NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

RICARDO MIGUEL FORTUNATO SILVESTRE

Bsc. in Electrical and Computer Engineering

AI-ENABLED ASSESSMENT OF GAIT PATTERNS VIA SMARTPHONE

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
October, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

AI-ENABLED ASSESSMENT OF GAIT PATTERNS VIA SMARTPHONE

RICARDO MIGUEL FORTUNATO SILVESTRE

Bsc. in Electrical and Computer Engineering

Adviser: Fernando Luís Lourenço Ferreira
Invited Assistant Professor, NOVA University Lisbon

Examination Committee

Chair: Rui Miguel Henriques Dias Morgado Dinis
Associate Professor, NOVA University Lisbon

Rapporteur: João Filipe dos Santos Sarraipa
Assistant Professor, NOVA University Lisbon

Adviser: Fernando Luís Lourenço Ferreira
Invited Assistant Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
October, 2023

AI-Enabled Assessment of Gait Patterns via Smartphone

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.10.2) [1].

Abstract

Musculoskeletal and neurological diseases greatly impact the quality of life of those afflicted. Symptoms typically worsen over time and thus early detection is critical. Gait alterations are a common early warning sign but tests to detect gait alterations require expensive clinical equipment and experienced physical-health workers. Thus, there has recently been great interest in finding lower cost alternatives to data collection and gait alteration detection. We believe that the ubiquity of smartphones and their ability to record motion data reflect an opportunity for democratizing data collection equipment. We found that a particularly smartphone-friendly clinical approach is the dual-task method. Dual-task simultaneously places a patient under both a cognitive and a motor load. The additional cognitive task causes affected individuals to exhibit decreased gait quality allowing for detection of the underlying disease.

In this dissertation, we develop *dualgAIt*, a system for the recording and analysis of a patient's gait using low-cost smartphones. Data acquisition is achieved through an Android application that leverages only common sensors such as the accelerometer, the microphone and the gyroscope. This data is recorded in a database and processed by a desktop module, which displays visualisations of spatial and temporal gait features in a dashboard for a clinician. Additionally, we train supervised classifiers on these features to provide further decision support through an initial guess presented in the dashboard. We implemented and evaluated gait features from prior work and propose *trial distance*, a novel feature based on dynamic time warping which has shown good clinical potential as it is correlated with the cognitive dual-task cost. We validate the system through two sets of experiments consisting of single- and dual-task trials, evaluating both the classifiers and the predictive capacity of the features, and show that the system is able to provide effective clinical decision support. Our work shows that it is possible to reduce the cost clinical diagnosis and improve decision support through the use of data-driven technology, while also providing insights into the most relevant gait features.

Keywords: decision support systems, gait, dual-task, smartphone, sensors, machine learning

Resumo

Doenças musculoesqueléticas e neurológicas têm um impacto substancial na vida dos pacientes. Dado que os sintomas tipicamente pioram com o tempo, a detecção prematura é crucial. Alterações na marcha são um primeiro sinal de alerta, no entanto, testes para as detetar requerem equipamento dispendioso e profissionais experientes. Consequentemente, soluções acessíveis para recolher dados e detectar alterações à marcha têm sido um objecto de estudo. O uso generalizado de *smartphones* e a sua capacidade para capturar dados de movimento apresenta uma oportunidade para democratizar equipamentos de recolha de dados. Consideramos que uma técnica clínica adaptável para uso em *smartphones* é o método de tarefa dupla. Este método combina no mesmo ensaio uma tarefa motora primária e uma tarefa cognitiva secundária. Em indivíduos afetados, a tarefa secundária adicional afeta a qualidade da marcha permitindo detetar a doença.

Nesta dissertação, desenvolvemos o *dualgAIt*, um sistema para recolher e analisar padrões de marcha, usando *smartphones* de preço acessível. A aquisição de dados é conseguida através de uma aplicação *Android* que usa sensores comuns, como o microfone, o acelerómetro e o giroscópio. Estes dados são guardados numa base de dados e processados por um software de *desktop* que extrai características temporo-espaciais da marcha. Estes dados são apresentados num *dashboard* para análise por um profissional de saúde. Usando classificadores de aprendizagem supervisionada, este *dashboard* oferece ainda uma sugestão de diagnóstico. Implementámos e avaliámos características da marcha da literatura e propomos uma nova característica baseada em *dynamic time warping*, que mostra potencial clínico por apresentar uma alta correlação com o custo cognitivo da tarefa dupla. Validámos o sistema através de duas experiências com ensaios de tarefa única e dupla, avaliando tanto os classificadores como a capacidade preditiva das características da marcha. Com esta dissertação, mostramos ser possível melhorar o apoio a decisões clínicas através de tecnologias orientadas a dados, ao mesmo tempo que oferecemos intuição sobre quais são as características da marcha mais relevantes.

Palavras-chave: sistemas de apoio à decisão, marcha, dupla tarefa, smartphone, sensores, aprendizagem automática

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AI Artificial Intelligence *(pp. 26, 33)*
AMW Average Moving Window *(pp. 18, 51)*
AUC Area under curve *(pp. 18, 50)*

CDSS Clinical Decision Support System *(pp. 2, 3, 6, 32)*
cDTC Cognitive Dual-Task Cost *(pp. 3, 66, 68, 80, 84, 86, 87, 89)*
CMI Cognitive Motor Interference *(p. 11)*
CRM Cyclic Rotation Metric *(p. 26)*
CSV Comma-Separated Values *(p. 64)*

DLS Double Limb Support *(p. 8)*
DSS Decision Support System *(pp. 3, 5)*
DT Dual-Task *(pp. 10–12, 28, 46, 58, 63–66, 85, 86)*
DTC Dual-Task Cost *(pp. 11, 63, 83, 89)*
DTREE Decision Tree *(p. 27)*
DTW Dynamic Time Warping *(pp. 2, 3, 17, 21, 23, 24, 26, 47, 56, 58, 76, 81–86, 89)*

ES Environment Sensors *(pp. 8, 9, 32)*

FN False Negative *(p. 30)*
FP False Positive *(p. 30)*

GUI Graphical User Interface *(pp. 33, 35, 66, 67, 71)*

HC Healthy Control *(pp. 25, 28)*
HMM Hidden Markov Model *(p. 27)*
HY Hoehn and Yahr *(p. 7)*

IMU inertial measurement units *(pp. 1, 14, 28, 32)*

k-NN k-Nearest Neighbour *(p. 28)*

LDA Linear Discriminant Analysis *(pp. 27, 28)*

mDTC Motor Dual-Task Cost *(pp. 66, 68, 82, 85, 89)*
MEMS micro-electro-mechanical-system *(p. 13)*
ML Machine Learning *(pp. 6, 26–30, 78)*
MV Machine Vision *(pp. 8, 9, 32, 65, 73, 78)*

NB Naive Bayes *(pp. 27, 28)*
NN Neural Network *(pp. 27, 28)*

PD Parkinson's Disease *(pp. 7, 10, 22, 24, 25, 27–29)*

RDTW Rotation Dynamic Time Warping *(p. 26)*
RF Random Forest *(pp. 27, 28, 64, 65)*
RMS Root Mean Square *(p. 26)*

SLS Single Limb Support *(p. 8)*
ST Single-Task *(pp. 10, 12, 58, 63, 66, 86)*
SVM Support Vector Machine *(pp. 27, 28, 64, 65)*

TN True Negative *(p. 30)*
TP True Positive *(p. 30)*

UPDRS Unified Parkinson's Disease Rating Scale *(pp. 1, 7)*

WS Wearable Sensors *(pp. 8, 9, 12, 14, 32)*

1 Introduction

1.1 Problem Statement and Motivation

Hundreds of millions of people worldwide are currently diagnosed with diseases that affect the peripheral and central nervous system, usually designated *neurological diseases* [2]. There is a wide range of such neurological disorders including epilepsy, Alzheimer's disease, cerebrovascular diseases, multiple sclerosis, neuroinfections, brain tumours and Parkinson's disease. On the other hand, *musculoskeletal disorders* (e.g. rheumatoid arthritis or osteoarthritis) affect a large and growing portion of the population[3]. These conditions can affect every part of the body including muscles, spinal cord, bones, nerve roots, autonomic nervous system, neuromuscular junction, cranial nerves and even the brain. People suffering from these diseases usually have their symptoms progress over time in both quantity and severity, inevitably worsening the patient's quality of life. If detected early, such conditions can often be combated, increasing the quality and longevity of the patient's life[4, 5]. An early indicator of these conditions is often an alteration in the *gait* (i.e. walk cycle) of affected individuals[6].

Gait alteration detection is typically performed by a medical professional who will evaluate patient's motor performance while executing specific tasks [7]. Unfortunately, these tests often require expensive equipment in the form of either cameras[8], treadmills[9] or multiple inertial measurement units (IMU)s[10] (sensors strapped to the body in many locations). Tests based purely on visual observation exist, but these depend on the inevitably biased opinion of the medical professional who will conduct them by following a rating scale such as the Unified Parkinson's Disease Rating Scale (UPDRS) [11]. As such, there is a need for more accessible alternatives to data collection, analysis and gait abnormality detection in order to offer accessible decision support for clinicians.

Mobile health technologies, such as wearable/portable sensors, are able to quantify mobility and have been emerging as complementary clinical assessments as of recently[12]. We believe that the ubiquity of smartphones and their ability to record motion data makes them a particularly promising platform for the democratization of data collection equipment. Modern smartphones, even accessible low-cost ones, are equipped with a suite

of sensors[13] that can be leveraged including microphones, accelerometers, gyroscopes and magnetometers. Prior work[14–16] has also validated that signal processing algorithms can be successfully applied to clean, analyse and detect steps in data collected via smartphones. To this end, we have chosen to focus on the *dual-task methodology*[17] as it has been shown to be particularly adaptable to a smartphone setting[14] and requires no special equipment. The dual-task method simultaneously subjects an individual to a cognitive and a motor test. The incorporation of the cognitive component induces cognitive motor interference, which is accentuated in affected individuals, leading to diminished gait quality and thus facilitating the detection of the underlying condition. Additionally, performance can be compared with single task trials for further clarification.

With this in mind, we seek to answer the following research questions: "Can smartphones combined with the dual-task methodology serve as a lower cost alternative to clinical gait alteration detection?", "What gait features are most useful in a clinical setting?" and "How can Machine Learning classifiers be applied to aid clinical decision support in this context?". We explore these questions through the development of *dualgAIt*, a system comprised of *dualgAIt: Android*, an Android application serving as the single data collection device, a real-time database for data aggregation and a *dualgAIt: Desktop*, a desktop application processes this data and presents it in a dashboard for a medical professional. In the process, we implemented several signal processing, step detection and gait feature extraction algorithms (e.g. step symmetry and cadence) from prior work[14–16]. Using the data collected by the system (in the form of gait features and final diagnosis), it becomes possible to train supervised machine learning classifiers[18] to automatically predict whether a patient is impaired. The aforementioned dashboard presents this prediction, along with visualizations of these gait features, thus offering different levels of clinical decision support. To the best of our knowledge, we are also the first to propose *trial distance*: a novel gait feature based on the application of Dynamic Time Warping (DTW) between the energy signals of single- and dual-task trials.

If successful, such a system would come at a crucial time where the prevalence of these diseases is the leading cause of disability[19] and is still increasing[20] due to an aging population, among other factors, such as (somewhat surprisingly) declining smoke rates. Furthermore, since our system is purely a software artifact built for ubiquitous low-cost smartphones, it can be quickly and effectively deployed anywhere it is necessary. Finally, since it is based on measurement instead of subjective observation, it reduces the bias in the final diagnosis and serves as a more objective Clinical Decision Support System (CDSS) tool. We validated the system through two sets of experiments conducted by trial. Trials were conducted on a group of 6 individuals who voluntarily participated. Since all individuals were healthy, a gait impairment was simulated when needed through the use of a boot to cause a disturbance in the gait profile.

Our first experiment investigates the performance of our signal processing, gait feature extraction and step detection algorithms on both impaired and unimpaired gaits. Algorithms are shown to perform desirably and a small number (~1) of missed step

detections per trial is achieved for the unimpaired gait case. Impaired gaits are found to be more difficult to properly segment, motivating the need for further research into these algorithms. The second experiment investigates the cognitive interference effect of dual-tasking in more detail. We also investigate the correlation of different gait features with Cognitive Dual-Task Cost (cDTC), i.e. how much the cognitive performance of an individual lowers during a dual-task trial. The novel feature we propose (trial distance) was found to have the highest correlation (0.86) with cDTC, while having a strong p-value of 0.03, demonstrating clinical potential and motivating the need for larger-scale trials. Throughout the experiments, the system was not only a helpful and easy-to-use platform that sped-up data collection, but it also correctly labelled 90% of individuals as healthy or impaired. The system also be used to track progress over time, as it allows for comparison of multiple trials in order to evaluate a user's gait change progression. Our results thus show that smartphones may be the answer to the growing need for low-cost gait analysis tools.

1.2 Main Contributions

The major contributions of the dissertation are as follows:

- An analysis of the state-of-the-art on Decision Support System (DSS), followed by some notions on gait assessments, smartphone sensors, signal processing and different supervised learning algorithms.
- The design and implementation of two software artifacts: (i) *dualgAIt: Android*, a smartphone application for data collection and (ii) *dualgAIt: Desktop* a desktop application for data processing, classification and visualization.
- A small-scale replication of prior work on gait features and analyse their performance.
- A novel gait feature (*trial distance*) based on the application of DTW between the energy signals of single- and dual-task trials which shows clinical promise.

1.3 Dissertation Structure

The remainder of this document is organized as follows:

- **Chapter 2** begins with a brief introduction by contextualizing this project as a CDSS system, reviewing what this entails. A lot of effort is then put into summarizing the literature on gait analysis assessment, the main topic of this dissertation. Then, the state-of-the-art in smartphone sensors is presented, along with some other work on necessary topics such as signal processing and machine learning.
- **Chapter 3** presents the design of the *dualgAIt* architecture at a high level. Following, a list of the proposed architecture requirements is reviewed. This chapter ends by

creating the bridge between the architecture and the implementation, describing the general objectives.

- In **Chapter 4**, the final implemented version of the architecture is presented. For each module, we detail how it was implemented and it's integration with the rest of the system. We validate the signal processing and gait analysis algorithms by presenting plots of the analysis on example data.
- In **Chapter 5** the evaluation and results will be presented. A mix of tests is prepared in order to evaluate the performance of the general system. Then, two technical experiments are showcased and discussed.
- In **Chapter 6** provides an overview of the strengths and gaps in the developed prototype. This leads to the future work discussion on improvements and future research directions.

2

State of the Art

In this chapter, our review of the State of the Art is presented. We start by positioning our work in the context of DSSs in Section 2.1. Then we introduce prior work on gait, gait analysis and types of assessments in Section 2.2, where we also introduce the dual-task method. In Section 2.3, we compile and review different gait-related topics. We first discuss the smartphone as a data collection tool in Section 2.3.1. Then in Section 2.3.2, signal processing and analysis methods used throughout the dissertation to analyse gaits are introduced. Finally, in Subsection 2.3.3 we provide a high-level introduction to the needed machine learning concepts.

2.1 Decision Support System

A DSS is a system designed to support and assist with decision making. It is an information system usually used to give responsible decision valuable insights as well as tools to analyse data[21]. These systems are based on decision-making and are usually used to support humans, not replacing them. The process of the decision is the knowledge-based choice among different alternatives[22]. Decisions reflect not just evidence, the logic and the probability but also the goals we are trying to achieve, our values and our available resources. Decisions can also come from different sources: deduction, where given a cause the consequences are inferred, abduction, when a list of chances explain the reason behind the cause and induction, when rules are defined based on observation. To the elements responsible for the decision making process, we denote of decision making factors[22]. The typical architecture of a DSS can be seen in Figure 2.1. It usually conveys system databases, users, software tools for database and model management, a communication structure and the system models.

2.1.1 Clinical Decision Support Systems

Moreira et al. [23] emphasizes the importance of DSS systems in the healthcare field. The author defends that to be able to provide high-quality care, the medical professionals not only need clinical skills but also to have an effective way of managing information.

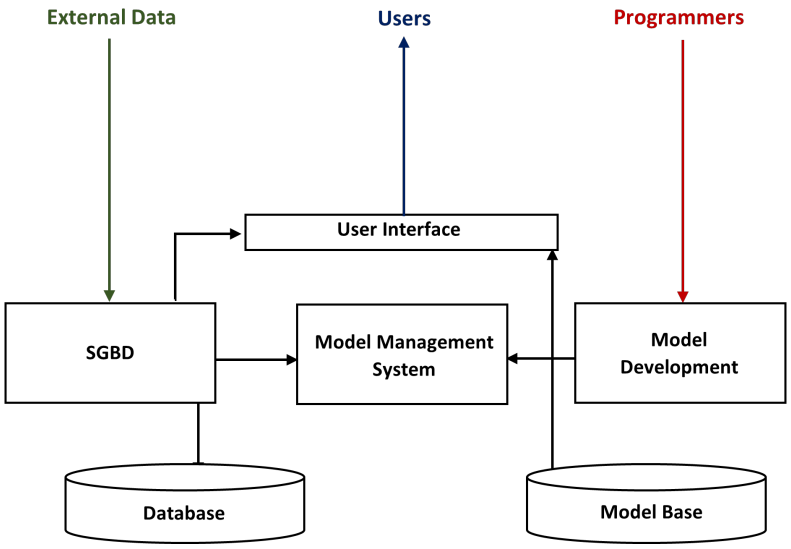

Figure 2.1: Conceptual architecture of a decision support system. Based on [22]

The use of knowledge- and data-driven systems can provide good insights to healthcare contributing to improve medical diagnosis. To provide insights, different techniques can be applied to generate model-driven systems such as mathematical programming and simulation.

Sloane and J. Silva [24] discussed about the importance of artificial intelligence in the context of CDSS systems. Applications include pattern detection, expert medical advice and automating surgical instruments. These systems can either be expert systems, using rule-based decision support by defining procedures for certain conditions or thresholds, or Machine Learning (ML)-based, using statistical inferences retrieved by finding patterns within data. CDSS software can provide reminders for preventative treatment and alert for specific medical events. As a result, employing a CDSS may save expenses and boost productivity.

2.2 Gait Analysis and Assessment

2.2.1 Gait Overview

Gait encapsulates our physiological systems capabilities while being, at the same time, a promising indicator of our health and functional status. Given that diminished motor skills are common on individuals with Parkinson's disease, gait-related features can be useful to access the progression of the disease. Alterations in these parameters are crucial to detect gait abnormalities and unlocking accurate diagnosis of neurological disease or of their progression over time. The assessment of what is disrupting a patient to walk in a certain way is defined as clinical gait analysis, which is a valuable resource for decision-making in a clinical setting [15].

As presented in Figure 2.2, these gait assessments started by being observational,

which are highly subjective given it directly depends on the previous experience of the professional analysing it.

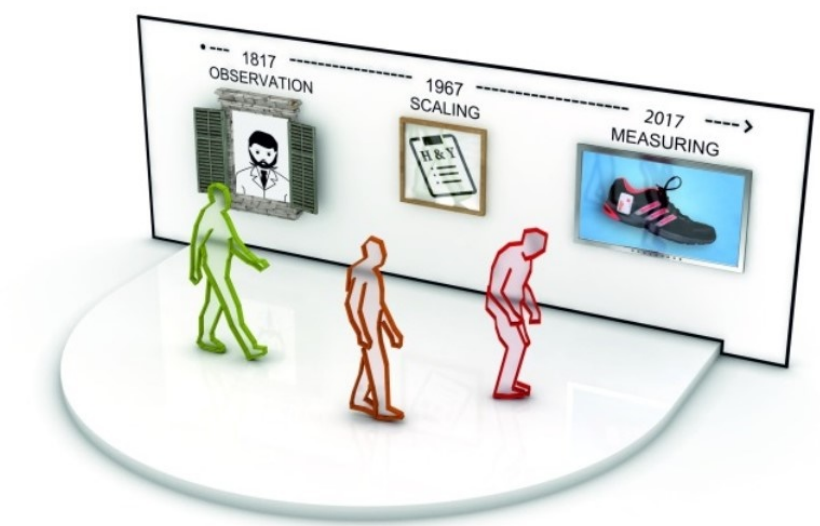

Figure 2.2: Evolution of gait assessment. Retrieved from [25] - Copyright licence in Annex II

Later, different standardized scales were developed to quantify different degrees of motor capability in Parkinson's Disease (PD) patients, such as the UPDRS[11] and the Hoehn and Yahr (HY)[26]. In the recent decade, there has been an increasing demand for instrumented assessments which even though objective, require measurement systems that can be highly complex [25]. When it comes to gait analysis, we can also highlight different approaches. Impairment-focused gait analysis focuses on the definition of the impairments most likely affecting the patient's gait performance. Function-based gait analysis aims to understand how well can patients achieve certain functions, not considering what impairments could be affecting them. Lastly, an approach in which the aim is to provide the professional with certain data that represents how a patient gait is progressing over time is defined as monitoring-focused gait analysis.

