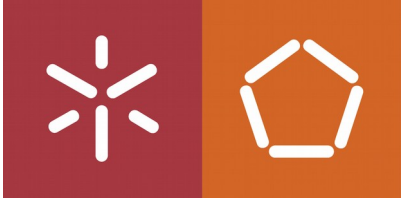
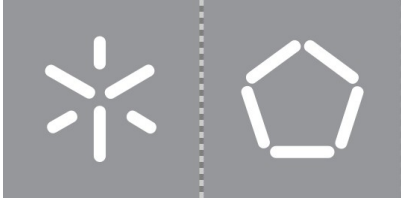


Universidade do Minho
Escola de Engenharia

João Miguel Santos Barbosa

Recognition of Gait Patterns in Human Motor Disorders Using a Machine Learning Approach

October 2019



Universidade do Minho
Escola de Engenharia

João Miguel Santos Barbosa

Recognition of Gait Patterns in Human Motor Disorders Using a Machine Learning Approach

Master Dissertation
Master Degree in Industrial Electronics and Computers Engineering

Dissertation supervised by
Professora Doutora Cristina P. Santos
Doutora Joana Figueiredo

December 2019

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor Cristina Santos, for her guidance and for giving me the opportunity to work on this project. I would also like to thank PhD. Joana Figueiredo for her endless guidance and support throughout this dissertation.

Thank you to everyone else in the laboratory for creating a very pleasant team-like work environment. I would like to thank Diogo Gonçalves for his knowledge and experience with Machine Learning and Ricardo Guimarães for showing me that good partners make all the difference in any challenge.

I would like to thank my fellow wonderful musicians and friends in our jazz group, André Pepe, José Esteves, Miguel Ferreira, Miguel Gonçalves, Sílvia Valente and our manager Rui Vieira, for their friendship and for creating a space were I could release stress and play quality music. The same goes to all members of the Fado group Gallus Gallus, thank you for creating a fun environment during each rehearsal.

Thank you to my friend Filipe Rocha for motivating and helping me through the first years of this degree. To João Costa for his medical knowledge and my fellow programmer, Carlos Daniel for our programming talks. Thank you to my slightly mad friend Ricardo Vilaça for persuading me to take breaks once in a while and to not take life too seriously.

Lastly, I would like to thank my girlfriend Inês Maia for her infinite love, patience and unconditional support through all our years together and my parents for always supporting me and never, ever giving up on me, this could not be possible without you.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

RESUMO

Com o avançar da idade, a ocorrência de distúrbios motores torna-se mais prevalente, conduzindo a patologias na marcha e aumentando o risco de quedas. Atualmente, muitos sistemas de monitorização de marcha extraem grandes quantidades de dados biomecânicos para apoio ao diagnóstico clínico, aumentando a quantidade de dados a ser processados em tempo útil. Para acelerar esse processo e proporcionar uma ferramenta objetiva de apoio sistemático ao diagnóstico clínico, métodos de Machine Learning são uma poderosa adição, processando grandes quantidades de dados e descobrindo relações não-lineares entre dados.

Esta dissertação tem o objetivo de desenvolver um sistema de reconhecimento de padrões de marcha com uma abordagem de Machine Learning para apoio ao diagnóstico clínico da marcha de vitimas de AVC. Isso inclui o desenvolvimento de uma ferramenta de estimação de dados biomecânicos e cálculo de features, a partir de sensores inerciais. Quatro redes neuronais foram implementadas numa ferramenta de classificação: uma rede Feed-Forward (FFNN), uma convolucinal (CNN), e duas redes recorrentes (LSTM e CLSTM). O desempenho de todos os modelos de classificação foi analisado. A métrica de desempenho usada é o coeficiente de correlação de Matthew. Os classificadores com melhor performance foram: Support Vector Machines (SVM), k-Nearest Neighbors (KNN), CNN, LSTM e CLSTM. Todos com uma performance igual a 1 no conjunto de teste. Apesar de os dois primeiros classificadores atingirem a mesma performance das redes neuronais, estas atingiram esta performance repetidamente e sem necessitar de métodos de redução de dimensionalidade.

Keywords: *Machine learning; patologias motoras humanas; reconhecimento de padrões de marcha; redução dimensional de dados*

ABSTRACT

With advanced age, the occurrence of motor disturbances becomes more prevalent and can lead to gait pathologies, increasing the risk of falls. Currently, there are many available gait monitoring systems that can aid in gait disorder diagnosis by extracting relevant data from a subject's gait. This increases the amount of data to be processed in working time. To accelerate this process and provide an objective tool for a systematic clinical diagnosis support, Machine Learning methods are a powerful addition capable of processing great amounts of data and uncover non-linear relationships in data.

The purpose of this dissertation is the development of a gait pattern recognition system based on a Machine Learning approach for the support of clinical diagnosis of post-stroke gait. This includes the development of a data estimation tool capable of computing several features from inertial sensors. Four different neural networks were be added to the classification tool: Feed-Forward (FFNN), convolutional (CNN) and two recurrent neural networks (LSTM and CLSTM). The performance of all classification models was analyzed and compared in order to select the most effective method of gait analysis. The performance metric used is Matthew's Correlation Coefficient. The classifiers that exhibit the best performance where Support Vector Machines (SVM), k-Nearest Neighbors (KNN), CNN, LSTM and CLSTM, with a Matthew's correlation coeficient of 1 in the test set. Despite the first two classifiers reaching the same performance of the three neural networks, the later reached this performance systematically and without the need of explicit dimensionality reduction methods.

Keywords: *Dimensional data reduction; gait pattern recognition; Human motor disorders; machine learning*

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ANOVA Analysis Of Variance. 24, 62–63, 77, 83–86

AHRS Attitude and Heading Reference System. 39, 41

ALS Amyotrophic Lateral Sclerosis. 15

AUC Area Under Curve. 65

BirdLab Biomedical Robotic Devices Lab. 1, 22, 26–27

CMEMS Center for MicroElectroMechanical Systems. 1

CAD Gait Cadence. 30, 33

CV Cross Validation. 24, 64, 75–76, 84, 89

CNN Convolutional Neural Network. 4, 19, 24, 61, 68, 71, 75, 80, 82–84, 86–87, 89

C-LSTM Convolutional Long-Short Term Memory Network. 4, 23–24, 61, 74, 76, 84, 86–87, 89–90

DNN Deep Neural Networks. 24, 66, 71

DAG Directed Acyclic Graph. 75

DA Discriminant Analisys. 4, 24, 63, 84–85, 87

DL Deep Learning. 2, 4, 20, 23, 51, 59, 71, 75, 77, 81, 83, 87, 89–90

DT Decision Trees. 18

DWT Discrete Wavelet Transform. 16–17

EMG Electromyography. 6, 11–13, 18

FSR Force Sensing Resistor. 6, 11–13

FoG Freezing of Gait. 10, 17

FFT Fast Fourier Transform. 13, 16

FSM Finite State Machine. 22, 28–29, 31, 34, 39, 41, 54, 90

FFNN Feed-Forward Neural Network. 4, 24, 61, 66, 68, 70, 72, 79, 84–87

FF Foot-Flat. 30, 44

FP False Positive. 65

FN False Negative. 65

GRF Ground Reaction Force. 8–9, 11, 13, 15, 18

GA Genetic Algorithm. 24, 62

HO Heel-Off. 30

HS Heel-Strike. 29–31 33–34, 52, 54,

HMM Hidden Markov Models. 19–20

ISIR Institute of Scientific and Industrial Research. 26

IMU Inertial Measurement Unit. 6, 11, 13, 20, 22, 35, 39, 42–43, 53, 55, 59

kNN k-Nearest-Neighbor. 4, 17–18, 24, 61, 63, 78–79, 84–85, 87, 89, 90

kPCA kernel Principal Component Analysis. 16

LSTM Long-Short Term Memory Neural Network. 4, 23–24, 71, 72–76, 82–84, 86–87, 89–90

LOO Leave-One-Out. 64

ML Machine Learning. 2–5, 17–18, 20, 24, 77, 80, 87, 89–90

MTC Minimum Toe Clearance. 8

MMSW Mid-Swing. 29–31, 33–34, 52–53

MMST Mid-Stance. 30–31

MARG Magnetic, Angular Rate, and Gravity. 35

mRMR Minimum Redundancy Maximum Relevance. 24, 62, 77, 84–87

MCC Mathew's Correlation Coeficient. 4, 65, 76, 83–87, 89–90

MAREA Movement Analysis in Real-world Environments using Accelerometers. 26

MIEEIC Integrated Masters in Industrial Electronics and Computer Engineering. 1

NN Neural Network. 2, 4, 16–18, 20, 61, 66, 80, 89

NB Naive Bayes. 18

NLP Natural Language Processing. 74

PCA Principal Component Analysis. 16–17, 19, 24, 62, 77, 84–87

RoM Range of Motion. 8, 13

RF Random Forests. 4, 18, 20, 61, 84–87

RBF Radial Basis Function. 17, 20

RNN Recurrent Neural Network. 19, 72

RL Reinforcement Learning. 90

ReLU Rectified Linear Activation Unit. 70, 74, 81

RMSE Root Mean Square Error. 49–50, 112

SCU System Control Unit. 7

SVM Support Vector Machine. 4, 15–18, 20, 24, 61, 63, 77–78, 84–85, 87, 89–90

TASMC Tel Aviv Sourasky Medical Centre. 10

TO Toe-Off. 29–31, 33–34, 53

TP True Positive. 65

TN True Negative. 65

UM University of Minho. 1

ZUPT Zero-Velocity Update. 40, 44, 46

1

INTRODUCTION

This dissertation was developed as part of the Integrated Masters in Industrial Electronics and Computer Engineering (MIEEIC) in Biomedical Robotic Devices Lab (BirdLab) at the Center for MicroElectroMechanical Systems (CMEMS), a research center from University of Minho (UM).

This work is related to the field of human gait recognition and its goal is the development of machine learning models capable of distinguishing between healthy and pathological gait. In this chapter the current work performed and existing challenges are addressed as well as the goals of this dissertation and the document structure.

1.1 Motivation

Human gait is a very complex human physical activity that involves major parts of the nervous, musculoskeletal and cardiorespiratory systems. The gait patterns of a subject are heavily influenced by several different factors such as age and motor condition.

As the field of medicine continues to develop the life expectancy of the population will also continue to grow. This increase in life expectancy will lead to a greater occurrence of several possible gait disorders at an advanced age. The prevalence of gait disorders increases from 10 % in people aged 60–69 years to more than 60 % in community dwelling subjects aged over 80 years [1].

Moreover, gait disorders are commonly observed after a stroke event. It affects the nervous system which in turn can lead to a degradation of the patient's gait and a decrease in quality of life. According to Witko [2], stroke is the leading cause of disability in adults in the Western world. Approximately 80% of post-stroke subjects experience walking difficulties 3 months after onset and about 70% of community-dwelling post-stroke individuals fall during the the first year [3]. Post-stroke subjects experience a decrease in spatiotemporal gait performance such as a lower walking speed, stride length and cadence. Kinematic

gait characteristics also become degraded: decreased hip flexion at initial contact, increased hip flexion at toe off, and decreased hip flexion during mid swing, more knee flexion at initial contact and less knee flexion at toe off and mid swing, more ankle plantarflexion at initial contact and mid swing and less ankle plantarflexion at toe off [4]. These impaired gait functions also affect the balance of the subject, which in turn increases the risk of falling, as previously mentioned.

Gait pattern analysis enables the clinical diagnosis of possible motor disorders. This can be performed by a trained health technician, although there is some degree of subjectivity in this method [5]. With the use of gait monitoring systems it is possible to obtain a more quantitative evaluation of a subject's condition. Recent advances in sensor technology have lead to the development of more powerful, efficient and smaller sensors which enables a greater precision in this analysis [6]. The development of gait monitoring systems has significantly increased the amount of bio-mechanical data that can be extracted from a patient's gait. The analysis of this data is usually performed by a doctor or therapist. However, with this increase in acquired data the time required to perform this analysis will also increase significantly as well as the probability of misdiagnosis [7]. The application of Machine Learning (ML) in this field can lead to an automatic, time-effective analysis of gait disorders aiming at a more effective diagnostic and personalized treatment. With a ML system, there will be a quantitative analysis of data without the subjective judgment of a clinician. With the ability to process great amounts of data, the time necessary for data analysis will shorten significantly [8]. Additionally, these systems are able to process complex, non-linear data with a high dimensionality [9] and thus uncover patterns in a patient's gait that are not easily detected by human visual inspection [10].

It is therefore of great value to explore and compare the performance of several ML and Deep Learning (DL) methods and unify them in a versatile and robust tool to support clinical-based diagnosis. Moreover, the employment of DL can greatly simplify the classification process due to the ability of Neural Networks (NNs) to work with different types of data and implicitly give more importance to the features that most affect its performance.

1.2 Problem Statement

The use of non-wearable gait monitoring systems is more prevalent due in part to their accuracy. However, one of the downsides of these systems are their limiting testing conditions, unlike wearable sensor systems, they are not able to perform in field testing (under real usage condition and not in a controlled laboratorial environment) [5]. Since the performance of ML systems is very dependent on the quality of

available data [11], there is a clear trade-off between the ability to perform tests under more realistic test conditions and the quality of the acquired data. It would be therefore beneficial to possess a ML based gait pattern recognition system that is robust enough to be able to handle less accurate data without any significant loss in performance. To further improve classification performance, dimensionality reduction methods are necessary to better select the most relevant data and organize it in terms of variance. It is also advantageous to explore a quantitative relationship between pathological and healthy gait patterns in order to better select the most appropriate features.

From the research carried out, not many studies focus the usage of these system specifically to post-stroke gait, which is one of the focus of this dissertation. Additionally, according to the research performed for this dissertation, most of the classification models chosen deal with non-sequential data (each training example is associated with a classification label). Very few studies explore the possibilities of sequential models or models that deal with image data when applied to human gait data obtained from inertial sensors.

1.3 Goals and Research Questions

The goal of this dissertation is the design, development and validation of a machine learning-based gait pattern recognition system to identify and distinguish pathological gait patterns in stroke survivors with motor disorders relying only on bio-mechanical data from wearable sensor systems. This system is composed of two different components: a bio-mechanical data estimation and feature determination tool and a gait pattern classification tool. This system aims to provide an automatic, time-effective, and objective analysis of gait disorders to support the clinical-based decision at a more effective diagnostic and personalized treatment. To accomplish this, four main objectives were defined:

- **Goal 1:** To identify relevant bio-mechanical features that characterize the motor disorders and abnormal gait patterns in post-stroke.
- **Goal 2:** To create a database of healthy and post-stroke bio-mechanical gait data and to implement a quaternion-based orientation filter tool for better estimation of segment orientation, position and joint angle.
- **Goal 3:** To develop a tool capable of computing relevant bio-mechanical features from raw gait data and store them in the same database for later use.

- **Goal 4:** To implement four NNs (Feed-Forward Neural Network (FFNN), Convolutional Neural Network (CNN), Long-Short Term Memory Neural Network (LSTM), Convolutional Long-Short Term Memory Network (C-LSTM)) in an available ML tool (Support Vector Machine (SVM), k-Nearest-Neighbor (kNN), Discriminant Analisys (DA), Random Forests (RF)) to distinguish between healthy and post-stroke gait.
- **Goal 5:** To identify an effective and accurate gait pattern recognition system through bench-marking evaluation of the used ML classifiers.

Among these goals, this dissertation will try to answer for the following research questions:

- **RQ 1:** Can the use of a quaternion-based orientation filter provide a better alternative to the estimation of sensor location/orientation?
- **RQ 2:** Is there an advantage in using DL methods in gait recognition when compared to the standard ML plus dimensionality reduction methods?

1.4 Main Contributions

With the work done during this dissertation, four main contributions can be mentioned:

- Development of a robust and accurate biomechanical data estimation tool based on a quaternion orientation filter for estimation of sensor orientation/position and joint angles
- Tool for feature determination from sensor data of healthy and stroke subjects
- Enhanced classification tool with four DL models, which proved to be superior with most of them reaching Mathew's Correlation Coeficient (MCC) scores very close to 1 and providing a significantly easier tuning process.

1.5 Dissertation Structure

This dissertation proposes an approach to gait disorder diagnosis based on a ML tool for gait recognition with several possible classification models.

Chapter 2 contains the current State of the Art related to the sensor systems used as a clinical diagnosis tool. Chapter 3 explains the overall system developed in this dissertation. Chapter 4 explains the process

by which the system extracts significant bio-mechanical gait data from raw sensor data. Chapter 5 presents the features chosen as input to the ML tool. Chapter 6 details the development of each classifier and the following results. Chapter 7 presents the final conclusions and possible future work.

2

STATE OF THE ART

In this chapter, the existing gait monitoring systems and gait recognition methods in literature will be examined. The presented literature analysis reviews the sensor systems, the used features, the dimensionality reduction methods, and the most used classification models. This research was focused mainly on gait pattern recognition systems. This chapter ends with a critical analysis of these reviewed topics.

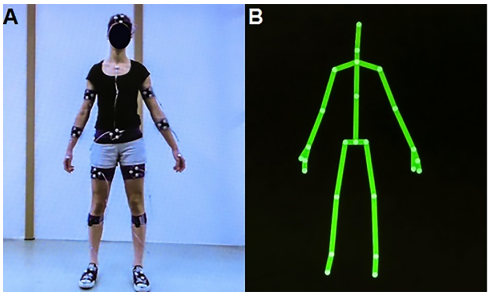
2.1 Sensor Systems

For any sensor system, sensor quality makes a considerable difference in the resulting acquired data accuracy. In the context of gait monitoring systems, these sensors can either be wearable or non-wearable. The non-wearable sensor systems addressed in this research are optical motion tracking systems and force plates. In terms of wearable sensors, inertial sensors, Force Sensing Resistors (FSRs) and Electromyographys (EMGs) sensors will be discussed. Among inertial sensors, accelerometers, gyroscopes and Inertial Measurement Units (IMUs) will be included.

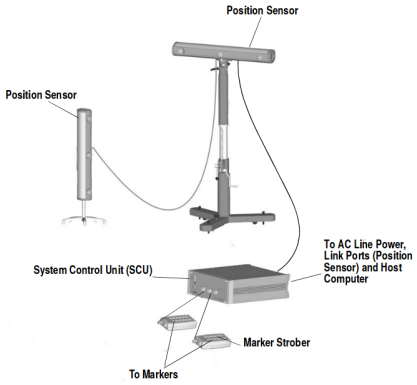
2.1.1 Optical Motion Tracking Systems

Optical motion tracking systems are one of the most utilized non-wearable sensor systems in literature [12, 13, 14, 15, 16, 17]. It consists of a set of markers placed in specific areas of a subject's body. The movement of these markers is then captured by a set of cameras that process the image and extract the markers' position thus tracking it in relation to time, enabling the monitoring of spatiotemporal and kinematic gait data. These are commercial systems and operate in controlled environments, an example is shown in Figure 1a. Although motion tracking systems are among the most used technology in the field of

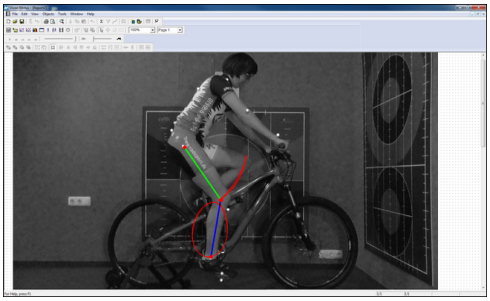
gait recognition, there are some disadvantages to its usage. Besides their complexity, these systems have a time-consuming operation [18] and function on a controlled environment which will limit the freedom of motion of the user. Several different products exist in the market such as Optotrak and Vicon/Peak Motus. These two systems were among the most common in literature so they will be discussed in this section.



(a) Optotrak motion tracking system [19].



(b) Optotrak overview [20].



(c) Vicon Motus [21].

Figure 1: Optical motion tracking system examples

2.1.1.1 Optotrack

This system consists of a set of markers with infrared light emitting diodes activated by a strober. The markers are tracked by optical position sensors and this information is collected and processed in a System Control Unit (SCU).

This system was used in Wu et al.[17] to acquire 3D gait data from tests realized with 24 young and 24 elderly patients with no previous injuries or abnormalities. Tests consisted of normal walking trials. Markers were attached to the right-hand side of each participant on the following anatomical landmarks: acromion of the scapula, greater trochanter, lateral femoral epicondyle, head of fibula, lateral malleolus, the posterior end of the lateral border of the calcaneus, the fifth metatarsal head of the right foot.

36 spatio-temporal and kinematic features per subject were determined: Stride length, stride duration, gait velocity, single support duration, stance duration, swing duration, gait cadence, hip, knee and ankle

angles (flexion/extension) at each gait event, angular Range of Motion (RoM) during stance and swing phases. These features were temporally normalized to each gait cycle.

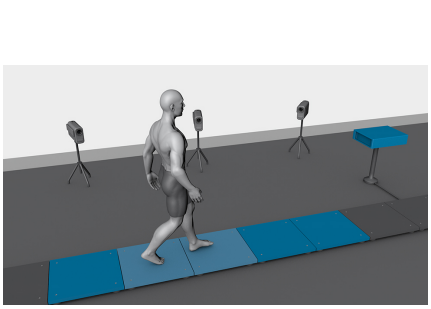
2.1.1.2 Vicon/Peak Motus

Commonly used in literature, this system can extract 2D or 3D data and uses reflective or black markers that contrast with the background in order to track each point through space. Despite taking longer to process the data, it can also be used without markers at all [22].

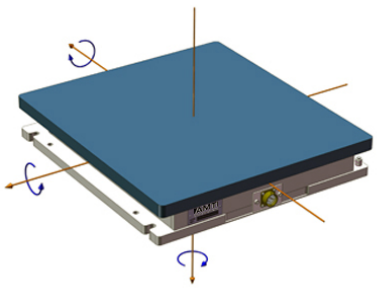
Toe clearance data was collected during walking trials on a treadmill using the PEAK MOTUS 2D motion tracking system in Lai et al. [15]. Two reflective markers were attached to each subject's left shoe at the 5th metatarsal head and the great toe. Minimum Toe Clearance (MTC) was calculated by subtracting ground reference (treadmill surface) from the minimum vertical coordinate of the virtual point during the swing phase. Bio-mechanical data was acquired from 23 subjects where 13 of them were healthy elderly with no fall history and 10 were elderly who had suffered more than one fall in the past year. The data acquired in this study was also used in Khandoker et al. [23].

2.1.2 Force Plates

Force plates are, namely Kistler and AMTI platforms, a non-wearable sensor system that measures the ground reaction forces generated by a body standing on or moving across them. These platforms are also sometimes used together with motion tracking systems [12, 9], enabling an acquisition of a greater volume of data by adding the force plate's Ground Reaction Force (GRF) to the spatiotemporal and kinematic bio-mechanical data of the motion tracking system. Other applications include the use of force plates in instrumented treadmills in order to reduce required test space.



(a) Kistler force platform [24].



(b) AMTI force plate [25].

Figure 2: Force plate examples

2.1.2.1 Kistler

Kistler force platforms are based on piezoelectric measurement technology that ensures that forces and moments are registered accurately in a variety of applications. These are very precise systems able to detect small changes in the gait pattern or shifts in the center of gravity [24].

In Su and Wu [26], GRF measured by two Kistler force platforms were used for gait-pattern recognition and used as classifier input data. These signals were recorded directly to a computer and after being post-processed, were used as input data to the classifier.

This system was also utilized in Lai et al. [9] together with a motion tracking system. GRFs were acquired through a Kistler force platform embedded in a 10 meter walkway centrally positioned to capture GRF data.

2.1.2.2 AMTI

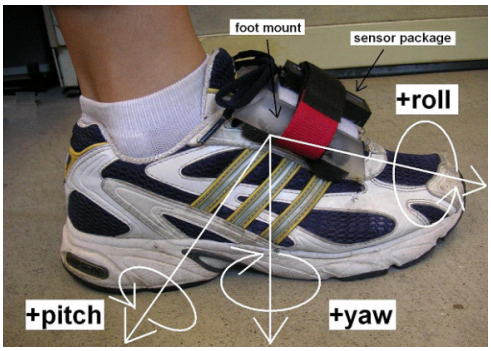
The force platforms developed by AMTI can have a sensing technology either based on Hall Effect or strain gauge. With Hall Effect technology several Hall Effect sensors/magnets are arranged inside the force platform. This type of force platform is less accurate but has got a simpler design, is less expensive and is also lightweight and portable. Strain gauge based force platforms have their working surface supported by thin-walled cylindrical sensing elements. Each element is instrumented with strain gauges excited by a constant voltage. This type of force platform has a better accuracy than its counterpart [25].

In Begg and Kamruzzaman [12] a motion tracking system is used along with two of these force platforms. This system was used to record GRFs in the vertical and anterior–posterior directions.

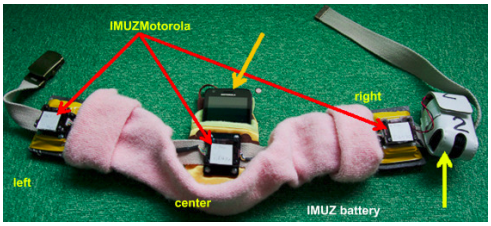
2.1.3 Inertial sensors

With the growing development in the field of microelectromechanical sensor technology, wearable inertial sensors have been increasingly more present in literature. An advantage of these sensors lies in their small size, portability and low-price. Despite not being as accurate as most optical motion tracking systems, these sensors make up for that fact due to their ability to enable in field gait monitoring [27].

Depending on its type, inertial sensors are capable of measuring acceleration, orientation and other bio-mechanical data. There are several types of sensors but in this section two of the most common in literature will be addressed: accelerometers and gyroscopes. Figure 3 shows some examples of systems that make use of inertial sensors.



(a) Accelerometer and gyroscope system [28].



(b) Inertial measurement unit [29].

Figure 3: Inertial sensors

2.1.3.1 Accelerometer

An accelerometer is an inertial sensor that measures acceleration forces. These forces can be static like the pull of gravity, or dynamic caused by movement or vibration. In Oung et al. [30], a public available dataset from the Laboratory for Gait and Neurodynamics, Department of Neurology, Tel Aviv Sourasky Medical Centre (TASMC) was created with data collected from 10 patient's with Parksinson's disease that encounter regular Freezing of Gait (FoG) during their daily activities. The data was acquired using triaxial accelerometers attached to the thigh, shank and lower back of each patient. This same data set was also used in El-Attar et al. [31].

In Ma et al. [32] foot-mounted triaxial accelerometers are used to monitor the gait patterns of 9 patients with glaucoma and 10 healthy controls under three different test procedures. The raw data was pre-processed to reduce its complexity and obtain a simplified signal with clear cyclic patterns ready for segmentation. This was achieved by calculating the signal vector magnitude which reflects the overall intensity of the movement during gait, reduces noise and highlights the cyclic nature of each signal.

2.1.3.2 Gyroscope

Gyroscopes are sensors that measure rotational motion by using the Earth's gravitational field. From what was found in literature, these sensors are always used in combination with other sensors such as accelerometers. In Barth et al. [33] triaxial gyroscopes and accelerometer were attached to the lateral heel of each shoe to monitor the gait of 18 patiens with Parksinson's disease and 17 healthy controls for 10 meter walking tests.

In Lee et al. [34] the gait of 20 hemiplegic patients and 20 healthy controls was studied by using a triaxial accelerometer and gyroscope located at the body's centre of gravity in the lumbar region during 20 meter gait trials.

2.1.3.3 Inertial Measurement Unit

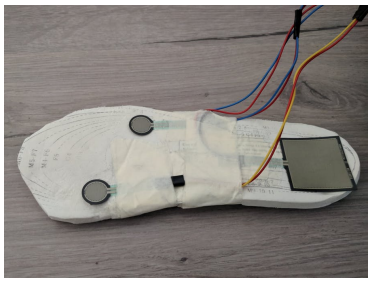
An IMU is able to measure a body's linear acceleration (accelerometer), rotational rate (gyroscope) and also its surrounding magnetic field as a heading reference (magnetometer).

In Mannini et al. [35] three IMUs are used together with GAITRite, an instrumented gait pressure mat to acquire data from the gait of 15 post stroke patients, 17 patients with Huntington's disease and 10 healthy elderly subjects. One IMU was placed on each shank and the third was located over the subject's lumbar spine. The GRF data was used to calculate the foot-strike and toe-off events in order to synchronize with the IMU data.

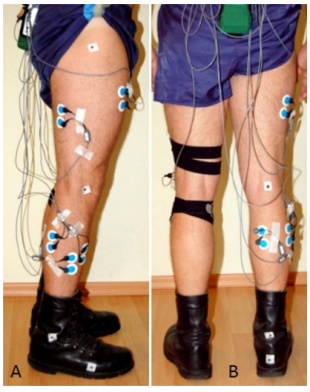
In Caramia et al. [7], an IMU-based motion capture system was used to acquire gait data from 25 patients with different levels of Parkinson's disease and 25 healthy controls. This system consisted of 8 IMUs. One IMU was placed on each foot dorsum, one on each shank, one on each thigh, one on the chest and one in the back side on the lumbar zone through elastic belts. From this data the angles between body segments are calculated, with the lumbar sensor as the body reference system.

2.1.4 Other Sensors

While not as common, some other types of sensors were also found in research. Figure 4 shows two of the most prevalent examples.



(a) FSR sensors [36].



(b) EMG Sensors [37].

Figure 4: Other types of sensors

2.1.4.1 Force Sensitive Resistors

FSRs are piezoresistive sensors, which implies that their electrical resistance varies in relation to the force applied under its surface. They are not commonly used in gait recognition systems due to their low precision.

Daliri [38] used a system of FSRs is used beneath the subject's foot to measure the pressure applied by each foot. Several time series data are created from these measurements for the left and right foot.

2.1.4.2 Electromyography Sensors

EMG measures the electrical activity of muscles and can be used to detect/monitor muscle weakness. Used in many clinical and biomedical applications, EMG sensors can be used as a diagnostics tool and also as a control signal for prosthetic devices.

In Ghasemzadeh et al. [39] an EMG system with two EMG sensors on each shank together with an accelerometer located on the subject's lumbar region is used to monitor the gait of 5 healthy subjects. Electromyographic data from the EMG sensors is used to monitor muscular activity duting gait.

In Nair et al. [40] six EMG electrodes were placed over six muscular groups of each leg of each subject to study the gait of 8 rheumatoid arthritis patients, 10 osteoarthritis patients and an unspecified number of healthy control subjects. EMG sensors recorded electromyographic data of six muscular groups of each leg of the rheumatoid arthritis patients: soleus, gactrocnemius medialis, peroneus brevis, tibialis anterior, vastus lateralis, and biceps femoris. For the osteoarthritis patients: soleus, gactrocnemius medialis, gluteus medialis, tibialis anterior, vastus lateralis, and biceps femoris. Data from seven muscles of each leg were recorded with the control subjects: soleus, gactrocnemius medialis, peroneus brevis, tibialis anterior, vastus lateralis, biceps femoris, and gluteus medialis.

2.2 Feature Determination

The types of features most commonly used in literature can be classified as either time-domain or frequency-domain features. In the context of ML, more specifically of gait recognition, a feature can be seen as a representative characteristic of a certain gait signal, such as the foot's angular velocity in the sagital plane. The determination of relevant features is a very important step to improve the classification performance and computational load of the classification model to be developed [41]. Part of the raw

bio-mechanical gait data is not relevant to the classifier and this can negatively affect its performance, therefore a good determination of the features that will be part of the initial feature space is mandatory.

2.2.1 Time Domain Features

Time-domain features are directly related to physical behaviors. In this context, they can represent spatiotemporal, kinematic, kinetic (inertial sensors and FSRs for temporal features) and electromyographical (EMGs) variables.

In Caramia et al. [7], 87 time-domain features were defined from data acquired from the gait of 25 patiens with Parkinson's disease and 25 healthy controls. Two different categories of features exist: kinematic and spatio-temporal. All kinematic features were RoM features which are defined as the difference between the maximum and minimum angle in the sagittal plane between two adjacent articular segments within one gait cycle. Features in this category include ankle, knee, hip and chest. Spatio-temporal features include step length, step time, stride time and stride speed.

In Asuroglu et al. [42], 16 time-domain features were defined from 8 GRF sensors: mean, median, minimum value and its index, maximum value and its index, range, root mean square, interquartile range, mean absolute deviation, skewness, kurtosis, entropy, energy, power, and harmonic mean. This data was available on the online gait database PhysioNet [43]. This database contains data from the gait of 93 Parkinson's disease patients and 73 control subjects.

2.2.2 Frequency Domain Features

By determining the frequency spectrum of a given signal or process it is possible to obtain a better understanding of its behavior by being able to extract data that is not easily accessible in the time-domain. Because of this it is common to use transforms such as the Fast Fourier Transform (FFT) to obtain frequency-domain features. These categories of features are usually employed together with time-domain features.

As stated earlier in Asuroglu et al. [42], 7 frequency-domain features were defined. A FFT algorithm was applied in order to extract these frequency-domain features, which were: mean, maximum value, minimum value, normalized value, energy, phase, and power.

In Mannini et al. [35], 90 total features were defined from data obtained from several IMUs and an instrumented gait pressure mat. This data was collected from 15 post-stroke patients, 17 Huntington's

disease patients and 10 healthy elderly subjects. From this feature set 42 were frequency-domain features. There were six types of features: power at first dominant frequency ($P1$), power at second dominant frequency, first dominant frequency, second dominant frequency, total power (PT) and $P1/PT$. These six features were computed for seven data segments thus giving a total of 42 frequency-domain features.

2.3 Dimensionality Reduction

In the context of ML, the feature sets or feature spaces are usually very big. With such feature spaces it is difficult to visualize and work on the training set. Besides the sheer size of the data there is also the occurrence of irrelevant features or features that are highly correlated with one another (redundant). This will affect negatively the performance of the classifier so, to mitigate these difficulties, there a vast array of dimensionality reduction methods that can be used.

Dimensionality reduction is the process by which the number of features or the dimension of the feature space is reduced. By doing this only the most relevant and less correlated features remain in the feature set which to some extent implies that some information will be lost. These remaining features are called principal components. Dimensionality reduction methods are separated into feature selection or feature extraction methods [41].

2.3.1 Feature Selection

Feature selection methods select the best features in the feature space. Some methods merely organize the features in the feature space by order of relevance while other methods actually create a different set containing the best features. Either way these methods do not alter in any way the feature space of the chosen features so they can still be interpreted in the same way as before.

2.3.1.1 Hill-Climbing

One of the most commonly found feature selection methods in literature is the Hill-Climbing algorithm. This algorithm is a heuristic search algorithm that finds a solution by searching for a local optimum for its cost function.

