*, 1:810–814, 2007. Cited on page 2.

[75] L. Mündermann, S. Corazza, and T. P. Andriacchi. The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. *Journal of NeuroEngineering and Rehabilitation*, 3:1–11, 2006. Cited on page 1.

[76] A. Nielsen. *Practical Time Series*. 2019. Cited on pages 24, 26, and 97.

[77] M. Nunes, E. Gerding, F. McGroarty, and M. Niranjan. The memory advantage of long short-term memory networks for bond yield forecasting. *Available at SSRN 3415219*, 2019. Cited on pages 46, 67, and 99.

[78] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. Cited on page 30.

[79] L. Pan, A. Song, S. Duan, and X. Shi. Study on motion performance of robot-aided passive rehabilitation exercises using novel dynamic motion planning strategy. *International Journal of Advanced Robotic Systems*, 16:172988141987323, 09 2019. Cited on page 2.

[80] S. Y. Park, H. Ju, and C. G. Park. Stance phase detection of multiple actions for military drill using foot-mounted imu. *sensors*, 14:16, 2016. Cited on page 29.

[81] G. Pascoli. L'effet Sagnac et son interpretation by Paul Langevin. *Comptes Rendus Physique*, 18(9-10):563–569, 2017. Cited on page 10.

[82] J. Patterson and A. Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017. Cited on pages 68 and 99.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. Cited on page 47.

[84] T. Raicharoen, C. Lursinsap, and P. Sanguanbhokai. Application of critical support vector machine to time series prediction. In *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, volume 5, pages V–V. IEEE, 2003. Cited on page 96.

[85] J. Rantakokko, P. Strömbäck, and P. Andersson. Foot-and knee-mounted ins for firefighter localization. In *Proceedings of the 2014 International Technical Meeting of the Institute of Navigation, San Diego, CA, USA*, pages 27–29, 2014. Cited on page 16.

[86] D. A. Rodríguez-Silva, F. Gil-Castineira, F. J. Gonzalez-Castano, R. J. Duro, F. Lopez-Pena, and J. Vales-Alonso. Human motion tracking and gait analysis: a brief review of current sensing systems and integration with intelligent environments. In *2008 IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pages 166–171. IEEE, 2008. Cited on page 1.

[87] D. Roetenberg, H. Luinge, and P. Slycke. Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 1, 2009. Cited on page 39.

[88] A. M. Sabatini. Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation. *Sensors*, 11(10):9182–9206, 2011. Cited on page 26.

[89] H. Sak, A. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014. Cited on pages 97 and 98.

[90] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, 2018. Cited on page 23.

[91] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687. IEEE, 2018. Cited on pages 35 and 37.

[92] A. L. Schwab and J. P. Meijaard. How to draw euler angles and utilize euler parameters. In *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 259–265. American Society of Mechanical Engineers Digital Collection, 2006. Cited on page 19.

[93] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017. Cited on page 97.

[94] S. Semeniuta, A. Severyn, and E. Barth. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*, 2016. Cited on page 53.

[95] D. K. Shaeffer. MEMS inertial sensors: A tutorial overview. *IEEE Communications Magazine*, 51(4):100–109, 2013. Cited on page 11.

[96] J. P. Silva do Monte Lima, H. Uchiyama, and R.-i. Taniguchi. End-to-end learning framework for imu-based 6-dof odometry. *Sensors*, 19(17):3777, 2019. Cited on pages 23, 24, 31, 33, 51, and 54.

[97] D. Simon. Kalman filtering. *Embedded systems programming*, 14(6):72–79, 2001. Cited on page 24.

[98] I. Skog, P. Handel, J.-O. Nilsson, and J. Rantakokko. Zero-velocity detection—an algorithm evaluation. *IEEE transactions on biomedical engineering*, 57(11):2657–2666, 2010. Cited on page 29.