2.2.2 Gait Cycle

The gait cycle is the main subject of gait analysis. A trial will usually contain multiple gait cycles therefore revealing information of a patient's gait, or in other words, how a patient walks. By comparing several gait cycles, we can perceive how variable the pattern is. The cycle itself is usually depicted as the moment when a foot makes direct contact with the walking surface, normally with the heel, followed by a step of the opposite foot, until this foot reaches the ground[15]. Two main phases can be defined, based on a single limb: the stance phase, when the foot is in contact with the ground, and the swing phase, when the foot is moving through the air, not in contact with the ground [15].

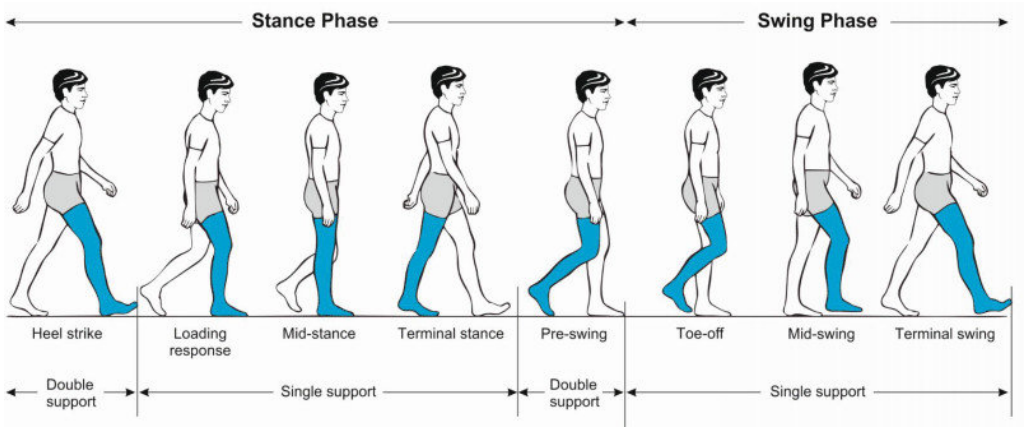

Figure 2.3: Gait cycle. Retrieved from [27] - Copyright licence in Annex II

As for the support, we can also distinguish two different settings: the Single Limb Support (SLS), when a patient has only one foot on the ground, and the Double Limb Support (DLS), when both foot are on the ground serving as support. While walking, when one foot it's during the swing phase, SLS occurs. On the other hand, DLS happens on the moment between the stances phases of each footstep. If someone's running, DLS does not occur. This makes DLS a great way of addressing the variability pattern of a patient's walking speed [15]. These different phases of a gait cycle can be visualized in Figure 2.3.

Step length is a representation of the distance between both feet after taking a step. As one would expect, both feet cover equal distances when the gait pattern is symmetric. This isn't observed otherwise. Stride length is the distance covered for a single gait cycle, measured by two steps. Stride time is the temporal equivalent of stride length, defined as the time taken to walk the stride length distance, the duration of one gait cycle. Knowing that the stance phase usually takes 60% of the gait cycle and the swing phase occupies the remaining 40%, we can combine this information with stride time to compute the duration of the stance and swing parts. Cadence, which is associated the number of steps per minute, is useful to calculate the patient's walking speed[16]. As gait general parameters, we can define step symmetry, measuring the disparity between feet and also gait regularity, which allows later comparison of sequences of gait or step cycles[15].

2.2.3 Gait Assessment Approaches

The gait cycle assessment can be performed in different approaches, not only in the type of data collecting system or the setting in which it is performed but also in the type of procedure defined for the patient to perform.

2.2.3.1 Gait Analysis Methods

There are three main approaches when it comes to the way we choose to obtain our gait data: Environment Sensors (ES) based, Machine Vision (MV) based and Wearable

Sensors (WS) based [28, 29]. The MV approach consists in the usage of cameras prepared in the room where the patient will perform a certain task, such as walking, for the assessment. Even though this approach is costly due to the highly specialized equipment involved, it has high accuracy. The computations necessary to process the images are also computationally expensive [30]. Even with these disadvantages, this approach is still considered the gold standard for gait assessment [25]. A reviewed study [8], developed an home integrated system for gait analysis by combining Microsoft Kinects and cameras, tracking the patient throughout the day detecting whether there was risk of fall.

The ES approach is usually done in a fixed location, for example, a treadmill can be used to simulate the task[9]. By using sensors in the fixed location, gait-related parameters can be collected. This approach doesn't require computationally expensive algorithms but treadmills can alter the natural gait pattern of the subject, which can be a disadvantage.

Lastly, the WS approach is of most interest for this dissertation context, since using today's smartphones to collect data falls into this category. It removes the problem of being constrained to a certain location and it is usually inexpensive in comparison to the previously seen approaches [31]. This price cut usually comes with the downside of a worse signal quality, even though that with the recent advancements in technology and filtering techniques, this approach has proven to be effective for gait analysis. A problem with WS is usually the obstruction of the patient's movement by their attachment. These sensors can be uncomfortable and difficult to set up. A study by Gouwanda and Arosha Senanayake [32] attached four wireless gyroscopes to patients with the objective of collect different gait parameters. This kind of approach also needs a prepared environment to assess and analyse the obtained signals.

This problem can be solved with a resource that most of us carry in our pockets everyday nowadays, the smartphone. Also, some of the most used sensors in the WS approach are accelerometers, gyroscopes and magnetometers [28], embedded in most of today's shelf smartphones. This will be revised in more detail in Section 2.3.1.

2.2.3.2 Supervised vs Unsupervised

For better understanding of both of these concepts, let's start by defining them. A supervised assessment is the conventional mode of accessing mobility in a clinical setting, such as a laboratory or a hospital, using standardised and mainly qualitative methods. On the other hand, an unsupervised assessment can be defined as the assessment of mobility in a daily living environment and is usually quantitative [12]. It is a continuous procedure possible by the usage of mobile health technologies.

Given that a supervised gait assessment is performed in a standardised environment, usually performing a task without distractions, these end up providing information of someone's best performance instead of their daily living performance. This means it reflects someone's capacity or ability to do some task and not how well they are able to do that exact task. Associated with the supervised approach, there are several factors that

can contribute to data variability as the Hawthorne effect (the change in results due to the fact that patient is aware of the tests) and the patient's alertness and motivation to perform the trial [12]. These can contribute to a worst reflection of the real performance of a patient during real daily activities.

Unsupervised gait assessments are usually only used as a complement to clinical assessments but promise great data viability given they can overcome limitations of the conventional clinical assessments by intervening during the patient's daily living environments, often capturing fluctuating and rare events. Not only this approach can save time but it can also cut costs which would otherwise be spent in expensive equipment and/or professionals. In rural areas or developing countries, this has proven to be even more important because of the lack of healthcare professionals relatively to the population size. In this kind of approach, patients also feel more actively involved, using their own devices, many times receiving feedback on their performance. These patients are measured for longer periods of time but have the ability to move freely and unsupervised. The variable paths with different obstacles encountered on the daily activities, in contrast with the clinical setting, also induce large asymmetry and variability in mobility patterns [12]. Performance for patients with PD and older adults seem to be comparable under both conditions, namely in gait speed.

2.2.4 Dual-task Assessments

2.2.4.1 Single-task vs Dual-task

In daily living, it is usual for one to be in multitasking situations but uncommon in supervised assessments [12]. This might mean that a single-task trial might not be best way to access a patient's performance or capability. A Single-Task (ST) trial for gait assessment should consist in asking a patient to perform a single motor task by itself. On the other hand, a Dual-Task (DT) trial entails the performance of two different tasks, at the same time. Usually a combination of two different motor tasks or a combination of a motor task with a secondary cognitive task [33].

As previously mentioned, performing simultaneous tasks may result in an impairment in patients with neurological diseases. This variation in results is usually called interference or DT effect and is mostly reflected in the primary motor task namely on postural control and quality of gait. This reflects a possibility to detect gait abnormalities in patients suffering from neurological and muscoskeletal. Because older adults and patients with other diseases may also see their performances affected while performing a dual-task trial, further differentiation is necessary. Studies also show that individuals with PD prioritize cognitive tasks over motor tasks, which could be denoted as an inadequate task prioritization. Even though earlier studies considered DT as risky during gait and balance training, recent articles reveal that the usage of DT in training doesn't seem harmful which means the same could apply for a gait assessment trial [33].

2.2.4.2 Dual-task cost

The division of attention between tasks caused by dual-tasking can have effects on both cognitive and motor performance usually denoted by Cognitive Motor Interference (CMI)[34]. There are different ways of performing dual-task gait assessments but also to quantify the associated performance. According to Leone et al. [35], in the clinical context multiple methods can be used to assess the mental state, attention and focus but there aren't a lot of norms defining these trials. The author defines a set of single cognitive tasks that have proven to be worth of study. Attention tasks such as answering to a certain pitch and answering, some memory tasks as well as verbal fluency are recurrent tests in this context. Arithmetic tasks such as subtracting by 3 and subtracting by 7 for 15 seconds have proven worthy of study because they are comparable and have different levels of cognitive demand because further mental regrouping is required to compute the results for the serial subtraction of 7. The subtraction tasks are simple tests to administer and easy to understand to the one performing it, not taking much time. The author also noted that it is possible to calculate the Dual-Task Cost (DTC) for both cognitive and motor interference. According to the author, the DTC may be computed in the following manner:

$$CognitiveDTC = \frac{STaccuracy - DTaccuracy}{STaccuracy} * 100 \tag{2.1}$$

$$MotorDTC = \frac{STfeature - DTfeature}{STfeature} * 100 \tag{2.2}$$

If the DTC value is greater than zero, the lower the DT ability this individual has[35]. The opposite also happens, where a low or negative DT value, represents higher dual-tasking ability. These formulae above were also supported by Postigo-Alonso et al. [34], who have studied the impact of different cognitive tasks.

Leland et al. [17] conducted a dual-task study to verify it's effects on motor, cognitive and balance in a group of veterans with traumatic injuries. The author reported that there was a 20% decrease on the stride length feature while dual-tasking, concluding that the DT paradigms slowed the gait down in his group of study. The author also denoted that incorporating dual-task into the rehabilitation has the capability of improving gait and attention capabilities over time. Some important reference values are also extracted from this article review such as the mean gait parameters while in single and dual-task conditions in a 6 meter long pathway. For cadence there was only a 2.23% increase while under DT conditions. For stride length and stride time, there weren't any significant changes. On the other side, there was a 20.23% increase on the number of steps.

The author Manor et al. [14] explored how feasible is acquiring gait data remotely using a smartphone testing the developed system in a group of healthy young adults. According to Manor et al. [14], serial subtraction is the most common cognitive task used in dual-task laboratory studies. An interesting note by the author is that the DT paradigms should affect the gait of healthy adults. According to the author, the duration of the strides

increased when under dual-task conditions (DT: mean 1.18 ± 0.16 seconds;ST: mean 1.05 ± 0.16 seconds.

2.3 Related Concepts

2.3.1 Smartphone as a Data Collecting Tool

As referred previously in section 2.2.3.1, the WS approach is of most interest for this dissertation context, where we explore the possibility of using smartphone as the data collecting tool. Smartphones have constantly increased in popularity over the last years. The expected number of acquired smartphones is of more than 3.8 billion by 2021 [36]. This expectancy combined with modern embedded sensors and communication technologies, enable a remote and continuous monitoring of a patient's health with barely any cost associated. Wearable sensor systems can be useful to follow closely the progression of gait problems by providing spatio-temporal parameters [37]. Also, in the latest years, an increase happened in moving into using sensors embedded in smartphones and smartwatches as gait analysis protocols [38][31].

Consequently, a mobile system that can run the aforementioned tests and collaborate with a central server for persistent storage and remotely accessible data would be useful for the ability of self-assessment, cost-effectiveness and progress tracking.

Smartphone's embedded sensors permit the active and passive sensing of multiple parameters and have been seeing significant advances when it comes to cost, dimensions, energetic needs and more importantly, sensitivity, allowing for multiple use-cases to monitor health issues [39]. Some examples of these are the usage of camera and microphone for monitoring cardiovascular activity, the camera, light sensor and GPS for cognitive function and mental health monitoring, and the usage of motion sensors, as the accelerometer, gyroscope and proximity sensor, combined with GPS for daily activity and fall monitoring.

According to Bouça-Machado et al. [38], out of all the analysed studies in "preclinical development and testing phase" (n = 23), only 8.7% (n = 2) used smart devices (using an accelerometer and gyroscope) as their data collecting tool for the studies, which might represent an opportunity for further research.

2.3.1.1 Motion Sensors

Recent smartphones are equipped with a long list of embedded components and technology which allows users to continuously collect data in an automatic way. Even if unnoticed, a lot of these sensors are used in a regular basis such as sensors related to proximity, GPS, accelerometer, humidity, microphone, temperature, pressure, fingerprint, magnetometer, touch sensitivity, gyroscope, along many others.

Given our work lands into the activity monitoring approach, as previously referred, inertial sensors as the accelerometer and gyroscope will be the main target of analysis. To allow cognitive testing, in dual-task approaches, the smartphone's microphone is used.

In order to perceive gait data, raw data about motion can be extracted from inertial sensors as the accelerometer and the gyroscope. This raw data can then be interpreted and analysed to extract gait-related features.

Smartphones incorporate three-axis accelerometers used to measure inertial dynamics in x, y, and z axis, measuring acceleration in m/s^2, as depicted in Figure 2.4. These work based on the basic principal of acceleration meaning the accelerations in the three axis reveal the changes in the linear velocity in the 3D space, therefore representing the user's movement by measuring the smartphone's acceleration, including gravity [40]. This way, accelerometers provide measurement of the dynamic accelerations of a body submitted to external forces, retrieving data about the device's orientation in relation to gravity [38]. Accelerometers function based on their electro-mechanical principal and are usually classified as piezoelectric, piezoresistive, capacitive and extensimetric. A transducer is part of their composition, detecting movement and transforming a mechanical signal into an electric signal, in the form of a micro-electro-mechanical-system (MEMS) [37].

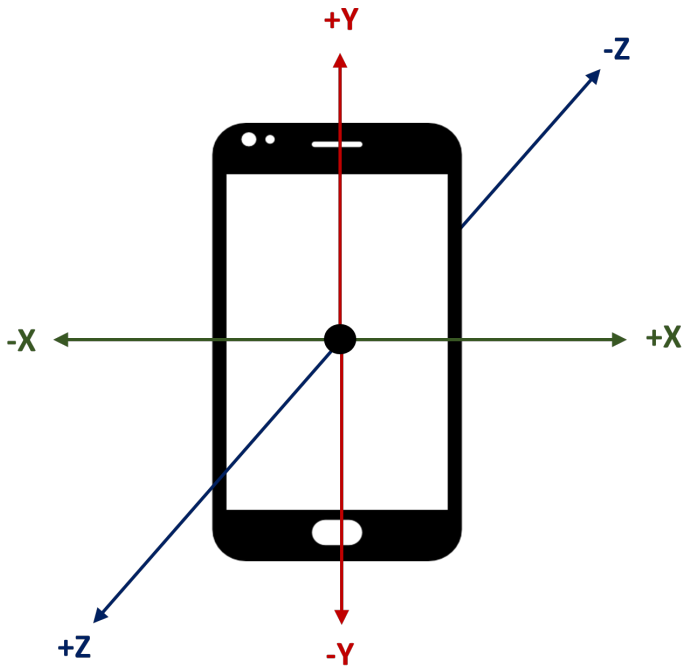

Figure 2.4: Smartphone's 3-D accelerometer axes

On the other hand, the smartphone's gyroscope is responsible for measuring angular velocities for three axis: yaw, pitch and roll [38][37], as presented in Figure 2.5. It's readings are in radians per second [41]. Based on it's rotation in space, it is useful to describe the body's movement pattern. Being a MEMS device, gyroscopes can be magnetic, cryogenic and electrostatic, using different electro-mechanical principles [37].

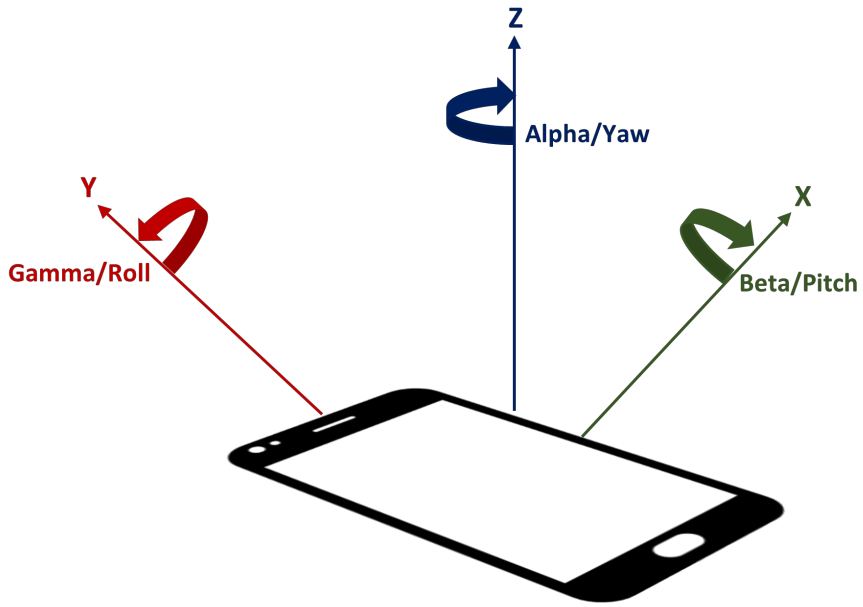

Figure 2.5: Smartphone's 3-D gyroscope axes

2.3.1.2 Reliability for Gait Assessment

According to Bouça-Machado et al. [38], to consider a measurement as accurate in a clinical sense, it shouldn't be above a 1% error deviation from the staple reference value. This is relevant, since according to Höchsmann et al. [42] studies using smartphones did not cross this line for step detection at a gait velocity of 4.8 km/h, while placed on shoulder bag or backpack and at 6.0 km/h for all smartphone positions. On the other hand, it also revealed that at a 1.6 km/h, a 3% error existed.

Proessl et al. [43] investigated the agreement of different spatio-temporal gait measures between inertial sensors, usually in the IMU form in WS approaches and a smart device. It concluded that during prolonged walking, the smart device was able to measure multiple spatio-temporal gait metrics as accurately as the IMU. This study placed a smart device strapped to the lateral malleolus (slightly above the feet) on each leg, and multiple wireless inertial units were attached on the sternum, wrists, lower back and dorsum of each foot. It concluded that stride duration, cadence, stride length and gait speed revealed high associations ($r < 0.9$) between the devices therefore concluding that smartphones can reliably collect different spatio-temporal gait features.

Another study by Furrer et al. [44] evaluated the concurrent validity of a smartphone application, for vertical center of mass displacement and step duration detection, by comparing the obtained values with a motion capture system. Most of the variables derived from the application had excellent reliability, with all variables correlating significantly with the ones collected from the motion capture system (from 0.61 to 0.92). It also concluded that there is a lot of potential for a smartphone application to become a valid tool for gait assessment.

Silsupadol, Teja, and Lugade [45] studied the reliability of a smartphone approach using an accelerometer to quantify spatio-temporal gait parameters. An important research point of this paper was the different placements for the smartphone that were used during the data acquisition, focusing on the body, inside a bag, attached to the belt, inside a pocked or on hand. This study focused on obtaining step length, step time, gait velocity and cadence, concluding that gait smartphone-base assessments get better results attached to the body, attached on a belt and inside a bag (15 cm width in relation to the 14.33 cm length smartphone). Manor et al. [14] also demonstrated that using a smartphone in the pockets of the trousers can viably be used to retrieve information about stride time.

According to Nickel [46], the use of mobile phones for obtaining a data collection for cycle extraction can lead to lower and irregular sampling rates. An important point to refer is that smartphone itself can provide synchronization between sensors, even though it can be slightly affected, in tolerable degrees, by the data recording process through the Android interface.

2.3.2 Data Processing and Analysis

Techniques to process signals as well as the selection of specific features play critical roles in designing a reliable and efficient system. Topics on some pre-processing and signal techniques are reviewed alongside with the main gait features extraction and machine learning algorithms applied in this context. In Figure 2.6, we can see the different types of gait analysis ramifications.

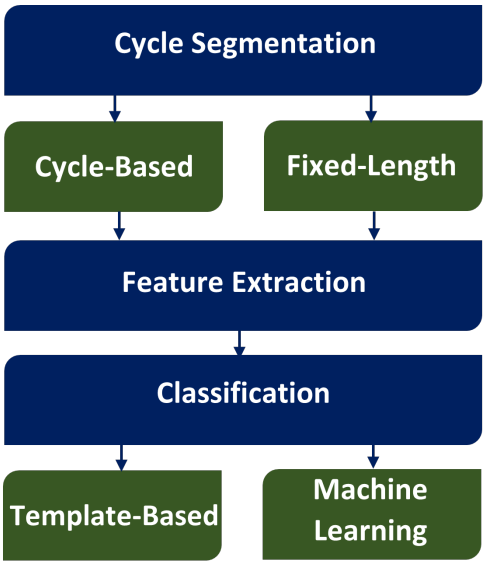

Figure 2.6: Gait analysis ramifications. Based on [46]

2.3.2.1 Signal Preparation

Signals from sensors acquired during walking require pre-processing and signal analysis techniques to extract value from them. To acquire valuable information from sensor data, signal pre-processing is necessary, which includes techniques such as filtering, normalizing, windowing and centering around zero and segmenting signals.