In Begg et al. [13] A hill-climbing feature selection algorithm was used along with a SVM classifier with 24 features. With hill-climbing feature selection, a small subset of only 3 to 5 gait features improved accuracy from 83.3% up to 90%.

In Lai et al. [9] it was found that GRF features resulted in a higher accuracy of the SVM classifier of 85.15% compared to 74.07% using only kinematic features. With a hill-climbing feature selection algorithm a subset of six features (two from GRF and four from foot kinematic features), improved accuracy up to 88.89%.

2.3.1.2 Genetic Algorithm

Genetic algorithms simulate the process of natural selection which means that those species (or possible solutions) that can adapt to changes in their environment are able to survive and reproduce.

In Daliri [38] a genetic algorithm was used to determine the most relevant features to improve the accuracy in the recognition of three different pathologies: Parkinson's disease, Amyotrophic Lateral Sclerosis (ALS) and Huntington's disease. In this study there were 24 total features. The performance of the classifier without feature selection was not specified. Classification results for Parkinson's disease were of 89.33% with three selected features. For ALS, classification accuracy was of 96.79% with three selected features. For the final test, which consisted in distinguishing between all three diseases, classification accuracy was of 90.63% with five selected features.

In Su and Wu [26] a genetic algorithm was combined with a neural network for the recognition of gait patterns. The accuracy of the neural network without the use of feature selection was of 89.7%. With the genetic algorithm classification accuracy was raised to 98.7%.

2.3.2 Feature Extraction

Much like feature selection methods, feature extraction methods provide a smaller feature space with the most relevant and less correlated features. However, unlike feature selection, these methods alter/transform the feature space. This means that completely new features are generated which can't be interpreted in the same way as the previous ones.

2.3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is highly used in literature and consists on an orthogonal transformation that converts a set of features, in which some of them are possibly correlated, into a set of linearly uncorrelated features. These features are called principal components and are organized in terms of their variance.

There is an extension to this method called kernel Principal Component Analysis (kPCA) which is a non-linear feature extraction method that makes use of kernel functions. This enables the creation of higher-dimension combination of features and thus the handling of more complex problems.

In Eskofier et al. [14] 84 features were extracted from each subject's data. A SVM classification model was used together with a PCA feature extraction algorithm. The performance of the classifier without feature extraction was not explored. With the PCA algorithm the maximum classification rate was of 95.8%. This performance was reached using 36 to 39 principal components. When using more than 39, classification performance is negatively affected.

In Wu et al. [17] with a feature set of 36 features the goal of this study was the analysis of the differences in performance between a SVM classifier combined with a PCA algorithm or combined with kPCA instead. Combined with PCA, the classifier achieved an accuracy of 87% with 14 principal components. With kPCA, the SVM classifier achieved 91% accuracy with 17 principal components. These findings showed that the non-linearity of kPCA results in better recognition outcomes when compared to PCA.

2.3.2.2 Discrete Wavelet Transform

Wavelet transforms are a mathematical method for analyzing a signal when its frequency varies over time. A wavelet can be seen as a 'brief oscillation' and for certain types of signals, wavelet analysis provides more precise information about signal data than other signal analysis techniques. Discrete Wavelet Transform (DWT) is a wavelet transform with a discrete sampling of the wavelets. An advantage over Fourier transforms is its temporal resolution.

El-Attar et al. [31] applied a DWT to raw sensor data in order to define the best feature set to improve classification. There were 8 total features defined and with this feature set an accuracy of 93.80% was achieved for the NN while an accuracy of 87.50% was achieved for the SVM classifier. For comparative purposes, feature extraction was also realized with an FFT algorithm. With the FFT instead of the DWT, the performance of the NN dropped significantly to 81.30%.

2.4 Classifiers used in Gait Pattern Recognition

Classification is the process by which the classifier recognizes or attributes a class to the given data points. Classification can be either binary or multi-class if there are two or more than two possible classes respectively. The focus of this dissertation is in supervised learning in which the whole data set consists of labeled data. This section encompasses the most used classification models in the context of gait recognition, as such this will be the main focus of the studies addressed in this section. The number of possible mathematical methods that can be used in a ML context is very vast although it is also application dependent. Out of all studies, the most common classification models found in literature were NNs, SVMs and kNNs.

In Badesa et al. [44] several machine learning methods were applied and their performances compared in order to select the best one to distinguish between different levels of post-stroke gait. Among others, this study also made use of an SVM (linear and Radial Basis Function (RBF) kernels) and a kNN algorithm. The best performance (91.43%) was achieved with a SVM classifier with a RBF kernel and 3 principal components. It achieved a significantly better performance than the kNN algorithm with only 80.95% for 3 or more principal components.

In Begg et al. [13], with 24 total features and a hill-climbing feature selection method, a SVM model and a NN were used. The SVM model using a linear kernel performed with 83.3% accuracy while the NN only showed an accuracy of 75%. The goal of this study was the distinction between gait patterns of young and elderly subjects.

Eskofier et al. [14] obtained an accuracy of 98.5% in distinguishing elderly subjects from young subjects using between 36 to 39 principal components by combining a linear kernel SVM classifier with PCA feature extraction.

In El-Attar et al. [31], a SVM with a linear kernel and an NN with a 20 neuron hidden layer were utilized. The total feature set was of 8 features and a DWT method was used. The NN outperformed the SVM model (87.50%) with an accuracy of 93.80%. The goal of this study was to improve the accuracy in the detection of FoG events in Parkinson's disease patients.

In Chan et al. [45], a NN along with other classifiers was evaluated and proved to have the greatest accuracy of 95.7% in distinguishing between walking up or down a flight of stairs and 80.6% for distinction between younger and older subjects.

In Pogorelc et al. [16], the main goal was the implementation of an early automatic recognition tool of distinct abnormal gait patterns. With 13 features from a motion capture system and six different classifiers:

SVM, Decision Trees (DT), kNN, RF, Naive Bayes (NB) and, NN. Each classifier was a multi-class classifier with five different possible classes, healthy, Parkinson's disease, hemiplegia, pain in the back and pain in the leg. kNN and NN achieved an accuracy of 100% and RF achieved an accuracy of 99.3%. SVM and NB achieved accuracies of 97.9% and 97.2% respectively.

In Alaqtash et al. [8], 19 features based on amplitude and temporal parameters of ground force reactions were defined. For classification a kNN algorithm and an NN were used. kNN proved to be more accurate than NN with an accuracy of 85% and 80% respectively. The goal of this study was recognition of gait patterns from multiple sclerosis, and cerebral palsy patients from healthy controls.

2.5 Critical analysis

2.5.1 Sensor Systems

Overall, the sensor systems here mentioned can be classified as either non-wearable (optical motion tracking systems and force plates) or wearable (inertial sensors, force sensitive resistors and EMG sensors). Non-wearable sensory systems can be more accurate and precise. However, these systems tend to be more expensive, complex and only operate in controlled environments, have a difficult time analyzing consecutive gait cycles for long-term applications, especially in a free-walking scenario [46].

Wearable sensor systems provide a cost-effective solution to the field of gait monitoring/recognition. Due to their portability these systems enable the possibility of 'in' field testing [27]. Table 20 in Appendix A.1 presents a summary of most of the studies discussed in this chapter. The most found non-wearable system were optical montion tracking systems, occasionally paired with force plates in order to also extract GRF data. In terms of wearable sensors, the vast majority of studies made use of inertial sensors, which can offer an accurate and reliable method to study human motion, but the degree of accuracy and reliability is site and task specific [18]. Due to the advantages of inertial sensors, the main focus of this research was based on data acquired from wearable sensory systems.

Taking into account the lower accuracy of wearable sensor systems and the need for high quality data in ML systems, one of the goals of this dissertation is the development of a bio-mechanical data estimation tool capable of accurately extract relevant features from a wearable sensor system.

2.5.2 Feature Determination

For the best possible performance of a classifier, data quality and abundance is of great importance [47]. But besides this, a correct choice of features from the total raw data set will have a major influence in classification performance.

This research has shown that there is a preference for the use of time-domain features in literature. A possible justification for this is the fact that this type of features are easier to interpret because they are usually directly related to the movement of the subject under test.

Additionally the trend for most studies to use non sequential classification models stood out. The vast majority of studies use models that deal with non-sequential data, were each training example is associated with a class label, and fail to explore other possibilities such as sequential models (Hidden Markov Modelss (HMMs), Recurrent Neural Networks (RNNs)) or models originally intended for image data (CNNs). This dissertation explores those possibilities by structuring the obtained data so as to fit all three types of models: non-sequential, image and sequential.

2.5.3 Dimensionality Reduction

With the growing size of the feature set, just like the number of relevant features increases so to will the number of irrelevant/redundant features, which will be detrimental to the classifier's performance. This is one of the main reasons why in a great majority of studies employs dimensionality reduction in order to maintain a greater percentage of relevant features in the feature set [48].

There are many methods available to reduce the dimensionality of a feature set. According to this research however, the performance of these methods is highly dependent on several factors like the data used or their intended goal. In literature, there is no clear preference between feature selection or feature extraction. The most found dimensionality reduction methods are the Hill-Climbing (feature selection) algorithm and the PCA (feature extraction).

2.5.4 Classifiers used in Gait Pattern Recognition

Table 21 in Appendix A.1, presents a summary of some of the studies addressed in this chapter. Most of these studies have the goal of distinguishing between gait patterns of young and elderly subjects while other studies focus more on the recognition of pathological gait patterns (Parkinson's, stroke, cerebral palsy, etc).

In literature, the two most popular classification models are SVM and NN although other models are also heavily used as well. From this research, there is no clear classifier that exhibits a superior performance under certain gait recognition goals. Much like with dimensionality reduction methods, there is no single classifier that outperforms all others. Depending on the context it is used in and more importantly on each models correct tuning of their hyper-parameters, its performance can vary greatly. Not only this, but a correct feature engineering can be decisive in obtaining a good performance from any classification model.

From all studies found, only three studies had the goal of post-stroke gait recognition. This implies that this condition is not sufficiently explored in the field of gait recognition when compared to conditions such as old age or Parkinson's disease.

Kaczmarczyk et al. [49] made use of a motion tracking system to determine 11 kinematic features: knee joint, sagital and frontal hip joint. This study used an Artificial Neural Network (ANN), with success rates from 100 % for the knee joint to 86 % for the frontal motion of the hip joint.

Mannini et al. [35] proposed a general probabilistic modeling approach for the classification of different pathological gaits (old age, post-stroke and Huntington's disease). This study used several IMUs and an instrumented gait pressure mat for the determination of features and used an HMM for the extraction of additional features as well, creating a total feature set of 90 time-domain, frequency-domain and HMM-based features. A SVM classifier with a RBF kernel achieved an accuracy of 90.5 %.

Lee et al. [34] used a RF classifier to improve the accuracy of hemiplegic gait classification of 15 healthy subjects and 20 hemiplegia patients undergoing stroke treatment, achieving an accuracy of 100 % in the process. The classifier used 165 time-domain features extracted from inertial bio-mechanical signals.

Additionally, no studies using DL in gait recognition were found. With DL models there would be no need for feature engineering or dimensionality reduction because these models implicitly perform feature extraction to a new more abstract feature space. This enables these methods to uncover abstract relationships between data by themselves, simplifying the classification process. The possibilities of DL in the context of gait pattern recognition were also explored in this dissertation by implementing four NNs in order to assess their performance in relation to regular ML methods.

3

RECOGNITION SYSTEM OVERVIEW

This chapter presents a high level perspective of the recognition system implemented in this dissertation. The several stages through which bio-mechanical data is processed and stored will be mentioned as well as the main algorithms of each building block of this system.

The system developed in this dissertation can be though of as two separate blocks: a bio-mechanical data estimation and feature determination tool and a gait pattern classification tool. The first block performs the task of extracting relevant bio-mechanical data from the sensor signals and determining relevant features. The classification block re-structures the resulting feature set, if necessary, and makes use of several ML methods to be able to differentiate between healthy and post-stroke gait patterns. The flowchart in Figure 5 presents a high level perspective of the whole system.

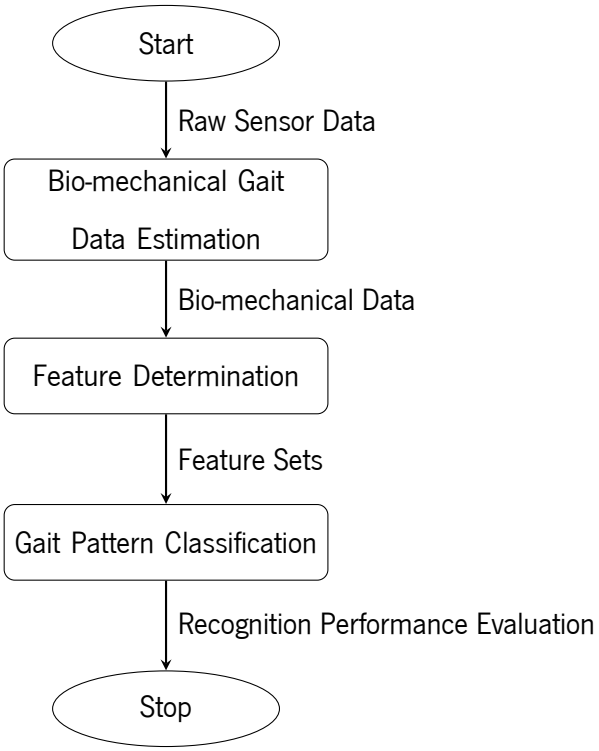


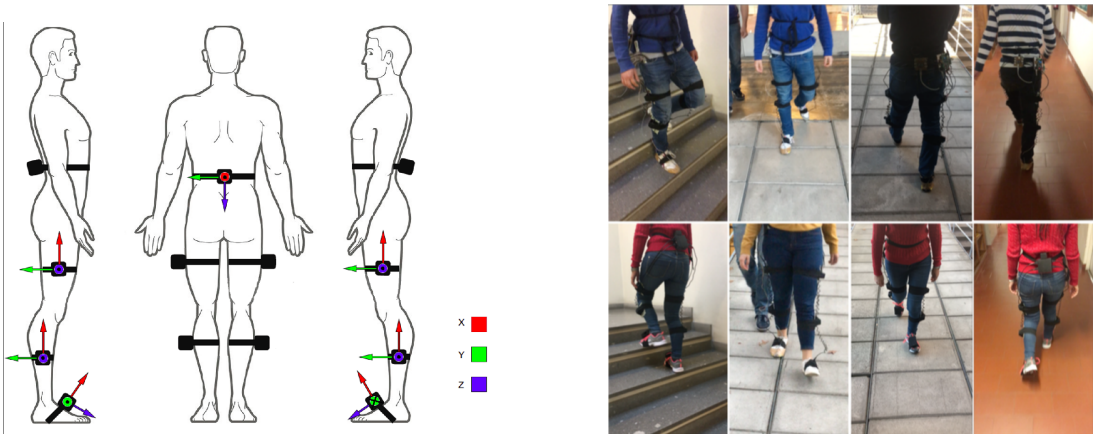
Figure 5: System Flowchart.

3.1 Bio-mechanical Data Estimation and Feature Determination Tool

The the raw sensor data was acquired from InertialLab, an inertial wearable sensor system developed at BirdLab, UM consisting of seven IMUs attached to a subject's body. A diagram of this system as well as photos of its usage during trials is shown in Figure 6.

Each IMU contains a triaxial accelerometer and a triaxial gyroscope, in total there are seven IMUs, one in each foot, shank, thigh and one in the trunk making up a total of 21 acceleration and 21 angular velocity signals acquired from the body.

The data obtained is stored in a database organized by type of gait pattern (healthy or post-stroke). This data is then fed to the bio-mechanical data estimation tool and run through a gait event detection Finite State Machine (FSM) [50]. This algorithm will identify the time instances were each gait event occurs so as to enable the computation of several non-sequential features per gait cycle and the segmentation of all signals into gait cycles, for the creation of a sequential and non-sequential feature set.



(a) InertialLab overview.

(b) InertialLab during trials.

Figure 6: InertialLab.

For feature determination, signals from each sensor are used to create a quaternion representation of their orientation relative to the earth frame. A quaternion is a four dimensional extension to complex numbers and found practical use in the calculations of rotations/orientation of bodies in three dimensional space. This four-dimensional representation of orientation is then used with the raw sensor data to compute the acceleration and orientation (in Euler angles) of the sensor relative to the earth frame. This way it is possible to compute the velocity/position of the sensor over time as well as the angle of each joint of the lower body, allowing for an accurate and effective estimation. This whole process is described in more detail in Chapter 4.

The gait events previously determined are now used to segment all signals by gait cycle and determine several non-sequential features that will serve as input to the classification tool for two sequential classifiers (LSTM and C-LSTM). Additionally, the raw sensor signals as well as the quaternion, joint angle, global acceleration, velocity and position signals are utilized to create a sequential feature set to serve as input to LSTM and C-LSTM networks, which are DL models. All feature sets will be stored for later use in the created database. Creation of each type of feature set will be further discussed in Chapter 5.

Figure 7, presents the main flowchart of the bio-mechanical data estimation and feature determination tool. This tool receives the raw sensor signals as input and computes each gait event for later segmentation. The sensor data is then used to determine the orientation of each sensor in quaternion form so as to compute the necessary bio-mechanical signals. After segmentation, all available data (raw sensor data, computed bio-mechanical and quaternion data) is used to create each feature set.

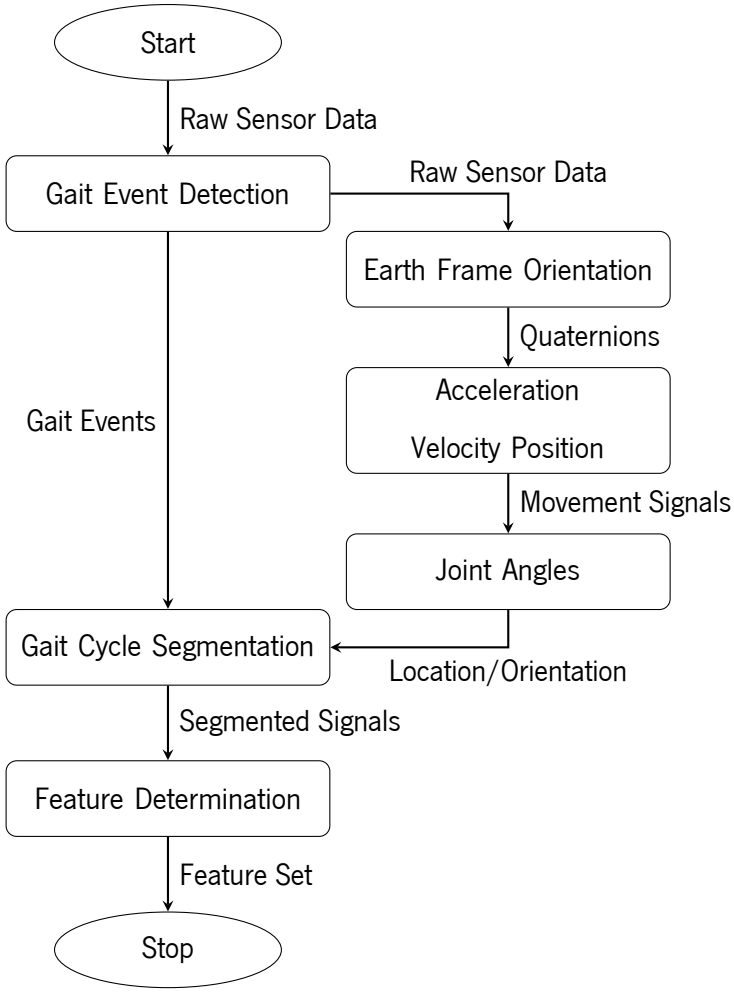


Figure 7: Bio-mechanical data estimation and feature determination tool.

3.2 Gait Pattern Classification Tool

The classification tool was initially developed in Gonçalves [41] and consisted of a normalization stage, a dimensionality reduction stage and a nested Cross Validation (CV) stage with 4 different classifiers available (SVM, kNN, DA and RF). The final developed tool in this dissertation consists now of a test/training split stage, a normalization stage, a dimensionality reduction stage and 8 different available classifiers: FFNN, CNN, LSTM, C-LSTM and the 4 previously mentioned classifiers. Dimensionality reduction is used depending on the choice of classification model made. In the case of Deep Neural Networkss (DNNs), dimensionality reduction methods are not required, as can be seen in the flowchart in Figure 8. After dimensionality reduction, the chosen classification model is trained and its performance evaluated. For the training of each model, nested CV is employed to obtain a worst case scenario of the performance of each model.

The data from the feature set is split into a training and a validation set in order to confirm that the performance of each model is within the range of the performance estimation obtained through CV. The train/test split was performed in two ways: a random split and a subject split. In the random split, a random 70% of the data was stored in a training set and the remaining 30% was stored in a test set. The data was stratified to avoid any unbalance of classes. In the subject split, one random healthy subject and one random post-stroke subject were selected and their gait data was entirely removed from the training set in order to create a test set. The remaining data was used to build the training set.

The resulting feature sets being used as input to the classification tool are restructured depending on each models specified input data. The original sequential and non-sequential feature sets will also be stored in this tool. Additionally, the non-sequential feature set will also be converted to an image set and the sequential feature set will be converted into a video set. This is due to the fact that CNN works with image data and the C-LSTM works with sequences of images or videos. There are a total of eight possible models to choose from in this tool: SVM, kNN, RF, DA, FFNN, CNN, LSTM and C-LSTM. The four NNs were implemented in this dissertation, with this process being described in detail in Chapter 6.

In Figure 8, the feature set is divided in a training and a test set with both sets later normalized. For each ML classifiers there is the possibility of dimensionality reduction (Genetic Algorithm (GA), Sequential Feature Selection, Minimum Redundancy Maximum Relevance (mRMR), PCA and Analysis Of Variance (ANOVA)) in order to reduce each feature set's dimension. The estimation of each classifier's performance is done through nested CV and later verified with the test set.

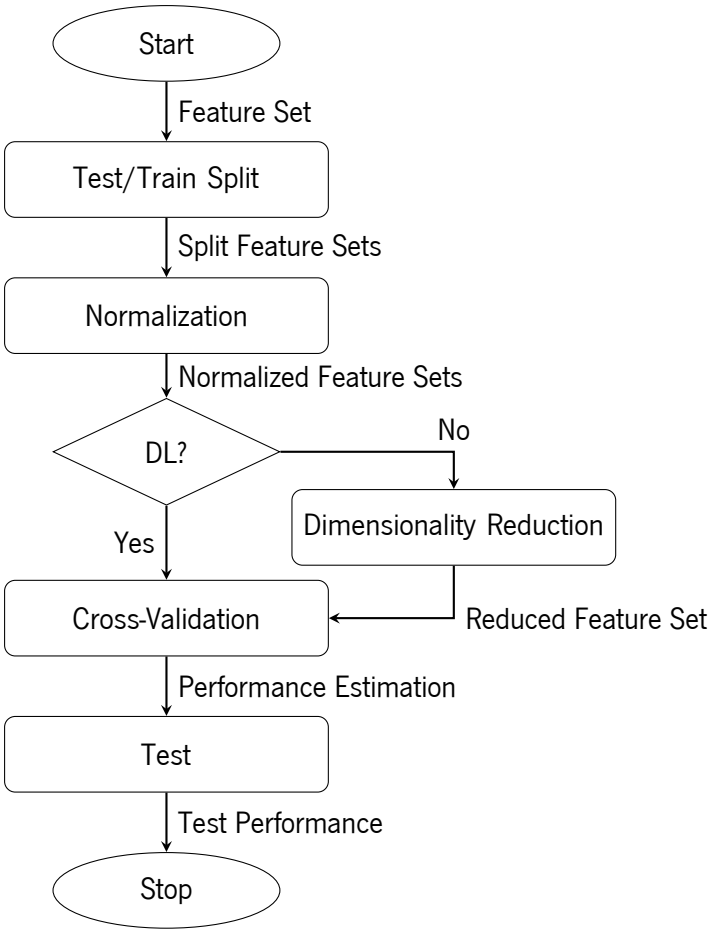


Figure 8: Classification Tool.

3.3 Database

The data used as input to this tool was organized in a database in terms of gait disorder and sensor system used. The gait disorder addressed in this dissertation is post-stroke gait however, this database was created to encompass all gait data from similar systems from the laboratory, so data from different disorders is also present in this database, as shown in Figure 9.

In the top directory there is a distinction between healthy or pathological gait patterns. Inside the *Pathological* directory there is one directory per disorder. Inside each disorder directory and inside the *Healthy* directory there is one directory per sensor system used. In each of these directories there is a directory named *Data* containing the acquired sensor data and another directory named *Features* containing the computed features for usage in the classification tool.

The data contained in this database was obtained from the InertialLab system, mentioned in the previous chapter, with 10 healthy, 7 post-stroke, 2 Parksinon's disease and 2 ataxia subjects. Additionally, a search

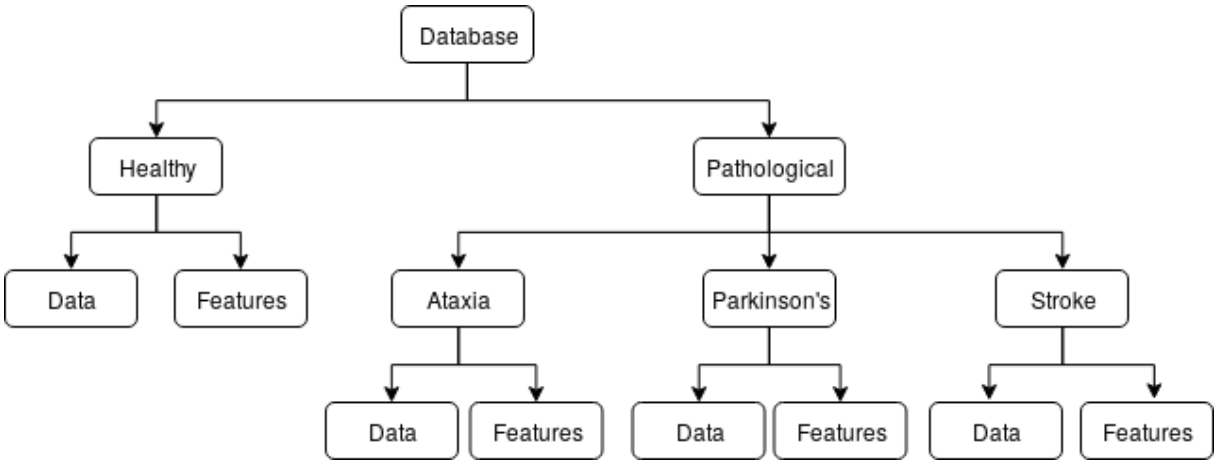


Figure 9: Database Layout.

was made for possible available online databases. From this search two different online gait databases were found. The first, Movement Analysis in Real-world Environments using Accelerometers (MAREA) [51], is a gait database from Halmstad University, Sweden. The data to this database was acquired with a somewhat similar system to InertialLab and has got data from 20 subjects. The second is an online biometric database, Institute of Scientific and Industrial Research (ISIR), with several different datasets, from Osaka University in Japan. Two of these datasets also use similar sensor systems and of these two, one contains data from forward walking trials. This is a very large dataset containing data from 744 subjects [29].

These databases were only available in a case-by-case basis. It is necessary to contact the database administrator and sign a release agreement. Unfortunately, after trying to establish contact by e-mail no answer was received.

A third database was also found in the Physionet website [52]. This is also a database consisting of several different datasets. All datasets contain physiological data but only a few of them contain data from gait trials. However, most of these gait datasets contain data acquired through sensor systems not similar to InertialLab.

Due to these limitations, this data base was created from a database available in BirdLab, built with data from trials done with InertialLab and other systems (Xsens, GaitShoe, etc). The data chosen from this database was taken from trials performed with InertialLab with healthy, Parksinson's disease, post-stroke and ataxia subjects. Table 1 presents a summary of some of the trials performed using wearable inertial sensor systems for several different physical and testing conditions.

Table 1: BirdLab trials

Physical Condition	No Subjects	Sensor System	Sampling Frequency (Hz)	Test Condition
Healthy	7	InertialLab	200	Forward Walking, Turning, Ramp, Stairs
Healthy	3	InertialLab + Xsens	60	Forward Walking, Turning, Ramp, Stairs
Stroke	7	InertialLab + Xsens	100 & 60	Treadmill, Forward Walking, Turning, Stairs
Parksinson's	2	Xsens	60	Not Specified
Ataxia	2	InertialLab	100	Forward Walking

4

BIO-MECHANICAL DATA ESTIMATION

This chapter details each method applied in the extraction of bio-mechanical data for determination of features. The first section describes the adaptation of method for gait event detection used to determine the stationary-foot instances. The following section details the adaptation of a quaternion-based orientation filter capable of estimating the orientation of each sensor relative to the earth frame. The final section describes the methods used for the estimation of earth frame acceleration, velocity and position signals using the orientation filter's output.

This bio-mechanical data estimation tool was developed in MATLAB and has the purpose of working as a stand alone system that could be used to estimate bio-mechanical data from any inertial sensor systems and not necessarily for a gait classification goal.

As stated in Chapter 3, this tool receives bio-mechanical signals from 7 sensors and is able to estimate their position/orientation relative to the earth frame. Each sensor is tracked by a conversion of the sensors output to a quaternion representation of its orientation relative to the earth frame. This method presents some advantages to using Euler angles that will be discussed ahead. The sensor data from the database was already pre-processed, so no filtering was required.

4.1 Gait Event Detection

The main purpose of gait event detection in this context, is related to the segmentation of each signal. Segmentation is necessary for the creation of both sequential and non-sequential feature sets, where each training example encompasses one full gait cycle. Gait event detection was performed using the y-axis angular velocity signal of each foot. The FSM used for this purpose, was adapted from Figueiredo et al. [50] to enable gait event detection for post-stroke gait as well. This adaptation was necessary due to the

fact that each gait pattern varies considerably from subject to subject. This difference is even greater when considering subject with motor disorders, which will negatively impact a correct detection of gait events.

4.1.1 Healthy Gait Patterns

The features determined by this system are computed for each gait cycle. To segment each bio-mechanical signal into several gait cycles, a gait detection FSM was used to detect the following gait events:

- **Toe-Off (TO):** Instant the foot leaves the ground.
- **Mid-Swing (MMSW):** Instant the swinging limb passes the opposite stance limb.
- **Heel-Strike (HS):** First ground contact with the heel of the leading limb.

The initial FSM was originally develop in Figueiredo et al. [50]. This algorithm needs only the input of the y-axis angular velocity signal of the foot to correctly identify each gait event. Figure 10 shows the gyroscope signal related to each phase of a single gait cycle. Adaptive thresholds were established to detect minimum and maximum (MIN_{thr} and MAX_{thr}).

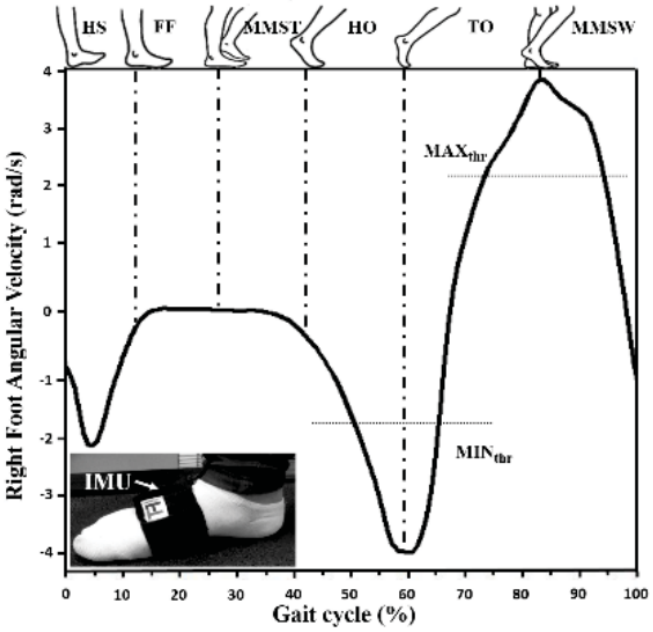


Figure 10: Angular velocity of instep of right foot related to each gait event [53].

The decision rules for this FSM were defined heuristically and were based on curve tracing techniques, such as thresholds crossing, local extrema detection and signal derivatives evaluation [53]. These rules are present in detail in Table 2

Table 2: Gait event detection decision rules using adaptive thresholds.

Condition	Decision Rule	State
1	$(gyro_n > MAX_{thr})$ AND $(derivative_n < 0)$ AND $(derivative_{n-1} > 0)$ AND $(gyro_{index} - MAX_{index} \in [0.7 * CAD_{prev}; 1.3 * CAD_{prev}])$	MAX/ MMSW
2	$((HS\_thr_{mean} - HS\_thr_{std} < gyro_n < HS\_thr_{mean} + HS\_thr_{std})$ OR $1st\_gyro\_min)$ AND $1st\_gyro\_max$ AND $(gyro_{index} - MAX_{index} \in [0; 0.4 * CAD_{prev}])$	HS
3	$(derivative_n \approx 0)$ AND $\lvert derivative_n \rvert < 0.2$ AND $1st\_gyro\_min$ AND $(gyro_{index} - MAX_{index} \in [1.5 * CAD_{prev}; 1.0 * CAD_{prev}])$	FF
4	$MMST\_counter > (HO_{indexPrev} - FF_{indexPrev})/2$	MMST
5	$(gyro_n < 0)$ AND $(derivative_n < 0)$ AND $(derivative_{n-1} < 0)$ AND $(derivative_n > 0.9 * derivative_{n-1})$ AND $(gyro_{index} - MAX_{index} \in [0.3 * CAD_{prev}; 1.0 * CAD_{prev}])$	HO
6	$(gyro_n > MIN_{thr})$ AND $(derivative_n = 0)$ AND $(derivative_{n-1} < 0)$ AND $(gyro_{index} - MAX_{index} \in [0.5 * CAD_{prev}; 1.1 * CAD_{prev}])$	TO

The detection of MMSW is defined as the local maximum above an established adaptive threshold (MAX_{thr}), HS as the angular velocity between a range empirically determined ($HS\_thr_{mean} \pm HS\_thr_{std}$) after the maximum corresponding to MMSW occurs. In Foot-Flat (FF), the angular velocity is nearly constant, so the first derivative is almost zero for several samples and Mid-Stance (MMST) is determined to occur n samples after the occurrence of FF, where n corresponds to the duration of the last valid MMST. For Heel-Off (HO), the angular velocity reaches negative values after a constant period and TO corresponds to the second minimum detected by an adaptive threshold (MIN_{thr}). These rules are also dependent on Gait Cadence (CAD), this way the algorithm can be sensible to changes in gait patterns [53]. With these rules, the algorithm is able to detect variations of step (duration of gait cycle) and speed (amplitude of angular velocity) and thus define intervals were each event should occur, depending on CAD, and adjust the adaptive thresholds MAX_{thr} and MIN_{thr} [53].