[99] A. Solin, S. Cortes, E. Rahtu, and J. Kannala. Inertial odometry on handheld smartphones. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1–5. IEEE, 2018. Cited on pages 24, 28, 29, and 33.

[100] A. Solin, S. Cortes, E. Rahtu, and J. Kannala. Pivo: Probabilistic inertial-visual odometry for occlusion-robust navigation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 616–625. IEEE, 2018. Cited on pages 23 and 28.

[101] M. W. Spong and M. Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008. Cited on page 18.

[102] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012. Cited on page 60.

[103] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999. Cited on page 96.

[104] J. A. Suykens and J. Vandewalle. Recurrent least squares support vector machines. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(7):1109–1114, 2000. Cited on page 96.

[105] D. H. Titterton and J. L. Weston. *Strapdown inertial navigation technology - 2nd edition - [Book review*, volume 20. 2004. Cited on pages 7, 8, 9, 10, 11, and 12.

[106] C. B. University of California. On the genera Dermacentor, Anocentor and M. 1960. *Fundamental studies of human locomotion and other information relating to design of artificial limbs : report to the Committee on Artificial Limbs*. Berkeley, Calif. : Calif. U.P, 1947. Microfilm. Cited on page 1.

[107] K. P. Valavanis. *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*, volume 33. Springer Science & Business Media, 2008. Cited on page 17.

[108] R. G. Valenti, I. Dryanovski, and J. Xiao. Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs. *Sensors (Switzerland)*, 15(8):19302–19330, 2015. Cited on page 20.

[109] B. Wagstaff and J. Kelly. Lstm-based zero-velocity detection for robust inertial navigation. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2018. Cited on pages 29, 51, and 54.

[110] L. Wang, Y. Hao, Z. Wei, and F. Wang. A calibration procedure and testing of MEMS inertial sensors for an FPGA-based GPS/INS system. *2010 IEEE International Conference on Mechatronics and Automation, ICMA 2010*, (1):1431–1436, 2010. Cited on pages 14 and 15.

[111] J. Watada and Z. B. Musaand. Tracking human motions for security system. In *2008 SICE Annual Conference*, pages 3344–3349. IEEE, 2008. Cited on page 1.

[112] W. Weber and E. F. W. Weber. *Mechanik der menschlichen Gehwerkzeuge*. Dieterich, 1836. Cited on page 1.

[113] G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995. Cited on page 24.

[114] J. M. Winters, Y. Wang, and J. M. Winters. Wearable Sensors and Telerehabilitation. *IEEE Engineering in Medicine and Biology Magazine*, 22(3):56–65, 2003. Cited on page 3.

[115] L. Xiaofang, M. Yuliang, X. Ling, C. Jiabin, and S. Chunlei. Applications of zero-velocity detector and kalman filter in zero velocity update for inertial navigation system. In *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, pages 1760–1763. IEEE, 2014. Cited on page 29.

[116] H. Xing, B. Hou, Z. Lin, and M. Guo. Modeling and compensation of random drift of MEMS gyroscopes based on least squares support vector machine optimized by chaotic particle swarm optimization. *Sensors (Switzerland)*, 17(10), 2017. Cited on page 97.

[117] H. Yan, S. Herath, and Y. Furukawa. Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods. *arXiv preprint arXiv:1905.12853*, 2019. Cited on page 31.

[118] H. Yan, Q. Shan, and Y. Furukawa. Ridi: Robust imu double integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 621–636, 2018. Cited on page 30.

[119] Y. Yao, X. Xu, and X. Xu. An imm-aided zupt methodology for an ins/dvl integrated navigation system. *Sensors*, 17(9):2030, 2017. Cited on page 29.

[120] P.-S. Yu, S.-T. Chen, and I.-F. Chang. Support vector regression for real-time flood stage forecasting. *Journal of Hydrology*, 328(3-4):704–716, 2006. Cited on pages 95 and 96.