Filters can be used for two main general purposes: signal separating, which is useful when the raw signal is corrupted by some sort of noise or interference, and signal restoration, useful when a signal has suffered some sort of distortion usually due to poor equipment [47]. For example, a study by Rovini et al. [7] applied a fourth-order low-pass digital Butterworth filter with 5 Hz cut-off frequency to remove high-frequency noise and frequency band associated to pathological tremor (3.5-7.5 Hz). Silsupadol, Teja, and Lugade [45] applied a Butterworth 4th order low-pass filter with 20 Hz cut-off frequency to accelerometer data in the three-axis. The same study also applied a high-band pass Butterworth 4th order filter with a 0.1 Hz cutoff frequency for remove integration drift. This was necessary after the double integration of the vertical acceleration data with the purpose of obtaining vertical position.

Windowing is extremely necessary. For example, for step segmentation, we need to be moving on the signal on a window that resembles the gait cycle. This necessity was enforced in the work of Perez and Labrador [15].

Even though **normalizing** is seen in some of the reviewed articles as an optional step in gait cycle processing when comparing multiple signals, it can be very useful. It's hard to compare two things that can't be put side by side. Nickel [46] refers that different lengths of extracted cycles can be an issue. A normalization of cycle length is required if for example, in the classification process of gait cycles, because some of the distance functions can require feature vectors of the same length as input.

Centering around zero might be useful for multiple reasons. For example, Nickel [46] refers that, by collecting sensor data with a smartphone, the acceleration that the mobile phone measures at a stable position (where there is no movement) is not exactly zero or gravity, as it should be in a vertical direction, and it is also not steady over times. This might be due to the mobile phone's poor calibration. The data is centered around zero to lessen the impact of this phenomena. This is accomplished by removing from the data the mean of the walk's acceleration values.

The author Marron et al. [48] developed a gait based system for activity detection in in-door environments. In agreement with the previous author, Marron et al. [48] estimated the mean of the acceleration in a segment of sure movement in order to perform a calibration routine in order to compensate for the bias introduced by the sensors. For each accelerometer axis (a_x, a_y, a_z) and with N being the total number of samples in a given segment, the estimation of the mean acceleration for that segment can be computed by using the Equation 2.3:

$$\overline{A(\alpha)} = \frac{1}{N}\sum_{i=0}^{N-1} a_{\alpha_i} \tag{2.3}$$

That ends up calculating A_$bias$ by using Equation 2.4 :

$$\overline{A\_calib} = \sqrt{\overline{A(x)}^2 + \overline{A(y)}^2 + \overline{A(z)}^2} \tag{2.4}$$

2.3.2.2 Gait signals segmentation

Segmenting is used if, for example, we want to analyse a certain signal and want to start by dividing it into parts with identical traits. This is done by signaling where an event starts and ends [47]. Most systems in gait assessment can be divided in two types: cycle-based segmentation, that needs to locate one step, or a complete gait cycle to process that data, and fixed-length time segmentation, that does not need to locate it within the sensor data. The first type is based on a estimation of cycle length. Maxima and minima of the signal are used to differentiate steps. The second type divides the data into segments with a fixed length. For example, Nickel [46] used a fixed-length time segmentation algorithm with segment lengths between 2 and 7.5s was used. These segments suffered from a overlap of 50%.

According to Schlachetzki et al. [25], the segmentation of the gait signals into single steps is crucial for gait analysis. In this study, a tri-axial accelerometer and gyroscope were attached to the side of each foot in order to determine gait parameters. To isolate strides from gait-related gyroscope signals, a stride detection algorithm was used to identify and segment strides, by using a multi-dimensional DTW. The algorithms used were previously reported and validated by Winkler, Klucken, and Eskofier [49]. Gait events were then detected based on the segmented strides enabling the computation of gait parameters.

Part of the research work of Perez and Labrador [15] was implementing a solution to detect the steps in a real-time fashion. Perez and Labrador [15] based the step detection implementation on previous work by Marron et al. [50]. The original article uses human activity to segment the signals. In order to improve the original algorithm, the author introduced state variables such as "prev high", that indicates that the previous sample had a high peak, "look fw", that indicates that the search window is looking to find a low peak and a counter that counts the remaining samples in the searching window. This alterations improve the efficiency and the precision of the detection by keeping the total count of steps in between segments.

The procedure processes acceleration data from the three axis in order to identify steps in the data. For a certain segment of N samples of the signal of the acceleration, $A = \{a1, a2, ..., a_N\}$, where each $a_i = \{a_{xi}, a_{yi}, a_{zi}\}$, the signal's energy can be computed by the Euclidean norm. It starts by computing the energy of the signal and the associated bias of the sensor, using Equation 2.5 and Equation 2.6, respectively.

First the author starts by calculating the magnitude of the acceleration signal by making the sum of the squares of each accelerometer axis for each sample.

$$E[a_i] = \sqrt{a_{x_i}^2 + a_{y_i}^2 + a_{z_i}^2} \tag{2.5}$$

Then, the bias of the acceleration is obtained by computing the mean value of all the samples.

$$A_{bias} = E[\overline{A}] \quad \text{where} \quad \overline{A(\alpha)} = \frac{1}{N}\sum_{i=1}^{N} a_i \tag{2.6}$$

Finally, using the previously calculated energy and bias, we compute the Average Moving Window (AMW) filter, for each sample, using Equation 2.7.

$$AMW[a_i] = \frac{1}{2 * \omega + 1}\sum_{j=i-w}^{i+w} E[a_j] - A_{bias} \tag{2.7}$$

The pseudo code for the algorithm developed by Perez and Labrador [15] can be seen in Code 2.7.

The presented procedure computes the energy and bias of the accelerometer before smoothing the signal using an AMW filter. For each sample, the AMW filter is applied. Using the state variables, a step is essentially detected when the "look fw" is true, meaning we are searching the window for a low peak, and the value of "s" is lesser or equal to -T. Or in other words, when a high peak is followed by a low one within the search window, a step was detected. The procedure will go on until all the samples are looped. The authors applied this algorithm for a sampling rate of 50 Hz limiting the segments to 500 ms. The threshold (T) for the energy was of 0.45 and the window size (ω) for the filtering to 5.

Manor et al. [14] created a smartphone application that using the sensors collects movement during both normal walking and dual-task setup. They placed the smartphone in the pants pocket. This paper also compares the obtained results with the gold-standard, the GAITRite mat. The project collected 3-D accelerometer and a 3-D gyroscope to collect data at a 100 Hz frequency and used turn detection as a way of segmenting the gait signals and remove turning intervals from the computation of stride time. This was a necessity because the turns would introduce unwanted variation into the data. By segmenting the periods without the turns, the authors were able to obtain preciser data about stride time resulting in an improved gait analysis system. Because the authors decided to transform the device coordinate referential to an earth coordinate referential, the z-axis ended up staying vertical to ground. One very interesting feature of this project was that the authors used an automatic way of detecting turns. When turning around, as in a 180º turn, angular velocity highly increases in one direction, as depicted in Figure 2.3.2.2.

To compute the total angular distance, in a particular direction, one should compute the Area under curve (AUC). This is done by integrating the angular velocity time series

```
1: prev_high, look_fw ← false
2: counter ← 2w
3: procedure StepDetector(A)
4:     Calculate Energy(Eq. 7) and Bias(Eq. 8) for A
5:     for i = 1, N do
6:         s ← AMW[a_i]
7:         if look_fw = false and s ≥ T then
8:             prev_high ← true
9:             next iteration
10:        else if look_fw = true then
11:            counter ← counter − 1
12:        end if
13:        if prev_high = true and s < T then
14:            prev_high ← false
15:            look_fw ← true
16:        end if
17:        if look_fw = true and s ≥ T then
18:            look_fw ← false
19:            prev_high ← true
20:            counter ← 2w
21:            next iteration
22:        end if
23:        if look_fw and s ≤ −T then
24:            Step detected
25:            counter = 2w
26:            look_fw ← false
27:            next iteration
28:        end if
29:        if counter = 0 then
30:            prev_high ← false
31:            look_fw ← false
32:            counter ← 2w
33:        end if
34:    end for
35: end procedure
```

Figure 2.7: Step detection algorithm by [15]

within an interval that is limited by two consecutive zero crossings as given by Equation 2.8.

$$AUC = \int_{zc_1}^{zc_2} \omega(t)\, dt \tag{2.8}$$

Where zc_1 is the first zero crossing and zc_2 is the second zero crossing. Then the time span between the two zero crossings is computed by Equation 2.9:

$$t_{span_{zc}} = t_{zc_2} - t_{zc_1} \tag{2.9}$$

The authors established that the product of the time span and the AUC must surpass the limit of 2.00 radian-seconds. To be considered a turn, the condition in the Equation below must be respected:

$$AUC * t_{span_{zc}} > 2.0 \tag{2.10}$$

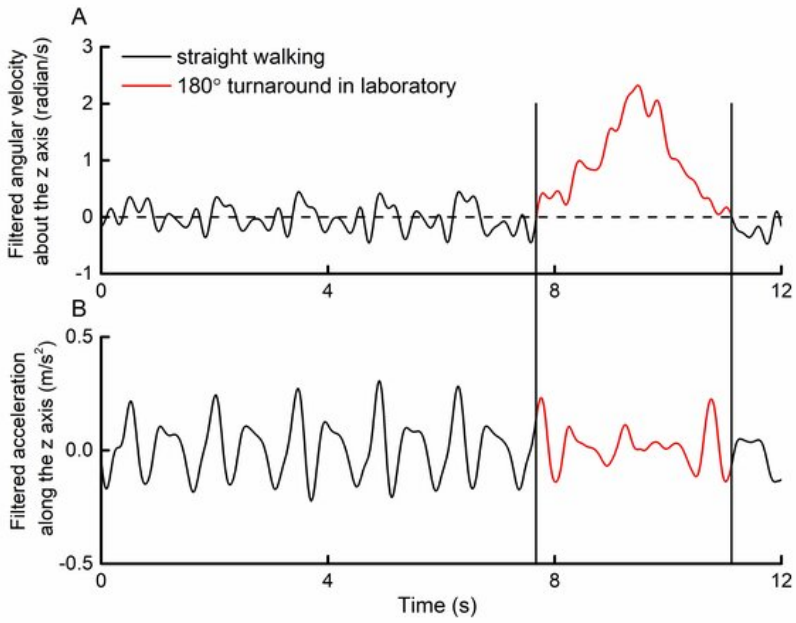

Figure 2.8: Angular velocity and acceleration of the vertical axis while walking and turning 180º. Retrieved from [14] - Copyright licence in Annex II

The paper concludes that smartphone is reliable for assessing stride time, during single and dual-task walking in laboratory and natural environments.

2.3.2.3 Auto-Correlation

Serial correlation or **auto-correlation** is defined as the correlation between a signal and a copy of it which is delayed. This technique is exceptionally useful to discover recurrent patterns in a certain signal. By combining the technique with the signals obtained from the smartphone's sensors it has been proven possible to obtain several gait related parameters [15].

It has been verified that if the coefficient of the first found dominant period is a low value, there is either a low step regularity or a repeating asymmetry between right and left steps. To differ these, the second dominant period should be examined. If it is also a low value, we should be facing low step regularity because there is irregularity between strides. On the other hand, if the second dominant period is higher than the first dominant period, we should be facing systematic asymmetry between different feet steps. If the values for the first and second dominant periods are close to 1.0, we can assume regularity in relation to steps and strides, respectively. The ratio between these two periods can also be used to retrieve information about symmetry, the closer to 1.0, the more symmetrical [15][51].

Cadence is also a gait parameter that can benefit from the auto-correlation technique. Cadence can be defined as the step rate per minute. The quantity of steps (NS) can be represented as the quantity of samples (TS) divided by the amount of coefficients (NC)

between the zero phase shift and the first dominant period. The period (T) on the other hand can be obtained by the total samples divided by the sampling rate (SR) [51] [16]. Cadence (C) is computed using the following expression in Equation 2.11:

$$C = 60 * \frac{NS}{T} = 60 * \frac{\frac{TS}{NC}}{\frac{TS}{SR}} \tag{2.11}$$

The auto-correlation sequence can be biased or unbiased. To compute these coefficients, the raw auto-correlation coefficient (A) must be calculated first. This can be obtained by the sum of the products of a time series X_i with $i = 1, 2, ..., N$ multiplied by a delayed copy of the time series X_{i+m}, where the phase shift in the number of samples is given by the delay parameter (m). This is represented by Equation 2.12:

$$A = \sum_{i=1}^{N-|m|} x_i x_{i+m} \tag{2.12}$$

Furthermore, to be able to compute the biased approach (Equation 2.13), we divide the raw auto-correlation coefficient (A) by the number of samples (N) in the time series:

$$A_{biased} = \frac{1}{N} \sum_{i=1}^{N-|m|} x_i x_{i+m} \tag{2.13}$$

The unbiased approach is often used in these studies because the biased approach can generate attenuation of the coefficient values after the zero phase shift, when using a limited amount of data. Due to this, to be able to get the dominant period values, both studies [51] [16] used a normalized unbiased auto-correlation function, which was computed by dividing the auto-correlation coefficient, in it's raw form, for the quantity of samples associated to that interval of the time series that overlaps and the delayed copy in the following manner 2.14:

$$A_{unbiased} = \frac{1}{N - |m|} \sum_{i=1}^{N-|m|} x_i x_{i+m} \tag{2.14}$$

Perez and Labrador [15] proposed a system that uses smartphone and its sensors to realize gait assessments for patients with motor disabilities arguing that the developed technology eliminates subjective errors from functional gait assessments. To assess the progress of the gait after the surgeries, the author uses different metrics such as DTW and correlation. This author defends the importance of evaluating symmetry within the gait analysis as well as the regularity by analysing sequences of steps. By doing this, the author extracts different temporal features such as stride time, cadence and walking speed. To do this the author followed the work of Moe-Nilssen and Helbostad [16] and Gouwanda and Arosha Senanayake [32] who defend that the autocorrelation function can be used to reveal the regularity between steps, represented by the coefficient A_{d1}, and the regularity between strides, represented by A_{d2}.Values for these coefficients near the value

of 1.0 represent regularity. The ratio of A_{d1}/A_{d2} close to 1.0 indicates perfect symmetry between right and left foot.

By using an unbiased normalized autocorrelation function, where X_i is the i_th value of the raw signal accelerometer, with N being the sample total number. m represents the amount of lag, with S_m being the correlation coefficient at lag m. At last A_m, the normalized coefficient can be obtained.

The author starts by applying a linear interpolation to the y-axis accelerometer raw signal. By assuming that the smartphone is an fixed position, the autocorrelation of the interpolated y-axis accelerometer is computed. Then, the peaks of the signal are found in order to identify the local dominant maxima, retrieving information about the samples in which each of this peaks occurs. Cadence was then extracted directly from the moment of the A1 by computing $Cadence = \frac{60}{t_{A_{d1}}}$. This is depicted on Figure 2.3.2.3.

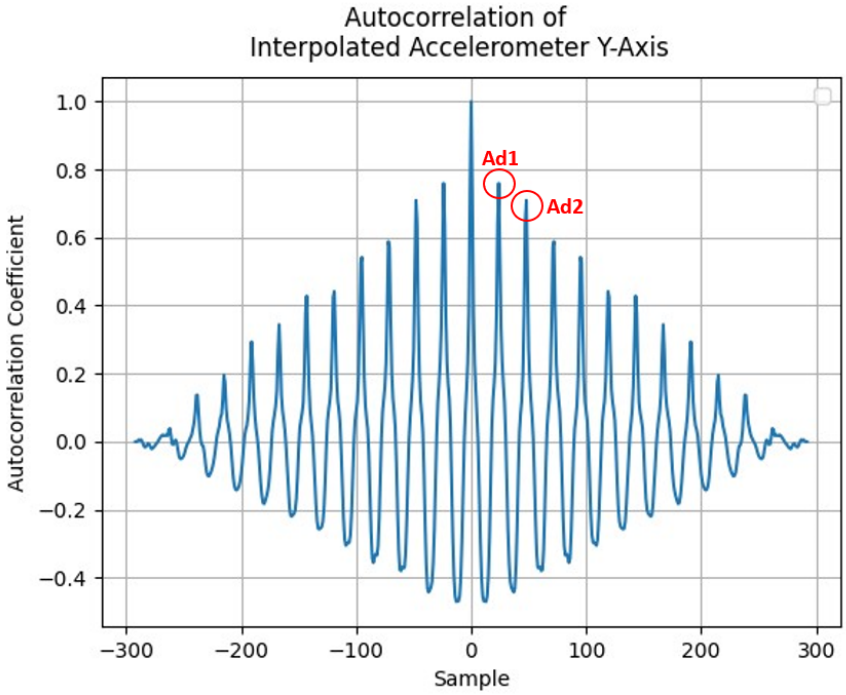

Figure 2.9: Identification of A_d1 and A_d2 in the autocorrelation function of the y-axis accelerometer signal.

According to the author, walking speed can also be calculated using the extracted cadence, following the Equation 2.15:

$$WalkingSpeed = \frac{cadence * stridelength}{120} \tag{2.15}$$

In the work of Yang et al. [51] we can find some common values for cadence which is around 105 steps/min. According to the author, the regular walking cadence is somewhere between 75 to 135 steps/min. By analysing the data from the developed experiment, PD patients exhibited more fluctuations and less regularity in their autocorrelation charts when in contrast to young healthy group.

2.3.2.4 Dynamic Time Warping

The **DTW** is a very useful algorithm used to compare and align two time series. It is useful to find a non-linear alignment between two numerical sequences, finding the similarity between two signals. By doing this, we can find patterns on events measured with different rhythms. The fact that the signals involved don't need to have the identical lengths, as opposed to the previously seen technique, is of great value for practical approaches. The similarity between signals is computed using the distance measure as a basis. Different distance functions can be used, even though the most widely used is the Euclidean distance [15]. The distance represents how similar these signals are, by computing the minimum warping path, meaning the lower the values are, the closer to similar the signals are.

To implement this technique, in a classical manner, a nxm distance matrix must be constructed [15]. As notation, the respective lengths of the signals S and T, can be defined as $S = s1, s2, ..., s_n$ and $T = t1, t2, ..., t_m$. To compute the (i,j)-th element of the distance matrix, the Euclidean distance is used, as follows in Equation 2.16:

$$d(s_i, t_j) = (s_i - t_j)^2 \tag{2.16}$$

To define the warping path, we can denote it as $W = \omega_1, \omega_2, ..., \omega_K$, where the k-th element is represented by $W_k = (i, j)_k$. K must be inserted between the sum of the lengths of both signals minus 1 and the maximum length between both signals, as follows in Equation 2.17:

$$max(n, m) \leq K < m + n - 1 \tag{2.17}$$

The warping path has certain conditions for which it has to comply with to be able to obtain a warping path closer to the matrix's diagonal. First, as a boundary condition, $\omega_1 = (1, 1)$ and $\omega_K = (n, m)$ must be respected. Then, for continuity, given $\omega_k = (i, j)$ and $\omega_{k+1} = (i', j')$, we must ensure that $i' - i \leq 1$ and that $j' - j \leq 1$. Finally, for the monotonicity, given $\omega_k = (i, j)$ and $\omega_{k+1} = (i', j')$, it must be ensured that, $i' - i \geq 0$ and $j' - j \geq 0$, resulting in $(i' - i) + (j' - j) > 0$. After assuring the complying of these conditions, multiple warping paths will still be viable. Given that we are only focused on the warping path with the minimum distance, an efficient manner to calculate the distance is necessary [15]. To compute the distance, dynamic programming with the usage of the following recurrent function can be used, Equation 2.18:

$$\rho(i, j) = d(s_i, t_j) + min[\rho(i - 1, j - 1), \rho(i - 1, j), \rho(i, j - 1)] \tag{2.18}$$

The expression for the DTW distance for S and T is, as follows in Equation 2.19:

$$DTW(S, T) = \sqrt{\rho(n, m)} \tag{2.19}$$

Derawi et al. [29] proposed a system using low grade smartphones placed at the subject's hip to extract accelerometer data during walking. The system detected and

segmented gait cycles, computing the average cycle for each user and using DTW to propose an authentication system based on gait analysis. To compute the average cycle, the author put to use the DTW method to compute the distances between all cycles. After having the distance values, the cycles with higher values for the distance, or in other terms, the cycles with lesser similarity are eliminated. Then, from the remaining cycles, the cycle with the least DTW distance to the others is picked as the average cycle. After detecting a cycle starting point, the system adds the average cycle length to this point, estimating where the ending point of the cycle would land.

Perez and Labrador [15] based a part of their work in the research of Derawi et al. [29]. The investigation started by segmenting the steps using the Listing in 2.7. This author used the DTW algorithm to compare different steps within a trial for one user. After segmenting each step, the aforementioned DTW algorithm was used to generate a comparison matrix representing the distance between each step with all the other detected steps.

To do this, for each step pair, the DTW was calculated to verify the alignment between the pair and the resulting value is placed in the comparison matrix. After that, all the obtained values are averaged out to obtain an average distance between all steps. Within the author's study context, the DTW average was also computed for all the differentiated left steps and right steps. This allowed the author to compare the resulting averages from each comparison matrix and perceive which of the feet/legs might have irregularities.