Figure 11 shows the flow chart of the algorithm, which is composed by five steps executed sequentially in each iteration. This algorithm starts by collecting and low-pass filtering the foot y-axis angular velocity. Each sample is then processed through the four following stages. The first derivative is computed, detecting when the velocity increases, decreases or becomes constant. Afterwards, the minimum/maximum values are determined to detect HS, MMSW, FF and TO. The next stage, computes CAD, using the three last valid steps, to establish statistic decision limits where the events must occur. In the beginning of the subject's gait, initial conditions are used until a valid CAD is obtained.

The developed FSM belongs in the final stage and is also depicted in Figure 11. This FSM has six states, one for each gait event (MAX/MMSW, HS, FF, MMST, HO, TO), and two additional states, the default state (DEF) and a reset state (R). The decision rules defined in Table 2 and an exit condition (E) are used to trigger each transition. The initial state is the R state where all variables are reset and the initial conditions (empirically tuned) are set.

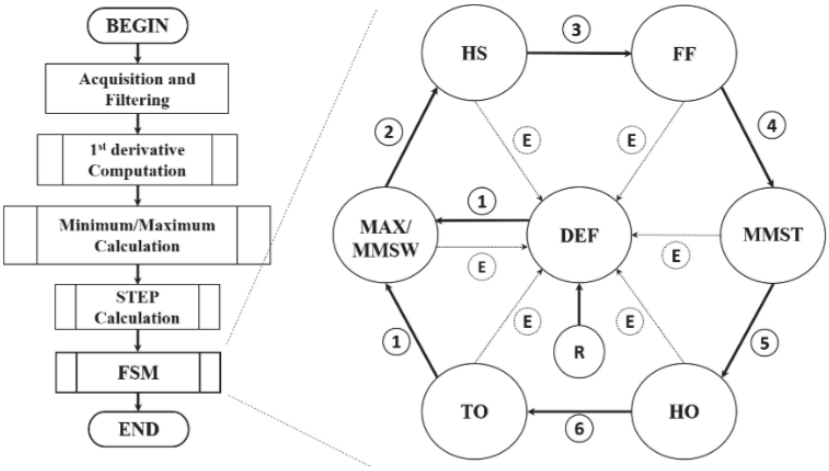
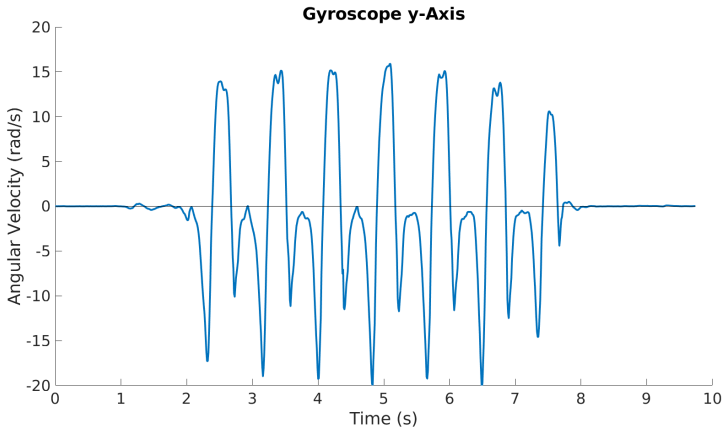


Figure 11: Gait detection algorithm [53].

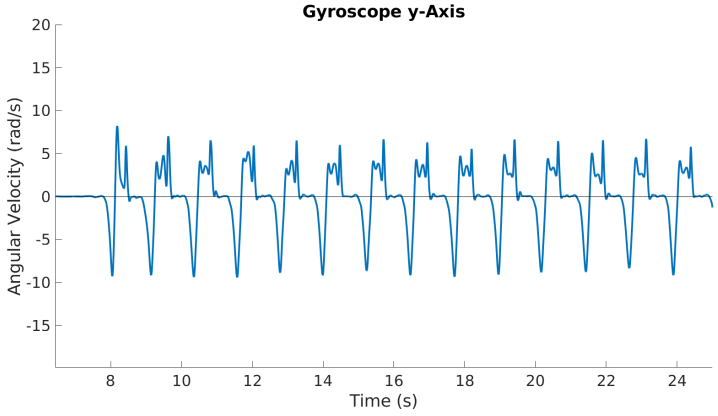
4.1.2 Post-Stroke Gait Patterns

Considering that the FSM previously described was developed with only healthy gait patterns in mind, some changes had to be made to the decision rules in order to adapt it to post-stroke gait present in this dissertation. The decision rules for post-stroke gait were heuristically determined. Firstly an analysis of the y-axis angular velocity signal of the foot of post-stroke subjects was compared to healthy subjects in order to assess the most crucial differences so as to tune the algorithm to these new signals.

Both healthy and post-stroke signals are shown in Figure 12. The first difference that is noticeable between both signals is the difference in amplitude, the foot's angular velocity in post-stroke gait is significantly lower than in healthy gait. Furthermore, in the maximum that corresponds to the MMSW event there is a sudden drop in post-stroke gait. In many other trials the signal even reaches negative values at this instant. In the instant where HS occurs in the stroke subject's foot, the magnitude of the negative velocity values is much lower, many times the signal does not even become negative, which will make detection of HS with the original decision rules very difficult.



(a) Healthy gait.



(b) Post-stroke gait.

Figure 12: y-axis foot velocity for both physical conditions.

These changes in angular velocity can be attributed to the lack of control of the foot that the subject has on the corresponding paretic leg, this symptom is usually called drop foot. On top of this, the signal from the paretic foot has an overall significantly lower amplitude when compared the the healthy foot's signal.

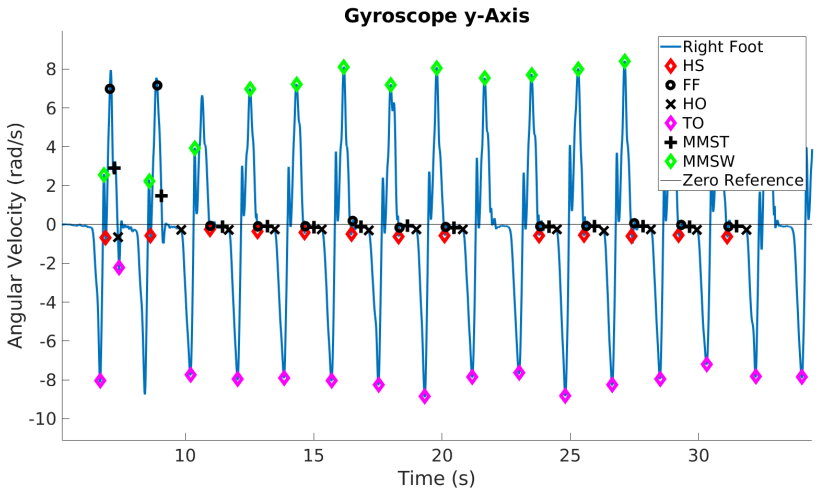


Figure 13: Post-stroke gait event detection with original rules.

Figure 13 shows another example of the foot's angular velocity of a post-stroke subject and the detected gait events using the original decision rules. In the positive part of the signal, there is a sudden drop in amplitude before a global maximum (MMSW) is reached, the gait algorithm falsely determines this amplitude drop as a HS which in turn corrupts the detection of following events and renders this algorithm useless without any tuning. Even by correcting the detection of this event, there still is the issue that during HS the angular velocity of the paretic foot does not reach zero in many cases (as seen in Figure 12b) which will in turn influence the algorithm to assign HS to the next minimum, which in reality would correspond to a TO event. The alterations made to the decision rules are present in Table 3 along with the values already used for healthy gait patterns.

Table 3: Changes in decision rules

Parameter	**Healthy**	**Stroke**
HS_thr$_{mean}$	-0.5	-0.51
HS_thr$_{std}$	0.05	0.5
CAD	$1.1 \times fs$	$1.4 \times fs$

The changes in these parameters were mostly achieved through trial and error. The angular velocity of the paretic foot during a HS event has a greater variability than in the case of healthy gait. For this reason, the value for HS_thr_{std} had to be increased while HS_thr_{mean} was slightly adjusted. The initial value of CAD, previously set to 70, was now determined by multiplying a constant value with the sampling frequency (fs) of the sensor data acquired. This multiplier term was heuristically determined by careful observation of the detection of gait event for all trials. The results of these changes are shown in the graph in Figure 14.

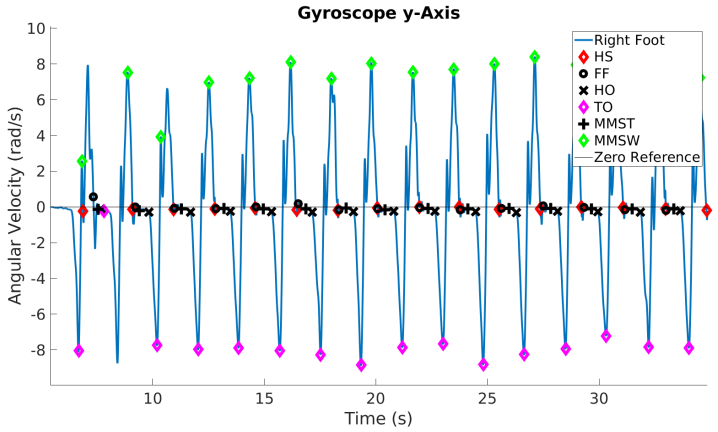


Figure 14: Post-stroke gait event detection with new rules.

With the new decision rules, the majority of gait events for post-stroke gait can now be correctly predicted, enabling this way the correction of drift, segmentation of gait and calculation of several features. With

these alterations, it is necessary for the algorithm to know beforehand the condition of the subject (i.e., healthy or stroke) in order to apply the correct decision rules. However, this FSM with the altered decision rules was not yet validated with stroke patients.

4.1.3 Gait Event Correction

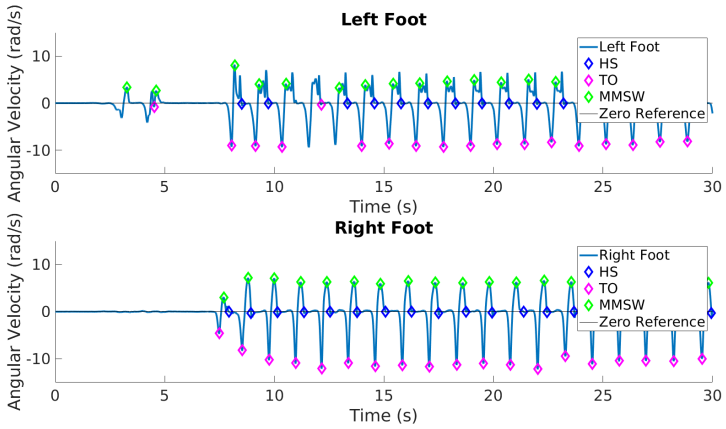
The gait event detection algorithm discussed earlier is not 100% perfect. Since a correct gait parameter estimation is highly dependent on a good prediction of gait events, the occurrence of false detection or no detection at all, must de addressed. Of all detected events, only HS, TO and MMSW will be corrected for these are the events that are essential to compute the necessary features.

Firstly, all events detected before the first HS and the first TO will be eliminated. This way every gait trial will begin with HS on the leg that initiates gait and TO on the other. Additionally, it must be ensured that all HS events are alternated between each leg, this event will be crucial to posterior segmentation of each signal. All valid gait cycles are required to have all three events in the following order:

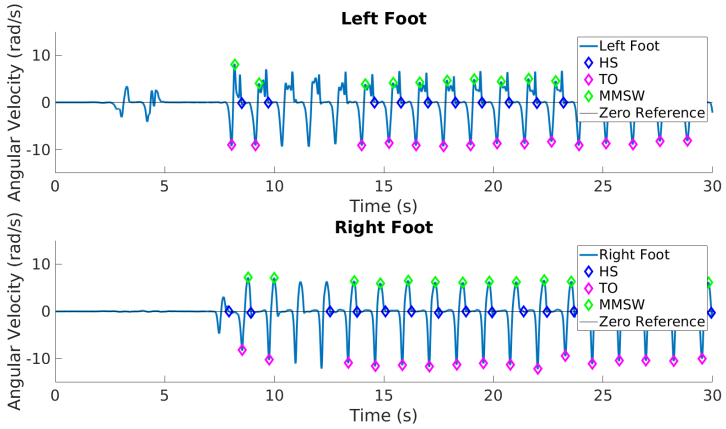
$$FirstLeg : HS \rightarrow TO \rightarrow MMSW$$
$$FollowingLeg : TO \rightarrow MMSW \rightarrow HS$$

Any gait cycle whose events do not respect these requirements will be discarded and the corresponding gait cycle of the opposite leg as well. This will ensure that all selected gait cycles will be in synchronicity with each other. The detected gait events before and after correction are present in Figure 15.

Based on the previous rules, on the corrected signal, the first event only occurs when the gait is initiated, after the 7 seconds mark. Gait cycle elimination can also be seen after the 10 seconds mark. With these corrections, it is possible to use each signal to compute the necessary features with a significantly reduced margin of error.



(a) Gait Events before correction.



(b) Gait Events after correction.

Figure 15: Elimination of false detections.

4.2 Inertial Sensor Tracking

In a practical scenario, with a wearable sensor system like InertialLab, each sensor's initial vertical axis is never perfectly aligned with the vertical earth axis. Due to this offset in orientation, it is advantageous to use an orientation filter to provide as estimation of each sensor's orientation relative to the earth frame. An orientation filter estimates the orientation of inertial sensors through the optimal fusion of gyroscope, accelerometer and magnetometer measurements. The Kalman filter is an example of an accurate and effective solution very used in literature. However, they are not easy to implement and have a heavy computational cost. Other solutions exist but they either also have a high computational cost or limited operating conditions.

The orientation filter used in this dissertation was developed in Madgwick [54] and employs a quaternion representation of the orientation of an IMU or Magnetic, Angular Rate, and Gravity (MARG) sensor. In this case, this orientation filter was adapted to the data acquired with InertialLab. One advantage of the use of

quaternions is that, unlike with Euler angles, Gimbal Lock does not occur. This occurence consists on the loss of a degree of freedom when the axes of two of the three axis are driven into a parallel configuration. This new filter improves the computational load required, can operate on a lower sampling rate, is easier to implement and tune without any loss in performance. This orientation filter will be used to compute the acceleration, velocity and position of each foot mounted IMU relative to the earth frame as well as the angles of each lower joint.

4.2.1 Theoretical Background

4.2.1.1 Quaternion Arithmetic

A quaternion is a four-dimensional complex number that can be used to encode the rotation of a body or frame in a three-dimensional space. A quaternion q is composed of a real number and three imaginary numbers and can be seen as the sum of a scalar q_1 and a vector $q = (q_2, q_3, q_4)$ [55]. This can be represented as shown in Equation 1:

$$q = q_1 + q_2 i + q_3 j + q_4 k \tag{1}$$

Considering two arbitrary orientation frames A and B, the orientation of B relative to A can be represented through a rotation of angle θ around an axis ${}^{A}\hat{r}$ defined in A.

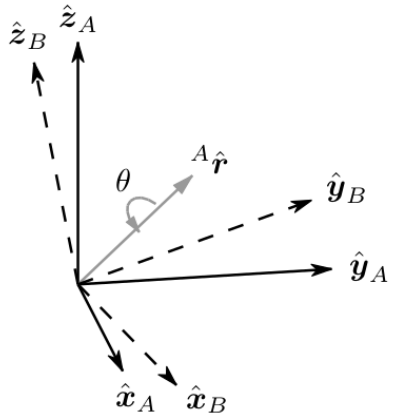


Figure 16: Rotation axis ${}^{A}\hat{r}$

The quaternion that describes this rotation is represented as

$$ {}^{A}_{B}\hat{q} = [q_1 \quad q_2 \quad q_3 \quad q_4] = [cos\frac{\theta}{2} \ -r_x sin\frac{\theta}{2} \ -r_y sin\frac{\theta}{2} \ -r_z sin\frac{\theta}{2}] \tag{2} $$

where r_x, r_y, and r_z are the x, y and z components of the rotation axis ${}^{A}\hat{r}$.

For the purpose of tracking the position of a body in space, it is necessary that the arithmetic is performed with normalized quaternions, therefore all quaternions must be of unit length.

The conjugate of a quaternion can be used to swap the relative frames in an orientation. For example, the orientation of frame A relative to B can be represented as

$$ {}^{A}_{B}\hat{q}^{*} = {}^{B}_{A}\hat{q} = [q_1 \ -q_2 \ -q_3 \ -q_4] \tag{3} $$

The product of two quaternions describes compund orientations. An orientation ${}^{A}_{C}\hat{q}$ can be achieved with two rotations ${}^{A}_{B}\hat{q}$ and ${}^{B}_{C}\hat{q}$.

$$ {}^{A}_{C}\hat{q} = {}^{A}_{B}\hat{q} \otimes {}^{B}_{C}\hat{q} \tag{4} $$

This product is determined using Hamilton's rule. Considering two quaternions a and b, their product is determined as

$$ \begin{aligned} a \otimes b &= [a_1 \quad a_2 \quad a_3 \quad a_4] \otimes [b_1 \quad b_2 \quad b_3 \quad b_4] \\ &= \begin{bmatrix} a_1 b_1 & -a_2 b_2 & -a_3 b_3 & -a_4 b_4 \\ a_1 b_2 & a_2 b_1 & a_3 b_4 & -a_4 b_3 \\ a_1 b_3 & -a_2 b_4 & a_3 b_1 & a_4 b_2 \\ a_1 b_4 & a_2 b_3 & -a_3 b_2 & a_4 b_2 \end{bmatrix} \end{aligned} \tag{5} $$

However it is necessary to keep in mind that, unlike with complex numbers, quaternions are not commutative. This means that $a \otimes b \neq b \otimes a$.

4.2.1.2 Rotation Matrix

The rotation of a three-dimensional vector from frame A to frame B is described in the Equation 6.

$$^{B}v = {}^{A}_{B}\hat{q} \otimes {}^{A}v \otimes {}^{A}_{B}\hat{q}^{*} \tag{6}$$

^{A}v and ^{B}v are the same vector described in frame A and frame B respectively where each vector contains a 0 inserted as the first element to make them 4 element row vectors.

This rotation can be described by a rotation matrix

$$^{A}_{B}R = \begin{bmatrix} 2q_1^2 - 1 + 2q_2^2 & 2(q_2q_3 + q_1q_4) & 2(q_2q_4 - q_1q_3) \\ 2(q_2q_3 - q_1q_4) & 2q_1^2 - 1 + 2q_3^2 & 2(q_3q_4 + q_1q_2) \\ 2(q_2q_4 + q_1q_3) & 2(q_3q_4 - q_1q_2) & 2q_1^2 - 1 + 2q_4^2 \end{bmatrix} \tag{7}$$

To convert this orientation back to Euler angles ψ, θ and ϕ, equations 8 to 10 are used.

$$\psi = \arctan(2q_2q_3 - 2q_1q_4, 2q_1^2 + 2q_2^2 - 1) \tag{8}$$

$$\theta = -\arcsin(2q_2q_4 + 2q_1q_3) \tag{9}$$

$$\phi = \arctan(2q_3q_4 - 2q_1q_2, 2q_1^2 + 2q_4^2 - 1) \tag{10}$$

4.2.1.3 Orientation From Gyroscope Data

With a tri-axial gyroscope it is possible to obtain the angular rate in the x, y and z axis of the sensor reference frame. The vector formed from each sensor frame axis can be converted to quaternion form by adding a 0 as the first element, according to Equation 11

$$^{S}\omega = [0 \quad \omega_x \quad \omega_y \quad \omega_z] \tag{11}$$

The rate of change of the earth frame relative to the sensor frame can be described by performing the quaternion derivative

$$ {}^{S}_{E}\dot{q}_{w,t} = \frac{1}{2} {}^{S}_{E}\hat{q}_{est,t-1} \otimes {}^{S}\omega_t \tag{12} $$

${}^{S}\omega_t$ is the three dimensional angular rate at time t and ${}^{S}_{E}\hat{q}_{est,t-1}$ is the previous estimate of orientation. By numerically integrating the quaternion derivative ${}^{S}_{E}\dot{q}_{w,t}$, it is possible to obtain the orientation of the earth frame relative to the sensor frame assuming the initial conditions are known. This is shown in the following equation

$$ {}^{S}_{E}q_{w,t} = {}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{w,t}\Delta t \tag{13} $$

where ${}^{S}_{E}q_{w,t}$ is the orientation of the earth frame relative to the sensor frame at time t and Δt is the sampling period.

4.2.2 Main Algorithm

The implementation of the orientation filter was adapted from the algorithm developed by Madgwick and also makes use of its library [54]. This orientation filter has the purpose of estimating the quaternions of the orientation of each sensor and, from that data, determine the earth frame acceleration, velocity and position of the foot IMUs and the angles of each lower joint.

The main algorithm is present in the flowchart in Figure 17. The structure remains very similar to the one developed by Madgwick apart from the inclusion of the FSM for gait event detection and a few changes in the drift compensation methods for calculating velocity and position.

The first block of the flowchart initializes the input data including accelerometer, gyroscope and time measurements as well as the sampling period. In the second block, the foot's zero velocity phases are calculated by computing the signal of the magnitude of the foot's acceleration over time by using all three accelerometer axis, as shown in Equation 14. A threshold was applied to this signal to separate stationary from non-stationary instants, just like in the original algorithm. The value of this threshold was equal to 1, as seen in Figure 18

In the third block, the methods used for tracking IMU orientation remain the same as in the original algorithm. An object of an Attitude and Heading Reference System (AHRS) class is initialized with the necessary parameters, the quaternion of the initial orientation of the sensor relative to the earth frame is

computed and then the same process is applied for every sample and stored in a quaternion array, used to rotate the accelerometer measurements to the earth frame.

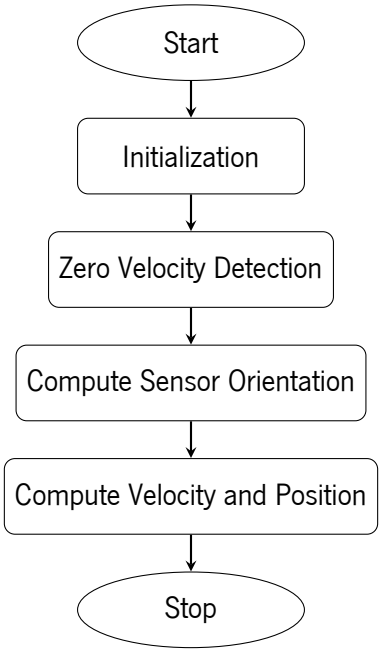


Figure 17: Main Algorithm

Lastly, in the forth block, from the acceleration of the sensor in the earth frame, the velocity of the sensor is computed through the Zero-Velocity Update (ZUPT) method. With the resulting values of velocity, the position of the sensor is calculated as well.

4.2.3 Zero Velocity Detection

For the detection of the stationary phases the magnitude of the acceleration of the foot is calculated with Equation 14 and using the acceleration measurements of all three axis.

$$acc_{mag} = \sqrt{acc_x^2 + acc_y^2 + acc_z^2} \tag{14}$$

A high-pass filter is applied to acc_{mag}. Afterwards, the absolute value of this signal is computed and low-pass filtered before the application of a threshold to it. However, as is shown in Figure 18, due to the high variation of this signal there are moments were a false detection or failed detection occurs, this can be seen in Figure 18 at 2 and 3 seconds respectively. Several different threshold values were experimented with but it was not possible to configure a specific threshold for all conditions. With this in mind, the

previously described FSM for gait event detection was added to this algorithm in order to improve zero velocity detection. This means that Equation 14 is used only to determine the instants were gait is initiated and terminated, the stationary instants during gait are all determined through the gait detection FSM.

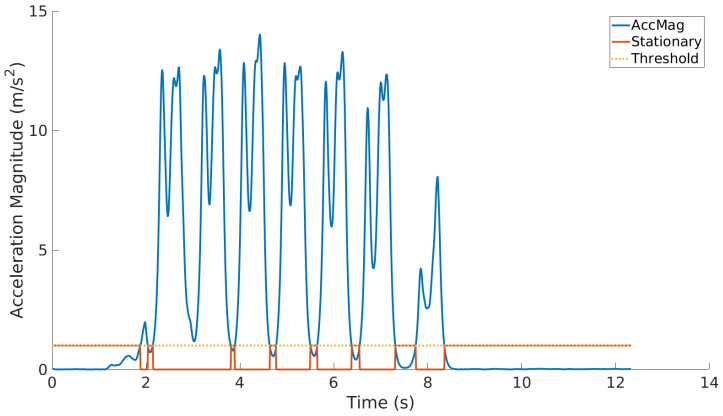


Figure 18: Accelerometer magnitude

In the original FSM the first transition was activated by the detection of a maximum. In this case, the initial transition was activated by the detection of a minimum. This enables the detection of the first TO event which, in this context, was determined to be the first event at the initiation of gait. This provides a better detection of all events present throughout each complete trial.

4.2.4 Compute Sensor Orientation

For this stage, the gyroscope and accelerometer signals are all used to compute the orientation sensor by using the orientation filter developed in Madgwick [54]. This filter was developed in MATLAB and consists of a class AHRS and several methods that belong to this same class.

Table 4: AHRS class

AHRS	
Public	
	q = [1 0 0 0]
	SamplingPeriod = 1/fs
	Kp = 1
Private	
	Quaternion = [1 0 0 0]
	IntError = [0 0 0]

The structure of the AHRS class is described in Table 4. The existing public variables that are used in this algorithm include the sampling period of the acquired data, the output quaternion array that contains

the sensor orientation relative to the earth frame on each sample and the proportional gain constant K_p. Two private variables are also used: the internal quaternion used in the computation of the orientation of the sensor and the integral error used to compute the integral feedback term.

The main method used from this library is represented in Algorithm 1. Firstly, the norm of the acceleration signal is computed and the method only resumes if it is non-zero in which case it aborts the operation.

if $norm(Accelerometer) == 0$ **then**
  $return$
else
  $Accelerometer = \frac{Accelerometer}{norm(Accelerometer)}$
end
$v = [2(q_2q_4 - q_1 * q_3) \quad 2(q_1q_2 - q_3 * q_4) \quad (q_1^2 - q_2^2 - q_3^2 + q_4^2)]$
$error = v \otimes Accelerometer$
$Ref = Gyroscope - K_p error$
$pDot = \frac{1}{2} \times quaternProd(q, \ [0 \ Ref_1 \ Ref_2 \ Ref_3])$
$q = q + pDot \times SamplePeriod$
$q = \frac{q}{norm(q)}$
$Quaternion = q$

Algorithm 1: Quaternion representation of orientation

Afterwards, the estimated direction of gravity v, computed with the initial value of the internal quaternion, is used in the cross product with the acceleration signal obtaining the error between the estimation and the measurement of the orientation of gravity. The product of this error with K_p corresponds to the proportional feedback term and is subtracted to the gyroscope signal obtaining ${}^S\omega_t$. This result is then used to compute the rate of change through Equation 12 and integrated using equation 13 to obtain the quaternion representation of the sensor orientation. This quaternion is then normalized and stored in the output quaternion.

4.2.4.1 Results

The position of the foot-mounted IMU used to acquired the data used in this dissertation was slightly different than the one used in the implementation of the orientation filter in Madgwick [54]. Instead of making alterations in the developed software, the axis of each sensor were adapted to fit the input data requirements for this filter. The acceleration, angular velocity, filtered total acceleration magnitude signal and the signal for zero-velocity detection for one foot are shown in Figure 42, in Appendix A.2.

The orientation filter computes these signals and returns a quaternion representation of the sensor's orientation relative to the earth frame. The resulting quaternion signal has four components, as stated in

Equation 1. These four components are present in the graphs in Figure 43 in Appendix A.2, with q_0 being the real component and q_1, q_2 and q_3 the imaginary components.

Knowing that a quaternion is a four dimensional complex number, the visualization of the orientation of the sensor can be difficult. Still, there are some details related to the sensor's orientation that can be found. For example, the component q_2 waveform is very similar to a derivative of the y-axis angular velocity of the foot. To prove this statement, both signals, q_2 and the foot's y-axis angular velocity were normalized and compared in Figure 19. As can be seen, q_2 accurately describes the rate of change of the foot's angular velocity, which corresponds to the sensor's orientation in the y-axis.

With the orientation of each sensor in quaternion form, it is possible to compute the acceleration and orientation relative to the earth frame through Equations 7 to 10, respectively. The resulting earth frame accelerations and the sensor orientation in Euler angles are shown in Figure 45 in Appendix A.2.

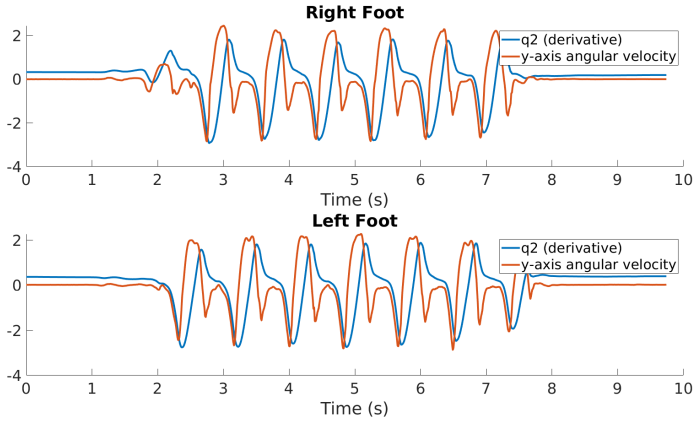


Figure 19: Derivative of angular velocity.

The results that are shown as example here are all from data acquired from foot-mounted IMUs but this process was applied to all other IMUs as well (shank, thigh and trunk). The x and y-axis component of acceleration is not zero due to the fact that the foot sensor is in an initial position were its z-axis is not parallel to the earth's gravitational field.

4.2.5 Compute Bio-Mechanical Signals

This subsection addresses the methods used to compute the necessary bio-mechanical signals for feature determination, as mentioned previously in Chapter 3. For this purpose, the signals of earth frame acceleration and orientation for each sensor, computed in the previous stage, are used to determine the velocity and position signals, as well as the angles of each joint.

4.2.5.1 Velocity and Position

With the acceleration of the foot in the earth frame, the velocity of the foot was computed by integration with integration step (Δt) equal to 1, as seen in Equation 15.

$$v_t = v_{t-1} + a_t \Delta t \tag{15}$$

With integration, any small existing offsets or errors in the acceleration measurements will be summed throughout the integration process, which can greatly increase the error of the resulting signal. This increasing deviation in a signal is known as drift and can be quite unpredictable and difficult to mitigate. From Figure 20, it can be seen that every velocity component is affected by an increasing deviation throughout time, with the x-axis velocity being the most affected signal.

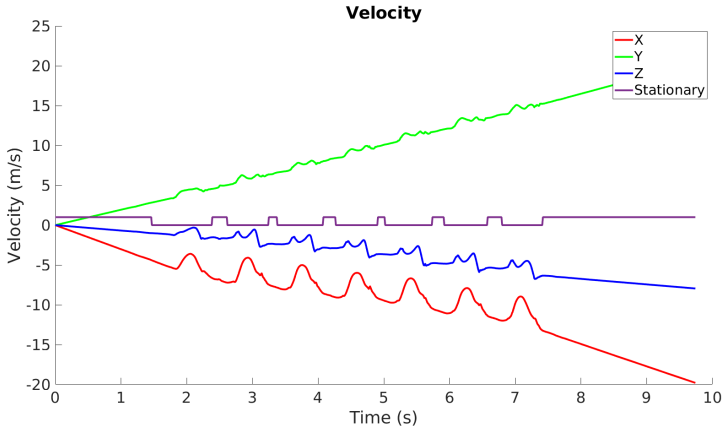


Figure 20: Velocity drift

It is known that during the FF event the foot is stationary so ZUPT can be performed. This will significantly minimize drift. This method of drift correction makes use of the previously discussed detection of stationary phases of gait. In Algorithm 2 the variable $stationary$ is a boolean array which is equal to 0 if the foot is in motion and 1 if it is stationary in which case the velocity is set to zero. This will significantly reduce drift which can be seen in Figure 21

while $i < length(vel)$ **do**
  $v(i) = v(i-1) + a(i) \times SamplingPeriod$
  if $stationary(i) == 1$ **then**
    $v(i) = 0$
  end
end

Algorithm 2: Zero Velocity Update

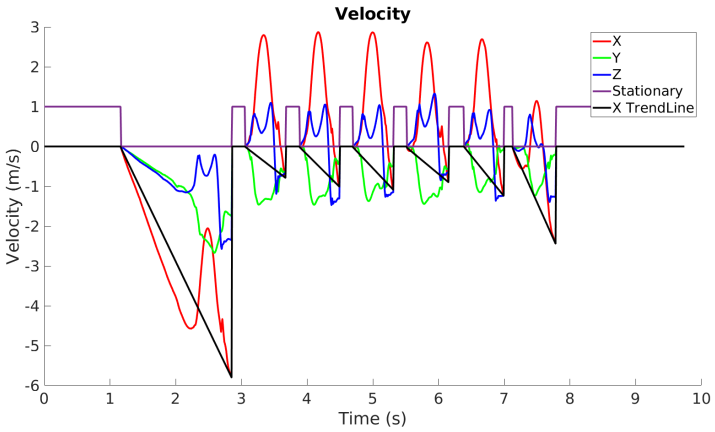


Figure 21: Improved velocity

Figure 21 shows a great improvement in the velocity signal however, some drift still exists during foot movement. To correct the existing drift Algorithm 3 is used.

Data: $statStart \quad statEnd$
for $i < length(statEnd)$ **do**
  $slope = \frac{(vel(statEnd(i)-1)-vel(statStart(i)))}{(statEnd(i)-1-statStart(i))}$
  for $(j = 1) < (statEnd(i) - statStart(i))$ **do**
    $enum(j) = j$
  end
  $drift = [enum^T \times slope]$
  $b = vel(statStart(i)) - slope$
  for $(j = statStart(i)) < (statEnd(i) - 1)$ **do**
    $velDrift(j) = drift + b$
  end
end
$vel = vel - velDrift$

Algorithm 3: Drift Correction

This algorithm stores each initial and final instant during each stationary phase in arrays *StatStart* and $StatEnd$ respectively. The time between the end of stationary phase and the beginning of the next represents the time during which the foot is in motion. This algorithm creates a linear trend line during this motion phase and subtracts it from the velocity signal thus further improving the signal. The improvements can be seen in Figure 22.