[121] R. Zhang, H. Yang, F. Höflinger, and L. M. Reindl. Adaptive zero velocity update based on velocity classification for pedestrian tracking. *ieee Sensors journal*, 17(7):2137–2145, 2017. Cited on page 29.

[122] Z. Zhang and D. Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018. Cited on page 60.

[123] J. Zhao. A review of wearable imu (inertial-measurement-unit)-based pose estimation and drift reduction technologies. In *Journal of Physics: Conference Series*, volume 1087, page 042003. IOP Publishing, 2018. Cited on page 26.

[124] H. Zhou and H. Hu. Human motion tracking for rehabilitation-A survey. *Biomedical Signal Processing and Control*, 3(1):1–18, 2008. Cited on pages 1, 2, and 3.

[125] R. Zhu and Z. Zhou. A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(2):295–302, 2004. Cited on page 2.

Appendix A

Machine Learning

A.1 Models

A.1.1 Logistic Regression (Log-R)

A Logistic Regression (Log-R) can be considered a probabilistic classifier since it uses the attributes variables to create a probability of class-membership using a discriminative function. This classifier constructs a linear hyperplane to separate both classes and, therefore, can be identified as a linear classifier [5]. For a binary classification problem, this hyperplane is described in Figure A.1.

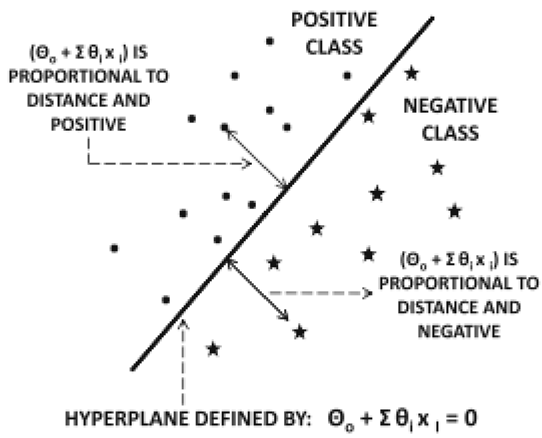


Figure A.1: Linear separator for a binary classification task, extracted from [5]

In a binary problem in which the classes are -1 and 1 the probabilities of belonging to each of the classes are modeled using a logistic function:

$$P(C = +1|\overline{X}) = \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}} \tag{A.1}$$

$$P(C = -1|\overline{X}) = \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}} \tag{A.2}$$

In which θ_i represents the coefficient associated with the i^{th} dimension of the data, and θ_0 the offset parameter. The calculation of those parameters is performed using a maximum likelihood methodology. The sum of both probabilities is equal to one. If a particular data point lies on top of the hyperplane, then the likelihood of belonging to each of of the classes is 0.5 [5].

A.1.2 Support Vector Machine (SVM)

Support Vector Machines (SVM) were initially projected to deal with binary classification problems. However, these can also be modified and applied to multiclass and regression problems. These can overcome some of the NN limitations like the lack of mathematical foundations and its low generalization ability. SVM, as initially designed, tries to identify the maximum margin hyperplane. This hyperplane acts as decision border guaranteeing the maximum separability between classes [73]. When the model is trained, the separation margin between classes is defined using examples from both classes that act as support vectors as described in Figure A.2.