2.3.2.5 Gait Feature Selection and Extraction

Feature selection can be very important but also very dependent of what exactly we are trying to achieve and what data we have available depending on the approach. Some of the typical features extracted from motion signals are presented below in table 2.1. Each column represents a different domain of a certain extracted feature.

Spatial	Temporal	Frequency	Statistical
Step length	Double support time	Spectral power	Correlation
Stride length	Stance time	Peak frequency	Mean
Step width	Swing time	Maximum spectral amplitude	Standard deviation
RMS acceleration	Step time		Covariance
Walking Speed	Stride time		Energy
SMV	Cadence		Kurtosis

Table 2.1: Typical features extracted from motion signals. Adapted from [52]

In the PD field, selecting which spatio-temporal features should be approached is a difficult task as there is a lack of consensus of which gait spatio-temporal parameters are clinically relevant [38][37]. That said, the study still concluded about the most frequent present features in the reviewed research, which were the gait velocity, stride length, step length and cadence. Other studies reported similar conclusions about parameter frequency [37].

Bouça-Machado et al. [38] also concluded about the main differences between the Healthy Control (HC) and PD test groups. PD patients presented decreased values on velocity, stride length, step length and swing time, in comparison to the HC group. On the other hand, this group presented increased values on stride time, stride time variability and dual support time. This value discrepancy follow the clinical expectancy for a patient with motor impairments such as being slower, short-stepped and feet shuffling, as well as postural instability [25, 37, 38].

According to Arora et al. [53], it might not be possible to extract every primary gait and postural sway metric from the smartphone accelerometer. In this study, it was not possible to compute stride length, trunk flexion and minimum foot clearance.

Proessl et al. [43] used two smart devices for it's approach. After manually entering the time step value, stride duration was computed by the difference of one gait cycle by it's preceding. Step duration was computed by halving the stride duration. To obtain cadence, in steps per minute, each stride duration was divided by 60 seconds and averaged them. The gait speed was obtained based on the distance covered, divided by 360 seconds. To derive mean stride length, gait speed and mean stride duration were multiplied.

Silsupadol, Teja, and Lugade [45] used a smartphone with a custom-build Android application, to collect tri-axial accelerometer data and compute spatio-temporal gait parameters. Computed parameters included step time, step length and gait velocity. Step time (ST) is the time difference between heel strikes. To compute step length (SL), the following formulae in Equation 2.20 was used:

$$SL = 2 * \sqrt{2 * h * l - h^2} \tag{2.20}$$

Where l is the leg length and h is the change in vertical position, or variation in height of the vertical position across each step cycle. h was computed by double integrating the vertical data from the accelerometer. Step velocity (SV) was also computed following Equation 2.21 :

$$SV = \frac{SL}{ST} \tag{2.21}$$

Gait velocity was computed by averaging SV across all steps. Cadence was defined as in Equation 2.22:

$$Cadence(steps/min) = \frac{NS}{D} \tag{2.22}$$

Where NS is given by the amount of steps and D is the duration of the trial.

The process of turning unprocessed raw data into numerical features that may be processed while keeping the original data set's information intact is referred to as **feature extraction**. This can be useful not only to simplify the problem but also to reduce data dimensionality. According to [46], this is an optional step in the gait analysis process. When it does exist, for each fixed-length segment or gait cycle, a feature vector is extracted. To reduce the dimensionality, different metrics are used such as: the mean, the

minimum/maximum, the standard deviation, the difference between the minimum and the maximum and the Root Mean Square (RMS).

2.3.3 Machine Learning

Artificial Intelligence (AI) is a branch of computer science focused on building smart algorithms capable of performing tasks that would otherwise need human intelligence. It's principle lies in the fact that we can define human intelligence in such way that a machine can mimic [54]. One of it's main characteristics is the ability to rationalize and act, in the best probabilistic manner, in the interest of achieving a specific goal.

ML is a sub-field of AI that focuses on using algorithms to extrapolate patterns in data. ML models are taught how to make informed decisions by being trained therefore adapting to new conditions. This can be done by retrieving a considerable data set and using algorithms to build a statistical model that uses it as it's basis. Different variations of ML algorithms can be defined and usually fall into three main categories: reinforcement learning, unsupervised learning and supervised learning [18].

In **reinforcement learning** algorithms, the trainer continuously provides input and the system constantly improves based on these, a lot like a child's mind [54]. These inputs are observations obtained from the perception of the target environment with the purpose of maximizing the average expected reward and minimizing the risk. It continuously learns in an iterative way. Main applications for this type of ML are robotics and resource management [18].

Unsupervised Learning consists of providing unlabeled data as input and expecting the system to discover patterns. This is useful for cases where we aren't aware of what to isolate in the data. This is mainly applicable for pattern detection and descriptive modeling typically for clustering problems [18].

At last, **supervised learning** consists of having a trainer who provides rules that make connections between inputs and outputs. This type of algorithm models relationships between a target output and respective input, learning from previous data how to define the output for a certain unlearned input.

2.3.3.1 Supervised Learning - Model Selection

Supervised Learning algorithms data sets are usually a set of labeled examples, each belonging to a specific class [18, 54]. This type of model is used for prediction, based on labeled data and it's main target problems are regression and classification. A real application for this kind of algorithm is medical imaging and classification.

Most gait classification systems stand on one of both: **template-based classification** and **stochastic classification** [46]. Template-Based Classifications approach include the Euclidean distance, DTW, Rotation Dynamic Time Warping (RDTW) and Cyclic Rotation Metric (CRM). The stochastic classification approach is the conventional machine learning

approach and includes algorithms such as Hidden Markov Model (HMM) and Support Vector Machine (SVM).

Gait data analysis using pattern recognition approaches has been investigated in the past with varied degrees of success. Among the methods most utilized are support vector machines and neural networks. The majority of these works focus on a binary choice, mainly if a disease is present or not [55].

According to Rovini et al. [7], sensors in combination with novel ML algorithms represent a useful way to aid clinicians in PD diagnosis, since the disease starts. It also refers that even though several classifiers were used in the reviewed studies, no consensus exists on the most suitable approach for classifying a PD assessment. From the three reviewed ML algorithm types, the one that seems more suitable for our study-case is the supervised learning for it's classification ability of labeled data and past medical applications. For better understanding, further research will be presented on this type of ML. Some of the common supervised learning algorithms include: SVM, Naive Bayes (NB), Neural Network (NN), Decision Tree (DTREE), Linear Discriminant Analysis (LDA) and Random Forest (RF).

With the objective of understand how each of these algorithms work, a description for each technique is presented. SVM is used as a classification method, plotting each data item as a mark in the N-dimensional space with certain values for each of the features, representing a particular coordinate. N is defined as the number of features of the classifier. Lines will be responsible for splitting the data between differently classified groups of data. Depending on where the new data lands on this N-dimensional space in relation to the decision boundary, a final classification is decided [18].

NB is a classification technique that uses an approach based on assuming independence between predictors, being the reason why it is called "naive". It is based on Bayes' theorem, assuming that a certain feature in a class is not related to any of the other features. It is a fairly simple model to construct and is a good tool for very large data sets. Even though it is a simple algorithm, this technique outperforms several more sophisticated classification methods [54].

DTREE is most commonly used for classification problems working for both categorical and continuous variables. The population is splat into homogeneous sets based on the most significant or independent variables, ensuring groups as distinct as possible using techniques like Gini, chi-square, entropy or information Gain. Each new object is classified based on it's attributes by the tree [18]. RF is fundamentally an ensemble of decision trees. The "Forest" represents a collection of decision trees using a "bagging" paradigm, creating slightly different copies of the training data, and the classification is done by majority of votes of the trees. Each tree classifies the new object and the final classification is given by the one with most votes among all trees in the forest, reducing the variance and the over fitting of the final model as well as usually increasing it's accuracy in contrast to the simple multiple DTREE[18].

LDA is a technique for dimensionality reduction used as a pre-processing step in

ML and used for pattern classification. The algorithm allows avoiding the curse of dimensionality by projecting features onto a lower-dimensional space while also reducing resources consumed. This is done by removing the redundant features present in the data. It can be used as a supervised classification tool and is used mainly for image recognition and predictive analysis [56].

At last, NN uses basic elements, known as neurons, that receive a value, multiply it by a certain weight and pass it on an activation function, usually non-linear. By using various layers of neurons, the neural network can process and learn increasing complex functions [54]. Even though it is very effective dealing with high dimensionality problems and complex relations, it is difficult to implement and, in most cases, requires a large training set to be effective.

From the reviewed state of art, some implemented classifiers by researchers were selected to understand which ones could be more adequate for our problem. For example, in a study [7], two supervised learning classifiers were implemented to classify different groups of people. By combining motion analysis with an olfactory screening test, SVM and RF classifiers were implemented and compared for two-group (HC and PD) and three-group classifications. Very good results were obtained for healthy vs. patients classification, obtaining classifiers with accuracy of 1.00 and 0.97 for RF and SVM, respectively.

Arora et al. [53] implemented a RF classifier with 98.5% average sensitivity and 97.5% average specificity in discriminating PD and HC groups. The data used was recorded by the build-in tri-axial accelerometer embedded in a consumer-grade smartphone and processed to extract features in the gait. Even though the approach was very successful, the authors noted the worth of researching further the incorporation of other study metrics, such as how long a patient takes to react, finger tapping and speech.

Quorum Voting Scheme: In Figure 2.10, it is independent of the features extracted and classification approaches, data can be further processed. Each single classifier votes for genuine or impostor. According to Nickel [46], this can be an advantage to the system due to short disturbances ended up being compensated in the obtained gait. Each of voters can also have more or less importance by defining importance weights for each classifier vote.

A study by Caramia et al. [10] used a IMU-based gait analysis system and evaluated different ML techniques to identify gait patterns of PD patients, discriminating them from the HC group. The different implemented techniques, and respective accuracy, were: SVM rbf (non-linear kernel) - 75.6%, k-Nearest Neighbour (k-NN) - 73.0%, NB - 72.7%, LDA - 72.5%, SVM linear - 72.0% and finally, DT - 68.9%. The study also implemented a seventh classifier, described as Majority of Votes (MV), using a weighted combination for the previous classifiers. The final classification accuracy for MV was always higher (in the range of 5% to 20%, on average) than the ones obtained by the basic classifiers, proving that by adding classification results from different ML techniques resolve into a relevant accuracy increase. It used a 5-fold cross validation procedure with 100 repetitions.

Given that the selection of an appropriate classification model is key, evaluation of the

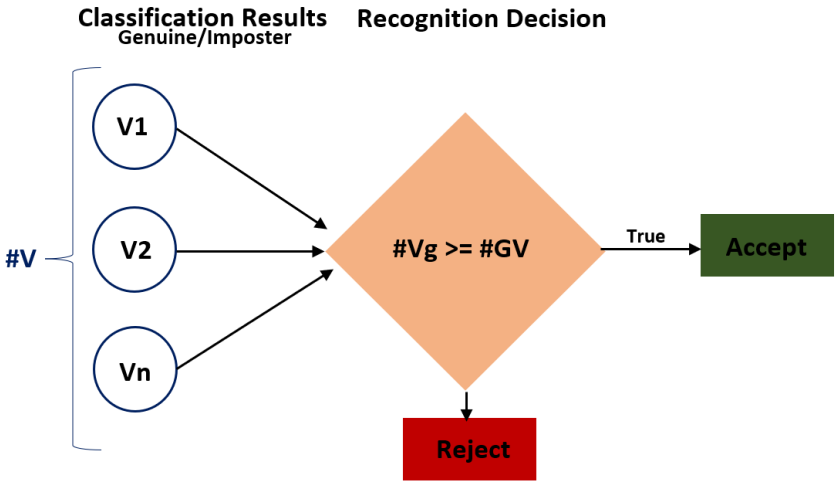

Figure 2.10: Quorum voting scheme. Based on [46]

performances of the classifiers is essential. With the objective to quantify the classifier's accuracy, some performance measures are defined and used by different studies such as the sensitivity, the proportion of correctly identified PD examples, the specificity, the proportion of controls identified correctly, and finally, the balance accuracy which represents the average between these last two measures [53][7].

2.3.3.2 Parameter Tuning

Even though a learning algorithm produces its results through the optimization of a set of training criterion to a certain set of ***parameters*** θ, the algorithm itself has its own set of control variables often denoted as ***hyperparameters*** λ.

For any given ML model, we can divide its configuration values into two classes: (regular) parameters and hyperparameters. Parameters are comprised of all the configuration values which an ML model will internally estimate from its training data, while hyperparameters are comprised of all the remaining configuration values which are decided by things external to the model.

The canonical example of the difference between parameters and hyperparameters is the neural network example: a properly trained neural network will learn the weights of the connections between its nodes, but the topology of those nodes (such as the number of nodes per layer or the number of layers) is decided by the designer of the network. Therefore, the weights are parameters of the network, while its topology is a hyperparameter.

It must be noted that the distinction between parameters and hyperparameters is not a hard boundary, but rather a qualitative *'I know it when I see it'* distinction. While the network topology is a hyperparameter in the neural network example above, in a hypothetical 'neural-network-like' algorithm which adjusted the topology of the network during training, for example, by automatically dropping nodes whose weights are so low

that they don't meaningfully affect the network's classification, the topology would be a parameter.

In developing a ML model, evaluating the results of its training will usually amount to running it on a data set kept separate from its training data set, computing some metrics for how well it's performing – the model's accuracy, precision, recall, and other assorted criteria, and qualitatively evaluating those metrics. As choosing the model's hyperparameters happens before the training takes place, this evaluation of the model's performance also serves to evaluate their quality and guide the model developer in their search for better hyperparameters.

2.3.3.3 Performance Measuring

For measuring the performance of the a certain ML, some main concepts have to be reviewed. The **True Positive (TP)** is the hit. On the other hand, the **True Negative (TN)** is the correct rejection. Following the same logic, **False Positive (FP)** is a type I error, meaning it is a false alarm for overestimation reasons. **False Negative (FN)** is a type II error, which associates with a miss or an underestimation.

The **Confusion Matrix** is a visual representation of the error in a supervised learning algorithm. It's typical representation can be seen in Figure 2.2.

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

Table 2.2: Example confusion matrix

These metrics serve for direct comparison of the relation between the actual condition *Vs* the predicted condition. Some of these are reviewed alongside with their formulas.

Precision is how accurate the model is. This means, how many of the predicted positives are actual real positives, as seen in Equation 2.23. It represents how effective the model is predicting specific categories.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{TruePositive}{TotalPredictedPositive} \tag{2.23}$$

Recall, also known as sensitivity, computes how many of the positives are true positives, as presented below in Equation 2.24.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.24}$$

The **F1 Score** captures the balance between precision and recall. If either the precision or the recall are low then the value of F1 score will be low, independently of if the other is high. It is defined by the harmonic mean of precision and recall, in accordance with the Equation 2.25:

$$F_1score = \frac{2 * TruePositive}{2 * TruePositive + FP + FN} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.25}$$

Finally, **accuracy** is how many times the system was correct in a general way. The computation is based on how many labels were correctly predicted.

3

System Design & Architecture

Recalling that we set out to find a lower cost CDSS tool to aid with gait analysis, monitoring and early gait alterations detection, in this Chapter we first justify the high-level design choices that were made in Section 3.1 and in Section 3.2 we present an overview of the resulting system architecture.

3.1 Design Choices

3.1.1 Clinical Gait Assessment Strategy

Our first decision was on what gait assessment approach to use. With the insights gained from our review of the literature in Section 2.2.3, we concluded we would like to avoid MV or ES based methods due to their higher equipment cost (treadmills, multiple cameras and depth sensors), leaving us with WS methods. Within the realm of WS, we knew we would like to avoid the need for multiple IMUs, for both the cost and inconvenience of set-up. Indeed, we quickly concluded that we needed to rely on the single most common measurement device that everyone has, a smartphone.

With this decided, we also understood the importance of offering support for both single and dual task trials, in order to measure the differential impact of cognitive interference on the patient's gait and a decision was made to offer support for both types of trials. In order to facilitate development of the data collection module, we fix a few parameters of the system, namely the trial length is fixed to 10 meters over a flat surface and the end of a trial is marked by the user turning 180 degrees anti-clockwise, allowing for automated termination of data collection.

3.1.2 Decision Support

For increased usability and decision support we chose to present all information on a single page dashboard. This dashboard ought to contain visualizations of the gait features, so that they may be analysed by a clinician. Buttons should be present to perform several desirable sub-tasks. The final layout of this dashboard is presented in Section 4.3.3.

We decided early on that this dashboard should also contain a suggestion of diagnosis. To achieve this, we chose to classify users into distinct categories based on their gait data For our purposes, supervised learning (Section 2.3.3) was the most appropriate choice of machine learning algorithm. For our purposes, binary classification is sufficient. Since different medical professionals may wish to classify users into categories based on different traits (e.g. healthy vs. arthritis or normal gait vs. impaired gait), we now begin to use the more general terminology "Control" vs. "Target" for the two classes.

3.1.3 Machine Learning Classifiers

We explicitly favoured simpler models over the extremely popular and powerful deep neural networks for three main reasons. First, *explainability* is very important in the medical field. Using AI models that are easy to understand and explain is very desirable, as it helps in communicating the reasoning behind the results back to both the patient and the medical professional. For this reason, our classifiers are not trained on the raw sensor signals, as they are not deep enough to meaningfully learn from them. Instead, they are trained on features extracted by the signal processing algorithms we discussed in Section 2.3.2.5. This feeds into to our second reason, which is cost. Deep neural networks are expensive to train, which would increase the cost of ownership of the system. Finally, we also needed to constrain the scope of the thesis where possible. Deep neural networks are not only more difficult to train correctly due to the fact that they learn from raw data end-to-end, but can also overfit more easily in the absence of big data. Since we do not expect to have big data, using deep neural networks could lead to overfitting.

In the end, we chose to explore three kinds of models: decision trees, decision forests and support vector machines. The first two are extremely explainable, as one can literally inspect the decisions taken at each level of the tree. The last one offers larger learning capacity, but comes at the cost of some explainability. All three are cheap to train and predict with, while requiring little memory, making them a good fit into our system. We additionally explore a quorum voting scheme between these models.

3.2 Overview & High-Level Architecture

We present an overview of the architecture of *dualgAIt* in Figure 3.1. Following software engineering conventions, the system is composed of three main Modules: Database, Processing and Presentation.

Trial data is collected using an Android application, *dualgAIt: Android*. This data is pushed to a database component, until a medical professional requests its processing. Through it's Graphical User Interface (GUI), a medical professional may request the processing of the data of the user in *dualgAIt: Desktop*. Processing the data involves pulling it from the database, applying different signal processing algorithms to clean and

segment it, generating all the plots for visualization and (possibly) invoking the classifiers for prediction of the user's class.

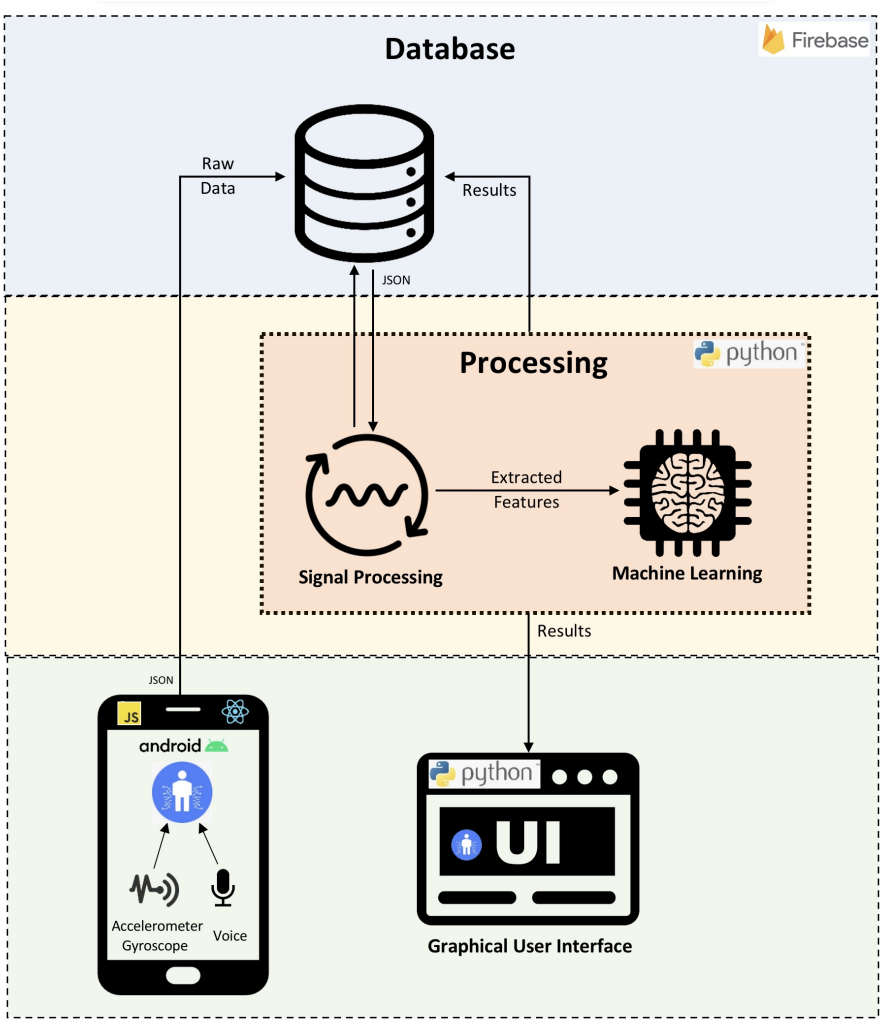

Figure 3.1: Implemented system architecture

The system has three main functionalities, which correspond to three main user workflows: **Training**, **Classifying** and **Evaluating Progress**.