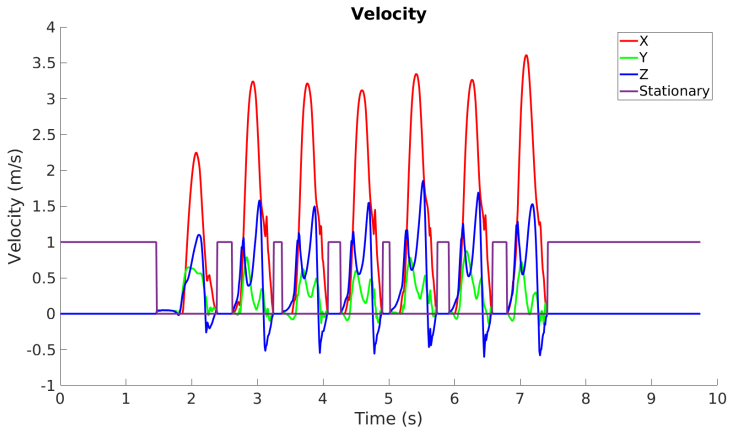


Figure 22: Corrected velocity

The position signal was obtained by integrating velocity through Equation 16, with integration step equal to 1. Due to this operation, the resulting signal will, once again, be affected by drift. In this case, it is not possible to apply ZUPT methods to all position components so it is of great importance to optimize the velocity signal as much as possible before taking this step.

$$p_t = p_{t-1} + v_t \Delta t \tag{16}$$

The computed position signals have little drift however, this drift cannot be eliminated due to the fact that none of these signals are expected to periodically reach 0, except for the z-axis component. When the foot is stationary the vertical position can be considered to be zero which can help reduce any existing drift in the z-axis signal using Algorithm 3. The improvements in the z-axis position signal can be seen in Figures 23 and 24.

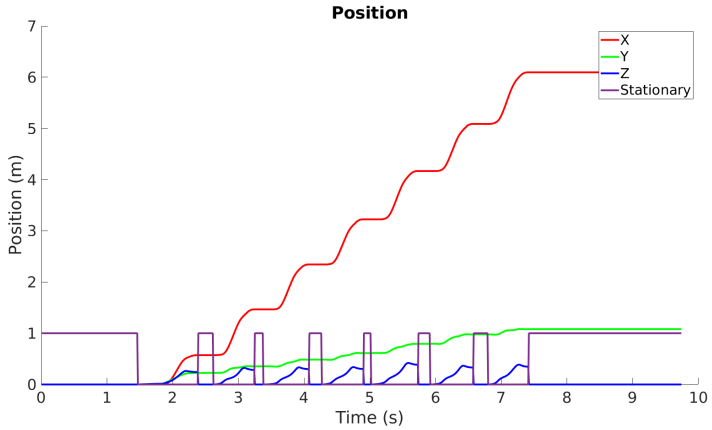


Figure 23: Position with drift.

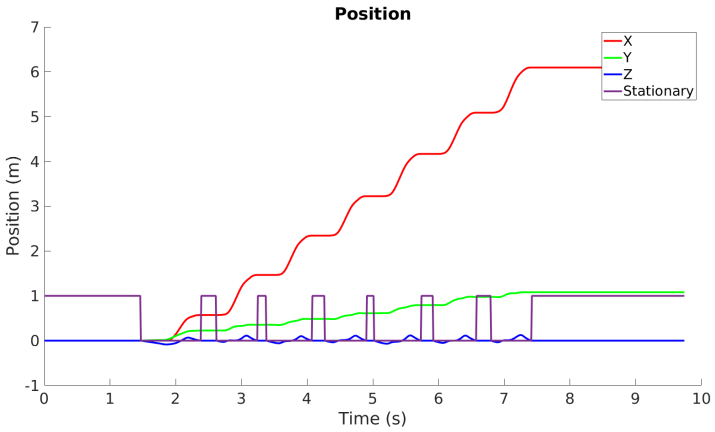


Figure 24: Position with corrected drift in the z-axis.

4.2.5.2 Joint Angles

The Euler angles representation in Figure 45a corresponds to the sensor's orientation relative to the earth frame. These measurements will be used to compute the values of all lower joints' angles in the sagittal plane for this will be a more useful signal component for the stage of feature determination.

Before the conversion to joint angles, the orientation of each sensor when the subject is at rest has to be in accordance to the conventions specified in the system InertialLab shown in Figure 25. It can be noted right away is that the joint signals in 45a need to be inverted. Besides this, it is also necessary to adjust the offset of each signal so that it's initial position is in accordance with Figure 25.

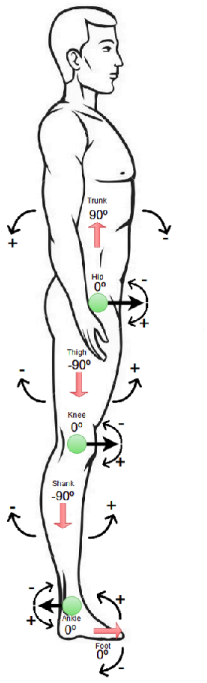
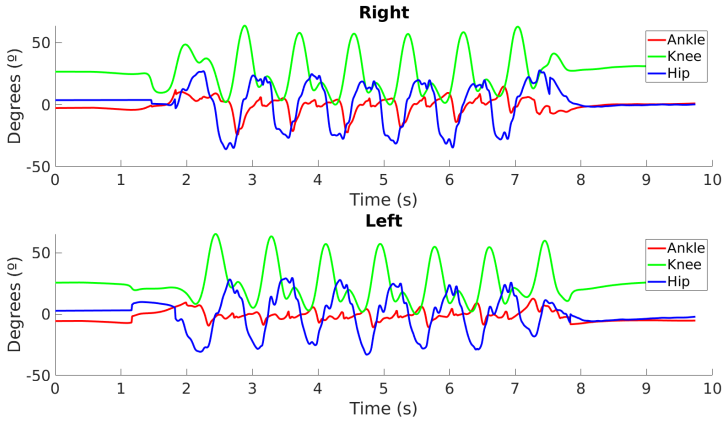


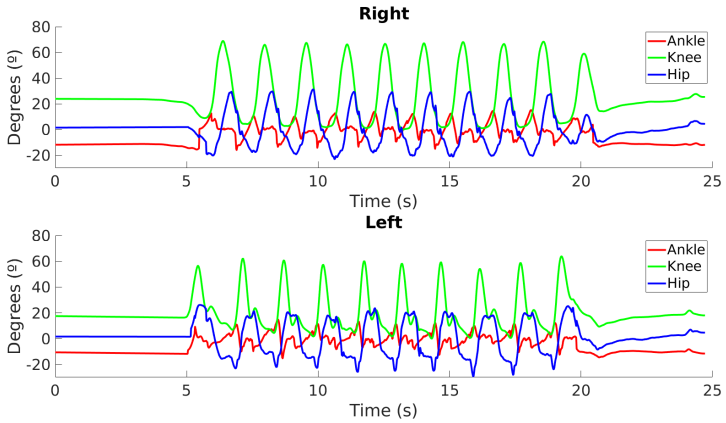
Figure 25: InertialLab conventions.

$$
\begin{aligned}
trunk &= trunk_{y-imu} - 90 \\
Rhip &= -(trunk_{y-imu} - Rthigh_{y-imu} - 180) \\
Lhip &= -(trunk_{y-imu} - Lthigh_{y-imu} - 180) \\
Rknee &= Rthigh_{y-imu} - Rshank_{y-imu} \\
Lknee &= Lthigh_{y-imu} - Lshank_{y-imu} \\
Rankle &= -90 - Rshank_{y-imu} + Rfoot_{y-imu} \\
Lankle &= -90 - Lshank_{y-imu} + Lfoot_{y-imu}
\end{aligned}
\tag{17}
$$

Afterwards, using Equation 17 it is possible to obtain the evolution of the angle of each joint over time. The resulting angle signals for each joint are present in the graph in Figure 26. The range of values of each healthy gait joint signal was compared to the ranges present in literature [56, 57] and all values seem to be within the ranges specified save for a few small deviation possibly caused by sensor drift.



(a) Healthy gait.



(b) Post-stroke gait.

Figure 26: Resulting joint angles.

4.2.6 Validation

Of all tests performed with InertialLab, some were performed with both this system and Xsens, which is a commercial system very similar to InertialLab. Xsens provides not only the basic sensor data that InertialLab also provides, but additionally also provides the quaternions of the orientation of each sensor and the joint angles as well. The data from Xsens can then function as a ground truth which will enable the comparison of the output signals obtained in this tool and the equivalent signals provided by Xsens.

Several trials were performed with a specific post-stroke subject (paretic right side) with both systems synchronized. The quaternion data obtained from one of those trials is shown in Figures 46 to 49, in Appendix A.2. The similarity between each component of different systems will be given by the corresponding Root Mean Square Error (RMSE) value, computed using Equation 18.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(y - y_{ref})^2}{N}} \tag{18}$$

The Xsens signals of the right foot experience an inversion of each signal when the foot is in motion, this occurrence was found in the majority of trials performed with Xsens. Appart from this, the q_0 components of the left foot show a significantly low RMSE value (0.0058), and seem to be very similar to each other, despite a slight offset between them at the beginning.

Figure 47 shows the q_1 component of each systems quaternions. In both cases the resulting RMSE values are below 0.08. While not as close to each other as the q_0 component of the left foot, each system's q_1 component seems to follow the same trend.

In Figure 48, the issue experience in Figure 46 is very obvious here. When the right foot is in motion, the q_2 component of the Xsens system is inverted, affecting the resulting RMSE results. Apart from this, there is a slight offset between each system's signals which will also increase the resulting RMSE. The left foot has a RMSE value close to 0.092. This value is acceptable for a signal that varies between -0.5 and 0.2. It can also be noted that both signals of both feet are very close to each other in terms of periodicity.

In Figure 49, the same inversion issue can still be observe, although on a smaller scale. The signals from the left foot, despite lokking similar, have a very large offset, raising the RMSE to 0.2151. The resulting RMSE values can also be seen in Table 5.

There were some considerable differences between the right foot's signals for each system due to the inversion in Xsens's quaternion when the foot was in motion. However, the left foot's quaternions seem to be in accordance with each other, apart from the offset between both q_3 components.

Table 5: RMSE values

Component	**Right Foot**	**Left Foot**
q_0	0.34294	0.0058315
q_1	0.072859	0.038341
q_2	0.24855	0.091707
q_3	0.082004	0.2151

5

FEATURE DETERMINATION

The performance of machine learning systems is highly dependent on the quality and processing/interpretation of available data. Therefore, a crucial step in the development of any classification system, besides a correct and robust data acquisition is good feature engineering. This is the process of using domain knowledge of the data to create an optimal feature set that best enhances a classifier/predictor's performance. To this end, it is preferable to have a majority of relevant features and as few irrelevant or redundant features as possible. This procedure is not perfect however, no matter how good the created feature set is, there will always exist some irrelevant or redundant information that can hinder classification performance. To mitigate these difficulties, dimensionality reduction methods can be employed to optimize the feature set. This is not necessary in DL methods for they already implicitly perform feature extraction.

Two types of feature set were created: a sequential feature set for two sequential classification models and a non-sequential feature set for all other non-sequential classification models. This chapter will address the steps taken throughout the creation of each feature set.

5.1 Non-Sequential Feature set

This is the most common type of feature set, for each training example there is one label associated with it. The structure of a non-sequential feature set for supervised learning is present in Table 6 below.

Table 6: Tabular feature set

X_1	X_2	X_3	...	X_n	Y
x_1^1	x_2^1	x_3^1	...	x_n^1	y^1
x_1^2	x_2^2	x_3^2	...	x_n^2	y^2
x_1^3	x_2^3	x_3^3	...	x_n^3	y^3
...	...	...	...	...	...
x_1^m	x_2^m	x_3^m	...	x_n^m	y^m

In this example, m and n represent the number of training examples and dimensionality of the feature set respectively. Each x_n columns represents the training values of a single feature. The last column, y_n, represents the labels of each training example. Since the aim of this work is gait pattern recognition, in this context the values of y_n correspond to class labels (healthy or stroke). Additionally, the features on each training example consist of gait parameters computed for each gait cycle.

5.1.1 Spatial Features

Spatial features are calculated parameters that can describe the spatial behavior of gait and consist of: Stride Length, Step Length, Stride Velocity and Foot Clearance. Table 7 shows the definitions of each of these features, adapted from Figueiredo et al. [56] These feature were chosen, not only due to their use in literature, but also because they will possibly provide a good way to identify several symptoms. Stride, Step Length and Stride velocity will generally decrease in pathological gait. In the case of post-stroke gait, these features will be able to show the existing difference between paretic and healthy legs during gait. Additionally, Foot Clearance of the paretic foot will be lower due to the symptom known as drop foot, in which the post-stroke subject has lower ankle control of the paretic foot.

Table 7: Spatial features.

Feature	**Definition**
Stride Length	Distance between successive HS events of the same foot
Step Length	Distance between successive MMSW events of alternating feet
Stride Velocity	Average speed of a single stride
Foot Clearance	Maximum height of the foot during each gait cycle

The calculation of Step Length usually consists of the distance between successive HS events of alternating feet. The problem with this definition however, is that in practice, this method will be prone to error. The x-axis position signal from where Step Length is computed was obtained from a double integration and although the drift of all signals was compensated, it cannot be completely eliminated. The result of this effect is shown in Figure 27.

Both signals should be "intertwined" with one another due to the fact that at the end of each gait trial, both feet are in the same x-axis position. Due to the effect of drift, these signals are slowly drifting apart from each other. This will increase the error associated with the calculation of Step Length for this feature is dependent of events from both legs. Due to this deviation and the uncertainty in the location of HS on each position signal, the value of Step Length will tend to zero on one leg and to the same value as the Stride Length on the other throughout each trial. To try and minimize this error, the event used to compute

this feature was MMSW for this is the easiest event to detect thus allowing for a more precise location along each gait cycle. For other spatial features, this procedure was not necessary for none of them are dependent on data from both feet.

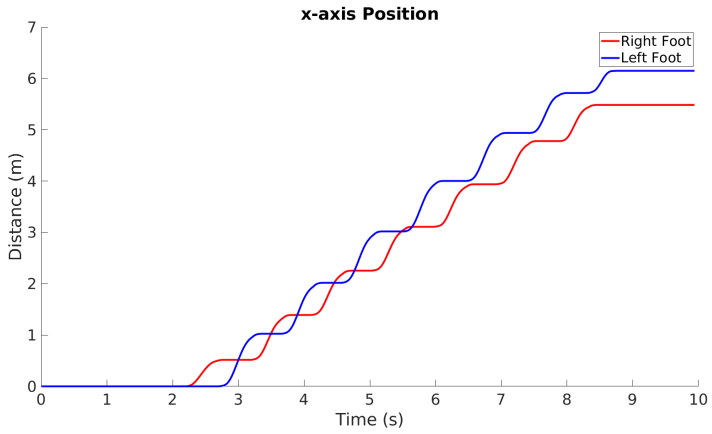


Figure 27: Drift between position signals.

In Appendix A.3, Table 22 shows some of the feature values obtained for the trials performed with healthy subjects for normal walking speeds. These features were obtained with some of the bio-mechanical signals previously calculated, namely the velocity and position signals of each foot mounted IMU. Table 23 also shows the values of the same features for trials performed with post-stroke subjects with a speed of 2 km/h. It is possible to se that some features present lower values for the paretic leg, subject S1 shows this behavior clearly in the values of Stride Length, Step Length and Stride Velocity.

5.1.2 Temporal Features

These features characterize the temporal behavior of gait and consist of the features in Table 8, most of which were also adapted from Figueiredo et al. [56] Single support refers to the part of a gait cycle were the subject only has one foot in contact with the floor while the other is in swing phase. Double support refers to the moments in a gait cycle were both feet are in contact with the floor. The stance phase corresponds to the percentage of each leg's gait cycle where it is supporting the body's weight and occurs between HS and TO events. The swing phase corresponds to the percentage of each leg's gait cycle during which the foot is on the air for limb advancement. This phase occurs between TO and HS. This can be better visualized in Figure 28.

In post-stroke gait, Step and Stride Duration decreased and since the velocity of gait decreases as well, Strides Per Minute and Cadence will also decrease. Due to the lack of control on the paretic leg, the subject

will not be able to spend much time with that leg in single support phase, so the duration will be smaller as well as the percentage that this leg spends in Stance Phase.

Table 8: Temporal features.

Feature	Concept
Stride Duration	Time between successive HS events of the same foot
Stride Per Minute	Number of strides per minute
Step Duration	Time between successive HS events of alternating feet
Cadence	Number of steps per minute
Single Support	Percentage spent in single support phase
Double Support	Percentage spent in double support phase
Per Stance	Percentage of gait cycle spent in stance phase
Per Swing	Percentage of gait cycle spent in swing phase

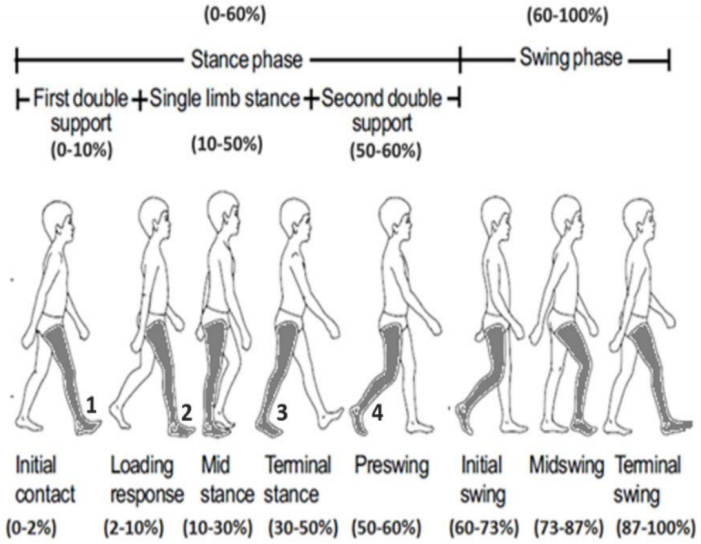


Figure 28: Gait phases, adapted from Figueiredo et al. [56]

Table 24 shows the resulting values of the temporal feature for three healthy subjects walking at a normal, self-selected comfortable speed. These features were determined through the gait events previously calculated, which were determined by using the gait event detection FSM with the foot's y-axis angular velocity as input. Table 25 shows the same results for post-stroke patients in 2 km/h trials, both tables are shown in Appendix A.3. This feature determination process is not flawless however, some features like Double Support take on negative values with some subjects, which is clearly not possible.

5.1.3 Kinematic Features

Kinematic features are usually computed from joint angle signals. The features present in Table 9 were also adapted from Figueiredo et al. [56] and were computed for each joint, both ankles, knee and hip joints. This will make up a total of 30 kinematic features.

Table 9: Kinematic features.

Feature	**Concept**
Peak of Flexion	Maximum angle during joint flexion
Peak of Extension	Minimum angle during joint extension
Range of Motion	Total range of movement of the joint in degrees
Angle of initial contact	Joint angle during HS
Angle of end contact	Joint angle during TO

Much like spatial features, these features were computed from signals that were the result of several operation on the original IMU signals so they can be negatively affected by the existence of signal drift. To verify the quality of these signals, their amplitude will be compared to the range of motion of each joint found in literature. The ranges for each joint found in Figueiredo et al. [56] are shown in Table 10, these values are merely approximations.

Table 10: Joint ranges of motion.

Joint	**Range of motion (Degrees)**
Hip	$-20 \leftrightarrow 30$
Knee	$0 \leftrightarrow 50$
Ankle	$-20 \leftrightarrow 10$

The values obtained for the features in Table 9 are present in Tables 26 and 27, which can be found in Appendix A.3. These features were determined with the use of the previously computed joint angles.

Figures 50, 51 and 52 in Appendix A.3, show all joint angles for a specific trial of a healthy and a post-stroke subject with a paretic right leg. Along with these signal are also represented as dashed horizontal lines, the maximum/minimum values for each joint found in literature.

In post-stroke gait, the range of motion of the both hip joints decreased while the range of motion of the knee joints is slightly increased, possibly as a form of compensation during gait. This differences in these kinematic features can greatly aid in the classification process.

Overall, despite some minor deviations, all joint angles appear to be in concordance with the values found in literature. By inspecting every trial, this seems to be the general case so no corrections are necessary.

5.1.4 Other Features

Besides the ones discussed already, some additional features were selected as well. The asymmetry of gait was computed and several synergies were calculated from joint angle signals to determine the contribution of each joint to the whole bio-mechanical chain of movement of each leg.

5.1.4.1 Gait Asymmetry

Gait asymmetry measures the percentage of symmetry in a subject's gait. This will enhance the differentiation of healthy and post-stroke gait pattern, especially due to the fact that the duration of the paretic leg's step is significantly smaller than the opposite leg. This feature was computed using the previously mentioned Step Duration present in Table 8 by using Equation 19, where *Rstep* and *Lstep* are the right and left step durations, respectively.

$$Asymmetry \leftarrow (1 - \frac{Rstep}{Lstep}) \times 100 \tag{19}$$

5.1.4.2 Synergies

A muscle synergy consists in the activation of a group of muscles contributing to a specific movement. A single muscle can be part of multiple muscle synergies, and a single synergy can activate various muscles [58]. Through factor analysis, like non-negative matrix factorization or PCA, it is possible to identify muscle synergies.

In this dissertation, the objective is to use non-negative matrix factorization to compute the joint synergies of each leg in order to identify the contribution that each joint makes during gait. The work done in Cunha et al. [59] makes use of a method of orthogonal non-negative matrix factorization proposed in Choi [60]. For a non-negative matrix V, this method finds non-negative matrix factors W and H such that:

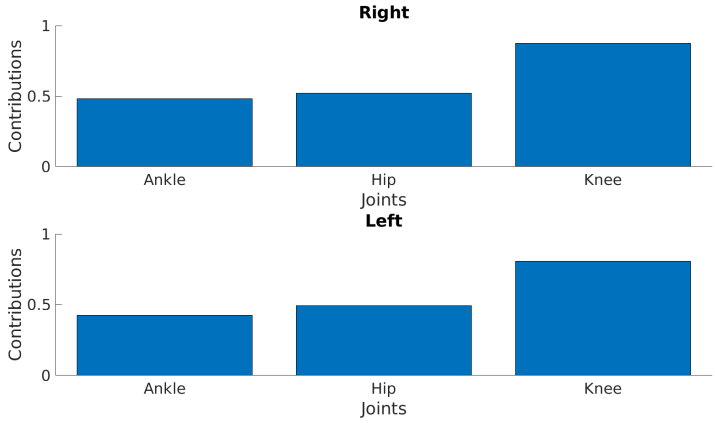
$$V_{N\times M} \approx W_{N\times K} H_{K\times M} \tag{20}$$

Where $V_{N\times M}$ is the non-negative input matrix with dimensionality N and number of samples M. $W_{N\times K}$ and $H_{K\times M}$ are respectively, the time-invariant activation profile and the time-variant activation coeficients of each synergy with k being the number of synergies. This method was used to extract synergies

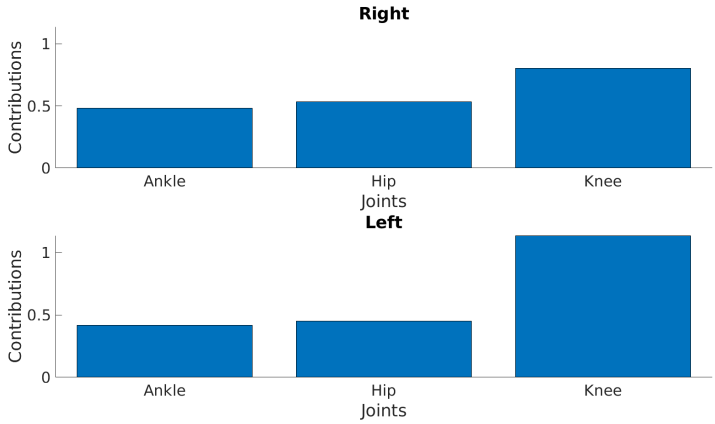
from all joint angles of each leg. In this context, V would be a matrix composed of all joint angle signals of one leg. In this case the dimensionality N, would be equal to 3 and the number of samples of V would be the same as the joint signals. Choi [60] states that K should be chosen to be smaller than N or M, so it's value was chosen to be 1 thus obtaining one synergy component.

Figure 29 shows the contribution of each joint for each corresponding leg. From the bar graphs it can be seen that the joint that most contributes to gait is the knee, the contribution of the other two joints remains relatively balanced. In post-stroke gait, the right knee contribution is noticeably lower thus implying that the subject possesses less control over this joint. Additionally, the contribution of the left healthy knee joint is higher than in the case of healthy gait, probably as a form of compensation.

This synergy extraction was performed for every gait cycle and , for the purpose of this document, the graphs present the average of those results, providing a more trustworthy analysis.



(a) Synergies for healthy gait.



(b) Synergies for post-stroke gait (paretic right leg).

Figure 29: Synergy of joint angles of each leg.

5.1.5 Feature Set

The resulting final feature set is a 1504×62 matrix, where each line corresponds to a gait cycle and each column is associated to a feature, with the last column being made up of the class labels of each trial. There are a total of 61 features: 4 spatial and 8 temporal features for each leg, 5 kinematic features for each joint, on feature related to gait asymmetry and 1 synergy for each leg. With the previous correction of gait events, some gait cycles were discarded. This will cause the calculation of some features to take into account more than one gait cycle. To prevent this, all feature values from a specific trial are averaged and any value that exceeds two standard deviations from the mean, will be discarded. The data extraction tool will extract one feature set per trial and will store it in its corresponding trial directory in the created database. Afterwards, when each feature set is extracted by the classification tool, the final full feature set is created and stored in the tool, for later use.

5.2 Sequential Feature set

In a sequential feature set each line also corresponds to a training example however, each feature is in itself a sequence of values. This means that instead of a classification label being associated with a set of values, it will be associated with a set of sequences as shown in Table 11.

Table 11: Sequential feature set.

X_1	X_2	X_3	...	X_n	Y
$x_1^1(1)$	$x_2^1(1)$	$x_3^1(1)$		$x_n^1(1)$	
$x_1^1(2)$	$x_2^1(2)$	$x_3^1(2)$	...	$x_n^1(2)$	y^1
...	...	...		...	
$x_1^1(k)$	$x_2^1(k)$	$x_3^1(k)$		$x_n^1(k)$	
$x_1^2(1)$	$x_2^2(1)$	$x_3^2(1)$		$x_n^2(1)$	
$x_1^2(2)$	$x_2^2(2)$	$x_3^2(2)$	...	$x_n^2(2)$	y^2
...	...	...		...	
$x_1^2(k)$	$x_2^2(k)$	$x_3^2(k)$		$x_n^2(k)$	
...	...	...	...	...	...
$x_1^n(1)$	$x_2^n(1)$	$x_3^n(1)$		$x_3^n(1)$	
$x_1^n(2)$	$x_2^n(2)$	$x_3^n(2)$	...	$x_3^n(2)$	y^n
...	...	...		...	
$x_1^n(k)$	$x_2^n(k)$	$x_3^n(k)$		$x_3^n(k)$	

5.2.1 Gait Segmentation

To build this feature set, the signals from each trial will be segmented into several gait cycles. In this feature set, each segmented sequence will correspond to a feature and each training set will consist of a gait cycle of the respective sequence/feature. To perform signal segmentation, the previously detected HS events were used to separate each signal in to several gait cycles were each cycle begins and ends with a HS event.

The classification models that will require this type of feature set are all DL models so dimensionality reduction methods will not be necessary. The signals used to create sequential features will be the InertialLab, quaternion and joint signals, as well as the time-variant activation coefficients of each synergy obtained in matrix H when previously extracting synergies from joint angle data. The signals used for the construction of this feature set are addressed in the following paragraphs.

InertialLab Signals The data acquired from each sensor will be used in this feature set, after segmentation. The used sensor system has 7 IMU sensors that each have a three dimensional accelerometer and gyroscope so there will be 6 signals from each IMU making up a total of 21 acceleration signals and 21 angular velocity signals and 42 InertialLab sequential features.

Quaternions The quaternion representation of the orientation of each sensor is four dimensional, so it is composed of 4 signals that can also be used as sequential features. The data extraction tool computes the earth frame orientation for each sensor, so a quaternion signal is obtained from each one. These signals will make up 28 sequential features from the total feature set.

Earth Frame Acceleration, Velocity and Position From the quaternion representation of the foot-mounted IMU's orientation, the acceleration, velocity and position were determined. These measurements were computed in the x, y and z-axis adding 9 sequential features per foot to the total feature set.

Joint Angles Additionally, the quaternion signals were also used to calculate the orientation of each sensor relative to the earth frame in Euler angles. From these measurements, the angle of each leg's joint is determined according to Equation 17. With the angles of all six joints, and also adding the trunk's orientation, 7 more sequential features are added.

Synergies The signals in the time-variant activation coefficient matrix H, obtained in the previous synergy extraction can be utilized as a feature as well. This matrix has dimensions $K \times M$, being K the synergy extracted from each leg. This makes up 2 sequential features.

5.2.2 Feature Set

In total the final feature set has 73 sequential features with 1189 gait cycles, each corresponding to a training example. This feature set was stored per trial in the database, for posterior extraction by the classification tool.

6

GAIT PATTERN RECOGNITION

This chapter is related to the development and validation of the classification tool used in distinguishing between healthy and post-stroke gait patterns. This tool was originally developed in Gonçalves [41] with four classification models: SVM, kNN, RF and DA. This dissertation also contributes with four more classifiers: FFNN, CNN, LSTM and C-LSTM. The available classification tool is initially described in more detail before addressing the development process of each NN added to this tool. The testing protocol will also be described as well as the test performance of each model, followed by a critical analysis of the results obtained.

6.1 Previous Classification Tool

The previous classification tool, developed in Gonçalves [41], can be seen as a system of three stages: Normalization, dimensionality reduction and performance estimation. Each method available in each stage will be briefly discussed in this section.

6.1.1 Normalization

Several features in a dataset are going to have different scales and different ranges of values. Some ranges can be quite different from one another for example, some features can vary between 0 and 1 while others can vary between 0 and 1000. This can be detrimental to classification performance, so data normalization can be of great benefit. In this context, data normalization can be seen as a rescaling of data in order to boost classification performance. The available methods for normalization in the classification tool are centering, z-score and min-max. In the context of this dissertation, the chosen normalization method was the min-max method.

This scaling method involves converting data into a range of values. This is usually used for algorithms that only work with data on a certain range of values like neural networks. For every feature, the minimum value of that feature can be transformed into a 0 and the maximum value transformed into a 1, and every other value gets transformed into a decimal between 0 and 1. This is merely an example for the min-max range can be defined by the user. The mathematical formula for this normalization method is represented in Equation 21.

$$x' = \frac{x - min(x)}{max(x) - min(x)}(b - a) + a \tag{21}$$

Here, x is the original variable value, $min(x)$ is the variable's minimum value and $max(x)$ is it's max value, a is the new minimum, b is the new maximum and x' is the normalized value.

6.1.2 Dimensionality Reduction Methods

In this tool there are a few dimensionality reduction methods available for selection: GA, Sequential Feature Selection, PCA, mRMR, and ANOVA. From all these, only the last three methods will be used due to the fact that the first two are very time-consuming to be using with all classifiers due to the fact that the first two methods actually build the model for each feature subset to evaluate the changes in performance. The implementation of each method was obtained from a MATLAB Toolbox called Feature Selection Library [61, 62, 63, 64].

Minimum Redundancy Maximum Relevance: This method selects features that are maximally dissimilar to each other creating a ranking that is immune to redundant features. Despite not being as heavy as a GA or sequential feature selection, it is still computationally expensive.

Principal Component Analysis: Converts the original feature set into a new set of linearly uncorrelated features called principal components. This results in an uncorrelated and orthogonal feature set. PCA is sensitive to the relative scaling of the original variables so data normalization is essential for this method to provide good results.

ANOVA: Test used in statistics that is used to determine the influence that independent variables have on the dependent variable in a regression study. The implementation of this method in this tool only selects as many features as the number of classes in the dataset. In this context, this method will only select two features so there is a possibility that this will hinder the performance of the classifier.

6.1.3 Classifiers

Four classification models are already included in this tool (SVM, kNN, DA and RF) and are briefly explained in this section and their performance will later be analyzed as well.

Support Vector Machines: Given labeled training data, outputs an optimal hyperplane which categorizes new examples. It can also be used with different kernels: linear, polynomial and gaussian among others. These project the data onto a new feature space to make the classes more separable [65].

k-Nearest Neighbor: One of the simplest classification algorithms, this model simply stores every train instance so that during classification it can compute the closest instances to the data point that is to be classified (the number of closest neighbors to which the point is compared to, designated k is defined by the user or by parameter tuning) and by using a majority vote on the neighbors' classes it obtains the new point class [41].

Discriminant Analysis: This model finds a combination of features of a certain order (linear, quadratic, cubic...) that best separates the data based on two or more classes of a response variable. Like the PCA principle, it does this by attributing coefficients to each of the original features to obtain a hyperplane that divides the data according to its classes [66].

Random Forests: These models are based on the concept of decision trees, rule-based classifiers that use the feature values to create a tree of decisions where the last step is the classification result. Random forests are an ensemble model constituted from a number of these decision trees built from random subsets of data and features. The number of decision trees is specified by the user [67].

6.1.4 Performance Estimation Methods

There are several ways to train and estimate the performance of a certain model. This is done to have an idea of how well the model will be able to deal with new, unseen data. The performance evaluation methods available in this tool are Hold Out, Resubstitution, Leave-One-Out (LOO) and K-Fold CV. This last method was the one used for evaluation performance. It is the most used and most recommended method for model validation. It operates by splitting the data into k folds, a user selected number, and training a model on all but one of the folds, which is used for testing. This is done until every fold was used for testing purposes. The performance is then obtained by averaging the results of each model. In the case when k is equal to the number of observations, this method is called LOO.

To reduce the inner variance that random data splitting may cause across folds it is recommended to use nested k-fold CV (averaging the results of each one of the procedures to get a good stable measurement of the model's performance). Stratification is also recommended meaning splits are created with the same proportion of each class label instances as the original dataset. Otherwise, some test sets may not include observations from all classes, especially in the case of a data set with an unbalanced classes.