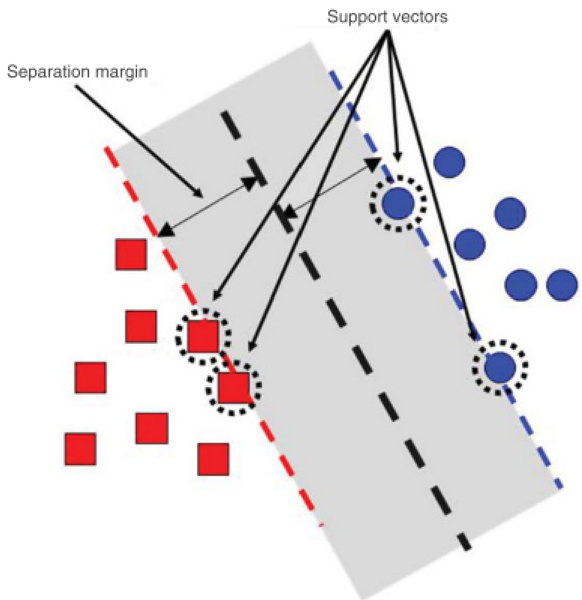


Figure A.2: Support Vector Machine separation margin for two classes (red and blue) with examples from both classes acting as support vectors, extracted from [73]

SVMs use the structural risk minimization as working principle. This way, SVMs avoid overfitting since the structural risk minimization looks for a compromise between the complexity of the model and the quality of the results. Therefore there is some error tolerance when training the model [120].

To separate classes that are not linearly separable is possible to use several kernels to transform data into higher dimensional feature spaces in which data is linearly separable. Some of the most used kernels are the Gaussian radial basis kernel, the polynomial kernel and the sigmoid kernel [73, 5]. One of the interesting properties of SVM is that its solution is globally optimal. The main disadvantages of this model are its computational cost and the high sensitivity of its hyper-parameters [3].

SVMs can also be used for regression problems with certain adjustments. Some interesting algorithms used for this purpose include Critical-SVM (CSVM) [84], Least Square-SVM (LS-SVM) [103], Recurrent Least Square-SVM [104] and Dynamic Least Square-SVM [41].

In the Support Vector Regression (SVR) case, the principle applied is the same. The goal is to create also a margin. But, instead of margin which separates two classes, the purpose is to include data points inside the margin. Again it is also possible to use other kernels if the linear one is not suitable, to transform the problem into a higher dimensional feature space [120].

A.1.3 Random Forest (RF)

RFs are a parallel ensemble of decision trees in which randomness in introduced in the building process of each decision tree. Firstly, the data used to train each decision tree is a bootstrap of the whole data. Additionally in each node only a set of the features is use to form the split rule. This building process is described in Figure A.3. This way, each decision tree of the random forest makes decisions using different samples and different features. After each decision tree generates its prediction, all the predictions are assembled to produce a final estimate. In a classification problem this decision is produced through major voting. Important to state that RFs can also be used in regression problems [5, 73].

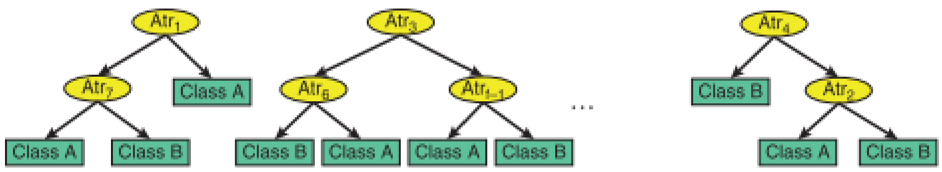


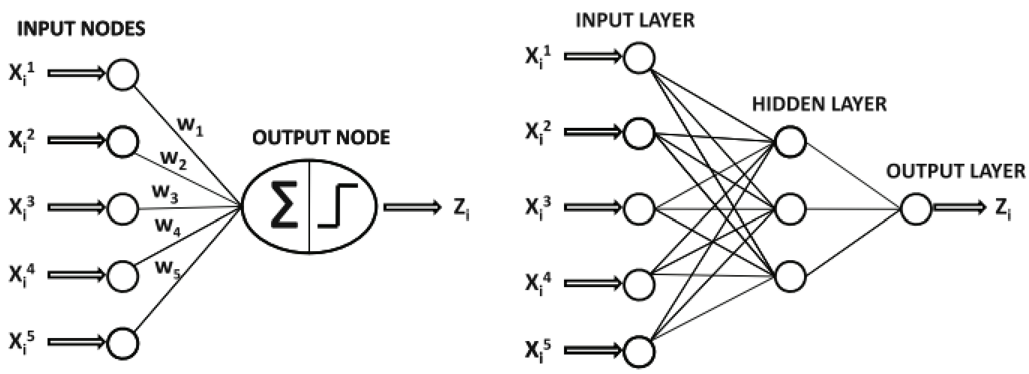
Figure A.3: Random Forest building process, extracted from [73]

The major advantages of this model are related with its good performance and interpretability (to a certain extent). The major drawbacks are related with its high computational cost (although it can be easily parallelized).