Training A trial is conducted by the user and medical professional by following a script. If it is the first time a user enrolls, the survey workflow, which requests basic details of the user, also occurs. Collected trial data is manually tagged with either the "control" or "target" label by the medical professional.

Classifying When classifying, no class is attributed to the trial data, but instead the classifiers are invoked to predict the users class. This involves invokeing each indidual classifier and then counting the votes of each. Note that, to prevent wasteful retraining of the classifiers, training is done *lazily*, being triggered only when classification is requested

and there is new trial training data. This flow is illustrated in Figure 3.2. As shown, it is the GUI that controls whether the main processing loop executes.

Evaluating Progress of a users progress can be requested from the GUI, which pulls all of the users previous trials from the database and produces a pdf report compiling all the important metrics.

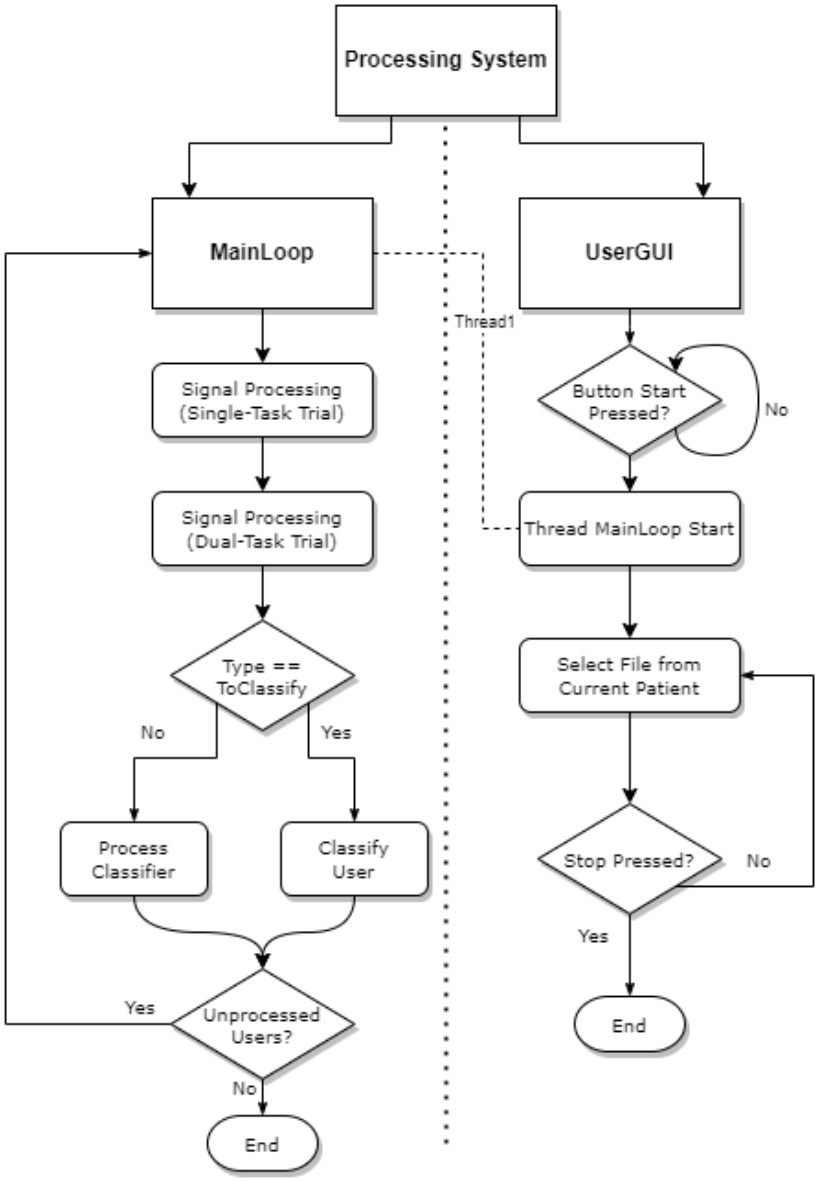

Figure 3.2: Flowchart of the Processing Module main loop and interaction with GUI

3.2.1 Data Model

Figure 3.3 presents the data model used to communicate between all the components. Each patient is represented by a unique ID that then links it to their survey data, cognitive test results, classification results and trial data as child references. Trial data itself is composed of accelerometer, gyroscope and audio readings, which are signals treated as float arrays. This data model is flexible enough for both the smartphone application to submit data and for the processing module to pull it. Signals are stored in raw format, meaning that if errors are later in the processing code or new algorithms are added, the original data can be reprocessed.

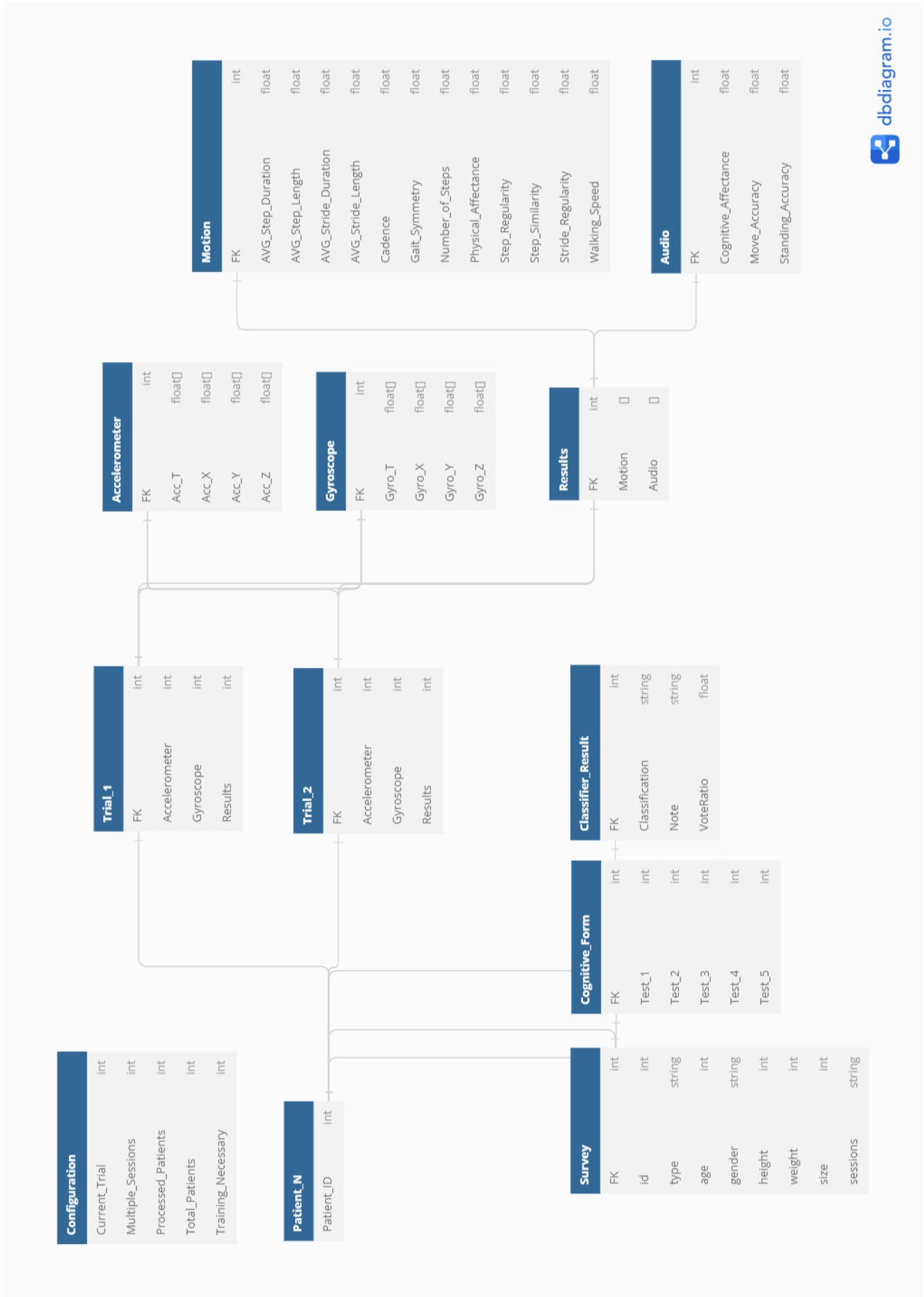

Figure 3.3: Project data model

4

Software Implementation

This chapter focuses on providing further detail on how the designed architecture was implemented. It includes, technologies used, algorithms and overall detailed information about system design, implementation, integration and how the system was dimensioned. To address the research problem that was proposed on Chapter 3, a prototype was developed and tested.

4.1 Database

The database was implemented using ***Firebase***, a No-SQL, non-relational database. Firebase is a Google's mobile app development platform with features for building and improving an application. Two factions of the Firebase technology are used, as can be seen in 4.1, *Firebase Real time Database* and*Firebase Cloud Storage*.

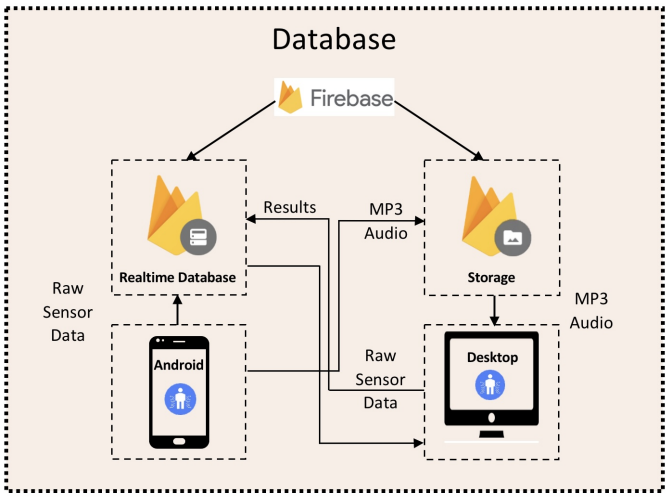

Figure 4.1: Database module

Data is encapsulated as JSON and is shared across all the existent clients in a real-time fashion, having the advantage of remaining available when the application is offline. This

implementation allows for real-time access which is stored in cloud, allowing for multiple clinicians to access a much more ample amount of data.

The ***Firebase Real-time Database***[1]. will be used to store data from the survey, the cognitive test and the sensors. Later on, it will be also used to store the results from both the feature extraction and the classification result, when applicable.

The ***Firebase Cloud Storage***[2]. will be used to store the audio recorded during the dual-task trial. When the application records the audio, this audio is sent to the cloud storage to the 'audios' folder. Later on, the system will download the audio corresponding to the user being processed by downloading the audio present in this user's storage path: '/audios/voice_signal_{userID}.mp3'.

4.1.1 Database Structure

To set the system up for later integration, the database compels a set of configuration variables. We will start by describing the function of each of these and their interactions with the system will be referred as necessary in the following implementation sections.

- **Total_Patients:** Is used as a counter to the total indexed users in the database. It's initial configuration is set to 0 and is incremented to 1, the moment the first user submits its survey.

- **Processed_Patients:** Is used as a counter to the total processed users in the database, which technically means the amount of users that have their results posted to the database already. It's initial configuration is set to 0 and it is incremented every time a full trial is complete.

- **Current_Trial:** It's value is either 1 or 2, representing the single-task trial and dual-task trial, respectively.

- **Multiple_Sessions:** It's a binary flag, which sets the system up for comparing multiple sessions from a single user. It is triggered when the user chooses the second functionality to evaluate progression, in the android application.

- **Training_Necessary:** It's a binary flag, which prepares the system for need to train the classification models. It triggers when there is a new user input, for the first functionality, training. Since there was a recent training, new data can be added to the original data set that is used to train the models. This flag is used to make sure the models aren't unnecessarily trained again if there isn't any new data since last training, for resource efficiency.

[1] https://firebase.google.com/docs/database
[2] https://firebase.google.com/docs/storage

4.2 Android Application

The *Android* application, or *dualgAIt: Android* was developed using *React-Native* in order to carry out the necessary data acquisition. React-Native has the added advantage of being portable across platforms, meaning that our application could also be easily ported to IOS. The Android application is divided into four main screens: the home screen, the survey screen, the cognitive test screen and the sensor screen. Each of these screens has a respective function that will be described in more detail.

4.2.1 Home Screen

Home Screen is the screen for welcoming the user and acquiring initial setting up information for the system.

These are the responsibilities of this screen:

- **Welcome user:** display to the user the application's logo and name, warning it will collect bio metrical data as well as voice signals.
- **Profile type selection:** the user is presented with three buttons. It allows to choose between the profile type, which includes, both training options groups (Control and Target) and Classification, which is associated with "To Classify". Choosing one of the training groups will store the processed data to later train its models. On the other option, if the classification mode is chosen, the rest of the system will retrain the models (if there is new data) or use the previously trained models to classify the user's gait.

The rendered screen can be seen in Figure 4.2.

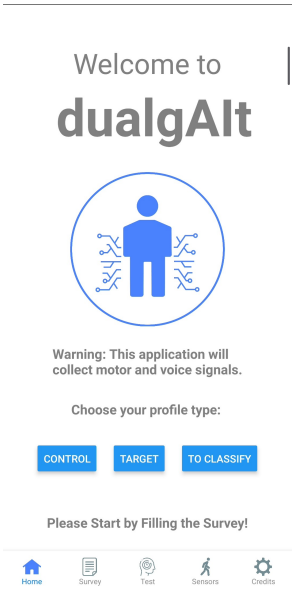

Figure 4.2: Android application opening screen

4.2.2 Survey Screen

Survey Screen is the tab responsible for retrieving initial information about the user. This tab will be split for each of the functionalities: First Session and Progress Evaluation. The system makes sure that the user chooses one functionality or the other. If it's the user's first session of any type, the user should start by filling the first option, **First Session**.

The responsibilities for this screen are:

- **Collect the information:** While displaying the selected profile type at the top of the screen, the user should start by filling the a survey containing basic information such as the age, gender, height, weight and shoe size.
- **Update the database:** when the 'Submit' button is pressed, the reference is pointed to the current user's ID and to it's 'Survey' child. The survey data is sent to the database alongside the user's selected type.

The rendered screens can be seen in Figure 4.3.

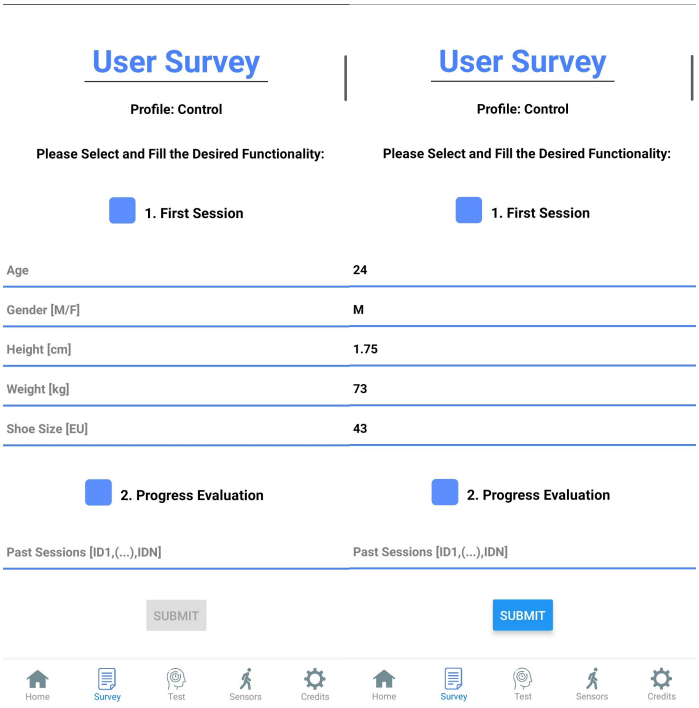

Figure 4.3: Android application survey screen - first session option

The survey information and profile type and successfully being sent to the firebase realtime database as seen in Figure 4.4.

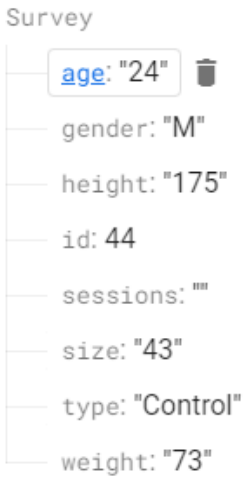

Figure 4.4: Android application posting survey to database

When the user chooses to fill the second option, **Evaluate Progress**, then the application will need a different set of procedures:

- **Collect list of identifiers:** the user is prompted to input the IDs of the last session or last multiple sessions (separated by a comma). This will set up the system for later comparison between sessions, used to assess progress evaluation.
- **Update the database:** when the 'Submit' button is pressed, the reference is pointed to the current user's ID and to it's 'Survey' child. The array of last sessions is set, leaving the rest of the user's survey information blank. This will be dealt by processing side later on where the different user sessions will be associated. The system is prepared to copy the old sessions survey information to the new survey using the given identifiers. The database configuration flag, *Multiple_Sessions*, is set to 1 at this point, preparing the rest of the system for the fact that this user has past sessions that will be necessary, for comparison, alongside with this new session of trials.

The rendered screen, for the second option, can be seen in Figure 4.5.

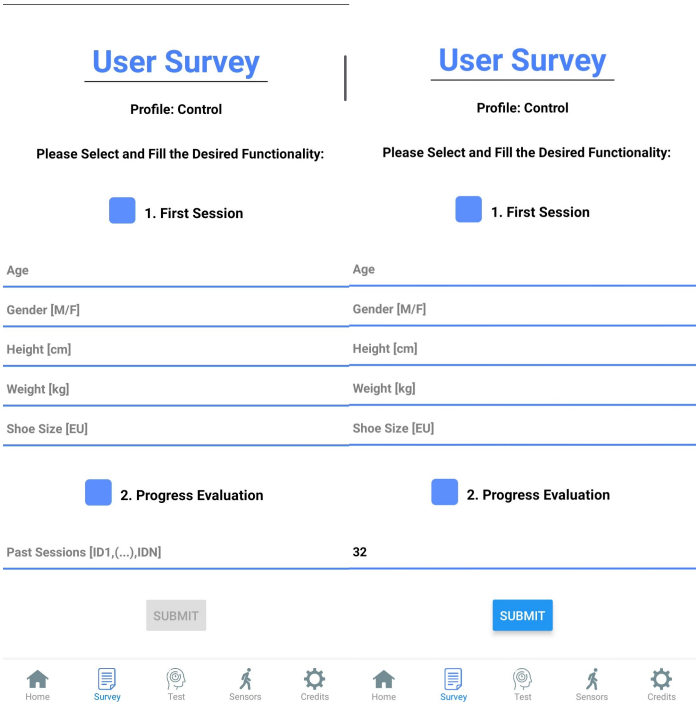

Figure 4.5: Android application survey screen - progress evaluation option

4.2.3 Cognitive Test Screen

Cognitive Test Screen is the responsible for running the standing (single-task) cognitive test. The screen is depicted in Figure 4.6. Below are some of it's functionalities:

- **Present the cognitive problem to solve:** The application explains that a serial subtraction problem will be run. It starts out by giving the user the initial number at the top of screen, from which users shall subtract 7. From that number on, the user should subtract 7 to the number inserted above.
- **Score calculation:** The system is prepared to accept answers in respect to the last answer in order to avoid propagating an input error. This means that if the user wrongly responds to the first number, they will still get a "point" if the next subtraction is correct.
- **End cognitive test:** The system has a timer of 1 minute that will be started when the 'start' button is pressed. The user can press submit when they are done with the test. If they don't, after the timer has reached 0 seconds, the cognitive test will automatically end and the answers will be sent to the database.

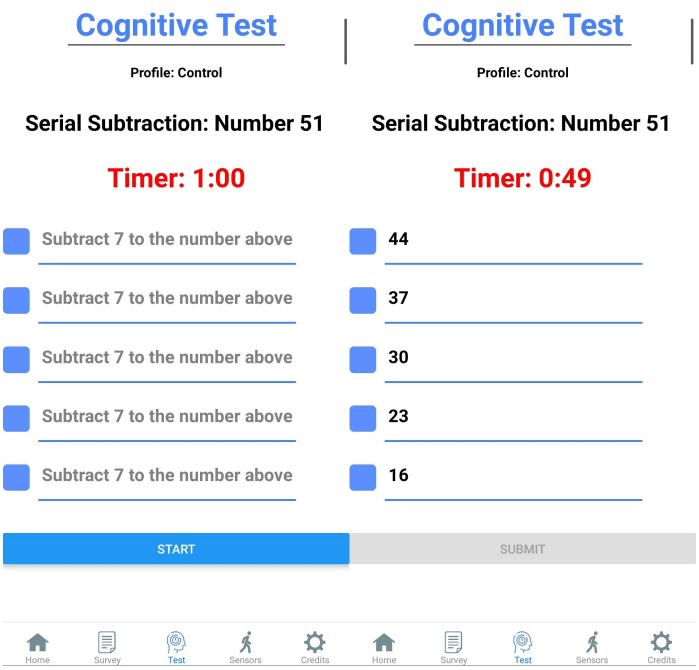

Figure 4.6: Android application cognitive test

When the cognitive test is submitted or the time is up, the application posts the cognitive test data to the database, as illustrated by Figure 4.7.