For predictive performance estimation, LOO or nested 50 and 20-fold CV are best but for model selection, repeated 2-fold CV is best for identifying the best candidate model since it inflates the error rate of each model, allowing for a better comparison. It is suggested that the minimum number of repetitions should be 10 to 20 times [68]. This validation method gives a pessimistically biased estimate of performance, lower than the true value because most statistical models will improve if the training set is increased. It is important to note that nested CV validates the entire model building process (including feature selection) so the entire process must be done inside each CV fold for a better estimate of the model's generalization performance [69].

6.1.5 Performance Metrics

In order to evaluate the performance of a model and compare it to other models' performances, there are several metrics available. Most of these metrics can be extracted from the confusion matrix (which is also included in this section).

Confusion Matrix Most of performance metrics are obtained from a confusion matrix, with $M \times M$ dimension, where M is the number of classes to be predicted. For binary classification problems, there

are two classes, thus the confusion matrix will have two rows and columns. The rows of the confusion matrix represent the target classes while the columns represent the output classes. The diagonal cells in each table show the number of cases that were correctly classified and the off-diagonal cells show the miss-classified cases.

Once all the instances are classified, the predicted results are compared to the actual values of the target variables. Table 12 shows the existing four possibilities for binary classification:

- **True Positives (TPs):** instances that are positive and are classified as positives.
- **False Positives (FPs):** instances that are negative and are classified as positives.
- **False Negatives (FNs):** instances that are positive and are classified as negatives.
- **True Negatives (TNs):** instances that are negative and are classified as negatives.

Table 12: Confusion matrix example.

	Positive (predicted)	**Negative (predicted)**
Positive (label)	TP	FN
Negative (label)	FP	TN

Some of the most used metrics are Accuracy, Precision, Sensitivity, Specificity, F-1 score, MCC, Area Under Curve (AUC). All but the Area under Curve can be obtained from the confusion matrix. These metric chosen for model evaluation was the MCC score. This is a correlation-based metric that is widely used, and generally perceived as the best one, for cases where the number of instances of each class are very different (unbalanced) since under these cases other metrics yield improper values. This metric has some properties that facilitate its interpretation, it ranges from -1 to 1 corresponding to all wrong guesses or all correct guesses, respectively. When the value 0 is achieved it means half of the instances were correctly classified (random guesses). However, from this metric it's hard to discern exactly which class labels were wrongly classified the most. This makes MCC worse than accuracy for balanced cases. Its formula is expressed in Equation 22.

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{22}$$

6.1.6 Class Labeling

For this classification task, every dataset contains only two classes: healthy and post-stroke. The values attributed to each are shown in Table 13.

Table 13: Class labels.

Class	Label
Healthy	1
Post-stroke	2

6.2 Neural Network Implementation

This section details the implementation process of each of the four NN that were added to this tool. The first is a FFNN, which is the most popular NN in literature and the easiest one to implement as well. The other added networks are all DNN, that already implicitly perform feature extraction on its input data. All of the models were developed using MATLAB 2019b and its Deep Learning Toolbox.

6.2.1 Feed-forward Neural Network

Neural networks are a set of algorithms (perceptrons), modeled loosely after the neurons of the human brain and are designed to recognize patterns and be able to approximate almost any function, generating predictions for complex problems. In the case of FFNN, they are composed of several layers: an input layer, one or more hidden layers and an output layer. The structure of this type of network is present in Figure 30.

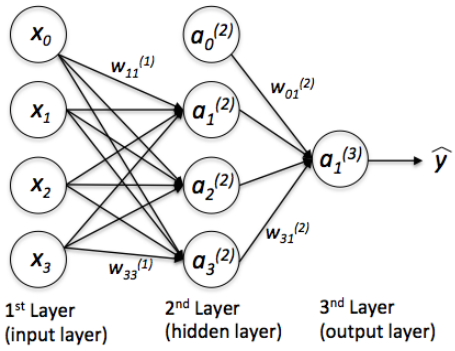


Figure 30: Feed-forward neural network [70].

Neurons x_0 to x_3 represent each feature of the input layer, $a_1^{(2)}$ to $a_3^{(2)}$ are the neurons of the hidden layer and $a_1^{(3)}$ is the output neuron of the third layer. $a_0^{(2)}$ is the associated bias neuron which is a special neuron added to each layer in the neural network, which simply stores the value of 1. This makes it possible to move or "translate" an activation function left or right on the graph, such as the sigmoid activation function in Figure 31.

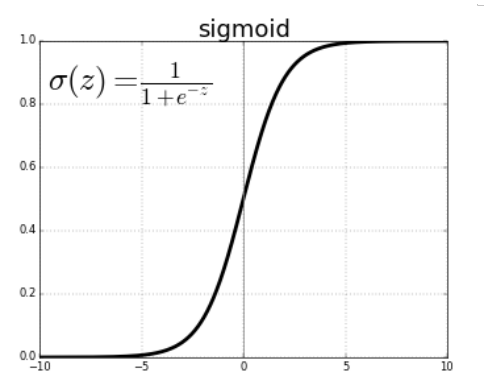


Figure 31: Sigmoid activation function [71].

Without a bias neuron, each neuron takes the input and multiplies it by a weight, with nothing else added to the equation. So, for example, it is not possible to input a value of 0 and output 2. In many cases, it is necessary to move the entire activation function to the left or right to generate the required output values—this is made possible by the bias.

The basic learning process of a neural network during a single epoch is as follows:

- **1:** Randomly initialize network parameters (weights and biases).
- **2:** Take a set of examples of input data and pass them through the network to obtain their prediction.
- **3:** Compare these predictions obtained with the values of expected labels and calculate the loss or classification error.
- **4:** Perform back-propagation in order to propagate this loss to each and every one of the parameters that make up the model of the neural network.
- **5:** Use this propagated information to update the parameters of the neural network with the gradient descent in a way that the total loss is reduced and a better model is obtained.
- **6:** Continue iterating in the previous steps until a good enough model is obtained.

During training, a neural network separates its training set into several mini-batches which the network processes and only updates itself (calculates the classification error) afterwards. The number of epochs

represents the amount of times that the network passes through the whole dataset, this means that in one epoch the network is able to pass through all mini-batches.

The FFNN for classification available in this toolbox allows the user to configure the number of existing hidden layers, how many neurons each layer has, which training function is used in the back-propagation stage and which function is used to access network performance.

The more hidden layers/neuron the network has, the more it is able to detect greater non-linear patterns. There is a limit to this however, according to Hagan et al. [72], for a network to generalize properly, the number of parameters should never surpass the number of data points in the training set otherwise the network will overfit. To avoid overfitting, the tool will test whether there are more parameters than training point, in which case, it will abort execution.

There are several available training functions in this toolbox however, by looking into the toolbox's documentation, the most advisable functions to use with a FFNN for classification are scaled conjugate gradient descent back-propagation and resilient back-propagation. The function that evaluates network performance will not be used in this tool, instead the previously mentioned metrics will be used for all models.

In order to implement this network the numbers in each class label had to be converted to binary base as shown in Table 14:

Table 14: Label conversion.

Regular Labels	**Binary Labels**
1 $(healthy)$	1 0
2 $(stroke)$	0 1

To further reduce network overfit, an early stopping method is used. For each epoch the data from the corresponding fold is split into a training and a validation set. After training, the network is evaluated on the validation set. Throughout each epoch, if the performance of the model on the validation set starts to degrade, the training process is stopped.

6.2.2 Convolutional Neural Network

A CNN is a type of neural network specialized in working with two-dimensional image data but they can also be used with one-dimensional and three-dimensional data. This network is comprised of one or more convolutional layers (often with a sub-sampling step) and then followed by one or more fully connected layers as in a standard FFNN. A layout of this network is shown in Figure 32.

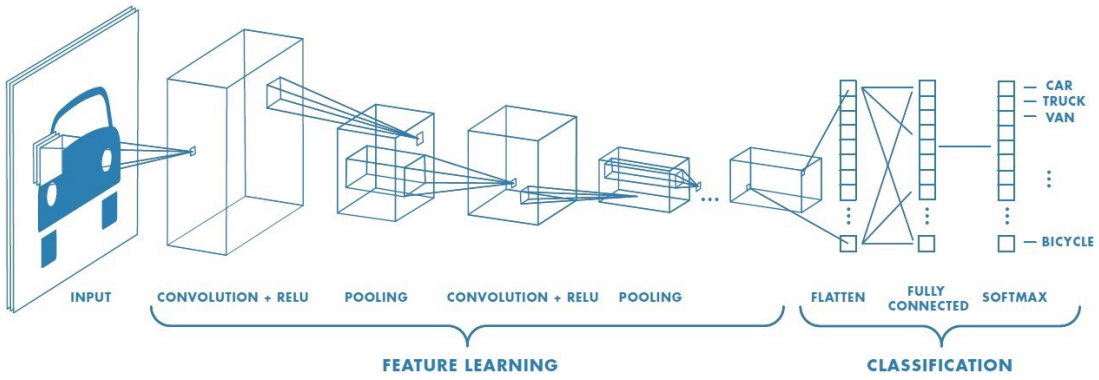


Figure 32: Convolutional neural network [73].

In the context of this neural network, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. However, given that the technique was designed for two-dimensional input, the multiplication is performed between a two-dimensional array of input data and a two-dimensional array of weights, called a filter or kernel. The kernel is smaller than the input data and the type of multiplication applied between a kernel-sized patch of the input and the kernel is a dot product. This process is exemplified in Figure 33.

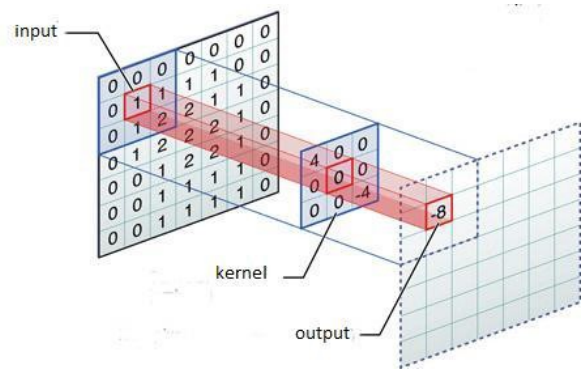


Figure 33: Convolutional kernel [74].

Using a kernel smaller than the input allows it to be multiplied by the input array multiple times at different points on the input. Specifically, the kernel is applied systematically to each overlapping part or kernel-sized patch of the input data, left to right, top to bottom. This systematic application of the same kernel across an image is a powerful idea. If the kernel is designed to detect a specific type of feature in the input, then the application of that kernel systematically across the entire input image allows the kernel an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance, which means that the general interest is in whether the feature is present rather than where it was present.

The output of this layer is a two-dimensional array of values that represent a filtering of the input. As such, the two-dimensional output array from this operation is called a “feature map“. Once a feature map

is created, it passes through a non-linear activation layer, such as a Rectified Linear Activation Unit (ReLU) layer. Other types of activation units, such as sigmoid or tahn units, can easily saturate and for deep networks the gradient can actually vanish. With ReLUs, this does not occur, despite looking and behaving as an almost linear function, it is in fact non-linear. The output of these units does not saturate and all negative values are forced to 0, which eliminates the vanishing gradient problem. The computational simplicity of these units also enables the creation of very deep networks.

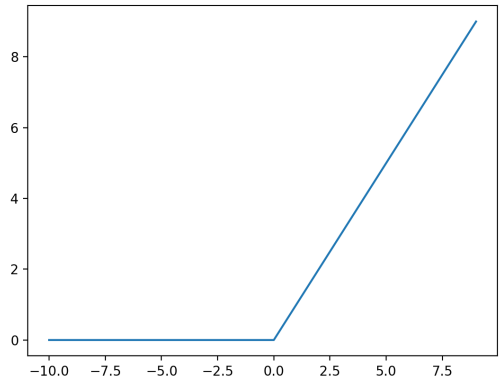


Figure 34: ReLU function [75].

At the output of a ReLU layer, is a pooling layer which operates upon each feature map separately to create a new set of the same number of pooled feature maps. This layer selects a pooling operation, much like a filter to be applied to feature maps, reducing the size of each feature map. This creates a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model's invariance to local translation. This three-layer pattern is commonly used for ordering layers within a convolutional neural network and may be repeated one or more times in a given model, thus being able of learning more abstract features.

For the purpose of classification a couple additional layers have to be added at the output. Firstly, a flatten layer is added to break the spatial structure of the image data into a one dimensional vector, to serve as input to a fully connected layer. This fully connected layer is the same type of layer that is found in FFNNs and its the layer that actually classifies the data, the previous convolutional groups of layers can be seen as feature extractors in this case. Lastly, the softmax layer uses a softmax activation function on its input to convert it to the range of 0 to 1, allowing it to be interpreted as a probability distribution were the sum of all outputs of this layer is equal to 1.

CNNs do not seem to be a common choice in the field of gait pattern classification. In this dissertation, the potential of this network will be explored in the context of gait pattern classification. Due to the nature of this network, the existing non-sequential feature set cannot be directly applied to the network. A restructuring of this feature set is therefore required. Each training example is converted to a single matrix that will then be converted to an image. To illustrate this, and example of a training set of 7 features is shown in Table 15.

Table 15: Non-sequential feature set example.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	Y
x_1^1	x_2^1	x_3^1	x_4^1	x_5^1	x_6^1	x_7^1	0
x_1^2	x_2^2	x_3^2	x_4^2	x_5^2	x_6^2	x_7^2	1

In this example, each training example is an array with a length of 7 features. These arrays will be converted to the smallest matrix that can fit all elements of the array. In this case it will be a matrix of 3×3 dimensions, with a total of 9 elements. The remaining elements will be padded with zeros. After conversion, each matrix will be associated with the same classification label of the corresponding training example. The result of this conversion is shown in Table 16.

Table 16: Image feature set example.

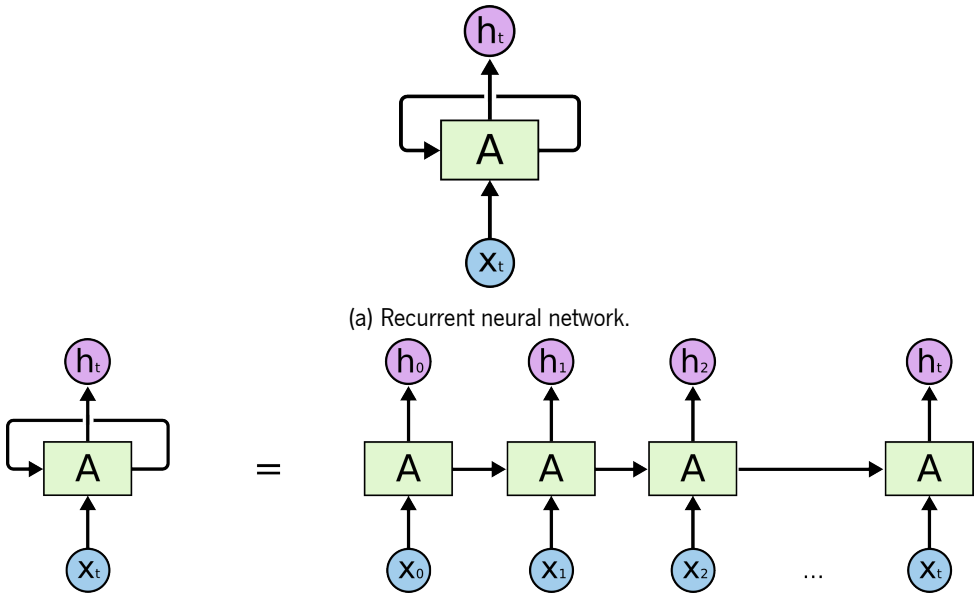
Image feature			Y
x_1^1	x_2^1	x_3^1	
x_4^1	x_5^1	x_6^1	0
x_7^1	0	0	
x_2^1	x_2^2	x_3^2	
x_4^2	x_5^2	x_6^2	1
x_7^2	0	0	

The input for the existing CNN in the DL toolbox needs to be a square matrix. In cases similar to this example, if the number of features is not enough to create a square matrix, zero padding can be used to fit the input requirements of the network.

Due to the fact that this is a DNN, no dimensionality reduction methods are necessary. Feature extraction is already performed implicitly, as shown in Figure 32. Additionally, to reduce possible overfit, a dropout layer could be added at the end of every pooling layer. This layer works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase, making the training process noisy and forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs, making the model more robust.

6.2.3 Long-Short Term Memory Neural Network

LSTMs are a type of RNN. RNNs, unlike FFNNs, have feedback connections which implies that they have memory by being able to remember past inputs.



(a) Recurrent neural network.

(b) Recurrent neural network through time.

Figure 35: RNN diagram [76].

LSTMs are designed to overcome the vanishing gradient problem and to retain information for longer periods compared to traditional RNNs. They can maintain a constant error, which allows them to continue learning over numerous time-steps and back-propagate through time and layers.

RNNs can be seen as a chain of repeating modules of neural network. In standard RNNs, each repeating module will have a very simple structure with only one layer, LSTMs however have a different structure in each module. Instead of a single neural network layer, there are four, as shown in Figure 36.

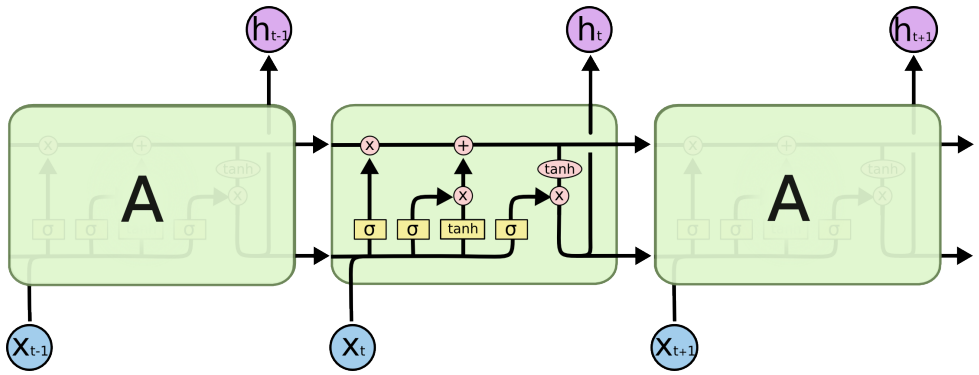


Figure 36: LSTM module [76].

The horizontal line running through the top of the diagram in Figure 37 is the cell state, which is like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged. This network has the ability to remove or add information to the cell state, carefully regulated by structures called gates. They are composed out of a sigmoid layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.

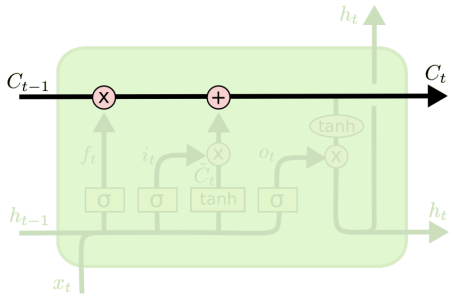
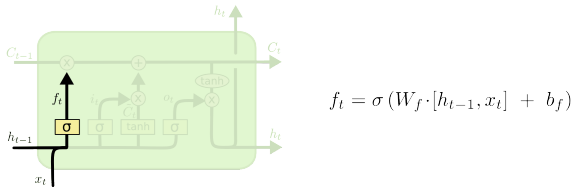
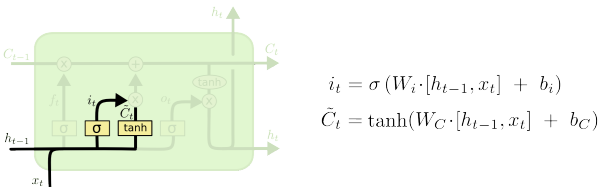


Figure 37: LSTM cell state [76].

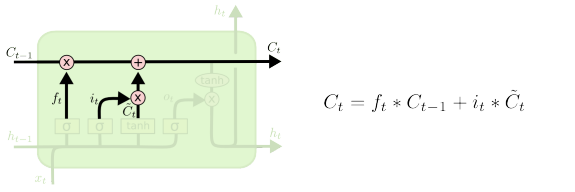
There are a total of three gates (Figure 38): forget, input and output gates. Figure 38a presents the forget gate, which decides what information will be thrown away from the cell state. By looking at the previous output, h_{t-1} and input x_t, the sigmoid layer σ outputs a value between 0 and 1 for each C_{t-1} where the value 0 corresponds to completely forgetting the previous cell state and 1 to fully preserving it.



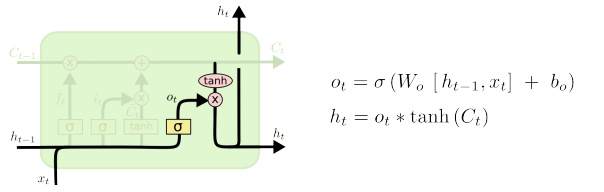
(a) LSTM forget gate [76].



(b) LSTM input gate [76].



(c) LSTM cell state update [76].



(d) LSTM output gate [76].

Figure 38: LSTM cell structure.

The input gate in Figure 38b is made up of a sigmoid layer and a $tahn$ layer. The sigmoid layer decides which values will be updated and the $tanh$ layer creates a vector , $\tilde{C}_t$,of new possible values, $\tilde{C}_t$, that could be added to the state. The previous state C_{t-1} is then updated to the current cell state C_t, as is shown in Figure 38c. The old state is multiplied by f_t and the result is then added to $i_t \times \tilde{C}_t$, thus creating

the new current cell state, C_t. In short, the first two gates decide which values will be forgotten which input values are relevant before updating the cell. The activation functions used in these networks are all either sigmoid or tanh functions, this is due to the fact that, if an activation function is not bounded, the network will not be able to converge. This makes activation functions like ReLUs improper for these networks [77].

The cell output in Figure 38d is based on a filtered version of the cell state, a sigmoid layer decides what parts of the cell state are passing to the output. The cell state goes through the $tanh$ layer (to push the values to be between -1 and 1) and is multiplied by the output of the sigmoid gate, so that only the chosen values are passed to the output.

These networks have found more and more usage since their surfacing in Hochreiter and Schmidhuber [78] in several fields such as Natural Language Processing (NLP) or weather forecast. The training of these networks however, tends to be quite slow due to the greater number of hyper-parameters that have to be tuned. One other factor to have into account is the fact that, with this many hyper-parameters, these networks tend to overfit easily according to the previously stated rule in Hagan et al. [72]. It is therefore advisable to include a dropout layer in these networks as well.

6.2.4 Convolutional Long-Short Term Memory Neural Network

A C-LSTM is a new architecture that involves using one or more convolutional layers for feature extraction and using that output as an input to a LSTM for sequence prediction. These networks are both spatially and temporally deep, and are usually used in video classification tasks. The architecture of a C-LSTM is shown in Figure 39.

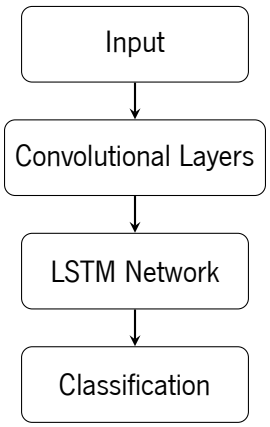


Figure 39: C-LSTM architecture.

Feature set: Taking into account that the initial layers of this new network are convolutional, they expect an image input or, in the very least, a matrix input. The LSTM is a sequential network, so a sequence of images will be necessary for this network. Because of this, each trial in the sequential feature set was converted into a video. Each instance from a sequential trial will be converted to an image, much like what was done in Tables 15 and 16.

Convolutional Feature Extraction: With the sequential feature set converted to an image sequence form, the convolutional layer is now capable of performing feature extraction. The convolutional layers are obtained from the previously developed CNN. Every layer until the last pooling layer will be used in this model, with the remaining layers being discarded for this purpose. This model is then used to convert the full feature set into a normal sequential feature set, to serve as input to the LSTM network.

6.2.4.1 Hybrid Network Method

The DL toolbox has got a functionality that enables the creation of networks with a more complex layer structure than the ones more traditionally used. These networks are known in MATLAB as Directed Acyclic Graph (DAG) networks. The layers from networks built this way, can have inputs from multiple layers and outputs to multiple layers, leading to the employment of the architecture shown in Figure 40.

With this new architecture, this model behaves as a single complex network were convolutional and sequential feature extraction are all performed simultaneously. A new type of layer was introduced with this new architecture. A sequence folding/unfolding layer converts a batch of image sequences to a batch of images and a batch of images to a batch of sequences, respectively. This enables the network to perform convolution operations on time steps of image sequences independently.

6.3 Results

This section documents the performance of all eight models under each conditions followed by a comparison between the performances of each one. Nested CV was used to estimate the performance of each model and since the goal is the selection of the model with the best performance, a repeated CV with 2 folds and 10 repetitions will be used. In nested CV, the full model building process is repeated for every fold and the resulting model is used to predict the test set from the outer loop.

The decision of using 2 folds is based on Zhang and Yang [68] that states that for model selection, the evaluation part has to be sufficiently large, as long as the ranking of the candidates in terms of risk at the

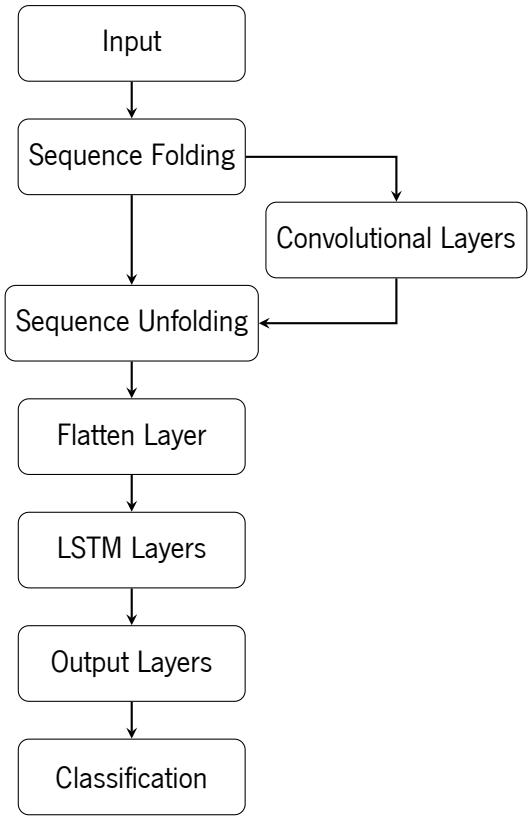


Figure 40: New C-LSTM architecture.

reduced sample size of the training part stays the same as that under the full sample size, which implies that the training sample size must not be too small. Additionally, the advised number of repetitions is between 10 to 20 repetitions. For reasons related to the time necessary to build certain models and the amount of time available to finish this project, the number of repetitions chosen was 10.

To verify the accuracy of the performance given by CV, two testing methods were employed. The first is a random split of the dataset into a training set and a validation set, where 70% of the data will go to the training set and 30% to the validation set. One downside that this method might have is that due to the fact that when splitting a randomly shuffled dataset, there will be data from nearly all subjects of all classes. This implies that the training data and test data will be very similar. To counterbalance this, the second method performs a subject split were all data from one subject of each class is completely removed from the dataset and is later used as a training set. Due to the high variable nature of gait signals, this unseen data could provide a better assessment of each model's performance. The metric used to quantitatively access model performance is MCC and was chosen due to its good representative properties of unbalanced classes classification results [79].

Several different model configurations were tested and the results are shown in the tables in Appendix A.4. Afterwards, the configuration of each model with the best performance is selected in order to compare all 8 models to each other. The testing conditions are different for ML and DL models.

Machine Learning Models: Performance will be estimated firstly without any dimensionality reduction methods and afterwords with the three chosen methods in the classification tool. For each case, the hyper-parameters of the model will be adjusted until it reaches its best performance within the established conditions. The dimensionality reduction methods to be used are mRMR, PCA and ANOVA.

Deep Learning Models: The hyper-parameters of each network will be adjusted until be best performance possible is achieved with the goal of keeping these networks as small as possible. The process of performance estimation will have to be more detailed due to the significantly greater amount of parameters and hyper-parameters to adjust, these will be detailed in the following section. However, greater importance will be given to the network's topology instead of their parameters. These will be the same for every network and were chosen according to the most used values found in literature. According to Zhang et al. [80], regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error. Additionally, there will be no need for dimensionailty reduction methods.

6.3.1 Support Vector Machine

For the SVM classifier in this tool, there are three available kernels to chose from: linear, Gaussian and polynomial. The number of hyper-parameters of this classifier depends on the type of kernel used. For linear kernels the only hyper-parameter is C, which represents the penalty of the error term. It controls the trade off between a smooth decision boundary and correctly classifying the training points. With non-linear kernels, an additional hyper-parameter, σ, is added and it represents how well the hyper-plane fits with the training data. If any of these hyper-parameters are too high the classifier will overfit due to the attempts to fit itself perfectly with the training data. For polynomial kernels, the second hyper-parameter is the order of the polynomial used to find the hyperplane to split the data. The higher this hyper-parameter, the higher the training times will be.

Linear Kernel: With this kernel, only C requires tuning. Table 28 shows the best results for each configuration (same model for different dimensionality reduction methods). The C hyper-parameter was

tuned by starting with zero and iteratively increasing it's value until the classifier's performance no longer increases.

Gaussian Kernel: With a gaussian kernel, σ, is added to the tuning process. In this, case both hyper-parameters need tuning so a grid-search is used and every hyper-parameter pair is tested with the pair yielding the best performance being chosen. The range picked for both hyper-parameters is an interval of exponents where the parameter takes the value 2^y and the value y ranges from $[-10 : 10]$ [81].

Polynomial Kernel: The polynomial kernel creates a feature hyperspace based on polynomials of the originals input features. With this kernel, the second hyper-parameter is not σ but instead the order of the polynomial, as mentioned previously. Only orders 2 (quadratic) and 3 (cubic) will be explored due to the fact that the greater the order of this kernel, the longer the required processing time will be.

6.3.2 k-Nearest Neighbors

As mentioned earlier, this model merely computes the closest instances to the test data point in order to classify it. This is a relatively simple algorithm to use, having only one main hyper-parameter k to tune, defining the number of closest neighbors. Additionally, to increase the robustness of the model, a distance function is used to give greater weight to points closer to the test point.

There are several different distance metrics that can be used, the most common is the euclidean distance. It is a good distance measure to use if the input variables are similar in type (all distance measurements) however, Manhattan distance is a good measure to use if they are not similar in type (measurements, weight, temperature, etc.). In this context, both distance metrics will be used and compared. The kNN models' performance was evaluated in the same manner as the SVM models, k is tuned by starting from a minimum value 1, and increasing until no improvement in performance is seen. The results are shown in Tables 31 and 32, in Appendix A.4.

6.3.3 Discriminant Analysis

When using discriminant analysis models, no hyper-parameters are tuned and so the default values provided by MATLAB are used. It is only necessary to select the type of kernel to be used, linear or quadratic. In each test condition, both kernels were tested with the best performance shown in Table 33.

6.3.4 Random Forests

In random forests the number of decision trees, or learners, is tuned, similarly to the kNN models, by incrementally increasing the number of trees starting with 1 until performance reaches the maximum value or starts decreasing. Additionally both linear and quadratic kernels were tested with the results shown in Tables 34 and 35.

6.3.5 Feed-forward Neural Network

The main hyper-parameters of a FFNN are the number of hidden layers and the number of neurons on each of those layers. These two hyper-parameters will be represented as an array were each position represents a hidden layer and the value in that position represent the number of neurons on that layer. The search for the hyper-parameters that give the best performance will begin with only one layer until the best number of neurons is found, afterwards the number of layers is increased and the process repeated. This is done until no significant improvement in performance is observed. Early stopping was also employed to reduce overfit, each fold was separated into a training set (70%) and a validation set (30%). The training set is used for computing the gradient and updating the network weights and biases, while the error of validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, just like the training set error. But when the network begins to overfit the data, the validation error usually begins to rise. After a specified number of iterations, if the validation error does not decrease, the training is stopped, and the weights and biases at the minimum of the validation error are returned. In this context, the maximum number of iterations, before the validation loss begins to rise, was set to 3. The results obtained are present in Table 36.

According to the toolbox's documentation, the resilient gradient descent function (*trainrp*) is the fastest algorithm for pattern recognition problems. However, it does not perform well on function approximation problems. Its performance also degrades as the minimum error to be reached is reduced. The memory requirements for this algorithm are relatively small in comparison to the other algorithms considered. However, Scaled Conjugate Gradient Descent (*trainscg*), in pattern recognition problems, works specially well and is almost as fast as *trainrp*. Its performance does not degrade as quickly as *trainrp* performance does when the error is reduced. For this reason the training function chosen was *trainscg*.

6.3.6 Convolutional Neural Network

When compared to other ML models, NNs have a lot of hyper-parameters to tune and CNNs are no exception to this rule. To simplify this process and eliminate the most possible number of variables when accessing the performance of this network, some fixed values will be chosen for a few parameters.

Kernel Size: There is a clear preference for odd-sized kernels (Figure 41) so that all the pixels in the layer's input are symmetrically around each output pixel computed by the kernel. In the case of an even-sized kernel, distortions across the layers would occur.

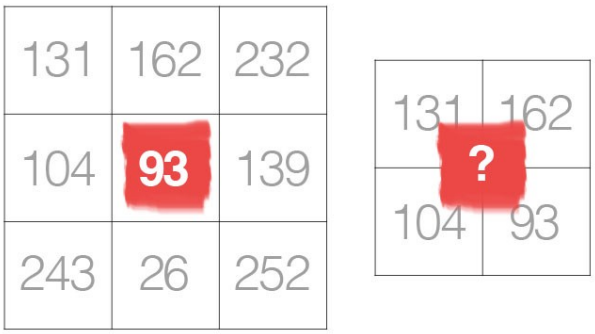


Figure 41: Odd-sized filter.

The size of the image input to this network is 9×9, so an appropriate size for the kernel would be of 3×3. This is the smallest possible kernel size and it was chosen due to the fact that there are very sharp differences in intensity between pixels. This was noticed by analyzing most images of the dataset and by using a small kernel, these sharp changes will be easier to detect.

Max Pooling Layer: The max pooling layer will have a pooling size of 2 so that the output image is downsampled to half its size. The stride of a pooling layer should not be smaller that the pooling size in order to avoid any possible overlap. A max pooling layer was chosen instead of an average pooling layer because max pooling is better at extracting relevant feature whereas average pooling smooths the output image.