A.1.4 Neural Networks (NN)

Neural networks (NN) are a deep learning solution that tries to mimic the human nervous system. These can recognize patterns and similarities in data and try to learn from a serious of committed errors. NN simulate the nervous system using several cells, which can be regarded as neurons, and reproduce the synapses as connections between neurons with changeable weights. The learning phase consists in modifying those weights. This modification occurs accordingly to an objective function which tries to minimize the prediction errors by altering those weights. This way, NN can be regarded as optimization-based learning algorithms [73, 3, 5]. The simplest NN is the perceptron, which consists of only two layers: an input layer and an output layer. A single node constitutes the output layer. A multi-layer perceptron (MLP) is composed by an input layer, an output layer and one or more hidden layers. MLPs are the most commonly used options for forecasting [3]. Both are described in Figure A.4.

The connections between neurons are associated with a specific weight. Each neuron possesses an activation function which is applied to the sum of its inputs. These inputs result from



(a) Perceptron (b) Multilayer

Figure A.4: a) Perceptron – Simplest Neural Network architecture; b) Multi-Layer Perceptron, extracted from [5]

the multiplication of the output of each neuron of the previous layer and respective associated weight. The most commonly used activation functions are the rectified linear unit (ReLU), logistic sigmoid, hyperbolic tangent and gaussian, among others. MLP must use a non-linear activation function, or else the network would collapse in a single perceptron [73].

As referred previously, the training phase of a NN consists of updating the weights of the connecting links. This phase is performed using a learning algorithm. Backpropagation is an example of an algorithm used for this purpose. It applies gradient descent to determine if each weight must be increased or decreased by comparing the outputs with their actual values.

NN have been used to model and compensate the MEMS IMUs errors. In theory, NN can be used to model complex non-linear functions. Nevertheless, these are often associated with a low generalization capability since they are very prone to overfitting. Other of the main disadvantages associated with NN is the fact that they are related to a long training time, which can be an obstacle to the implementation of algorithms that operate in real-time based on NN [15, 116].

Using these solutions for time-series analysis there is no need to determine the model order or parameters as in the methods described in sections **??** and **??**, and there is no need to create a hypothesis about the dynamics of the system as in the Kalman filter method described in 3.1.1. Another important characteristic of deep learning solutions is that these do not require the data to be stationary [76].

For time-series analysis and modelling, a particular focus has been put recently on Recurrent Neural Networks (RNN). So in the next section, it is presented an insight on this architecture of NN.

A.1.4.1 Recurrent Neural Networks (RNN)

RNNs are one of the most exciting methods in what concerns time-series modelling. These have been widely used with success in fields like speech recognition, stock price forecasting or text classification [89, 93, 62].

An example of RNN is illustrated in Figure A.5. Analysing the figure, it is possible to conclude that, since the network possesses loops, the data persists and is retained from memory cell to memory cell. The main disadvantages of this simple RNN are that it has trouble dealing with long-range term dependencies and can experience vanishing or exploding gradient when trained with backpropagation through time algorithm. [89].