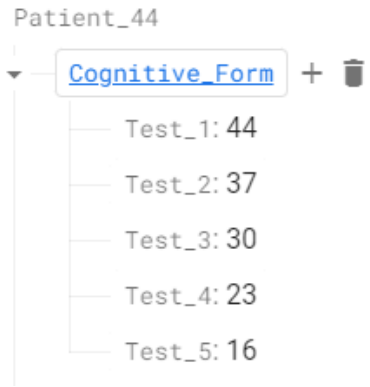

Figure 4.7: Android application posting standing cognitive test to the database

4.2.4 Sensor Screen

Sensor Screen is the tab responsible for starting and stopping both the single-task trial, depicted in Figure 4.8, and the dual-task trial. The way the system deals with the events of *onTrialBegin()* and *onTrialEnd()* change depending on the trial we will be executing.

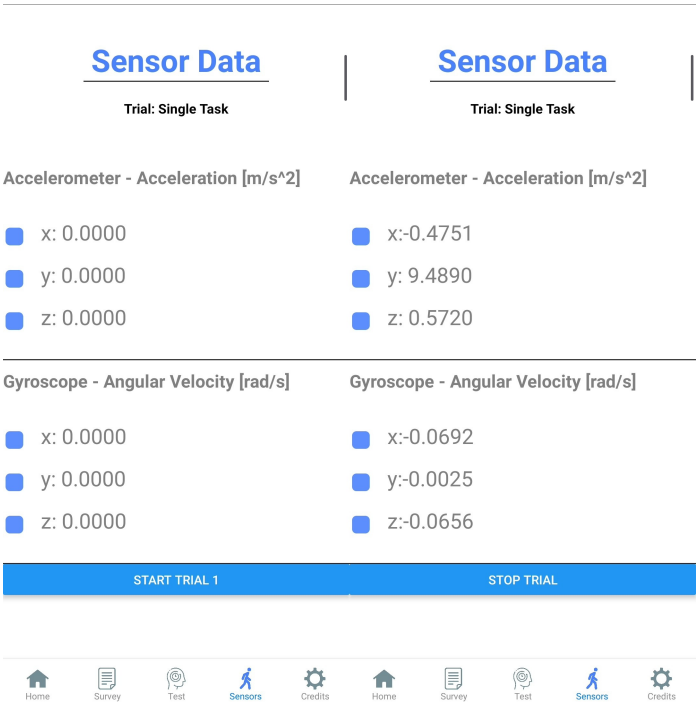

Figure 4.8: Android application sensor screen - Single-task

The script main tasks are as described:

- **Prompt for Android Permissions:** the first time the application is executed after install, the user is prompted to give Android permissions. In order to collect data from the sensors, the microphone, storing the audio in the 'root' and reading from it, some android permissions have to be given to the application. These are permissions to *BODY_SENSORS*, *RECORD_AUDIO*, *WRITE_EXTERNAL_STORAGE* and *READ_EXTERNAL_STORAGE*.

- **Update the trial type:** On trial stop, the system updates the current trial which is represented in the top screen.

- **Control sensor subscription and database update:** On trial begin, we subscribe to the sensors at a 50 Hz sampling rate, storing the obtained values in an array in memory. When stop trial is pressed, the sensors are unsubscribed and the stored arrays are sent to the Firebase database using dataStorage.saveTrial(). For each sensor the reference is pointed to the current user's ID, current trial and respective sensor entry child node. At this point, for each of the sensor axis (x,y,z) array and

the timestamps array are posted to the database, transferring all the trial data to the database. If a trial is restarted, the old trial data is overwritten by the new one.

- **Render sensor values:** This screen also renders the sensor values. The application streams the read values by using requestAnimationFrame() and using setAcceleration() and setAngularVelocity(), while the sensors are subscribed. This was useful for prototyping the application but it's also useful to make sure the that the reading of the sensors are properly working when running the trials.
- **Record audio:** During the second trial, the DT run, audio recording[3] begins when the start trial button is pressed. When the 'Stop Trial' is pressed, the recording stops. At this point, the generated mp3 audio file is located in the smartphone's 'root/downloads' path and is put to the Firebase Cloud Storage, for later retrieval.

The screens for the DT trial are illustrated in Figure 4.9.

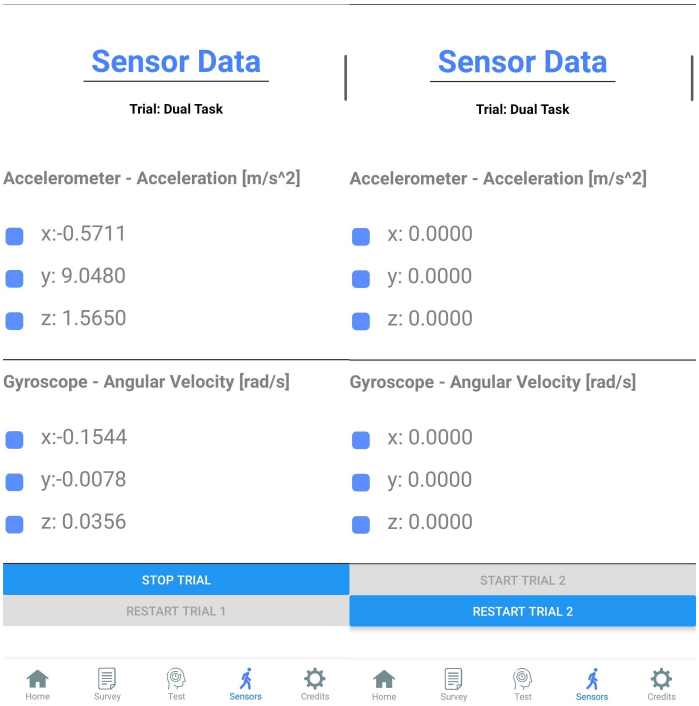

Figure 4.9: Android application sensor screen - Dual-task

When the trial ends, the application sends the sensor data to the database.

[3] Library used for audio recording

4.3 Processing System

The **processing system** or *dualgAIt: Desktop*, was developed using Python. The implemented architecture for the processing system module is presented in Figure 4.10. The implementation can be divided in two main parts, the **motion signal processing** and the **classification**.

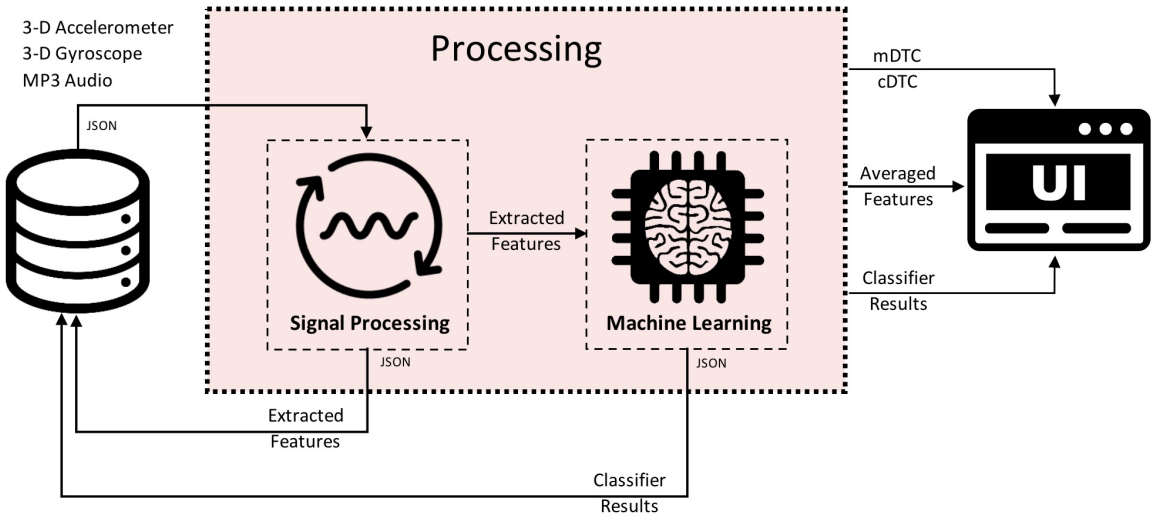

Figure 4.10: Processing module

4.3.1 Signal Processing

Signal Processing is the module responsible for handling all the feature extraction from the collected sensor data. This module's diagram is illustrated in Figure 4.11. The main objective of this module is to extract the final array of extracted features, preparing it for the next step, the classification. In this Subsection, we will review the implementation for each of the applied algorithms and what features were extracted from each of them.

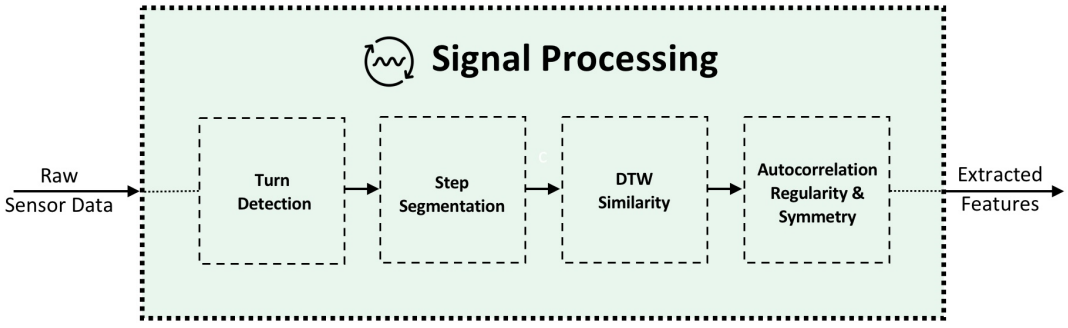

Figure 4.11: Signal processing module

This module is mainly responsible for handling the **motion processing**, which can be divided in four main algorithms: turn detection, step segmentation, DTW analysis and

auto-correlation analysis.

In this module's Section, we will be presenting a full run of processing the data for a single trial of 10 meters. For each of the trials, the flow of the signal processing is repeated, with the exception of a feature (trial distance) that is only computed after terminating both trials. The system will process the dual-task trial when the single-task trial is done being analysed. The systems starts retrieving the signals from the database for the current user. The retrieved signals correspond to the charts seen in Figures 4.12 and 4.13, that correspond to the three-dimensional accelerometer and three-dimensional gyroscope raw readings.

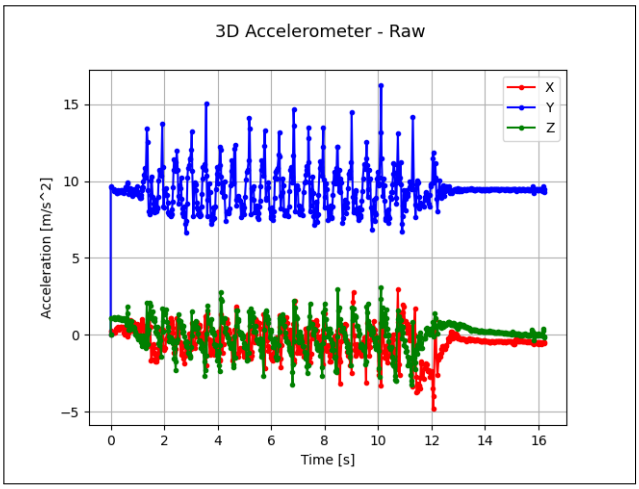

Figure 4.12: Single-task trial raw 3-D accelerometer readings

From Figure 4.13, it is noticeable that there is a spike in the gyroscope Y-axis, at the end of the trial. This is due to the format of the experiences being run that always finish with turning around, facing the starting point, ending the trial. This will be detectable by our implementation.

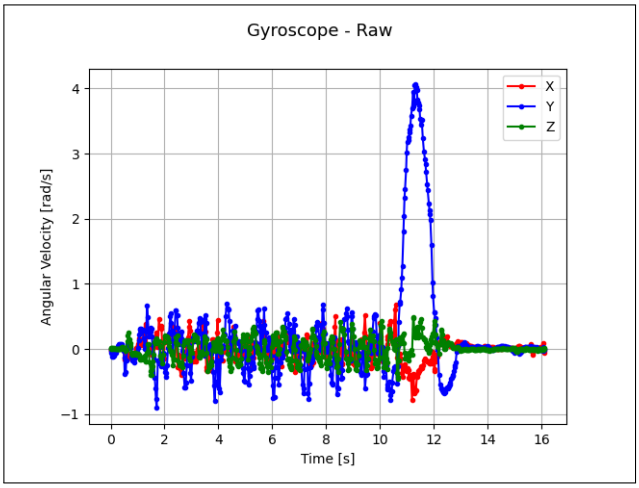

Figure 4.13: Single-task trial raw 3-D gyroscope readings

Turn Detection: The users end the trial facing the opposite direction they have started it by rotating. This is detected by our implemented algorithm that was a reproduction of the algorithm presented by Manor et al. [14]. Alterations to the original work consist of using the y-axis gyroscope instead of the rotated z-axis gyroscope. This is because our smartphone is placed in the lower back and is already vertical to the ground, forming a line between the center of the earth and the device. The original implementation uses the rotated z-axis gyroscope in order to accomplish the same. The idea to use a change of heading as a way of segmenting the signal was also inspired by the work of Gonzalez Rodriguez et al. [41], who used the turns to detect different human activities. The implemented pseudo-code can be seen in Algorithm 1.

Listing 1 Our implementation of Turn Detection by Manor et al. [14]

```
def detect_turn(ang_vel_and_ts):
  #indices where sign changes (i.e. we "cross" the y axis at 0)
  zero_crossing_idxs = []
  last_sample = ang_vel_and_ts[0]
  for sample_idx, sample in enumerate(ang_vel_and_ts[1:]):
      if math.sign(sample[0]) != math.sign(last_sample[0]):
          zero_crossing_idxs.append(sample_idx)
      last_sample = sample

  zero_crossing_times = []
  for zc_idx in zero_crossing_idxs:
    before_cross = ang_vel_and_ts[zc_idx - 1]
    after_cross = ang_vel_and_ts[zc_idx]

    #interpolate to find exact zero cross time
    ts_range = [before_cross[1], after_cross[1]]
    ang_vel_range = [before_cross[0], after_cross[0]]
    f = interpolate.interp1d(ts_range, ang_vel_range)
    halfway = before_cross[1] + (after_cross[1] - before_cross[1])/2
    zero_cross_ts = optimize.root(f, halfway).x[0]
    zero_crossing_times.append(zero_cross_ts)

  all_ts, all_ang_vels = ang_vel_and_ts[:,1], ang_vel_and_ts[:,0]
  f = interpolate.interp1d(all_ts, all_ang_vels)

  start_ts = all_ts[0]
  for end_ts in zero_crossing_times:
      area_under_curve = abs(integrate.quad(f, start_ts, end_ts)[0])
      metric_from_the_paper = area_under_curve * (end_ts - start_ts)
      if metric_from_the_paper > 2.0:
        return (start_ts, end_ts, metric_from_the_paper)
      start_ts = end_ts

  return -1 #No turn detected
```

In that procedure, we start by reaching for the y-axis gyroscope signal. Then we detect when the zero crossings happen in the signal. After obtaining the zero crossings indexes, we compute the AUC using the Equation 2.8. After integrating the angular velocity time series in the interval between the two consecutive zero crossings, we compute the time span between these crossings by applying Equation 2.9.

Finally, we multiply the time span by the AUC to verify if it checks the condition present in 2.10. If the obtained value surpasses the threshold of 2.0 rad/s, as defined by Manor et al. [14], then we have detected a turn. In Figure 4.14, we can see the detected turn by identifying the timestamps for the starting and ending of the rotation in the y-axis gyroscope signal.

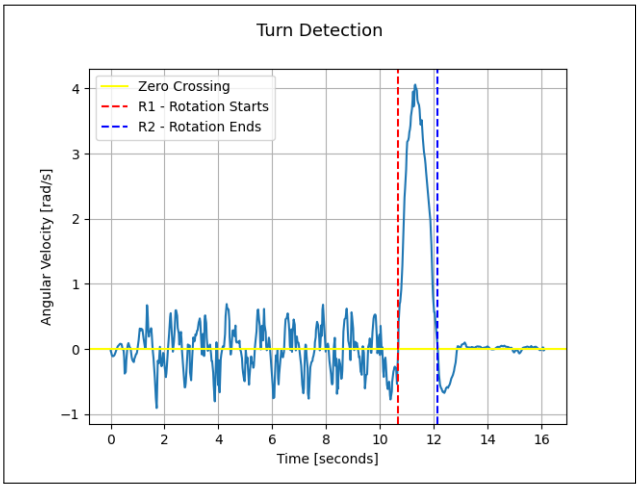

Figure 4.14: Turn detection in the y-axis gyroscope signal

Then, we test if the total angular distance is greater then a certain defined threshold. In our case, reproducing the [14] work, they used 2.0 for the threshold and so did our system. Then, we did some resampling to guarantee that the number of samples between the gyroscope and the accelerometer would be the same. Now, we know when the rotation starts and ends in both sensors sample indexes. Using these indexes, we can find the rotation start timestamp, we take the accelerometer signal and remove the signal after the rotation starts timestamp, as depicted in Figure 4.15. This is very useful to segment all the users signals in the same way, making the trials more even.

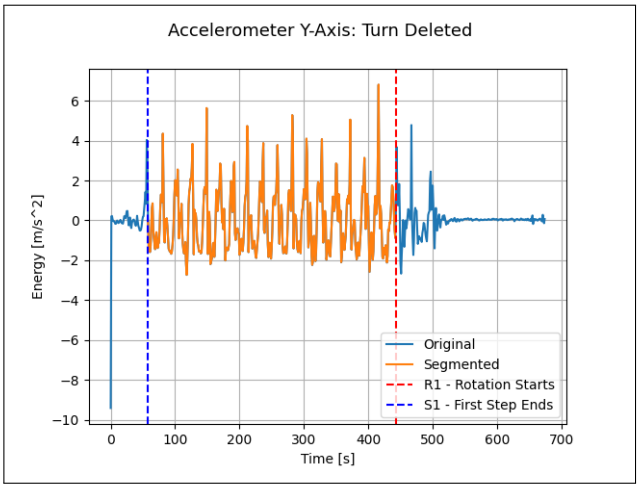

Figure 4.15: Segmented accelerometer signal after turn and first step deletion

Step Segmentation: We start by implementing a way of detecting the steps by segmenting the signal into step cycles. The step segmentation is implemented using a the step detection procedure that is a reproduction of part of the work of Perez and Labrador [15]. On contrary to most reviewed system implementations, this implementation fixes the covered distance instead of the stride length or a limited time for the trial, given that our data acquisition is done in a controlled setting.

Our implementation uses Perez and Labrador [15]'s algorithm, that can be found in Figure 2.7, but instead of calculating the bias for each segment, we calculate the bias for a single segment that was selected to guarantee the presence of movement (because of our trial methodology), without the drift in the beginning of the signal and without the rotation at the end of the signal. This alteration to the work of Perez and Labrador [15] comes with the fact that in our project we also don't need live detection because our data processing is posterior and our trials have a maximum distance of 10 meters. Because of this we have adapted it to our project concerns. After testing, we came up with a general segment for all trials between the samples $N = 30$ and $N =$ rotation starts index. To calibrate the system in order to nullify this bias, we use the equations given by Marron et al. [48].

For each accelerometer axis, we start by estimating the mean of the acceleration in the segment using the Equation 2.3. Then, the bias or the calibration is computed by summing the squares of each accelerometer axis mean accelerations for the selected segment, using the Equation 2.4. The obtained A_{calib} is then sent as an argument to our step detection function.

At this point, we apply the work of Perez and Labrador [15] and calculate the energy of the accelerometer signal, using Equation 2.5, which is obtained by computing the Euclidean norm for each given A_i , where $A_i = a_{i_x}, a_{i_y}, a_{i_z}$. Then, the AMW for each sample is computed, as given by the Equation 2.7, obtaining the energy and filtered energy depicted in Figure 4.16 and Figure 4.17. Then we store the sample indexes for each step

beginning and end. We also store the energy of that step.