Training Options: Several training conditions were established for the evaluation of each network. Each fold will be separated into several mini-batches, the network will use each mini-batch to estimate its classification error before updating itself. This method is called Mini-Batch Gradient Descent. The size of each mini-batch will be of 32, as recommended in literature [82, 83]. The number of epochs, which is

the number of times that the network passes through the whole data set, is usually large, often hundreds or thousands, allowing the learning algorithm to run until the error from the model has been sufficiently minimized. In this case, the epoch size chosen is 500, with a shuffling of data in every epoch. To avoid overfitting, early stopping is also employed so that if the loss on the validation set becomes larger than or equal to the previously smallest loss does not decrease within a specified number of iterations, the training stops. The optimization function to reduce the network's error is the ADAM function [84]. The selected training options are shown in Table 17.

Table 17: Training Options

Mini Batch Size	Max Epochs	Optimization Function	Initial Learning Rate	Gradient Threshold	Validation Patience
32	500	ADAM	0.01	1	3

The gradient threshold parameter limits the gradient's value to 1 to avoid any possible exploding gradient as the network increases in depth. The validation patience parameter determines the amount of validation evaluations during which the validation error can increase, or fail to decrease, before the training is interrupted. Since these results are meant for model selection/comparison, less focus was given to the optimization of each model. It is preferable to maintain constant as many parameters as possible in order to access each models capabilities. For this reason, the learning rate for each network will remain constant, at 0.01. The advised range of values in Bengio [82] is between 0.1 and 0.01. By looking at the DL toolbox documentation for the $ADAM$ optimizer, a value of 0.01 was suggested.

The hyper-parameters that will be explored in these tests are the number of kernel has got on the networks performance and the number of convolutional layers. Throughout each convolutional layer, the size of the image eventually gets smaller. By using a max pooling size of 2, the output image in each convolutional layer will be half the size of its input. Seeing that the input image used has got size 9×9, this can be a problem for more than one convolutional layer. To avoid this, zero padding is used in each pooling layer to keep the output image with its original dimensions. Additionally, between each convolutional layer and ReLU layer, a batch normalization layer will be applied to speed up training and reduce overfit. In most cases where more than one convolutional layer is used, each layer has double the amount of kernels than the preceding layer. This relationship between layers will also be applied to this network. It is necessary that the number of kernel increases in each layer because the feature maps' representation of the input of the convolutional layer is much richer than the inputs themselves. To properly encode this information in the following convolutional layers it is necessary that they possess a large enough amount of kernels. The performance of this network is shown in Table 37.

6.3.7 Long-Short Term Memory Neural Network

The implementation of this network in MATLAB is as follows: (i) sequence input layer, (ii) recurrent layer, (iii) dropout layer, (iv) fully connected layer, (v) softmax layer.

The input layer of this network has got 73 inputs, each one receiving a sequence of values. The following layer is a recurrent layer which gives the network its characteristic behavior. In the toolbox used, these layer can be one directional or bi-directional. A one directional layer only takes into consideration the past states of the network while a bi-directional layer can look into data in both ways, from beginning to end and vice-versa. This enables the bi-directional LSTM to preserve information from the past and future at any given point in time. However, in most real world applications, this is not realistic. Still, both types of layer were evaluated and the results are shown in Tables 38 and 39.

As was mentioned earlier, due to the great amount of parameters of these networks, they can easily overfit the training data. To mitigate this issue, a dropout layer is used , if necessary. With this layer, during training, some number of layer outputs are randomly ignored, making the training process noisy and forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs, making the model more robust. Because the outputs of a layer under dropout are randomly subsampled, it has the effect of reducing the capacity or thinning the network during training. As such, a wider network, may be required.

Just like with CNNs, LSTMs also need a fully connected layer and a softmax layer at the output, for classification. The recurrent layer function as feature extractors, just like the convolutional layers.

Uni-Directional Recurrent Layer Uni-Directional LSTMs are only capable of learning from present and past inputs so generally their performance can be inferior to their bi-directional counterpart. However, they find much more usage in real-world applications.

Bi-Directional Recurrent Layer These Bi-Directional layers can learn from data in both ways, from past to future and from future to past. This will provide the network with a better perspective of the data however, these networks can't find many real-world application due to their need to have data from future events. The training parameters are all the same as the ones used in the One-Directional network. The training conditions for both these networks are shown in Table 18.

There is an additional parameter, sequence length, with represents the maximum sequence length from the input that the network can accept. Any larger sequence will be truncated and any smaller sequence

Table 18: Training Options

Mini Batch Size	Max Epochs	Optimization Function	Initial Learning Rate	Gradient Threshold	Validation Patience	Sequence Length
32	500	ADAM	0.01	1	3	400

will be padded to this length. According to the documentation of the DL toolbox, it is advisable to select a length between 200 and 400. Since most of the sequences in the feature set were closer to the upper limit, the value 400 was chosen to avoid any unnecessary loss of information.

6.3.8 Convolutional Long-Short Term Memory Neural Network

For a more accurate comparison of each network, the training options for this network will be exactly the same as the ones used for the LSTM network, shown in Table 18.

Just like with the previous network, the performance of this model will be accessed with one-directional and bi-directional layers. Since the objective of this analysis is model selection, the tuning of this network will not be very thorough. There will be only one convolutional layer with the same number of kernels as the one used in the CNN in order to access the effects of associating a convolutional layer with a recurrent layer. The results of both uni-directional and bi-directional networks are shown in Tables 40 and 41, respectively.

6.4 Critical Analysis

This section analyzes the collected performance results of each classifier. The results for the test performed under a random and subject split will be analyzed separately. In the end, the best configuration of each model will compared. To select the best model in each case, the test performance of each will be compared. In the case that two or more models exhibit the same performance, the less computationally expensive model will be selected.

6.4.1 Random Split

From the results obtained with the random train/test split it can be seen that the MCC score of most classifiers were very close to 1, except when using ANOVA for feature selection. This was expected due to the selected number of features being very small (only 2 features).

The classification models that have shown lowest performance were DA and RF. The highest MCC score with DA was of 0.9899 in the test set with mRMR for feature selection. In this condition, only incorrectly classifying 2 stroke gait cycles as healthy gait cycles (false positive). In RF overal, there was no difference between the usage of a linear or quadratic kernel. Both CV performances were similar the best models being RF with linear kernel and no dimensionality reduction with an MCC of 0.9945 (one false positive) and RF with quadratic kernel and mRMR feature selection with a MCC of 0.995 (one false negative).

With SVM models, all of them achieved MCC values of 1 with the test set. The only exceptions were the instances were ANOVA feature selection was used and the case of a linear SVM combined with PCA feature extraction, which only reached an MCC of 0.9895 with 2 false positives.

Regarding kNN, most models achieved MCC values of 1 apart from every instance were ANOVA was used and in the case were a kNN model with the Manhattan Distance metric and PCA feature extraction, which only achieved a MCC of 0.9145 on the test set.

With the FFNN the outcome was very similar. Very good results were shown, apart from the instance were the model was combined with ANOVA for feature selection. None of the models achieved a MCC of 1 but the FFNN with the full feature set and the network combined with mRMR both achieved a very close value of 0.9949 with only 1 false positive.

The performance of the developed CNN is slightly dependent on the amount of layers that each network has. The best performances on the test set , with MCCs of 1 and 0.9947 (one false positive), were achieved with 3 and 4 layers, respectively.

For the LSTM networks, there was very little change in the obtained MCC values with the test set, ranging from only one incorrect classification to a perfect classification. One thing to note however, is that there is a greater variability of the estimated performances in CV with the one-directional networks.

With the final network, C-LSTM, a perfect MCC score (1) was achieved in every test case, with both uni-directional and bi-directional recurrent layers.

The point of optimal performance for each classifier was very easy to achieve because an MCC of more than 0.95 was achieved with almost no tuning at all. As mentioned before, this can be attributed to the fact that both train and test set have data from all subjects. Furthermore, since each instance of the feature set represents a full gait cycle, data from the same trial can be present in both train and test sets when performing the random split. This can negatively affect the confidence in the test set results and is the main reason for redoing the tests with a subject data split, in order to access if these suspicions were true.

One very common occurrence that can be noted in these results is that when applying any dimensionality reduction methods, the model's performance usually decreases, which is slightly unexpected. However,

this is not a significant drop in performance and having into account that to reduce the number of features used some information will inevitably be lost. The performance of PCA and mRMR were very similar, with mRMR showing a slightly higher performance.

6.4.2 Subject Split

Due to the fact that this second round of tests was created as a consequence of the suspiciously good results obtained in the previous tests, not much time was available to explore the tuning of each model. With this in mind, more time consuming models, such as RF and FFNN, were tuned to its best performance without any dimensionality reduction method and then the same tuning was used with the three different dimensionality reduction methods to evaluate changes in performance. The overall results with this new dataset split were significantly smaller, confirming the hypothesis that a random split is not appropriate for this dataset.

For each SVM kernel, the model with the best performance did not use any dimensionality reduction methods. The best SVM models were the gaussian kernel SVM and the polynomial kernel SVM, both models used the full feature set and achieved a MCC score of 1. From all SVM classifiers created, the ones with best performance employed no dimensionality reduction. From the models that did employ dimensionality reduction, the method that provided best results was mRMR and the worst method was ANOVA. The hyper-parameters for each SVM model (C, σ and the order of the polynomial kernel), were determined through a grid search.

The kNN models with best performance were a kNN model with euclidean distance metric and a kNN model with Manhattan distance metric, with both models using the full feature set and reaching a MCC score of 1. The kNN model with mRMR feature selection and euclidean distance metric also reached an MCC of 1 but was not selected due to the fact that it is much more computationally expensive than the other two models. Off all three dimensionality reduction methods used, the best performing one was mRMR and the worst was ANOVA. In each case, the best performance for each kNN model was reach with a small k (no larger than 5), apart from the euclidean kNN model with PCA feature extraction which required that $k = 13$.

The best performing DA model had a quadratic kernel and used mRMR feature selection, reaching a MCC score of 0.9594. In this case mRMR actually improved the classifiers performance, unlike PCA and ANOVA, which decreased it to 0.6281 and 0.4014, respectively. From these results, DA models with quadratic kernels seem to exhibit a superior performance.

The RF model with best performance was a RF model with a quadratic kernel and mRMR feature selection, achieving a MCC score of 0.8467. Overall, RFs with quadratic kernels seem to perform better than with linear kernels. For each kernel, mRMR boosted performance while PCA and ANOVA lowered it. The number of decision trees was set by finding the best performing RF model, when using the full feature set. The same number of decision trees was used on the following models due to time constraints.

From the FFNN created, the best performing network used PCA feature extraction, achieving a MCC score of 0.9797. In this case, mRMR was the method to obtain the worst performance while ANOVA reached an MCC score of 0.8564, very close to the score obtained with the FFNN with the full feature set (0.8765). Like with the RF models, due to time constraints, it was not possible to explore the performance of each network with a greater number of hidden layers. The number of hidden neurons was set with the full feature set and the same configuration was used with all dimensionality reduction methods.

All CNN models reached a MCC test score of 1, even with only one layer and 8 kernels. This network has the advantage of being quite fast to train when compared the the other networks (LSTM and C-LSTM). The network chosen to be included in Table 19 was the smallest network, with one convolutional layer and 8 kernels.

All LSTM also achieved a MCC score of 1 in the test set. The networks with a uni-directional recurrent layer did exhibit a greater standard deviation than their bi-directional counterparts, possibly due to the fact that uni-directional network cannot learn from future training examples. These networks require a significantly higher training time than other classifiers, which is a disadvantage. To speed up training time, the performance of the network could be analyzed for a smaller number of units, the values chosen in this dissertation might have been greater than what was necessary for this dataset. In the case of C-LSTM networks, all of the models created also achieved MCC scores of 1. They also needed a slightly longer training time than LSTMs due to the added convolutional layer.

This new data split proves that, with random split the presence of gait cycles on both datasets (train and test) negatively affects the confident in the test results. Apart from the models that use ANOVA feature selection, the remaining models have very high MCC scores. There was not much difference in the performance of PCA feature extraction and mRMR feature selection. Furthermore, from these results, it can be concluded that the performance of each dimensionality reduction method also depends on the classifier it is used with. In the case of FFNN and RF, the best method was PCA, while with the other ML classifiers mRMR showed a superior performance.

6.4.3 Comparison

With a subject split, the differences between classifiers and dimensionality reduction methods was more visible and greater care was needed to tune each classifier. Because of this, only the classifiers tested with a subject data split will be included in Table 19.

DL models achieved MCC scores of 1 as well as the selected SVM and kNN models, both of which use the full feature set. The best dimensionality reduction was mRMR, with PCA proving to be superior with the FFNN. The worst performing model was RF, with a MCC score of 0.8467. The selected DA and FFNN models achieved MCC scores of 0.9594 and 0.9797, respectively. There was not much time available to tune the FFNN and RF models, there is a possibility that with more time spent tuning, their performance could increase. kNN and CNN models have the advantage of being faster to train when compared to LSTM, C-LSTM and SVM using grid search. Taking into account Tables 28 to 32 and 37 to 41, DL methods have a systematic test MCC score of 1 for every model. Ttaking into account MCC score and speed, the best classification models were kNN (with the full feature set) and CNN. This implies that DL method are not universally superior to ML methods, the performance of each is very application dependent.

Table 19: Classifier comparison

Model	**D.R.**	**Hyper Parameters**	**Cross-Validation**				**Test**		
			MCC	**Std**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
SVM (Gauss)	None	Sigma: 4 C: 1024	0.9983	0	430	0	1	31	0
					1	920		0	120
KNN (Eucl)	None	k: 1 Weighted: No	0.9963	0.0013	430.7	1.9	1	31	0
					0.3	918.1		0	120
DA	mRMR	Kernel: Quadratic	0.985	0.0049	424.1	1.9	0.9594	30	1
					6.9	918.1		1	119
RF	mRMR	No Trees: 80 Kernel: Quadratic	0.971	0.0035	420.7	6.7	0.8467	31	9
					10.3	913.3		0	111
FFNN	PCA	Layers: 1 Neurons: 7	0.9807	0.0058	424.8	5.1	0.9797	30	0
					6.2	914.9		1	120
CNN	N/A	Conv. Layers: 1 Kernels: 8	0.9855	0.01	424.4	1.9	1	31	0
					6.4	918.1		0	120
LSTM	N/A	Hidden Units: 50	0.7968	0.2359	241.4	8.8	1	28	0
					78.6	738.2		0	94
CLSTM	N/A	Hidden Units: 50	0.982	0.028	312.6	0.7	1	28	0
					7.4	746.3		0	94

7

CONCLUSIONS

In this dissertation, a gait pattern recognition system was developed with the objective of distinguishing between healthy and post-stroke gait patterns.

In Chapter 2, a study of the available work in the field of gait recognition systems was carried out. In this research, all building blocks of these systems were analyzed. The existing sensor systems to acquire data, the nature of the features used in literature, the existing dimensionality reduction methods and also the classification models used all aided in setting the correct path to take for a better development of this system. The data was acquired through a wearable sensor system due to its lower cost and ease of in terms of testing conditions when compared to most commercial solutions. The choice of features found in literature significantly influenced the determination of features in this work.

The bio-mechanical data estimation tool implemented in Chapter 4, makes use of an orientation filter that capitalizes on the advantages that quaternions provide related to three dimensional rotation when compared to Euler angles. This method provided a good estimation of features, after some de-drifting procedures on the joint angles and the foot's velocity and position signals. There was less confidence in the position signals due to the fact that they were obtained by double integrating the acceleration signal, which will heavily increase any existing drift. The quaternion signals obtained with the sensor data from InertialLab were compared against the quaternion signals from the Xsens commertial system. There was an issue with the Xsens quaternion signals from the right foot that generated an increase in the RMSE values between InertialLab's and Xsens's signals. Despite this, the overall quality of the signals for the left foot was acceptable.

The features computed in Chapter 5 were stored in several feature tables in each trial directory of the database. With this it is possible to make the bio-mechanical data extraction tool a stand-alone system capable of easily being integrated with other inertial sensor systems. Additionally this tool also has a

function that properly extracts all feature tables and arranges them into two full feature sets, sequential and non-sequential.

These feature sets are stored in the classification tool developed in Chapter 6, and in the case of some NN, are structurally modified to fit each network's input requirements. This tool is very versatile in the sense that, with the several normalization options, dimensionality reduction and CV methods, performance metrics and classification models available, it provides a great array of possibilities for the optimization of the building process of a classification model. Additionally, with the addition of the four NN developed in this dissertation, this system is now capable of handling not only tabular datasets, but also image, sequence and video datasets making the possible addition of several other different classification models much easier.

As mentioned in Chapter 6, when analyzing the results obtained with random train/test data split on all models, nearly all classifiers obtained a MCC score very close to 1. These results proved to be suspicious, especially due to the fact that very little tuning was required to obtain good results. The separation of two subjects' complete data from each class to create a test set provides a better separation of training and test data in order to better evaluate the performance of each model on truly unseen data. With this new separation, the results were much more realistic with each classifier requiring a more careful tuning process, as expected. The results were quite satisfactory with kNN, SVM, CNN, LSTM and C-LSTM classifiers reaching MCC scores of 1. However, it is necessary to take into account that this classification tool is merely performing binary classification, it is possible that in the context of multi-class classification the resulting MCC scores will not be as high.

In light of the research questions asked at the beginning of this dissertation, these questions will be answered below:

RQ 1: Can the use of a quaternion-based orientation filter provide a better alternative to the estimation of sensor location/orientation?

The resulting quaternion signals from InertialLab, were compared to those from Xsens for a specific trial. For the left foot, where a correct acquisition was performed, the InertialLab signals were close to those obtained in Xsens, indicating that this method is a good alternative to other approaches, like ones where Euler angles are used for estimation of orientation. Despite this statement, an extensive validation procedure would have to be carried out to confirm this.

RQ 2: Is there an advantage in using DL methods in gait recognition when compared to the standard ML plus dimensionality reduction methods?

In general, DL methods did achieve a superior and more consistent performance, all of them reached MCC scores of 1 however, so did some SVM and kNN models. This implies that DL are not entirely superior in every aspect to other ML approaches. However, DL methods need no previous selection of a good dimensionality reduction method to obtain good results, greatly simplifying the classification process. In reality, the advantages really depend on the application. DL methods do not require that the user possess a good domain knowledge to perform classification tasks, this is an advantage for increasingly complex applications. In the context of gait classification, ML methods also show a lot of potential but DL methods have the advantage of internally performing feature extraction but the training of each models also has a greater computational cost.

7.1 Future Work

As mentioned previously, the gait event detection FSM was not validated for post-stroke patients. A thorough validation process would be necessary for this FSM. Additionally, to confirm the conclusions made from the resulting comparison between the quaternions of both InertialLab and Xsens, the same analysis could be made for gait trials with both systems being used simultaneously and synchronized with each other, in order to validate the answer given to RQ1.

A few considerations could also be taken to further improve the performance of this system. Firstly, as with all ML systems, the more data available for training, the better the generalization capabilities of the model. So more gait trials should be carried out thus increasing the created database, to increase the confidence in each developed model. Additionally, existing data in the laboratory from tests performed with other inertial sensor systems could be added, systems such as Xsens.

The data available in the laboratory is not limited to healthy and post-stroke gait, some trials were also performed on patients suffering from parkinson's disease or ataxia. If more trials were carried out on more patients there could be enough data to further increase this systems capabilities to recognition of other pathologies. Not only that, but multi-classification could also be attempted, both with different disorders and with different stages of the same disorder.

Seeing that the LSTM and C-LSTM networks that were developed in this dissertation are both sequential models, they could be very good candidates to be used with Reinforcement Learning (RL). This could be applied in a real-time context, providing that the networks are not to large and that the real-time system had enough processing power.

BIBLIOGRAPHY

[1] Walter Pirker and Regina Katzenschlager. “Gait disorders in adults and the elderly: A clinical guide”. In: *Wiener Klinische Wochenschrift* 129.3-4 (2017), pp. 81–95. issn: 16137671. doi: 10.1007/s00508-016-1096-4.

[2] Andrezej Witko. “Edited by Edited by”. In: *World* 3.February 2004 (2003), pp. 53–60.

[3] C. Beyaert, R. Vasa, and G. E. Frykberg. “Gait post-stroke: Pathophysiology and rehabilitation strategies”. In: *Neurophysiologie Clinique* 45.4-5 (2015), pp. 335–355. issn: 17697131. doi: 10.1016/j.neucli.2015.09.005. url: http://dx.doi.org/10.1016/j.neucli.2015.09.005.

[4] Sandra J Olney and Carol Richardsb. “Hemiparetic gait following stroke . Part I : Characteristics”. In: *Gait and Posture* 4.2 (1996), pp. 136–148.

[5] Mengxuan Li et al. “Gait Analysis for Post-Stroke Hemiparetic Patient by Multi-Features Fusion Method.” In: *Sensors (Basel, Switzerland)* 19.7 (2019). issn: 1424-8220. doi: 10.3390/s19071737. url: http://www.ncbi.nlm.nih.gov/pubmed/30978981http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6479843.

[6] Alvaro Muro-de-la Herran, Begoña García-Zapirain, and Amaia Méndez-Zorrilla. “Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications”. In: *Sensors (Switzerland)* 14.2 (2014), pp. 3362–3394. issn: 14248220. doi: 10.3390/s140203362.

[7] Carlotta Caramia et al. “IMU-Based Classification of Parkinson’s Disease from Gait: A Sensitivity Analysis on Sensor Location and Feature Selection”. In: *IEEE Journal of Biomedical and Health Informatics* 22.6 (2018), pp. 1765–1774. issn: 21682194. doi: 10.1109/JBHI.2018.2865218.

[8] Murad Alaqtash et al. “Automatic classification of pathological gait patterns using ground reaction forces and machine learning algorithms”. In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. 2011, pp. 453–457. isbn: 9781424441211. doi: 10.1109/IEMBS.2011.6090063.

[9] Daniel T.H. Lai, Rezaul K. Begg, and Marimuthu Palaniswami. “Computational intelligence in gait research: A perspective on current applications and future challenges”. In: *IEEE Transactions on Information Technology in Biomedicine* 13.5 (2009), pp. 687–702. issn: 10897771. doi: 10.1109/TITB.2009.2022913.

[10] Fei Jiang et al. “Artificial intelligence in healthcare: Past, present and future”. In: *Stroke and Vascular Neurology* 2.4 (2017), pp. 230–243. issn: 20598696. doi: 10.1136/svn-2017-000101.

[11] Venkat N Gudivada, Amy Apon, and Junhua Ding. “Data Quality Considerations for Big Data and Machine Learning : Going Beyond Data Cleaning and Transformations”. In: *International Journal on Advances in Software 10.1* 10.1 (2017), pp. 1–20.

[12] R. Begg and J. Kamruzzaman. “A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data”. In: *Journal of Biomechanics* 38.3 (2005), pp. 401–408. issn: 00219290. doi: 10.1016/j.jbiomech.2004.05.002.

[13] Rezaul K. Begg, Marimuthu Palaniswami, and Brendan Owen. “Support vector machines for automated gait classification”. In: *IEEE Transactions on Biomedical Engineering* 52.5 (2005), pp. 828–838. issn: 00189294. doi: 10.1109/TBME.2005.845241.

[14] Bjoern M. Eskofier et al. “Marker-based classification of young-elderly gait pattern differences via direct PCA feature extraction and SVMs”. In: *Computer Methods in Biomechanics and Biomedical Engineering* 16.4 (2013), pp. 435–442. issn: 10255842. doi: 10.1080/10255842.2011.624515.

[15] Daniel T.H. Lai et al. “Detection of tripping gait patterns in the elderly using autoregressive features and support vector machines”. In: *Journal of Biomechanics* 41.8 (2008), pp. 1762–1772. issn: 00219290. doi: 10.1016/j.jbiomech.2008.02.037.

[16] Bogdan Pogorelc, Zoran Bosnić, and Matjaž Gams. “Automatic recognition of gait-related health problems in the elderly using machine learning”. In: *Multimedia Tools and Applications* 58.2 (2012), pp. 333–354. issn: 15737721. doi: 10.1007/s11042-011-0786-1.

[17] Jianning Wu, Jue Wang, and Li Liu. “Feature extraction via KPCA for classification of gait patterns”. In: *Human Movement Science* 26.3 (2007), pp. 393–411. issn: 01679457. doi: 10.1016/j.humov.2007.01.015.

[18] Antonio I. Cuesta-Vargas, Alejandro Galán-Mercant, and Jonathan M. Williams. “The use of inertial sensors system for human motion analysis”. In: *Physical Therapy Reviews* 15.6 (2010), pp. 462–473. issn: 1743288X. doi: 10.1179/1743288X11Y.0000000006.

[19] Daphne J. Geerse, Bert H. Coolen, and Melvyn Roerdink. “Kinematic Validation of a Multi-Kinect v2 Instrumented 10-Meter Walkway for Quantitative Gait Assessments”. In: *PLOS ONE* 10.10 (2015). Ed. by Alfonso Fasano, e0139913. issn: 1932-6203. doi: 10.1371/journal.pone.0139913. url: http://dx.plos.org/10.1371/journal.pone.0139913.

[20] Shuyang Han, Shirong Ge, and Hongtao Liu. “Modeling of human lower limb and technical analysis of athlete’s high leg lift”. In: *2010 3rd International Conference on Biomedical Engineering and Informatics*. IEEE, 2010, pp. 951–954. isbn: 978-1-4244-6498-2. doi: 10.1109/BMEI.2010.5639927. url: http://ieeexplore.ieee.org/document/5639927/.

[21] Vicon Motus. *Vicon Motus: History of Peak Motus motion analysis software*. url: http://www.motus10.com/motion{_}analysis{_}motus{_}2d.aspx (visited on 11/14/2019).

[22] Vicon Motus. *Vicon Motus: Features*. 2018. url: http://www.motus10.com/motion{_}analysis{_}software{_}motus.aspx (visited on 09/30/2019).

[23] Ahsan H. Khandoker et al. “Wavelet-based feature extraction for support vector machines for screening balance impairments in the elderly”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 15.4 (2007), pp. 587–597. issn: 15344320. doi: 10.1109/TNSRE.2007.906961.

[24] Kistler. *Gait Analysis | Kistler*. 2019. url: https://www.kistler.com/en/applications/sensor-technology/biomechanics-and-force-plate/gait-analysis/ (visited on 09/30/2019).

[25] AMTI. *Hall Effect and Strain Gage Sensing Technology for Multi-axis Force Plates and Force Sensors | AMTI Products*. 2019. url: https://www.amti.biz/fps-sensor-tech.aspx (visited on 09/30/2019).

[26] Fong Chin Su and Wen Lan Wu. “Design and testing of a genetic algorithm neural network in the assessment of gait patterns”. In: *Medical Engineering and Physics* 22.1 (2000), pp. 67–74. issn: 13504533. doi: 10.1016/S1350-4533(00)00011-4.

[27] B. Najafi, T. Khan, and J. Wrobel. “Laboratory in a box: Wearable sensors and its advantages for gait analysis”. In: *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 2011. IEEE, 2011, pp. 6507–6510. isbn: 978-1-4577-1589-1. doi: 10.1109/IEMBS.2011.6091605. url: http://www.ncbi.nlm.nih.gov/pubmed/22255829http://ieeexplore.ieee.org/document/6091605/.

[28] Iris Tien, Steven D. Glaser, and Michael J. Aminoff. “Characterization of gait abnormalities in Parkinson’s disease using a wireless inertial sensor system”. In: *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC’10* August 2010 (2010), pp. 3353–3356. doi: 10.1109/IEMBS.2010.5627904.

[29] Thanh Trung Ngo et al. “The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication”. In: *Pattern Recognition* 47.1 (2014), pp. 228–237. issn: 0031-3203. doi: 10.1016/J.PATCOG.2013.06.028. url: https://www.sciencedirect.com/science/article/pii/S003132031300280X.

[30] Qi Wei Oung et al. “Objective Evaluation of Freezing of Gait in Patients with Parkinson’s Disease through Machine Learning Approaches”. In: *2018 International Conference on Computational Approach in Smart Systems Design and Applications, ICASSDA 2018* (2018), pp. 1–7. doi: 10.1109/ICASSDA.2018.8477606.

[31] Amira El-Attar et al. “Discrete wavelet transform-based freezing of gait detection in Parkinson’s disease”. In: *Journal of Experimental and Theoretical Artificial Intelligence* 00.00 (2018), pp. 1–17. issn: 13623079. doi: 10.1080/0952813X.2018.1519000. url: https://doi.org/10.1080/0952813X.2018.1519000.

[32] Yuchao Ma, Ramin Fallahzadeh, and Hassan Ghasemzadeh. “Glaucoma-specific gait pattern assessment using body-worn sensors”. In: *IEEE Sensors Journal* 16.16 (2016), pp. 6406–6415. issn: 1530437X. doi: 10.1109/JSEN.2016.2582083.

[33] Jens Barth et al. “Combined analysis of sensor data from hand and gait motor function improves automatic recognition of Parkinson’s disease.” In: *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference* 2012 (2012), pp. 5122–5125. issn: 1557170X.

[34] Junseok Lee, Sooji Park, and Hangsik Shin. “Detection of hemiplegic walking using a wearable inertia sensing device”. In: *Sensors (Switzerland)* 18.6 (2018). issn: 14248220. doi: 10.3390/s18061736.

[35] Andrea Mannini et al. “A machine learning framework for gait classification using inertial sensors: Application to elderly, post-stroke and huntington’s disease patients”. In: *Sensors (Switzerland)* 16.1 (2016). issn: 14248220. doi: 10.3390/s16010134.

[36] Hackster.io. *A DIY Smart Insole to Check Your Pressure Distribution - Hackster.io*. url: https://www.hackster.io/Juliette/a-diy-smart-insole-to-check-your-pressure-distribution-a5ceae (visited on 11/14/2019).

[37] Schulze Christoph. “The Influence in Airforce Soldiers Through Wearing Certain Types of Army-Issue Footwear on Muscle Activity in the Lower Extremities”. In: *The Open Orthopaedics Journal* 5.1 (2011), pp. 302–306. issn: 18743250. doi: 10.2174/1874325001105010302. url: http://benthamopen.com/ABSTRACT/TOORTHJ-5-302.

[38] Mohammad Reza Daliri. “Automatic diagnosis of neuro-degenerative diseases using gait dynamics”. In: *Measurement: Journal of the International Measurement Confederation* 45.7 (2012), pp. 1729–1734. issn: 02632241. doi: 10.1016/j.measurement.2012.04.013. url: http://dx.doi.org/10.1016/j.measurement.2012.04.013.

[39] Hassan Ghasemzadeh, Roozbeh Jafari, and Balakrishnan Prabhakaran. “A body sensor network with electromyogram and inertial sensors: Multimodal interpretation of muscular activities”. In: *IEEE Transactions on Information Technology in Biomedicine* 14.2 (2010), pp. 198–206. issn: 10897771. doi: 10.1109/TITB.2009.2035050.

[40] Sumitra S. Nair et al. “The application of machine learning algorithms to the analysis of electromyographic patterns from arthritic patients”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18.2 (2010), pp. 174–184. issn: 15344320. doi: 10.1109/TNSRE.2009.2032638.

[41] Diogo B. Gonçalves. “Real-time recognition and prediction of human and humanoid robot locomotion mode”. PhD thesis. Universidade do Minho, 2018.

[42] Tunç Aşuroğlu et al. “Parkinson’s disease monitoring from gait analysis via foot-worn sensors”. In: *Biocybernetics and Biomedical Engineering* 38.3 (2018), pp. 760–772. issn: 02085216. doi: 10.1016/j.bbe.2018.06.002.

[43] Physionet. *PhysioNet Databases*. 2019. url: https://www.physionet.org/about/database/ (visited on 09/30/2019).

[44] Francisco J. Badesa et al. “Auto-adaptive robot-aided therapy using machine learning techniques”. In: *Computer Methods and Programs in Biomedicine* 116.2 (2014), pp. 123–130. issn: 18727565. doi: 10.1016/j.cmpb.2013.09.011. url: http://dx.doi.org/10.1016/j.cmpb.2013.09.011.

[45] Herman Chan et al. “Assessing Gait Patterns of Healthy Adults Climbing Stairs Employing Machine Learning Techniques”. In: *International Journal of Intelligent Systems* 28.3 (2013), pp. 257–270. issn: 08848173. doi: 10.1002/int.21568. url: http://doi.wiley.com/10.1002/int.21568.

[46] Joana Figueiredo, Cristina P. Santos, and Juan C. Moreno. “Automatic recognition of gait patterns in human motor disorders using machine learning: A review”. In: *Medical Engineering and Physics* 53 (2018), pp. 1–12. issn: 18734030. doi: 10.1016/j.medengphy.2017.12.006. url: https://doi.org/10.1016/j.medengphy.2017.12.006.

[47] Valerie Sessions and Marco Valtorta. “The Effects of Data Quality on Machine Learning Algorithms”. In: *undefined* (2006). url: https://www.semanticscholar.org/paper/The-Effects-of-Data-Quality-on-Machine-Learning-Sessions-Valtorta/4bff8c41f74696d28b67f39859fa858f86c3b3fa.

[48] C. O. S. Sorzano, J. Vargas, and A. Pascual Montano. “A survey of dimensionality reduction techniques”. In: (2014), pp. 1–35. arXiv: 1403.2877. url: http://arxiv.org/abs/1403.2877.

[49] Katarzyna Kaczmarczyk et al. “Gait classification in post-stroke patients using artificial neural networks”. In: *Gait and Posture* 30.2 (2009), pp. 207–210. issn: 09666362. doi: 10.1016/j.gaitpost.2009.04.010.

[50] Joana Figueiredo et al. “Gait Event Detection in Controlled and Real-Life Situations: Repeated Measures From Healthy Subjects”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26.10 (2018), pp. 1945–1956. issn: 1534-4320. doi: 10.1109/TNSRE.2018.2868094. url: https://ieeexplore.ieee.org/document/8452990/.

[51] Siddhartha Khandelwal and Nicholas Wickström. “Evaluation of the performance of accelerometer-based gait event detection algorithms in different real-world scenarios using the MAREA gait database”. In: *Gait & Posture* 51 (2017), pp. 84–90. issn: 0966-6362. doi: 10.1016/J.GAITPOST.2016.09.023. url: https://www.sciencedirect.com/science/article/pii/S0966636216305859?via{\%}3Dihub.

[52] A L Goldberger et al. “PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.” In: *Circulation* 101.23 (2000), E215–20. issn: 1524-4539. doi: 10.1161/01.cir.101.23.e215. url: http://www.ncbi.nlm.nih.gov/pubmed/10851218.

[53] PAULO FÉLIX et al. “ADAPTIVE REAL-TIME TOOL FOR HUMAN GAIT EVENT DETECTION USING A WEARABLE GYROSCOPE”. In: *Human-Centric Robotics*. WORLD SCIENTIFIC, 2017, pp. 653–660. isbn: 978-981-323-103-0. doi: 10.1142/9789813231047_0079. url: http://www.worldscientific.com/doi/abs/10.1142/9789813231047{_}0079.