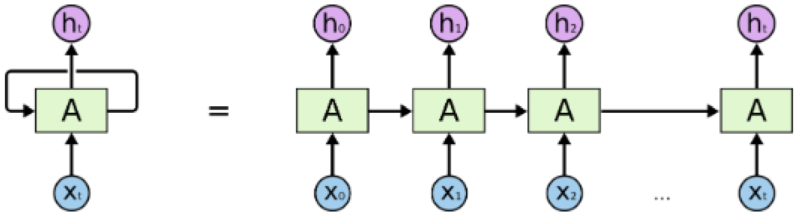


Figure A.5: Different representations of a Recurrent Neural Network, extracted from [1]

Long Short Term Memory (LSTM) Neural Networks

To overcome these limitations, Long Short Term Memory - RNN (LSTM-RNN) have emerged. LSTMs deal well with long term dependencies and with the vanishing or exploding gradient problem. LSTMs are composed of one input layer, one output layer, and one hidden layer which contains several memory cells that possess a recurrent hidden state whose activation depends on previous cycles. This hidden state is known as cell state and is responsible for passing information between modules. To adjust or maintain the information present in the cell state, LSTMs have multiple memory cells each with three gates: forget gate, input gate, and output gate [13].

A representation of a single memory cell is presented in Figure A.6. The forget gate is used to decide what information to maintain from the cell state. The input gate decides what new information is going to be inserted in the cell state, and the output gate decides what information from the cell state is going to be output.

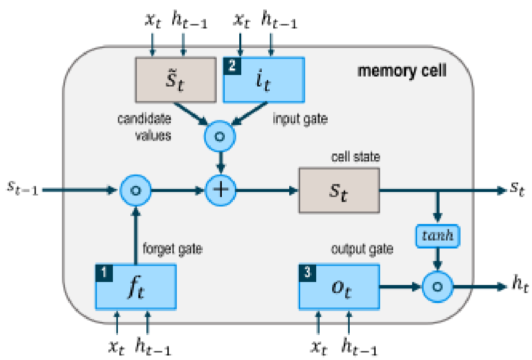


Figure A.6: Representation of a single memory cell from a LSTM, extracted from [44]

The number of cells or blocks from a LSTM layer should not be confused with the number of units. The number of units is representative of the dimension of the output and of the cell state as it is well represented in Figure A.7. In this image a timestep with 3 dimensions is introduced in a LSTM cell with 2 units.

[1]Understanding LSTMs. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/` Accessed on 27-01-2020