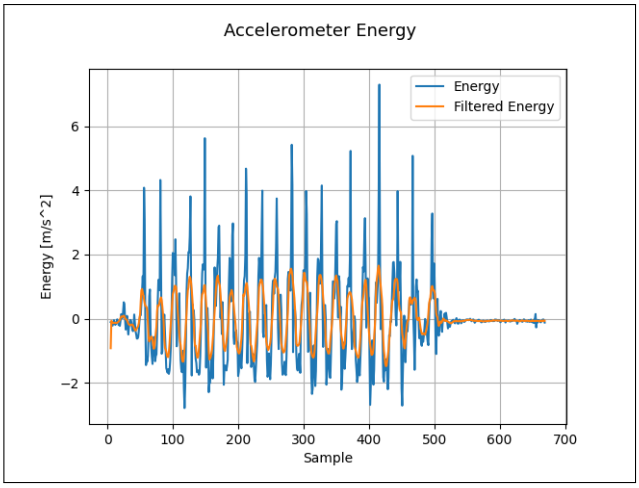

Figure 4.16: Energy and filtered energy

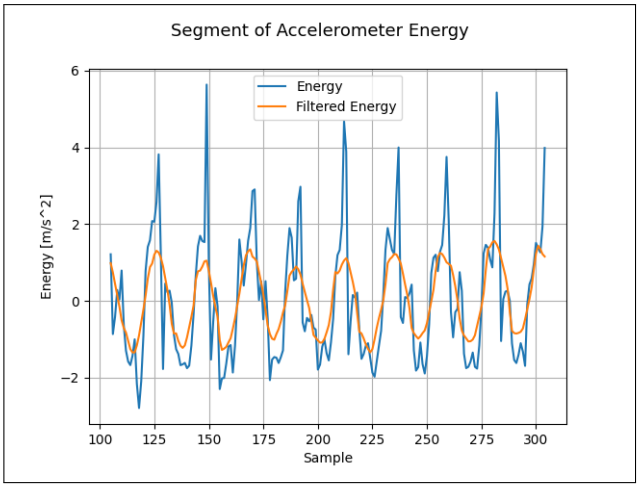

Figure 4.17: Energy and filtered energy plotted from sample 100 to sample 300

A step is detected when a low peak within a certain distance, or window, was preceded by a high peak. We have decided to use the same parameters as the author[15]: the window size is $\omega = 5$ and the threshold of detection is $T = 0.45$. This can be observed in Figure 4.18.

The result of the application of this algorithm can be seen in Figure 4.19, showing when each step ends.

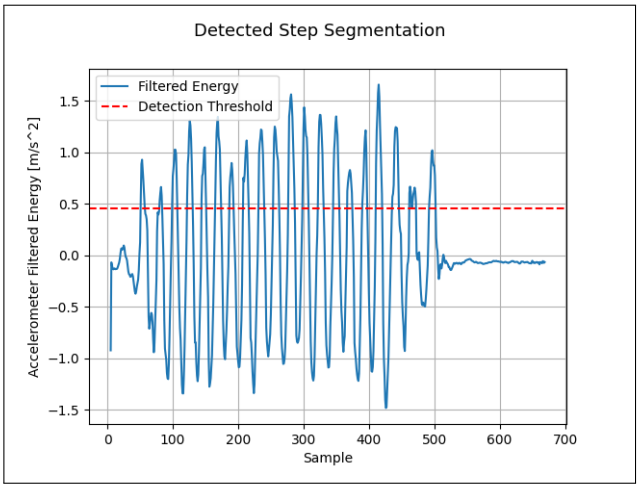

Figure 4.18: Filtered energy and detection threshold T = 0.45

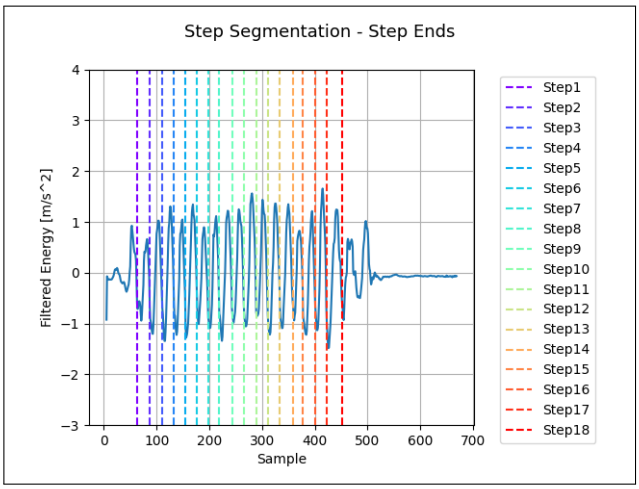

Figure 4.19: Detected sample indexes for each step ending

The steps resulting from this detection can be seen isolated as seen in Figure 4.20.

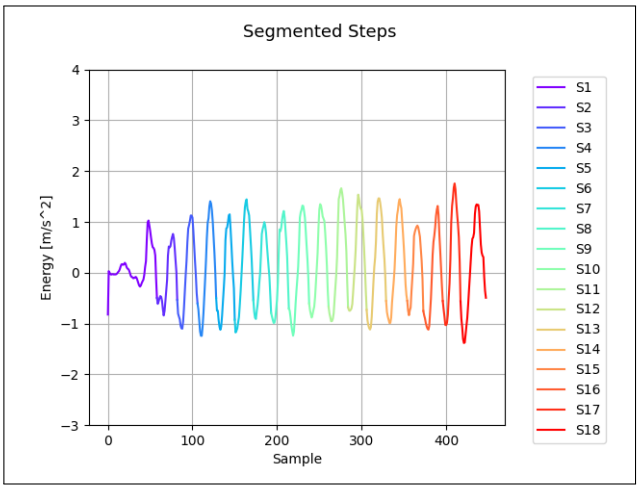

Figure 4.20: Trial detected steps

In order, to compare the signals side by side, the detected steps were equalized and plotted by percentage of their completion[15], as seen in Figure 4.21.

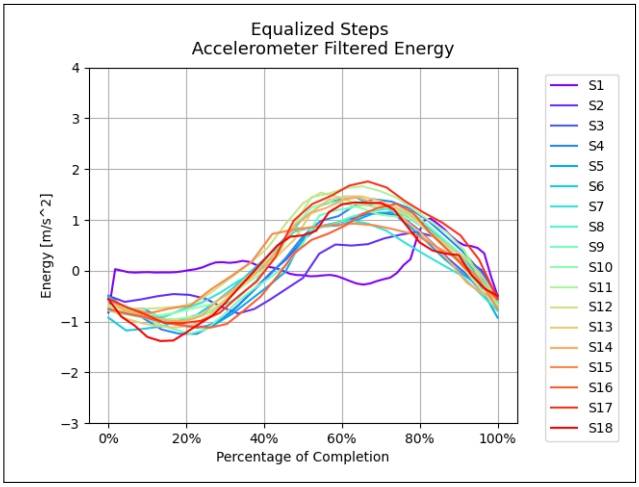

Figure 4.21: Equalized steps plotted with percentage of completion

At this point, we are aware of the **number of steps** the system has detected which was a total of 18 steps. Because we are aware that the first step always gets detected alongside some noise from the beginning of the trial, we have decided to remove it from future average calculations. It is now possible to estimate the average step length by dividing the trial covered distance by the number of steps. Then the average **stride length** is computed bearing in mind that a stride is composed by two steps, obtaining a total of 1.11 meters for this case. The step duration is calculated subtracting the accelerometer timestamp when the step starts from the timestamp when the step ends. This is repeated for every step and averaged out resulting in the average step duration for the user. The average **stride time** is then computed by doubling this value which obtained 1.27 seconds for this trial.

Listing 2 Our implementation of Step Detection by Perez and Labrador [15]

```
def step_detection_algo(As, A_calibration, T, w):
    def E(a): # Signal energy (Euclidian Norm)
        return math.sqrt(a[0] ** 2 + a[1] ** 2 + a[2] ** 2)
    def AMW(i):
        #Average Moving Window, where w is the window size
        summation = 0
        for j in range(i - w, i + w + 1):
            summation += E(As[j]) - A_calibration
        return summation/(2 * w + 1)

    AMWs = [AMW(a_idx) for a_idx in range(w, len(As) - w)]

    counter = 0 #Counts the remaining samples in the low peak search window
    prev_high, look_fw = False, False #state variables
    step_start_idx, step_end_idx = [], []
    step_start_idx.append(0) #First step timestamp

    for s_idx, s in enumerate(AMWs):
        if not look_fw and s >= T: #if high peak and within energy threshold
            prev_high = True
            continue
        elif look_fw:
            counter = counter - 1

        if prev_high and s<T: #after high, look for low
            prev_high = False
            look_fw = True
        if look_fw and s >= T:#after low, look for high
            look_fw = False
            prev_high = True
            counter = 2*w
            continue
        if look_fw and s <= -T: #after low, look for low
            step_end_idx.append(s_idx) #end of step
            step_start_idx.append(s_idx) #start of next step
            counter = 2*w
            look_fw = False
            continue
        if counter == 0: # back to initial state
            prev_high = False
            look_fw = False
            counter = 2*w

    return step_start_idx, step_end_idx
```

4.3.1.1 Dynamic Time Warping:

DTW between users steps: This algorithm was a partial reproduction of the work of Perez and Labrador [15]. In order to compare different steps from a user, the DTW is applied between the two step signals in order to compute the minor distance between them. This can be repeated for each detected step resulting in a comparison matrix of all the distances between all the step combinations, resulting in an average distance value between all the steps. This is important to study the progression of a user's gait over time because this can be used to compare if the user's step pattern is getting more or less identical over time. During the course of this dissertation, we designated the average DTW distance between all step combinations as **step similarity**.

In our implementation we have decided to remove the first and last steps from the computations. The first step usually has lower acceleration and carries noise from the trial beginning. The last step is influenced by the turn at the end of the trial. Like this, we try to make sure all the steps are from a rhythmic gait pattern. Using these steps would lead to unrealistic values for the average distance between all step combinations since the normal rhythmic steps are very different in pattern from these two.

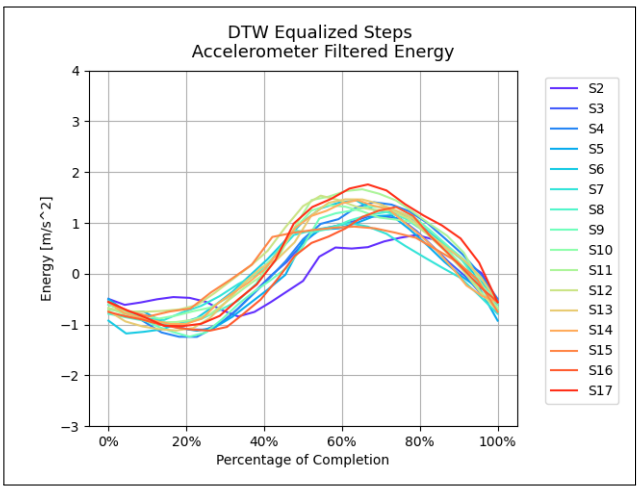

Figure 4.22: Selected equalized steps for Dynamic Time Warping computation

The chart in Figure 4.22 has indexes starting from 2 on purpose in order to show which steps have been removed when comparing to Figure 4.21, which were the first and last steps. By partly reproducing the work Perez and Labrador [15], we calculate the DTW distance between each of the selected steps with each other, resulting in a comparison matrix. The obtained distance values are then laid in an array in order to compute the average of all the step combinations distances, computing "step similarity". The developed procedure for this computations can be seen in Code 3.

For example, for this user, by computing the DTW of each step combination and averaging the obtained list, the resulting average distance (**step similarity**) for all the user's steps was of 0.8. Because this is a distance metric, the lower this value gets, the more similar the user's steps are between each other. The step combinations with higher

DTW values might be important to review since these might be the exact moments when a patient is limping or having a hard time walking overall.

Listing 3 Our implementation of DTW

```
  function DTW(S, T):
    initialise_table(Distances, len(S), len(T))
    initialise_table(DTW, len(S), len(T))

    //Store distances in table.
    for idxS, sampleS in S:
        for idxT, sampleT in T:
            Distances[idxS][idxT] = (sampleS - sampleT)**2

    for idxS in S:
        for idxT in T:
            //Accessing a position outside table bounds returns 0
            DTW_prev_row_prev_col = DTW[idxS - 1][idxT - 1]
            DTW_prev_row = DTW[idxS - 1][idxT]
            DTW_prev_col = DTW[idxS][idxT - 1]

            DTW[idxS][idxT] = Distance[idxS][idxT] + \
                min(DTW_prev_row_prev_col, DTW_prev_row, DTW_prev_col)

    return sqrt(DTW[len(S) - 1][len(T) - 1])

function DTW_all_steps(steps: number[][]):
    DTWs = []

    //Assumes at least 1 step
    for step in steps:
        stepDTWs = []

        for nextStep in range(S, indexOf(step) + 1):
            stepDTWs.add(DTW(step, nextStep))

        if(stepDTWs.length != 0):
            DTWs.add(stepDTWs)

    return DTWs
```

DTW between ST and DT signal : Inspired by the DTW use-cases presented by Derawi et al. [29] and Perez and Labrador [15], we have decided to test a new metric out of the thought that if we can compare the distance between any two signals, then maybe this could be applied directly to the dual-task paradigm.

This can be used to evaluate the effect of the secondary task during the DT trial. The further from 0, the more distinctive these signals are and the more affected we assume the user was from the secondary cognitive task. To the best of our knowledge of the reviewed articles and research, we don't believe the DTW technique has been directly implemented to determine the distance between single and dual-task trials signals of the same person. This might constitute an interesting point of study that we will be reviewing later on in Chapter 5.

The procedure for this calculation was very simple. We removed the bias from each of the accelerometer signals, from the ST and DT trials, and calculated the energy for each of those. Then, these signals were sent directly to the DTW procedure that was used before. The result is a distance metric that can be comparable between users.

By using DTW on these signals, and comparing them point by point, we can obtain a distance metric to evaluate how different these signals were, as seen in Figure 4.23.

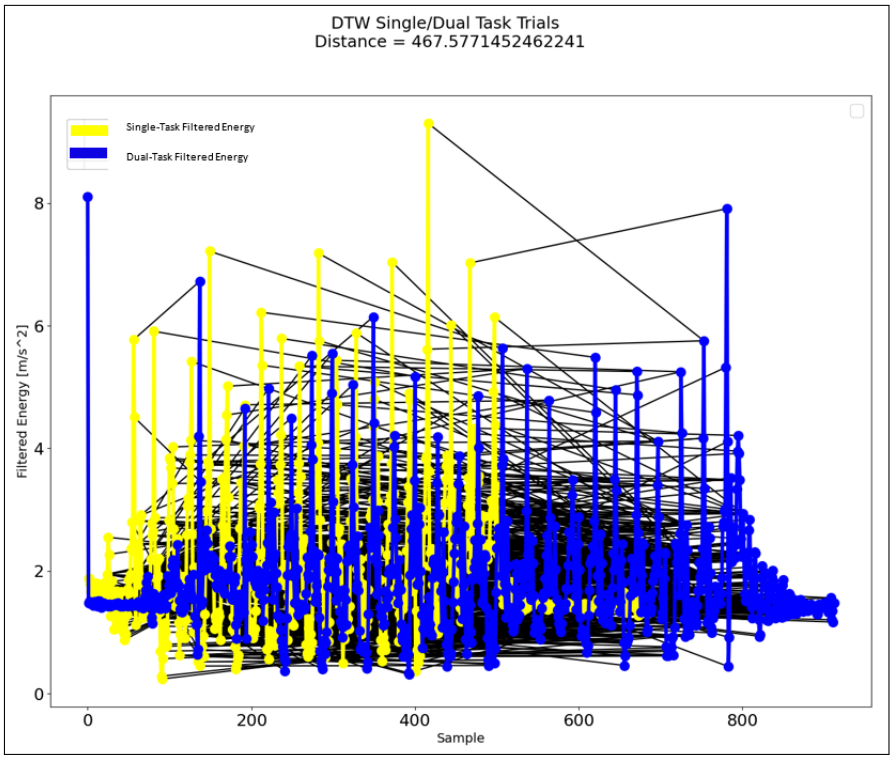

Figure 4.23: Single and Dual Trials Similarity

This is used to compute what we called of **trial distance** in this project, which obtained a distance value of 467.58. Even though this is a metric under test, this could present very useful, to study if a user was rather affected or not by the presence of a simultaneous cognitive task while walking. This strategy can be used as an indicator for affected motor

skills based on the cognitive task demand. We hypothesize that the higher the similarity, the less affected a user is by the cognitive task and the lower the distance between the signals is.

Autocorrelation: Reproducing the work of Perez et al. [15] and Moe-Nilssen et al. [16], previously reviewed in Chapter 2, we performed autocorrelation analysis to the retrieved y-axis accelerometer signal.

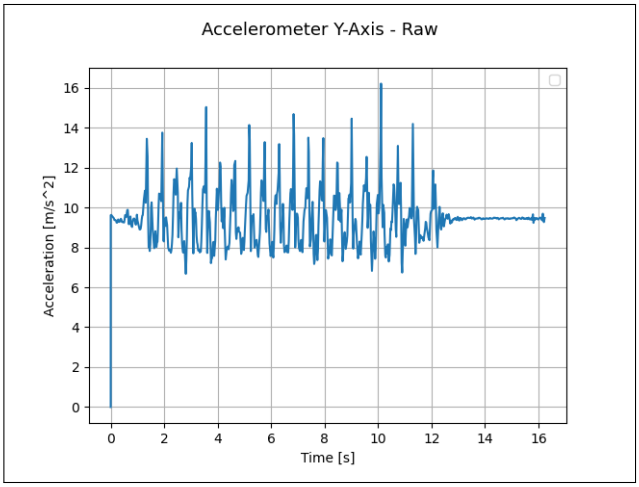

Figure 4.24: Raw y-axis accelerometer readings

Alterations to the original algorithm include segmenting the signal before anything else. This was necessary in our implementation since initial tests have showed that the initial noise before the first step and the rotation noise at the end of the signal were interfering with the obtained values. Because of this fact, we started by segmenting the signal to only keep the samples between the end of the first step and the start of the rotation, making sure that the new signal only contains steps with normal patterns. Because we have segmented the signal, the corresponding timestamp array must be corrected. To do this, we subtract the timestamp of the new first position of the time array, corresponding to the timestamp associated with the sample when the first step ends.

After the preparation, the segmented y-axis accelerometer bias is removed by subtracting the average of acceleration values of the segmented y-axis accelerometer signal. At this point, We are ready to apply a linear interpolation to the signal, which is illustrated in Figure 4.25.

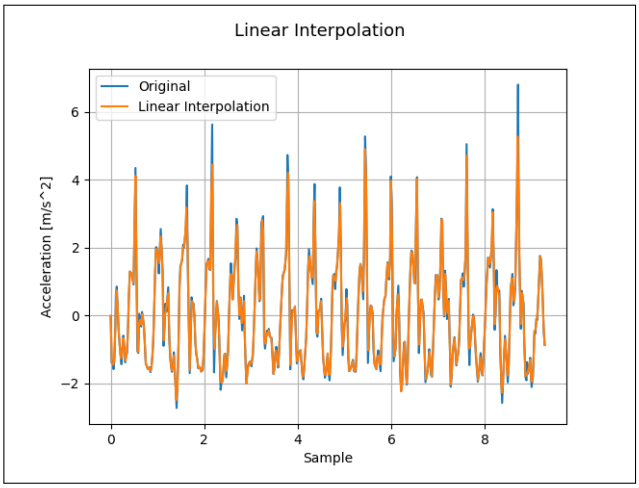

Figure 4.25: Resulting segmented interpolated y-axis accelerometer signal

Then the correlation function was applied using the segmented interpolated y-axis accelerometer signal as both arguments which is equivalent to the autocorrelation, obtaining the chart in Figure 4.26.

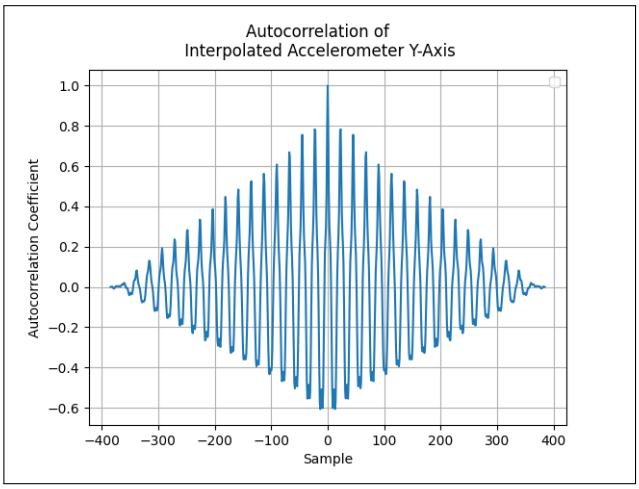

Figure 4.26: Interpolated Y-axis accelerometer autocorrelation

Following the procedure presented by the authors, we detect the two dominant peaks equivalent to the dominant local maxima. The dominant local maxima is discovered and associated with ad1 and ad2 that represent **step regularity** and **stride regularity**, respectively. We detect the two dominant local maxima as seen in Figure 4.27.

The associated values with A_{d1} and A_{d2} are 0.78 and 0.76, representing normal values for a single-task trial. From these values we can also calculate **step symmetry** by assessing the ratio of $\frac{A_{d1}}{A_{d2}}$ which is equal to 1.04 in this case. Values closer to 1.0 represent perfect symmetry between left and right steps. To finish the procedure, we investigate the sample indexes with the corresponding peaks in the time-domain in order to extract the values for each timestamp, as indicated in Figure 4.28.