[54] S.O.H. Madgwick. *An efficient orientation filter for inertial and inertial/magnetic sensor arrays*. 2010, pp. 1–32.

[55] Jia Yan-Bin. “Quaternions and Rotations in E^4”. In: (2013), pp. 1–12.

[56] Joana Figueiredo, Cristina P Santos, and Juan C Moreno. “Human walking and abnormal gait patterns : a tutorial for the rehabilitation field”. In: ().

[57] C. Frigo, P. Crenna, and L.M. Jensen. “Moment-angle relationship at lower limb joints during human walking at different velocities”. In: *Journal of Electromyography and Kinesiology* 6.3 (1996), pp. 177–190. issn: 1050-6411. doi: 10.1016/1050-6411(96)00030-2. url: https://www.sciencedirect.com/science/article/pii/1050641196000302.

[58] Tytus Wojtara et al. “Muscle synergy stability and human balance maintenance”. In: *Journal of NeuroEngineering and Rehabilitation* 11.1 (2014), p. 129. issn: 1743-0003. doi: 10.1186/1743-0003-11-129. url: http://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-11-129.

[59] Tomas Cunha et al. “Looking for motor synergies in Darwin-OP biped robot”. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), pp. 1776–1781. issn: 10504729. doi: 10.1109/ICRA.2016.7487322.

[60] Seungjin Choi. “Algorithms for orthogonal nonnegative matrix factorization”. In: *Proceedings of the International Joint Conference on Neural Networks* 1 (2008), pp. 1828–1832. doi: 10.1109/IJCNN.2008.4634046.

[61] Giorgio Roffo and Simone Melzi. “Ranking to learn: Feature ranking and selection via eigenvector centrality”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10312 LNCS (2017), pp. 19–35. issn: 16113349. doi: 10.1007/978-3-319-61461-8_2. arXiv: 1704.05409.

[62] Giorgio Roffo. “Ranking to Learn and Learning to Rank: On the Role of Ranking in Pattern Recognition Applications”. In: (2017). arXiv: 1706.05933. url: http://arxiv.org/abs/1706.05933.

[63] Giorgio Roffo et al. “Infinite Latent Feature Selection: A Probabilistic Latent Graph-Based Ranking Approach”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 1407–1415. isbn: 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.156. url: http://ieeexplore.ieee.org/document/8237418/.

[64] Giorgio Roffo et al. “Infinite Latent Feature Selection: A Probabilistic Latent Graph-Based Ranking Approach”. In: (2017). arXiv: 1707.07538. url: http://arxiv.org/abs/1707.07538.

[65] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. “A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin | Request PDF”. In: *BJU International* 101.1 (2003), pp. 1396–1400. doi: https://doi.org/10.1177/02632760022050997. url: https://www.researchgate.net/publication/2926909{_}A{_}Practical{_}Guide{_}to{_}Support{_}Vector{_}Classification{_}Chih-Wei{_}Hsu{_}Chih-Chung{_}Chang{_}and{_}Chih-Jen{_}Lin.

[66] Mohammad Haghighat, Mohamed Abdel-Mottaleb, and Wadee Alhalabi. “Discriminant Correlation Analysis: Real-Time Feature Level Fusion for Multimodal Biometric Recognition”. In: *IEEE Transactions on Information Forensics and Security* 11.9 (2016), pp. 1984–1996. issn: 1556-6013. doi: 10.1109/TIFS.2016.2569061. url: http://ieeexplore.ieee.org/document/7470527/.

[67] Danielle Denisko and Michael M. Hoffman. “Classification and interaction in random forests”. In: *Proceedings of the National Academy of Sciences* 115.8 (2018), pp. 1690–1692. issn: 0027-8424. doi: 10.1073/PNAS.1800256115. url: https://www.pnas.org/content/115/8/1690.

[68] Yongli Zhang and Yuhong Yang. “Cross-validation for selecting a model selection procedure”. In: *Journal of Econometrics* 187.1 (2015), pp. 95–112. issn: 18726895. doi: 10.1016/j.jeconom.2015.02.006.

[69] Gavin C. Cawley and Nicola L.C. Talbot. “On over-fitting in model selection and subsequent selection bias in performance evaluation”. In: *Journal of Machine Learning Research* 11 (2010), pp. 2079–2107. issn: 15324435.

[70] Quora. *What is the role of the activation function in a neural network? How does this function in a human neural network system? - Quora*. url: https://www.quora.com/What-is-the-role-of-the-activation-function-in-a-neural-network-How-does-this-function-in-a-human-neural-network-system (visited on 12/11/2019).

[71] Saugat Bhattarai. *Saugat Bhattarai | Data Science, Machine Learning and Computer Vision*. url: https://saugatbhattarai.com.np/what-is-activation-functions-in-neural-network-nn/ (visited on 12/11/2019).

[72] Martin T. Hagan et al. *Neural network design*. 2014. isbn: 0971732116. url: https://books.google.pt/books/about/Neural{_}Network{_}Design.html?id=4EW9oQEACAAJ{\&}redir{_}esc=y.

[73] Towards Data Science. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. url: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 12/11/2019).

[74] Mc.ai. *Convolutional Neural Networks from the ground up - mc.ai*. url: https://mc.ai/convolutional-neural-networks-from-the-ground-up/ (visited on 12/11/2019).

[75] Machine Learning Mastery. *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. url: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ (visited on 12/11/2019).

[76] Colah’s Blog. *Understanding LSTM Networks – colah’s blog*. url: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 12/12/2019).

[77] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. “A Simple Way to Initialize Recurrent Networks of Rectified Linear Units”. In: (2015). arXiv: 1504.00941. url: http://arxiv.org/abs/1504.00941.

[78] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. issn: 08997667. doi: 10.1162/neco.1997.9.8.1735.

[79] Giuseppe Jurman, Samantha Riccadonna, and Cesare Furlanello. “A comparison of MCC and CEN error measures in multi-class prediction.” In: *PloS one* 7.8 (2012), e41882. issn: 1932-6203. doi: 10.1371/journal.pone.0041882. url: http://www.ncbi.nlm.nih.gov/pubmed/22905111http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3414515.

[80] Chiyuan Zhang et al. “Understanding deep learning requires rethinking generalization”. In: (2016). arXiv: 1611.03530. url: http://arxiv.org/abs/1611.03530.

[81] Huichuan Duan et al. “A Method to Determine the Hyper-Parameter Range for Tuning RBF Support Vector Machines”. In: *2010 International Conference on E-Product E-Service and E-Entertainment*. IEEE, 2010, pp. 1–4. isbn: 978-1-4244-7159-1. doi: 10.1109/ICEEE.2010.5661082. url: http://ieeexplore.ieee.org/document/5661082/.

[82] Yoshua Bengio. “Practical recommendations for gradient-based training of deep architectures”. In: (2012). arXiv: 1206.5533. url: http://arxiv.org/abs/1206.5533.

[83] Dominic Masters and Carlo Luschi. “Revisiting Small Batch Training for Deep Neural Networks”. In: (2018). arXiv: 1804.07612. url: http://arxiv.org/abs/1804.07612.

[84] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (2014). arXiv: 1412.6980. url: http://arxiv.org/abs/1412.6980.

A

APPENDIX

A.1 State of the Art

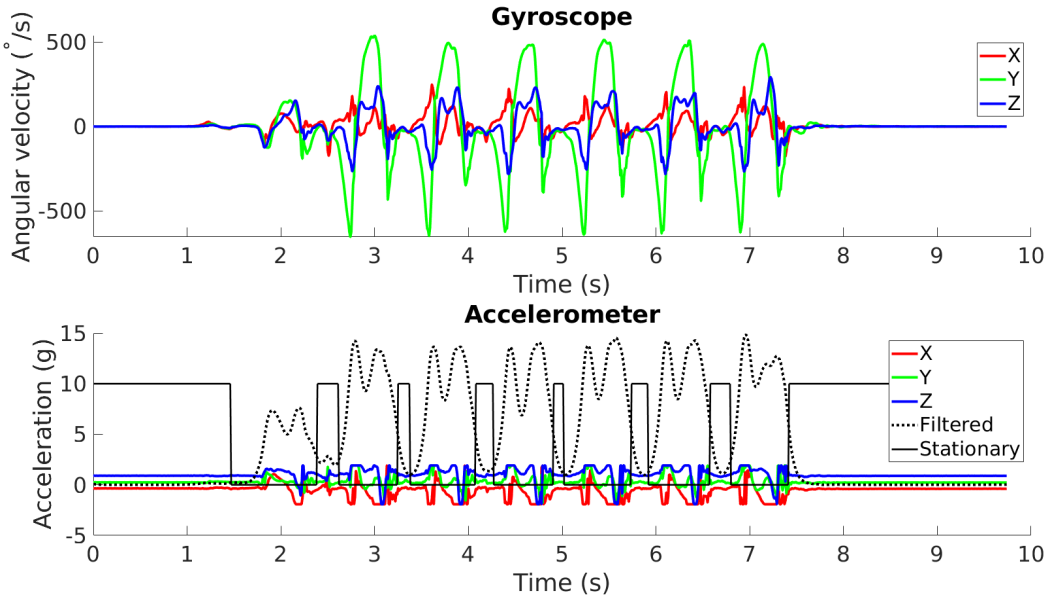
Table 20: Sensor system state of the art (summary)

Study	Subjects	Sensor System	Wearable	Bio-mechanical Data
Eskofier et al.	24 healthy young and 24 elderly subjects	Optical motion tracking system	No	84 spatiotemporal features
Lai et al.	13 healthy subjects and 10 fall subjects.	PEAK MOTUS 2D motion analysis system	No	512 minimum toe clearance (MTC) values
Pogorelc et al.	5 healthy, 3 right hemiplegia and 1 left hemiplegia subjects.	Smart IR motion capture system (3D Markers).	No	13 kinematic & spatiotemporal features.
El-Attar et al.	10 Parkinson's disease patients.	3 accelerometers (ankle, thigh, hip).	Yes	8 time-domain features.
Ma et al.	9 glaucoma patients and 10 healthy participants.	3-axis foot mounted accelerometers.	Yes	40 statistical features & 21 spatio-temporal features.
Lee et al.	15 healthy and 20 hemiplegia subjects.	Three-axis accelerometer & gyroscope.	Yes	165 time-domain features extracted from sensor's signals

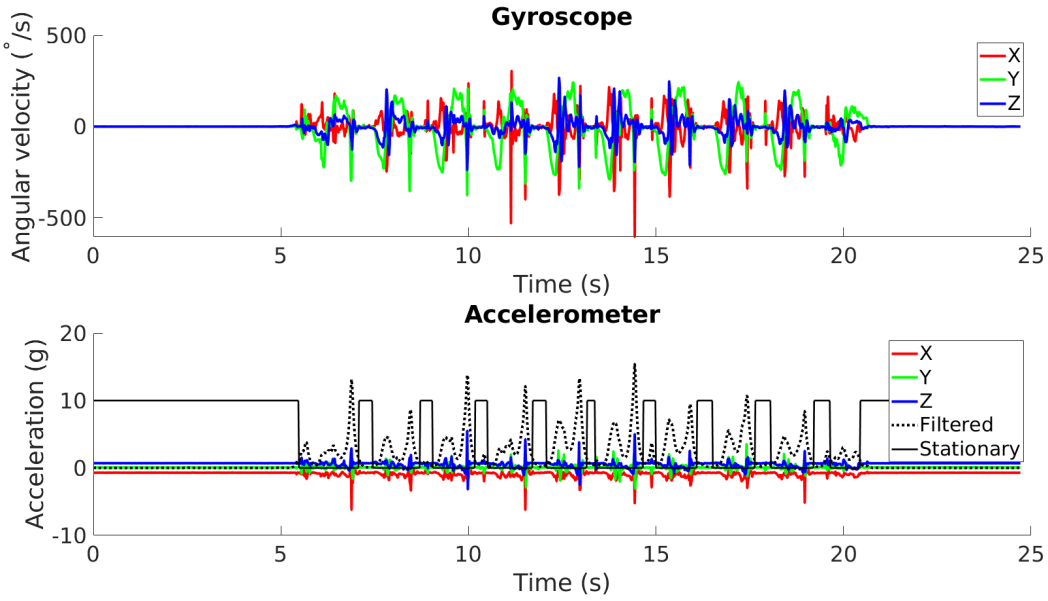
Table 21: Classifier state of the art (summary)

Study	Subjects	Dim. Red.	Classifier	Results
Badesa	7 subjects ages between 26 and 42	PCA	Perceptron, Logistic Regression (LR), Discriminant Analysis (DA), SVM (linear, RBF), Naive Bayes, kNN, Radial Basis Function (RBF).	Perceptron: 83.05% LR : 85.71 % DA (Quadratic): 78.1 % SVM (RBF): 91.43 % NB: 66.67 % kNN: 80.95 % RBF: 58.1 %
Begg	30 young healthy and 28 elderly participants	Hill-Climbing	ANN and SVM (linear, polynomial and RBF kernels).	SVM (linear): 83.3% ANN: 75%.
Eskofier	24 healthy young and 24 elderly participants	PCA	SVM (linear kernel)	98.5% accuracy when using between 36 to 39 principal components
El-Attar	10 Parkinson's disease patients.	Discrete Wavelet Transform (DWT)	SVM (linear kernel) and an ANN with 20 neurons in its hidden layer.	ANN: 93.80% SVM: 87.50%
Chan	13 healthy younger adults and 12 healthy older adults	Hill-Climbing	ANN, KStar, SVM, Naive Bayes (NB), Random Forests (RF), Decision Trees (DT)	(AGE/STAIRS) SVM: 71%/92% ANN: 80.6%/95,7% KStar: 79.6%/82.8% NB: 76%/90.4% RF: 76.3%/88.3% DT: 76.3%/86.2%
Pogorelc	5 healthy, 3 right hemiplegia and 1 left hemiplegia subject.	N/A	Five-class classification with SVM, DT, kNN, RF, NB, ANN.	SVM: 97.9% DT: 90.1% kNN: 100% NB: 97.2% RF: 99.3% ANN: 100%

A.2 Bio-Mechanical Data Extraction

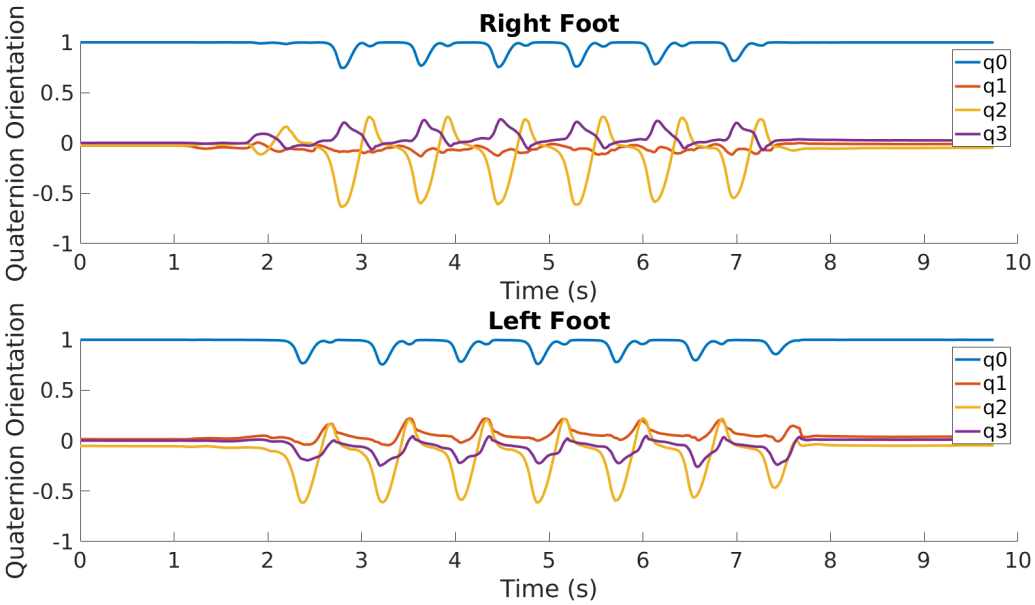


(a) Healthy gait.

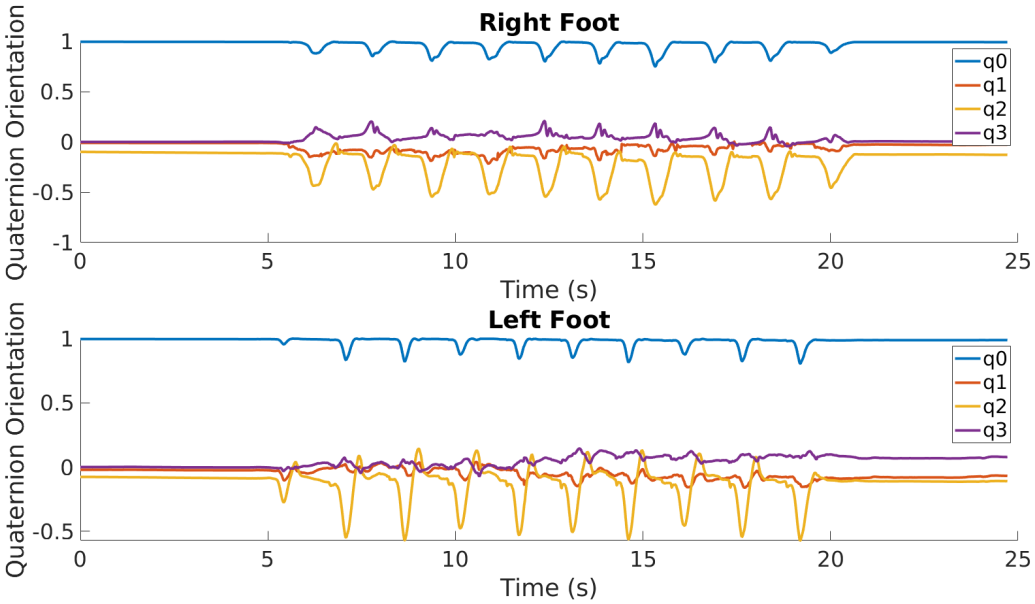


(b) Post-stroke gait.

Figure 42: Sensor accelerometer and gyroscope measurements.

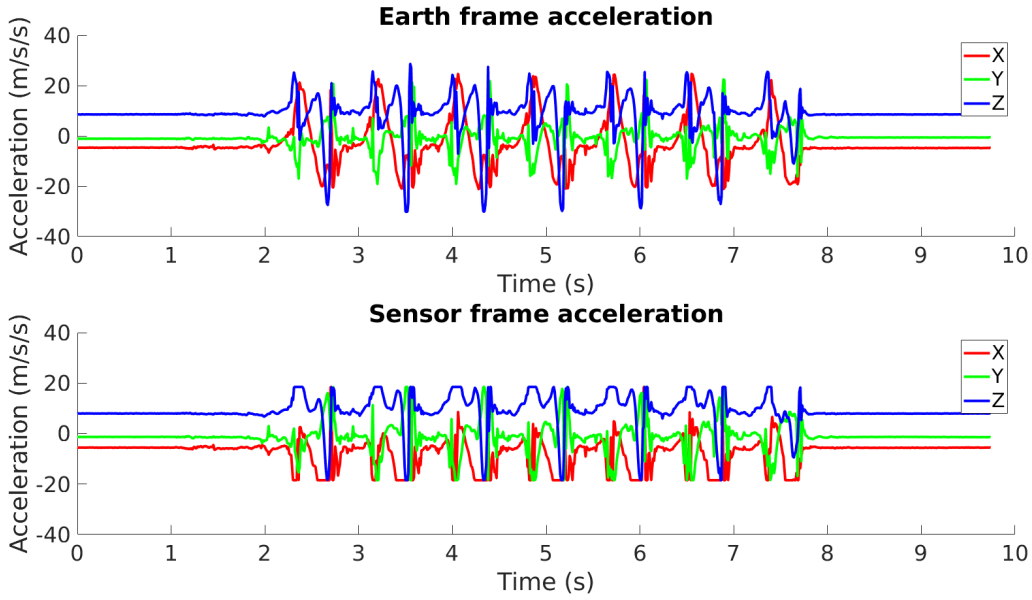


(a) Healthy.

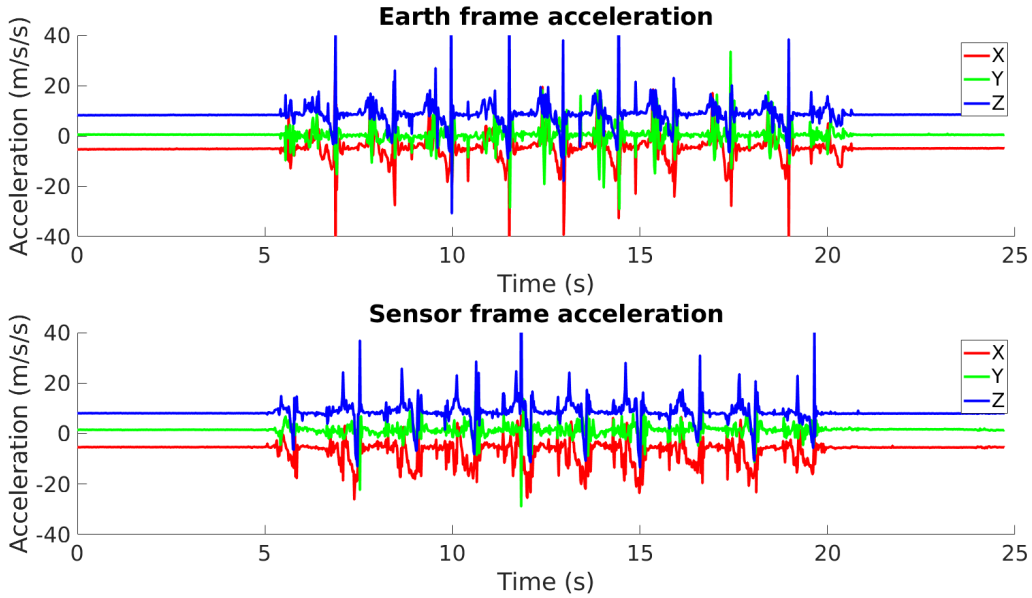


(b) Stroke.

Figure 43: Quaternion orientation.

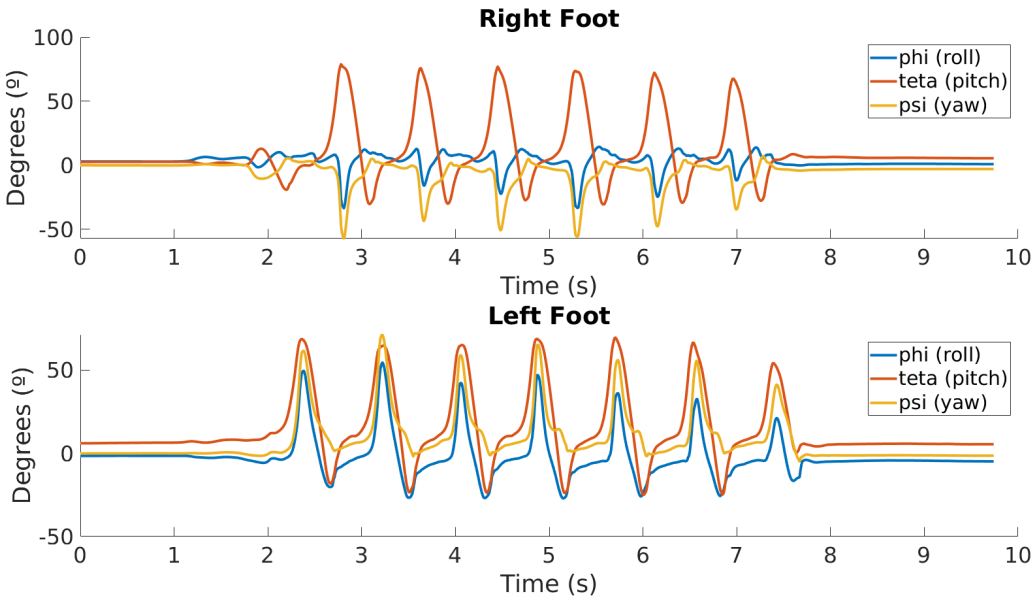


(a) Healthy gait.

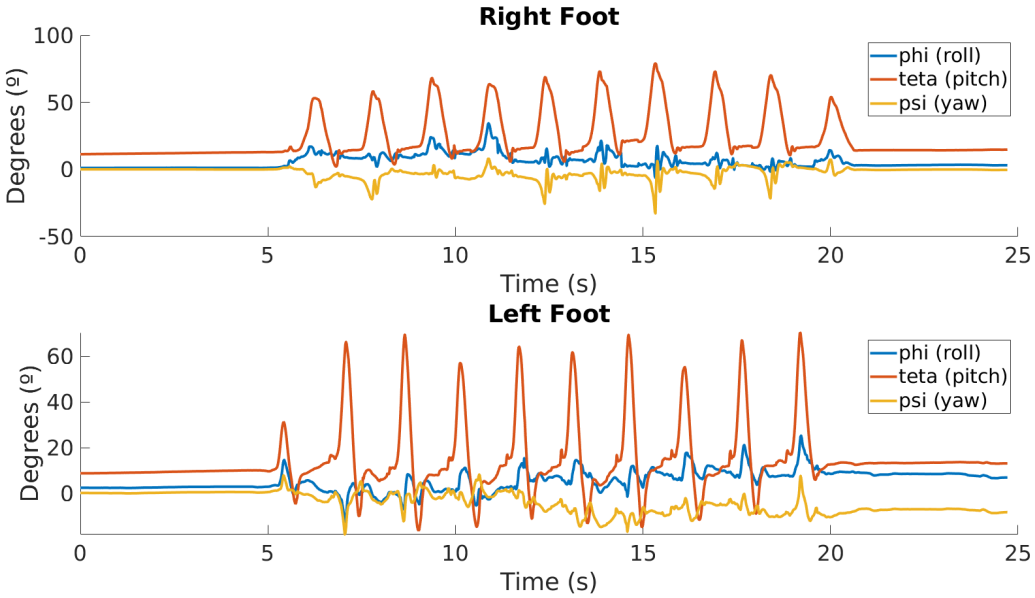


(b) Post-stroke gait.

Figure 44: Earth frame accelerations.



(a) Healthy gait.



(b) Post-stroke gait.

Figure 45: Earth frame orientation.

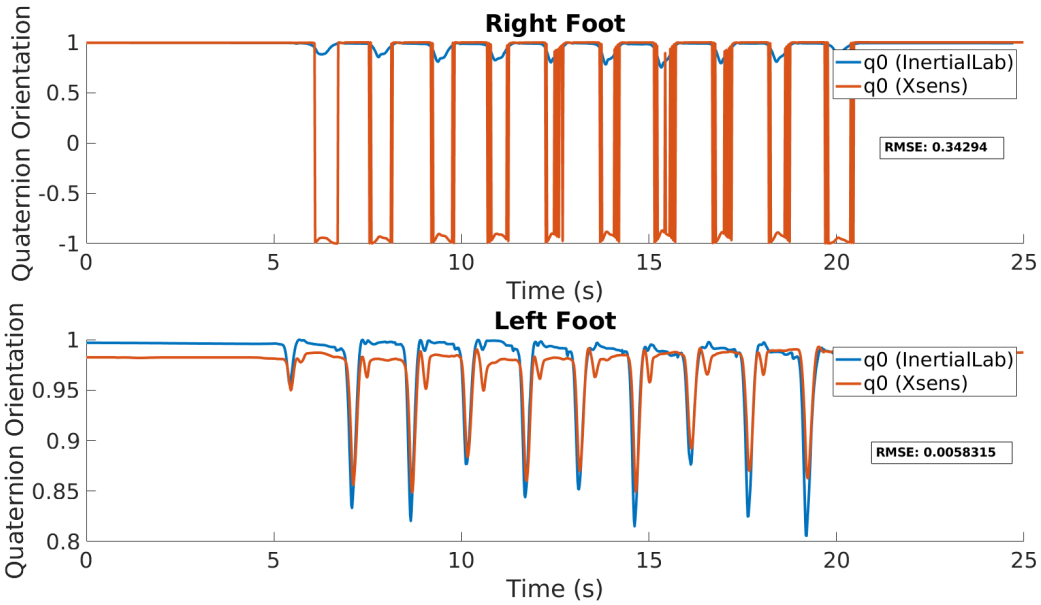


Figure 46: Comparison of q_0 component.

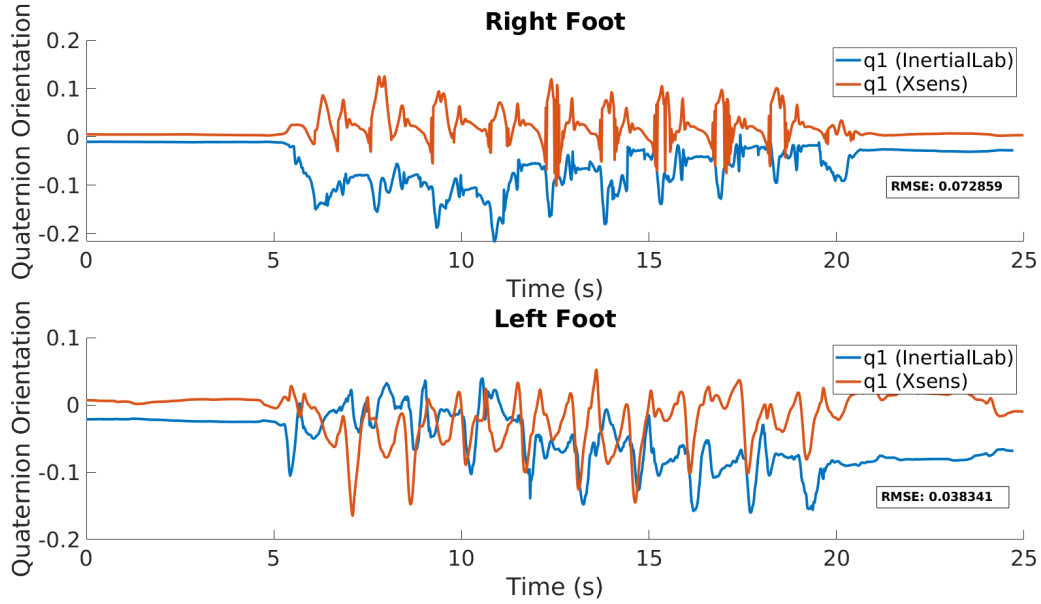


Figure 47: Comparison of q_1 component.

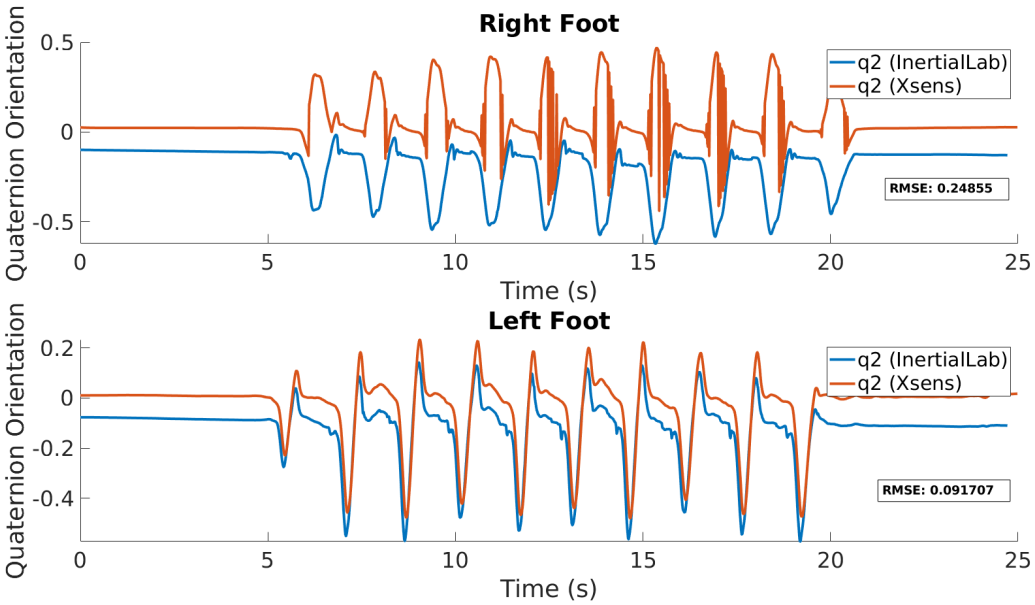


Figure 48: Comparison of q_2 component.

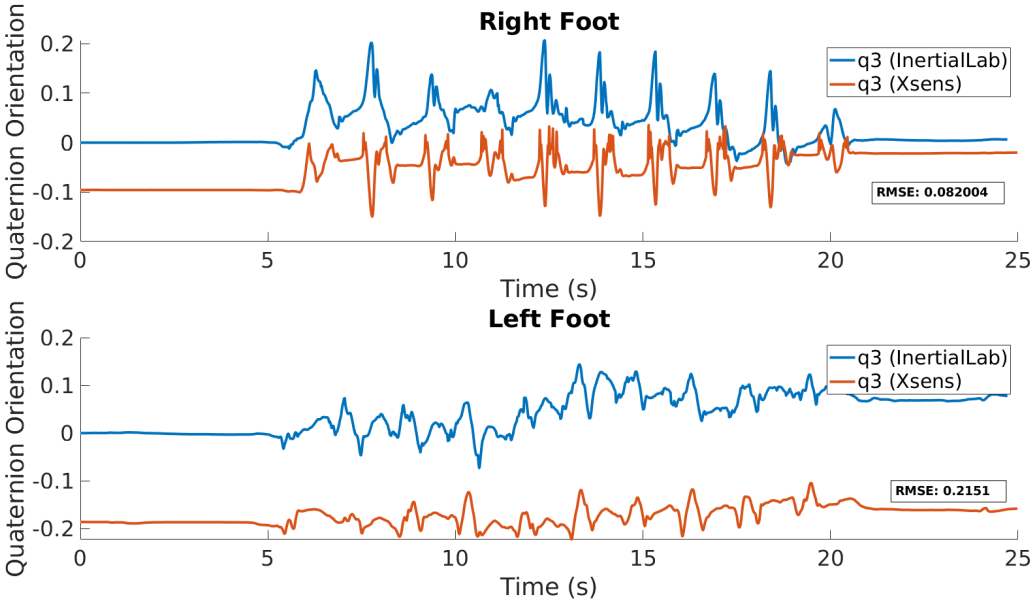


Figure 49: Comparison of q_3 component.