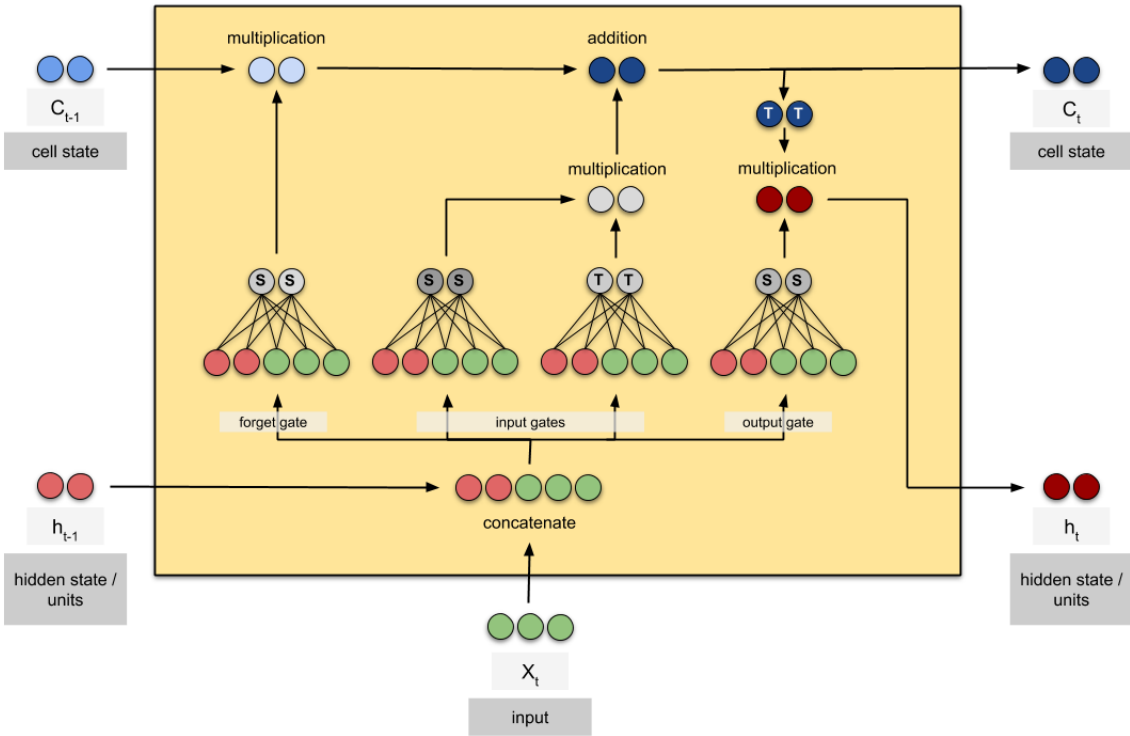


Figure A.7: Representation of a single memory cell from a LSTM with a visual representation of the number of units, extracted from [2]

When compared with simpler RNNs, LSTMs can deal better with sequential data since these can capture long-term dependencies, without suffering as much with problems as vanishing or exploding gradients [77]. Vanishing gradients appears when the gradient becomes too low making the model uncapable of learning long-time dependencies. Exploding gradients are the opposite: the gradients becomes too large increasing the loss function values. This could harm the previously learned features. LSTMs can deal well with vanishing gradient problems, but sometimes can be affected by exploding gradient problems. A possible solution is to perform gradient clipping or gradient renormalization to avoid large gradient values [82].

[2] Animated RNN, LSTM and GRU. https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45 Accessed on 05-06-2020

Appendix B

Results of the Zero-Velocity Detection using inertial data - One placement to all

Table B.1: Accuracy performance of the Logistic Regression Classifier trained with data from a single placement and evaluated in several placements (RSHO- Right Shoulder,LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Train \ Test	RSHO	LSHO	RUPA	LUPA	RTOE	LTOE	Mean
RSHO	0.8903	0.9031	0.7216	0.8056	0.8589	0.8913	0.8451
LSHO	0.8933	0.913	0.7412	0.8160	0.8581	0.8927	0.8524
RUPA	0.8018	0.8222	0.8154	0.8899	0.8899	0.9120	0.8552
LUPA	0.8328	0.8584	0.8247	0.8793	0.8885	0.9140	0.8663
RTOE	0.5576	0.5756	0.6096	0.6475	0.9367	0.9497	0.7128
LTOE	0.5548	0.5719	0.5985	0.6297	0.9243	0.9506	0.70450
Mean	0.7551	0.7740	0.7185	0.7780	0.8927	0.9184	0.8061

Table B.2: Accuracy performance of the SVM Classifier trained with data from a single placement and evaluated in several placements (RSHO- Right Shoulder,LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Train \ Test	RSHO	LSHO	RUPA	LUPA	RTOE	LTOE	Mean
RSHO	0.9485	0.95410	0.8998	0.9070	0.8814	0.9167	0.9179
LSHO	0.946	0.95347	0.9175	0.9235	0.8909	0.9214	0.9255
RUPA	0.9403	0.9502	0.9495	0.9485	0.9045	0.9282	0.9369
LUPA	0.9351	0.9466	0.9426	0.9451	0.9098	0.9296	0.9348
RTOE	0.5899	0.6018	0.6327	0.6645	0.9460	0.9562	0.7319
LTOE	0.6515	0.6691	0.686	0.7103	0.9397	0.9608	0.7696
Mean	0.8352	0.8459	0.8380	0.84982	0.91205	0.9355	0.8694

Table B.3: Accuracy performance of the Densely Connected Classifier trained during 10 epochs with data from a single placement and evaluated in several placements (RSHO- Right Shoulder,LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Train \ Test	RSHO	LSHO	RUPA	LUPA	RTOE	LTOE	Mean
RSHO	0.9533	0.9557	0.8675	0.8877	0.8844	0.9190	0.9113
LSHO	0.9492	0.9567	0.8661	0.9027	0.8827	0.9127	0.9117
RUPA	0.9410	0.9515	0.9331	0.9377	0.9052	0.9332	0.9336
LUPA	0.9415	0.9471	0.9088	0.9445	0.8934	0.9287	0.9273
RTOE	0.6591	0.6505	0.7468	0.7530	0.9422	0.9477	0.7832
LTOE	0.6670	0.6284	0.7014	0.7591	0.9364	0.9601	0.7754
Mean	0.8519	0.8483	0.8373	0.8641	0.9074	0.9336	0.8738

Table B.4: Accuracy performance of the Single Layer LSTM Classifier trained during 10 epochs with data from a single placement and evaluated in several placements (RSHO- Right Shoulder,LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Train \ Test	RSHO	LSHO	RUPA	LUPA	RTOE	LTOE	Mean
RSHO	0.9434	0.9461	0.8485	0.8487	0.8805	0.9065	0.8956
LSHO	0.9349	0.9454	0.8522	0.8819	0.8738	0.909	0.8995
RUPA	0.9423	0.9395	0.9192	0.9359	0.902	0.9408	0.9300
LUPA	0.9155	0.9286	0.9184	0.9431	0.9047	0.9311	0.9236
RTOE	0.6558	0.716	0.7199	0.7558	0.94	0.9495	0.7895
LTOE	0.6786	0.7038	0.7118	0.7424	0.9413	0.958	0.78932
Mean	0.8451	0.8632	0.8283	0.8513	0.9070	0.9325	0.8712

Table B.5: Accuracy performance of the 6-Layer LSTM Classifier trained during 10 epochs with data from a single placement and evaluated in several placements (RSHO- Right Shoulder,LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Train \ Test	RSHO	LSHO	RUPA	LUPA	RTOE	LTOE	Mean
RSHO	0.9418	0.9444	0.8384	0.8695	0.8764	0.9164	0.8978
LSHO	0.9429	0.9481	0.8337	0.8633	0.8755	0.9088	0.8954
RUPA	0.8392	0.8522	0.8656	0.8817	0.9103	0.9262	0.8872
LUPA	0.9024	0.9166	0.8797	0.9103	0.9023	0.9261	0.9062
RTOE	0.7552	0.7648	0.7547	0.7715	0.8926	0.9121	0.8085
LTOE	0.7077	0.7275	0.7541	0.7922	0.9258	0.9393	0.8078
Mean	0.8482	0.8589	0.8210	0.8481	0.8972	0.9215	0.8658

Table B.6: Accuracy performance of the Convolutional+LSTM Classifier trained during 10 epochs with data from a single placement and evaluated in several placements (RSHO- Right Shoulder,LSHO- Left Shoulder, RUPA- Right Upper Arm, LUPA- Left Upper Arm, RTOE- Right Toe, LTOE- Left Toe)

Train \ Test	RSHO	LSHO	RUPA	LUPA	RTOE	LTOE	Mean
RSHO	0.9375	0.9418	0.8218	0.8632	0.8625	0.9094	0.8894
LSHO	0.9372	0.9446	0.8412	0.867	0.88	0.9153	0.8976
RUPA	0.9188	0.9312	0.916	0.9213	0.8935	0.9271	0.9180
LUPA	0.9308	0.944	0.9147	0.9372	0.8934	0.93	0.9250
RTOE	0.5981	0.6137	0.6468	0.6833	0.9416	0.9509	0.7391
LTOE	0.6274	0.6529	0.6615	0.7252	0.9361	0.9588	0.7603
Mean	0.8250	0.8380	0.8003	0.8329	0.9012	0.9319	0.8548