A.3 Feature Determination

Table 22: Spatial Features (Healthy)

Subject	Foot	Stride Length (m)		Step Length (m)		Stride Velocity (m/s)		Foot Clearance (m)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L	0.89	0.039	0.48	0.004	1.07	0.04	0.11	0.009
	R	0.91	0.001	0.62	0.017	1.09	0.008	0.11	0.0008
S2	L	0.88	0.015	0.54	0.057	1.01	0.06	0.14	0.0015
	R	0.85	0.053	0.51	0.123	0.97	0.12	0.11	0.024
S3	L	0.96	0.07	0.63	0.079	1.02	0.017	0.14	0.019
	R	0.9	0.044	0.56	0.046	0.95	0.014	0.13	0.007

Table 23: Spatial Features (Stroke)

Subject	Foot	Stride Length (m)		Step Length (m)		Stride Velocity (m/s)		Foot Clearance (m)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L (Paretic)	0.83	0.11	0.34	0.067	0.48	0.032	0.093	0.017
	R	0.97	0.25	0.39	0.03	0.5	0.09	0.092	0.007
S3	L	0.81	0.04	0.32	0.03	0.48	0.004	0.13	0.05
	R (Paretic)	0.83	0.19	0.29	0.15	0.46	0.057	0.09	0.009
S5	L	1.18	0.3	0.45	0.09	0.52	0.029	0.09	0.029
	R (Paretic)	0.99	0.13	0.6	0.21	0.51	0.035	0.06	0.018

Table 24: Temporal Features (Healthy)

Subject	Foot	Stride Duration (s)		Stride p/ Min (s)		Step Duration (s)		Cadence (step p/ min)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L	0.83	0.006	72.4	0.5	0.41	0.004	3.7	1.5
	R	0.82	0.007	72.6	0.63	0.42	0.003	3	2.6
S3	L	0.94	0.15	64.5	9.5	0.62	0.34	17	26
	R	0.89	0.055	67.9	4.3	0.26	0.3	49	79
S5	L	0.96	0.02	62.9	1.7	0.54	0.07	17	5.1
	R	0.95	0.057	63.4	4	0.41	0.12	11.8	27

Subject	Foot	Single Support (%)		Double Support (%)		Per Stance (%)		Per Swing (%)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L	42	0.47	7.8	0.47	58.2	0.12	41.6	0.29
	R	42	0.36	7.9	0.45	57.5	0.54	42.5	0.54
S2	L	41	3.9	-9.2	29.6	59.6	4.1	43.9	2.29
	R	47	7	22.1	27.4	57	0.43	42.8	0.43
S3	L	41	0.34	3.13	10.7	59.4	0.45	41.8	1.88
	R	42	3.9	14.3	6.7	58.8	1.57	41.2	1.57

Table 25: Temporal Features (Stroke)

Subject	Foot	Stride Duration (s)		Stride p/ Min (s)		Step Duration (s)		Cadence (step p/ min)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L (Paretic)	1.7	0.116	34.8	2.4	-1.17	2.4	40	13.3
	R	2	0.409	30	5.4	3.1	2.9	29	8.8
S3	L	1.7	0.071	35.9	1.4	-0.49	0.96	54	20.4
	R (Paretic)	1.9	0.197	32	2.7	2.27	1.18	22	10.5
S5	L	2.2	0.476	27.7	6.1	1.73	2.1	46	11.8
	R (Paretic)	2.1	0.174	29.1	2.6	0.54	1.7	38	11.6

Subject	Foot	Single Support (%)		Double Support (%)		Per Stance (%)		Per Swing (%)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L (Paretic)	41	9.8	133.8	146.9	63.8	0.95	36.1	0.95
	R	31	4.5	-75.4	90.2	65.8	6.47	34.7	5.73
S3	L	42.8	2.4	88.8	63.7	67.8	5.63	32.2	5.6
	R (Paretic)	28.5	3.7	-45.3	42.6	61.5	5.98	37.8	4.42
S5	L	31.2	8.5	4.36	74.4	64.3	20.82	35.6	20.82
	R (Paretic)	40.1	27.3	42	71.5	66.7	4.38	32.5	4.38

Table 26: Kinematic Features (Healthy)

Ankle

Subject	Foot	HS Angle (º)		TO Angle (º)		Peak DorsiFlexion (º)		Peak PlantarFlexion (º)		RoM (º)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L	-0.92	0.93	-6.8	2.24	8.66	0.7	-7.98	1.49	16.6	1.71
	R	2.45	0.41	-13.7	1.36	9.67	0.27	-18.6	0.98	28.3	1.07
S2	L	0.31	1.06	-9.02	1.92	11.87	1.08	-13.1	0.62	22.3	5.1
	R	-0.49	2.56	-13.6	0.46	6.82	1.11	-15	1.28	23.7	4.2
S3	L	-0.37	0.64	-11.5	2.35	14.1	0.12	-17.3	1.82	28.7	5.7
	R	1.65	2	-13	2.19	14.3	0.42	-19.4	0.65	30.4	5.8

Knee

Subject	Foot	HS Angle (º)		TO Angle (º)		Peak DorsiFlexion (º)		Peak PlantarFlexion (º)		RoM (º)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L	3	1.03	31.2	4.43	57	1.56	0.97	0.21	56	1.73
	R	4.3	0.77	31.1	0.48	59.1	1.23	1.03	0.22	58	1.01
S2	L	1.91	0.98	35	3.31	62.5	0.56	0.66	0.19	60	3.38
	R	2.72	1.37	32	0.94	54.6	0.33	0.98	0.19	55	2.47
S3	L	2.73	0.71	35	0.34	56.6	0.47	1.14	0.6	57	2.31
	R	3.95	1.06	29.3	3.53	52.9	0.75	1.33	0.18	52.6	2.26

Hip

Subject	Foot	HS Angle (º)		TO Angle (º)		Peak DorsiFlexion (º)		Peak PlantarFlexion (º)		RoM (º)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L	20.6	1.32	-17.3	2.29	22.4	0.76	-25.1	1.44	47.4	1.12
	R	18.2	1.17	-18.8	0.45	21.3	0.59	-25.6	0.26	47	0.34
S2	L	14.3	5.59	-13.7	3.22	17.6	0.14	-20.3	0.46	41.3	5.34
	R	15.4	2.35	-18.8	0.14	15.6	0.41	-22.7	0.14	41.1	5.07
S3	L	11.8	2.12	-14.1	0.38	13.9	0.43	-20.7	0.68	36.5	4.15
	R	14.6	0.74	-17.8	0.9	17.1	0.61	-25.6	0.53	41.8	1.01

Table 27: Kinematic Features (Stroke)

Ankle

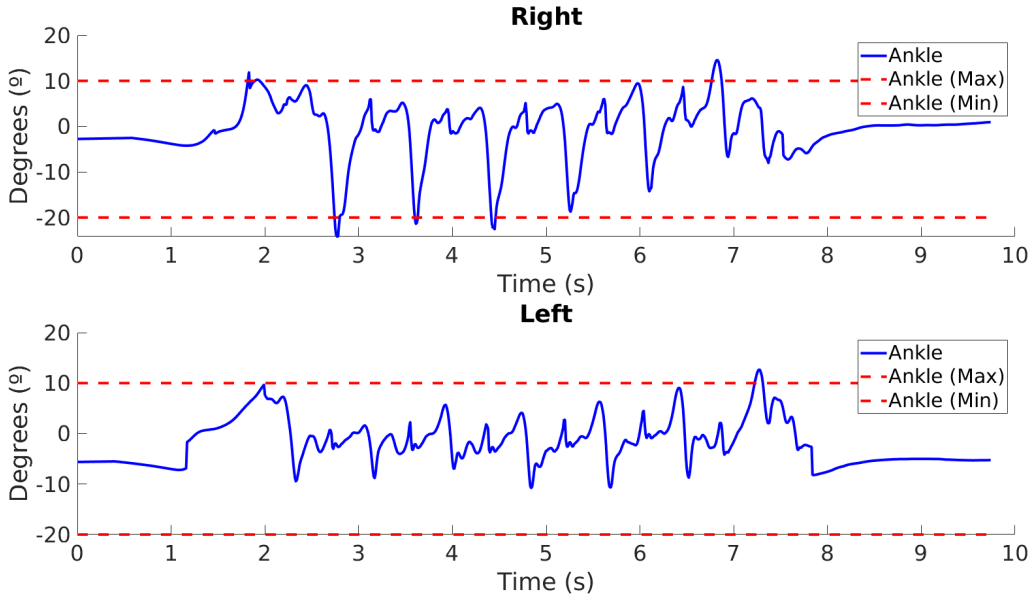
Subject	Foot	HS Angle (º)		TO Angle (º)		Peak Dorsi-Flexion (º)		Peak Plantar-Flexion (º)		RoM (º)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L (P)	-4.3	1.57	9.66	1.18	16.85	0.31	-10.37	0.26	27.22	0.39
	R	3.1	1.06	-8.36	3.08	8.55	0.68	-18.5	6.17	27.06	5.76
S3	L	-2	3.13	1	12.24	13.18	0.9	-15.1	0.19	28.13	1.29
	R (P)	-1.1	5.91	-5.44	4.13	9.97	0.35	-11.38	0.22	24	4.32
S5	L	-0.78	1.1	-2.27	2.64	13.59	1.53	-13.12	2.43	27.49	3.62
	R (P)	-4.2	2.1	-1.58	2.59	12.31	0.65	-10.54	0.8	23.05	1.49

Knee

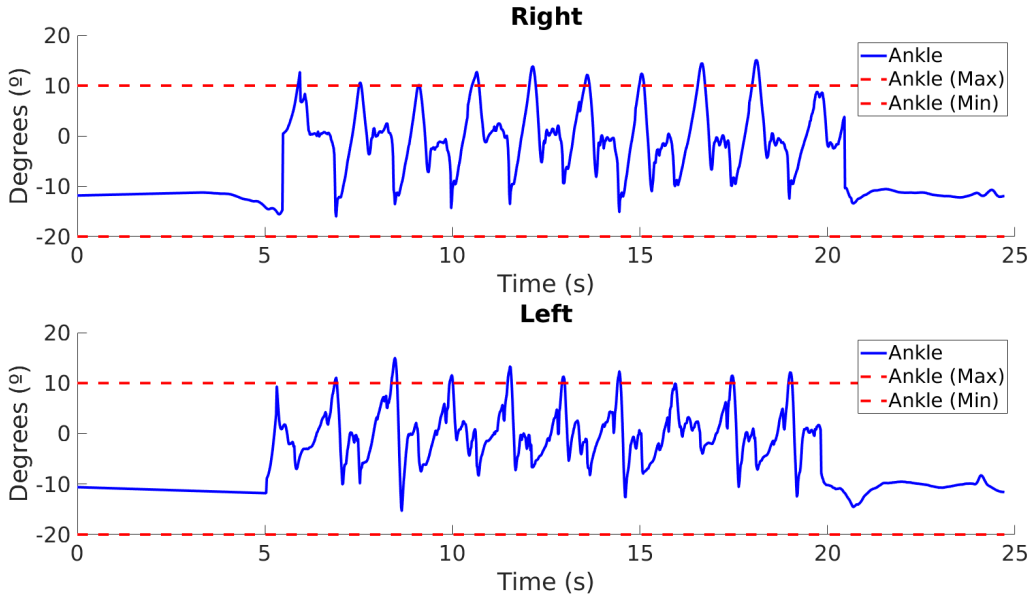
Subject	Foot	HS Angle (º)		TO Angle (º)		Peak Dorsi-Flexion (º)		Peak Plantar-Flexion (º)		RoM (º)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L (P)	21.27	1.54	43.2	4.98	59.58	3.96	1.99	1.83	57.6	2.53
	R	22.95	3.45	32.82	14.67	57.78	6.42	2.28	0.21	55.5	6.56
S3	L	12.73	12.08	61.03	25.22	76.98	2.8	2.85	0.23	66.77	12.97
	R (P)	14.11	12.5	23.33	13.43	30.46	1.68	2.54	1.39	41.6	19.65
S5	L	13.93	4.16	47.65	10.79	61.75	2.6	3.33	1.32	60.7	4.41
	R (P)	18.7	3.67	38.89	10.48	71.73	1.4	0.89	0.14	63.26	14.48

Hip

Subject	Foot	HS Angle (º)		TO Angle (º)		Peak Dorsi-Flexion (º)		Peak Plantar-Flexion (º)		RoM (º)	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
S1	L (P)	22.49	1.09	-10.14	1.28	39.66	2.85	-24.62	0.97	64.28	0.38
	R	17.62	4.08	-9.32	4.74	20.9	1.67	-19.92	3.39	40.82	0.5
S3	L	2.11	28.83	-0.94	13.02	21.37	1.29	-22.62	0.001	53.68	15
	R (P)	14.53	4.38	-13.42	5.79	25.44	1	-23.25	0.12	44.36	5
S5	L	18.1	10.67	-6.68	3.87	24.69	0.64	-20.22	1.05	47.58	4.21
	R (P)	17.83	2.58	-10.51	2.14	26.73	1.5	-14.29	1.45	41.78	3.34

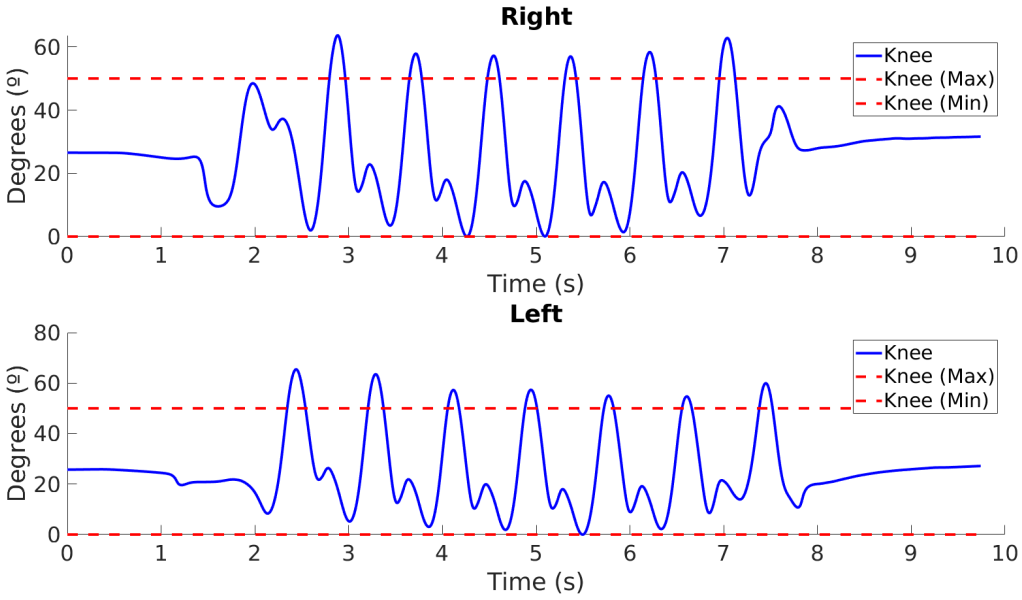


(a) Healthy ankle joint.

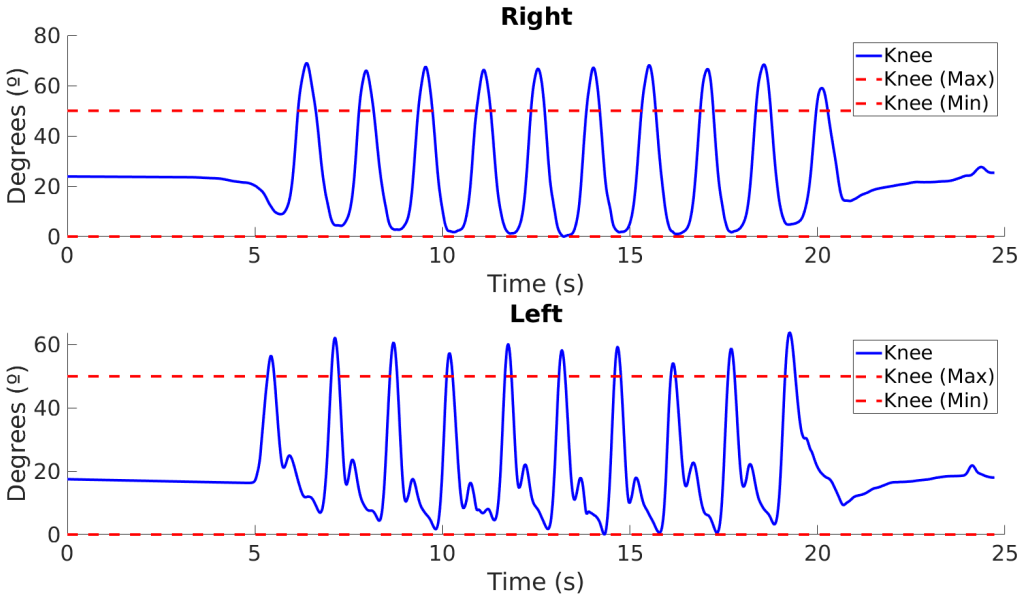


(b) Post-stroke (right side) ankle joint.

Figure 50: Ankle joint angles with maximum and minimum limits.

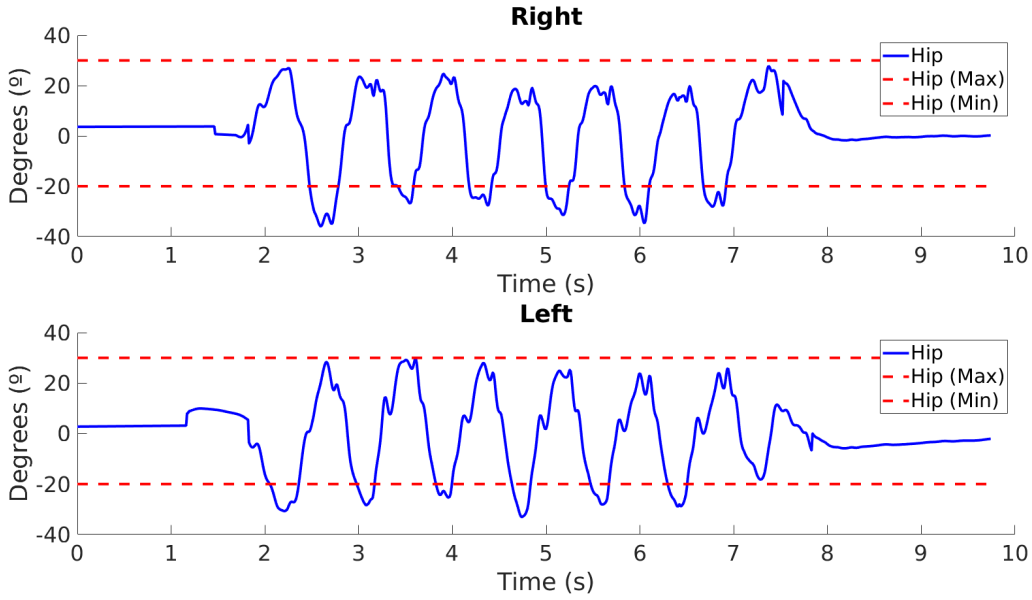


(a) Healthy knee joint.

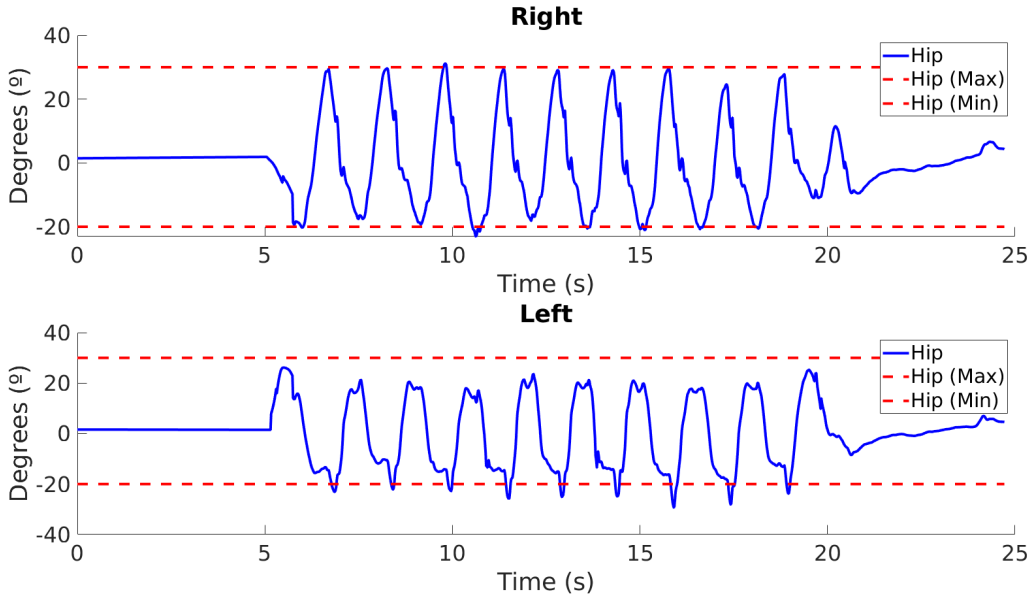


(b) Post-stroke (right side) knee joint.

Figure 51: Knee joint angles with maximum and minimum limits.



(a) Healthy hip joint.



(b) Post-stroke (right side) hip joint.

Figure 52: Hip joint angles with maximum and minimum limits.

A.4 Test Results

Table 28: SVM (linear kernel) results.

Random Split										
Dim. Red.	**Parameters**			**Cross-Validation**				**Test**		
	C	**sigma**	**order**	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	4	N/A	N/A	0.9967	-	330.5	0	1	130	0
						1.5	720		0	322
PCA	5	N/A	N/A	0.9933	-	320.1	2.6	0.9895	135	0
						4.9	724.4		2	315
mRMR	2	N/A	N/A	0.993	-	329.4	0.6	1	130	0
						2.6	719.4		0	322
ANOVA	$9.7e^{-4}$	N/A	N/A	0.7523	-	246.5	27.6	0.8145	99	3
						85.5	692.4		31	319
Subject Split										
Dim. Red.	**Parameters**			**Cross-Validation**				**Test**		
	C	**sigma**	**order**	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	4	N/A	N/A	0.9952	0.0016	429.4	1.2	0.9801	31	1
						1.6	918.8		0	119
PCA	2	N/A	N/A	0.9843	0.0033	426.1	4.3	0.6715	31	24
						4.9	915.7		0	96
mRMR	0.0313	N/A	N/A	0.9686	0.0407	415.4	2.9	0.9611	31	2
						15.6	917.1		0	118
ANOVA	$9.7e^{-4}$	N/A	N/A	0.8055	0.0082	323.4	7	0.4014	31	62
						107.6	913		0	58

Table 29: SVM (gaussian kernel) results.

Random Split										
Dim. Red.	**Parameters**			**Cross-Validation**				**Test**		
	C	**sigma**	**order**	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	1024	2	N/A	0.9977	-	313	0	1	148	0
						1	738		0	304
PCA	3	1	N/A	0.9928	-	319.5	1.7	1	141	0
						1.5	729.3		0	311
mRMR	2	1	N/A	0.9946	-	311.7	0.1	1	148	0
						2.3	737.9		0	304
ANOVA	1024	0.0313	N/A	0.7793	-	239	20.2	0.7861	115	9
						75	717.8		33	295
Subject Split										
Dim. Red.	**Parameters**			**Cross-Validation**				**Test**		
	C	**sigma**	**order**	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	1024	4	N/A	0.9983	0	430	0	1	31	0
						1	920		0	120
PCA	2	1	N/A	0.9968	0.0022	430.4	1.3	0.8087	28	7
						0.6	918.7		3	113
mRMR	64	1	N/A	0.9971	0.0011	429.6	0.3	0.8467	31	9
						1.4	919.7		0	111
ANOVA	2	1	N/A	0.8102	0.0056	327.2	7.9	0.4014	31	62
						103.8	912.1		0	58

Table 30: SVM (polynomial kernel) results.

Random Split										
Dim. Red.	**Parameters**			**Cross-Validation**				**Test**		
	C	**sigma**	**order**	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	0.0313	N/A	3	0.9967	-	330.5	0	1	130	0
						1.5	720		0	322
PCA	1024	N/A	2	0.9789	-	303.4	4.6	1	154	0
						4.6	739.4		0	298
mRMR	0.0078	N/A	3	0.9916	-	328.9	0.7	1	130	0
						3.1	719.3		0	322
ANOVA	0.002	N/A	2	0.729	0.089	219.9	15.6	0.5666	137	120
						104.1	712.4		1	192
Subject Split										
Dim. Red.	**Parameters**			**Cross-Validation**				**Test**		
	C	**sigma**	**order**	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	1024	N/A	2	0.9966	0.0027	429.5	0.5	1	31	0
						1.5	919.5		0	120
PCA	0.5	N/A	2	0.9872	0.0032	427	3.5	0.6715	31	24
						4	916.5		0	96
mRMR	0.002	N/A	2	0.9929	0.0031	429.3	2.5	0.8922	31	6
						1.7	917.5		0	114
ANOVA	$9.7e^{-4}$	N/A	2	0.5072	0.2857	231.6	102.2	-0.4014	0	58
						199.4	817.8		31	62

Table 31: kNN euclidean results.

Random Split									
Dim. Red.	**Parameters**		**Cross-Validation**				**Test**		
	Weighted	k	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	No	1	0.9986	-	313.9	0.5	1	148	0
					0.1	737.5		0	304
PCA	No	1	0.9877	-	318.5	3	1	144	0
					2.5	728		0	311
mRMR	Yes	3	0.9957	-	312.2	0.1	1	148	0
					1.8	737.9		0	304
ANOVA	Yes	5	0.7541	-	245.3	37.5	0.791	117	10
					68.7	700.5		31	294
Subject Split									
Dim. Red.	**Parameters**		**Cross-Validation**				**Test**		
	Weighted	k	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	No	1	0.9963	0.0013	430.7	1.9	1	31	0
					0.3	918.1		0	120
PCA	Yes	13	0.9914	0.0035	430.5	4.6	0.6715	31	24
					0.5	915.4		0	96
mRMR	No	4	0.9963	0.0028	429.3	0.5	1	31	0
					1.7	919.5		0	120
ANOVA	No	3	0.7759	0.0056	350.4	49.1	0.4014	31	62
					80.6	870.9		0	58

Table 32: kNN manhattan results.

Random Split									
Dim. Red.	**Parameters**		**Cross-Validation**				**Test**		
	Weighted	k	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	No	1	0.9989	-	314	0.5	1	144	0
					0	737.5		0	304
PCA	Yes	11	0.988	0.0041	321.7	3.1	0.9145	135	14
					2.3	724.9		3	298
mRMR	No	1	0.9986	-	310.5	2.9	1	144	0
					5.5	733.1		0	304
ANOVA	No	18	0.7773	0.0047	245	19.1	0.6814	92	13
					79	708.9		46	299
Subject Split									
Dim. Red.	**Parameters**		**Cross-Validation**				**Test**		
	Weighted	k	**MCC**	**Std. Dev**	**Conf. Mat.**		**MCC**	**Conf. Mat.**	
None	No	1	0.998	0.0007	431	1.2	1	31	0
					0	918.8		0	120
PCA	No	4	0.9899	0.0031	431	6	0.6224	29	24
					0	914		2	96
mRMR	Yes	5	0.999	0.0012	430.4	0	0.9084	30	5
					0.6	920		0	115
ANOVA	No	3	0.7759	0.0056	350.4	49.1	0.4014	31	62
					80.6	870.9		0	58

Table 33: DA results.

Random Split								
Dim. Red.	Kernel	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	Linear	0.9551	-	307.3	11.2	0.9391	138	4
				8.7	724.8		8	302
PCA	Quadratic	0.9611	-	309.5	5.8	0.9587	136	3
				11.5	725.2		5	308
mRMR	Linear	0.981	-	310.5	2.9	0.9899	144	0
				5.5	733.1		2	306
ANOVA	Quadratic	0.8024	-	242.4	12.4	0.7594	106	7
				73.6	723.6		40	299
Subject Split								
Dim. Red.	Kernel	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	Linear	0.9677	0.0031	417.3	5.2	0.8055	31	12
				13.7	914.8		0	108
PCA	Quadratic	0.9716	0.002	424.4	10.1	0.6281	30	26
				6.6	909.9		1	94
mRMR	Quadratic	0.985	0.0049	424.1	1.9	0.9594	30	1
				6.9	918.1		1	119
ANOVA	Both	0.808	0.0059	330.1	11.5	0.4014	31	62
				100.9	908.5		0	58

Table 34: RF (linear kernel) results.

Random Split								
Dim. Red.	Trees	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	6	0.9504	-	321	8.6	0.9945	126	0
				14	708.4		1	325
PCA	4	0.8024	-	256.6	30	0.92	137	3
				55.4	710		13	299
mRMR	4	0.9564	-	322.5	13.4	0.9734	131	3
				6.5	709.6		2	316
ANOVA	3	0.673	-	254	73.7	0.7445	116	32
				74	650.3		18	286
Subject Split								
Dim. Red.	Trees	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	80	0.9693	0.0063	416.2	3.2	0.6239	26	17
				14.8	916.8		5	103
PCA	80	0.926	0.0074	408.8	21.2	0.5065	15	6
				22.2	898.8		16	114
mRMR	80	0.9712	0.0053	421.3	7.2	0.662	31	25
				9.7	912.8		0	95
ANOVA	80	0.7207	0.0095	349.5	82.6	0.3958	31	63
				81.5	837.4		0	57

Table 35: RF (quadratic kernel) results.

Random Split								
Dim. Red.	Trees	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	9	0.9535	-	304.7	4.4	0.9638	136	2
				16.3	726.6		5	309
PCA	9	0.8462	-	283.8	23.6	0.9539	126	5
				45.2	699.4		7	314
mRMR	5	0.95	-	294.4	7.3	0.995	149	1
				14.6	731.7		0	302
ANOVA	8	0.6865	-	263.4	76.9	0.7213	109	30
				66.6	645.1		23	290
Subject Split								
Dim. Red.	Trees	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	80	0.9692	0.0055	415.9	3	0.6367	27	18
				15.1	917		4	102
PCA	80	0.9304	0.0092	408.9	18.7	0.5538	16	5
				22.1	901.3		15	115
mRMR	80	0.971	0.0035	420.7	6.7	0.8467	31	9
				10.3	913.3		0	111
ANOVA	80	0.7207	0.0095	349.5	82.6	0.3958	31	63
				81.5	837.4		0	57

Table 36: FFNN results.

Random Split								
Dim. Red.	Hidden Neurons	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	3	0.9946	-	315.3	1.7	0.9949	145	0
				0.7	734.3		1	306
PCA	4	0.9816	-	302	3	0.9902	153	0
				5	742		2	297
mRMR	3	0.991	-	313.3	1.3	0.9949	145	0
				2.7	734.7		1	306
ANOVA	5	0.7906	-	232.8	11.4	0.7836	114	7
				79.2	728.6		36	295
Subject Split								
Dim. Red.	Hidden Neurons	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
None	7	0.9940	0.0033	329.3	1.8	0.8765	31	7
				1.7	918.2		0	113
PCA	7	0.9807	0.0058	424.8	5.1	0.9797	30	0
				6.2	914.9		1	120
mRMR	7	0.9922	0.005	428.1	1.7	0.7679	31	15
				2.9	918.3		0	105
ANOVA	7	0.8101	0.0049	326.1	7	0.8564	27	3
				104.9	913		4	117

Table 37: CNN results.

Random Split								
No. Layers	No. Filters	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
1	16	0.9838	0.0128	320.8	1.1	0.9947	134	0
				6.2	723.9		1	315
2	32,64	0.9918	0.0054	324.6	1.3	0.9947	134	0
				2.4	723.7		1	315
3	8,16,32	0.9862	0.0097	322.3	1.5	0.9947	134	0
				4.7	723.5		1	315
Subject Split								
No. Layers	No. Filters	Cross-Validation				Test		
		MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
1	8	0.9855	0.01	424.4	1.9	1	31	0
				6.6	918.1		0	120
2	2,4	0.9675	0.0128	420.4	8.5	1	31	0
				10.6	911.5		0	120
3	1,2,4	0.9389	0.0189	407.5	12.3	1	28	0
				23.5	907.7		0	94

Table 38: LSTM (Uni-Directional) results.

Random Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.8766	0.2017	209.8	6.4	1	104	0
			34.2	582.6		0	252
100	0.8883	0.1406	220	18	1	104	0
			24	571		0	252
150	0.8671	0.1464	200.7	2.1	0.9933	104	1
			43.3	586.9		0	251
Subject Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.7968	0.2359	241.4	8.8	1	28	0
			78.6	738.2		0	94
100	0.8537	0.1629	258.4	3.3	1	28	0
			61.6	743.7		0	94
150	0.8970	0.1552	277	2.6	1	28	0
			43	744.4		0	94

Table 39: LSTM (Bi-Directional) results.

Random Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.9791	0.0125	242.7	6	1	104	0
			1.3	583		0	252
100	0.9763	0.0079	242.7	7	0.9933	104	1
			1.3	582		0	251
150	0.985	0.098	242.3	3.5	0.9933	104	1
			1.7	585.5		0	251
Subject Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.9834	0.0111	317.8	5.3	1	28	0
			2.2	741.7		0	94
100	0.9898	0.0063	318.1	2.7	1	28	0
			1.9	744.3		0	94
150	0.9879	0.007	319.2	4.7	1	28	0
			0.8	742.3		0	94

Table 40: C-LSTM (uni-directional) results.

Random Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.9849	0.008	240	1.2	1	104	0
			4	587.8		0	252
100	0.9587	0.1123	230.2	0.3	1	104	0
			13.8	588.7		0	252
150	0.9627	0.038	240.9	10.5	1	104	0
			3.1	578.5		0	252
Subject Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.982	0.028	312.6	0.7	1	28	0
			7.4	746.3		0	94
100	0.9739	0.0173	312.8	4.6	1	28	0
			7.2	742.4		0	94
150	0.9181	0.1468	284.4	1	1	28	0
			35.6	746		0	94

Table 41: C-LSTM (Bi-Directional) results.

Random Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.9931	0.0068	242.9	1.3	1	104	0
			1.1	587.7		0	252
100	0.9911	0.0034	243.2	2.3	0.9933	104	1
			0.8	586.7		0	251
150	0.9829	0.0219	239.5	1.4	1	104	0
			4.5	587.6		0	252
Subject Split							
Hidden Units	Cross-Validation				Test		
	MCC	Std. Dev	Conf. Mat.		MCC	Conf. Mat.	
50	0.9955	0.026	319.3	1.3	1	28	0
			0.7	745.7		0	94
100	0.9967	0.028	319.9	1.4	1	28	0
			0.1	745.6		0	94
150	0.9953	0.016	319.3	1.4	1	28	0
			0.7	745.6		0	94