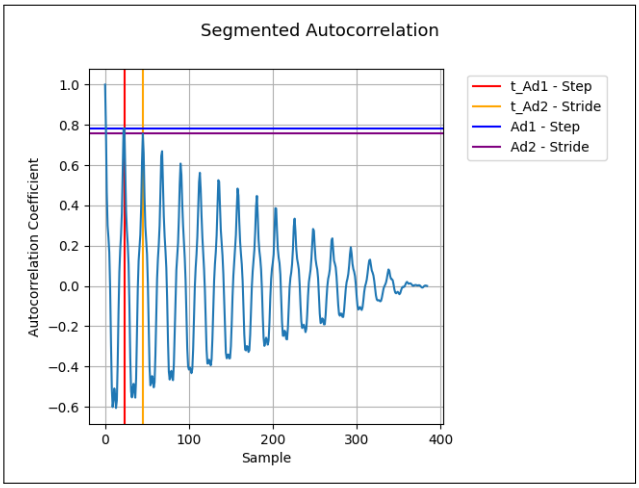

Figure 4.27: A_{d1} and A_{d2} detection in frequency-domain

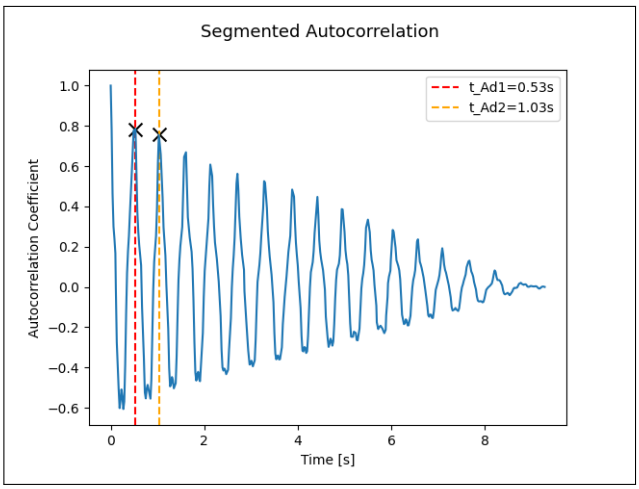

Figure 4.28: A_{d1} and A_{d2} detection in time-domain

From the chart, we can extract the values of the associated timestamps which are at 0.53 seconds and 1.03 seconds. From the timestamp associated to A_{d1}, we can directly extract **cadence**, which by computing $\frac{60}{t_{A_{d1}}}$, obtaining a total of 113.21 steps per minute. From the extracted cadence and using the previously calculated average **stride length**, we can also derive **walking speed**, using Equation 2.15, which is equal to 1.05 meters per second for this case. Each of these signal computations presented in this Subsection are then repeated the same exact way for the dual-task trial.

Listing 4 Our implementation of Autocorrelation

```
def autocorrelation(acc_y, acc_ts, start_idx, end_idx):
    #Cleaning the signal after the first index of complete step
    acc_ts = acc_ts[start_idx:end_idx]
    acc_y = acc_y[start_idx:end_idx]
    A_bias_y_segmented = AVG(acc_y)
    #Remove the bias from the segmented y-axis accelerometers
    acc_y = [origin - A_bias_y_segmented for origin in acc_y]

    segmented_idx = [origin + start_idx for origin in segmented_idx]

    #Adjust timestamps after segmentation
    acc_ts = [ts - acc_ts[0] for ts in acc_ts]
    num_ts = len(acc_ts)

    #Compute the linear interpolation for the y-axis accelerometer signal
    linear_interp = interp1d(acc_ts, acc_y, \
                                kind='linear', fill_value='extrapolate')

    delta_t = acc_ts[-1] - acc_ts[0]
    N = (len(acc_ts)-1)
    equally_spaced_times = [i * delta_t/N + acc_ts[0] for i in range(0, num_ts)]

    linear_results = linear_interp(equally_spaced_times)

    def acf(x):
        x = x - np.mean(x)
        corr = np.correlate(x, x, 'full')
        return corr/np.max(corr)

    acf = acf(linear_results)
    lags = [i-(len(acf)-1)/2 for i in range(0, len(acf))]

    acf_after_zero_phase = acf[int(len(acf)/2):]
    acf_time_zero_phase = lags[int(len(acf)/2):]

    peak_detect_threshold = 0.2
    peak_timestamps = []
    peak_values = []
    peak_index = detect_peaks(acf_after_zero_phase, \
                 mph=peak_detect_threshold, mpd=2, edge='falling',show=True)

    for peaks in peak_index:
        peak_timestamps.append(acf_time_zero_phase[peaks])
        peak_values.append(acf_after_zero_phase[peaks])
```

Feature	Unit	Single-task trial	Dual-task trial
Number Steps	Step	Yes	Yes
Stride Duration	s	Yes	Yes
Stride Length	m	Yes	Yes
Walking Speed	m/s	Yes	Yes
Step Regularity	NA	Yes	Yes
Stride Regularity	NA	Yes	Yes
Step Symmetry	NA	Yes	Yes
Cadence	Steps/min	Yes	Yes
Step Similarity	NA	Yes	Yes
Trial Distance	NA	No	Yes

Table 4.1: Structure of the extracted feature array for the user

Final Feature Array: After the computation of each feature, the ones that are composed by arrays of values are then extracted in the form of a mean reducing the dimensionality. This module is also responsible for setting the processed users final features into a database child, which we called of results containing all the calculated features for both trials.

Even thought other features, such as step length, step time, cognitive and motor DTC, among others are computed, not all features were used in the classifier implementation. The DTC should already be reflected in the differences between the trials and both of these are in the feature vector that will be used in classifiers. By this point, the final array of features for each user should then be, as seen in table 4.1. Each user computations lead to a final feature vector with a total of 19 features. Of these features, 9 are extracted from processing the ST signals, 9 are extracted from processing the DT and trial distance.

4.3.2 Classifiers

Classifiers is the module responsible for creating, training and employing the machine learning models used by the system. Figure 4.29 depicts the architecture of the machine learning module. This module implementation consists of two main parts: the training of the classifiers and the prediction process itself.

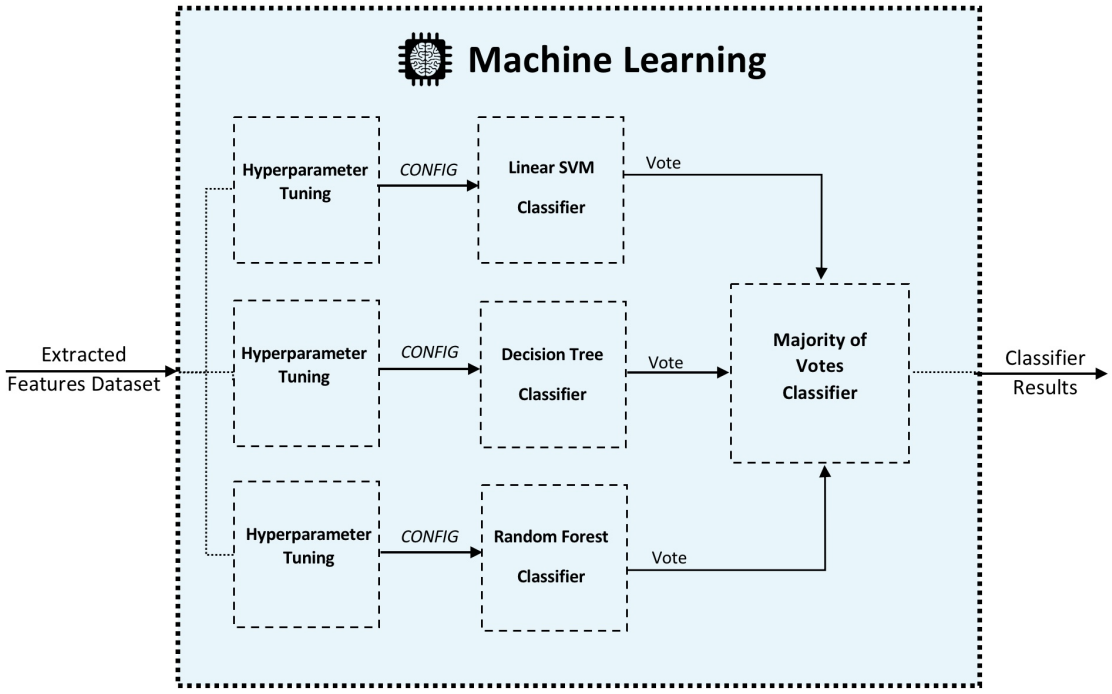

Figure 4.29: Machine Learning Module

The final feature array of each user in training mode are split between two separate Comma-Separated Values (CSV) files, one for 'control' data and one for 'target' data. Once there is a new request for classification, these files will be used as input to re-train the models if, and only if there were any new entries since the last training event. This verification is tested using the database configuration flag 'Training_Necessary'. This flag is set every time new training data is available. Then, as seen in Figure 4.29, different supervised learning algorithms were implemented using the *sklearn* library. These were: the Linear SVM Classifier[4], the DT Classifier[5] and the RF classifier[6].

For each of these, we started by setting their hyperparameters, which were selected by running a grid search[7] over a range of reasonable values for each of the hyperparameters. The developed strategy trained all the selected classifiers with all possible combinations, or in other words, the Cartesian product between the provided ranges of values. For

[4] Scikit LinearSVC documentation
[5] Scikit Decision Tree Classifier documentation
[6] Scikit RandomForestClassifier documentation
[7] Scikit GridSearch documentation

each hyperparameter configuration, the system trains, tests and uses a k-fold validation[8], with $k = 5$. The reasoning behind using k-fold validation is that given our small dataset size, this method allows us to train and test the entire dataset by splitting it into k bins and averaging the results of k experiments. Importantly, the classifiers undergo parallel training which reduces the duration of this step. Finally, the script outputs the best parameter configuration for each classifier. These values, outlined below, were then substituted in the models that will be used in the project:

SVM For SVM we used a "linear" kernel and C was defined to 0.1, the degree to 2 and the gamma is "scale". These parameters obtained a final accuracy of around 0.67.

DT Then, for DT, the best parameters used a "Gini" criterion, had a maximum depth of 10 with the minimum number of samples necessary to be at leaf node was of 1 and the minimum number of samples required to split a node was of 3. This classifier obtained an accuracy of around 0.83.

RF Finally, for RF, the bootstrap was defined to 'True', the maximum depth was set to 10, the minimum number of samples necessary to be at leaf node was of 3, the minimum number of samples required to split a node was 4 and we should use 50 estimators. This classifier obtained a perfect final accuracy of 1.0.

With our classifiers and parameters selected, we are now ready to implement the MV classifier. This classifier uses a voting system, as seen in Section 2.3.3.1, where each of the implemented classifiers votes for a class as their final prediction. These votes are averaged down into a single final score which is rounded to obtain the prediction. For example, if two of our classifiers vote for 'target' class, then we will obtain a vote score of around 0.67, which will be rounded to 1 and this will be the prediction label outputted our final classifier. Based on this final decision, the database is then updated with the classifier prediction result as well as the vote ratio score.

[8] Scikit k-fold documentation

4.3.3 Clinician GUI

The **clinician GUI** module was created to visualize and interact with the processed data. When executed, the opening format of it can be seen in Figure 4.30. In the *dualgAIt: Desktop* side, the clinician responsible for handling the software, will be able to inspect the collected data and all the proceedings that were dealt with in-between, giving the transparency that is needed to connect the bridge between technology and clinical expertise. The GUI was implemented using the library 'PySimpleGUI'[9]. It's opening screen can be seen in Figure 4.30.

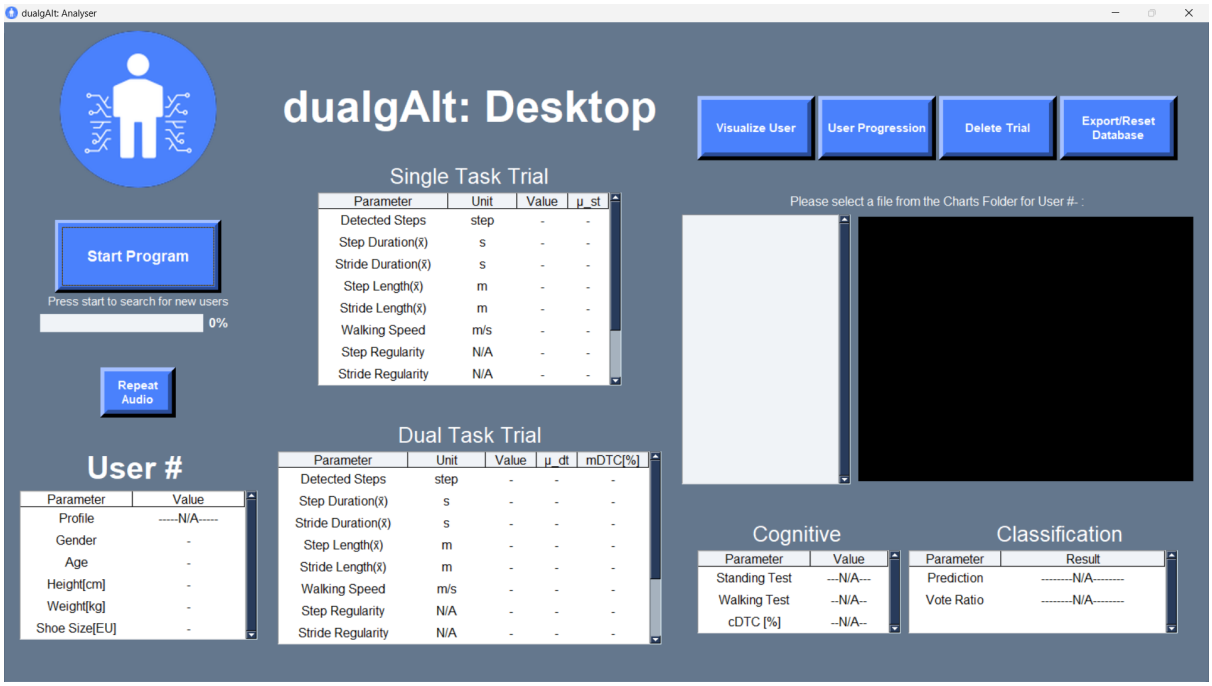

Figure 4.30: Opening screen

The system starts by launching the GUI thread. Then the system constantly waits for unprocessed user data. When new data is available, the system is ready to process the next user. It waits for the user to start the system, and when it does, it sets an event allowing for the start of a new thread, the main loop thread. The system runs the previously described process. The integration of the system's main loop with the graphical part was made using threads.

When new data is available, the system starts by prompting the clinician to listen to the audio, as depicted in Figure 4.31, and to input the correct number of serial subtractions for the DT trial, presenting the static solution array.

After the clinician input, we are now set to compute every metric in an automatic way. The system starts by processing the ST trial. Then processes the DT trial in a similar way, as illustrated in Figure 4.32. After this, the system computes metrics for Motor Dual-Task Cost (mDTC) and cDTC.

[9] PySimpleGUI Documentation

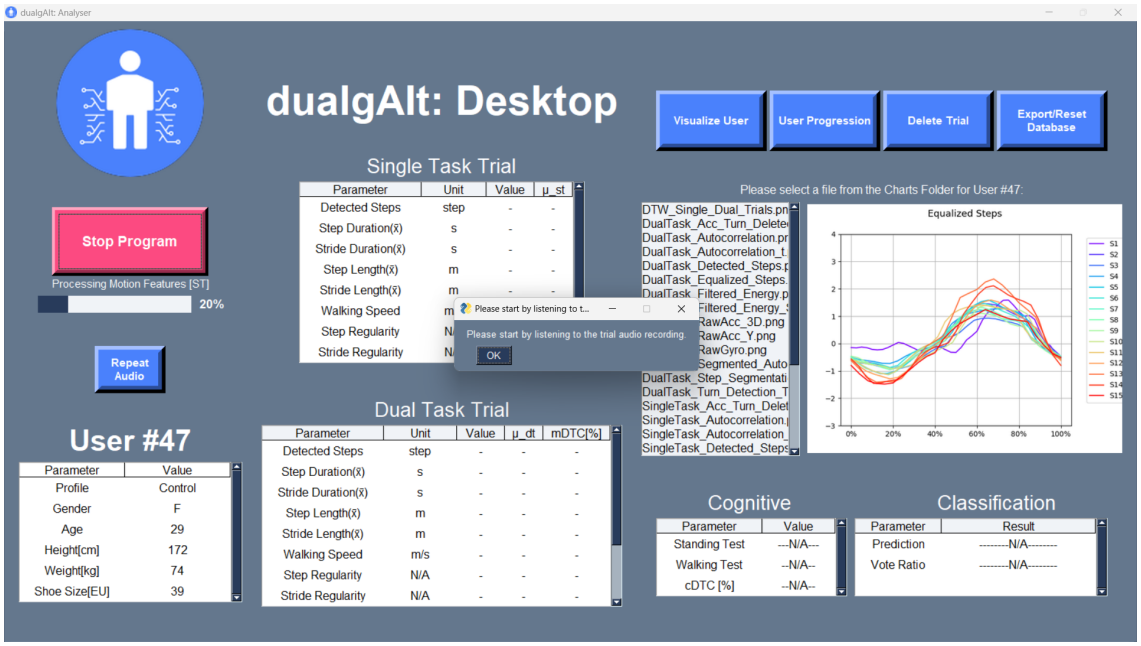

Figure 4.31: Dual-task cognitive test audio recording prompt

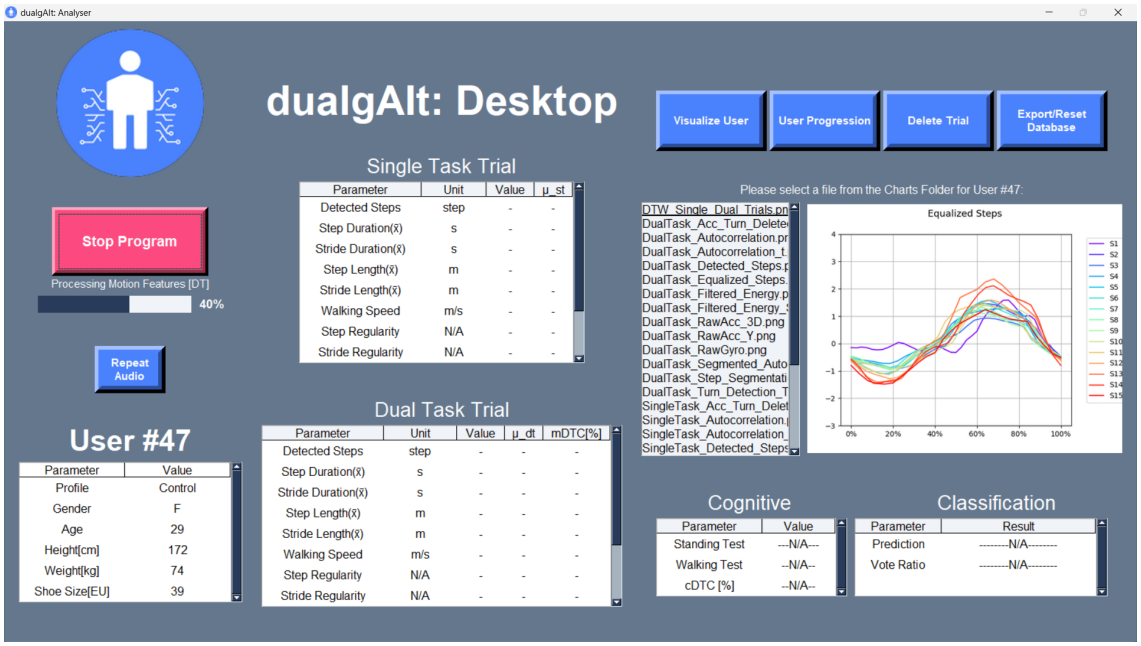

Figure 4.32: Motor and cognitive metrics being processed for both trials

Below, a list of the main components displayed by the GUI alongside with their major functionalities can be found:

- **Start program button:** This button is responsible for controlling the system flow. When it is pressed the system starts checking for unprocessed users in the database. When this happens, it starts processing the user, running the main processing loop.

- **Progression bar:** This bar represents the percentage of completion of the processing,

by checking the triggering of certain completion events. It has levels of 40% for completion of motion processing on both single and dual-task trials, 60% for the completion of the cognitive tests computations, 80% for the train-classifiers or classify-user task and finally, 100%, when the system is done computing all the data and is ready to update it's dynamic components.

- **Table for user survey:** This table represents the survey information input by the user in the android application.
- **Table for single-task:** This table displays the obtained results for the feature array obtained by processing the obtained signals from the single-task trial. The table includes the feature/parameter, it's unit, the obtained value for the trial and the average of all the already processed user's for the single-trial.
- **Table for dual-task:** This table displays the obtained results for the feature array extracted from the dual-task trial. It follows the same structure as the Table for single-task but it also has a column that indicates the mDTC, in percentage, for each feature.
- **Table for cognitive tests results:** This table shows the cognitive tests results obtained by the system. These include the accuracy during the standing and walking tests as well as the computed cDTC.
- **Table for classification:** This table shows the classification results obtained by the system if the current user being processed, has the 'To Classify' profile type.
- **Image display:** To keep the computations as transparent as possible to the team performing the trials using the system. It displays a set of images, allowing for visualization of the internal computations and the origin of the generated feature array data.
- **Repeat audio button:** This component is associated with the cognitive test. Pressing it repeats the audio from the dual-task trial, allowing the clinician to listen and process how many correct entries were during the cognitive task, as many times as necessary. When the audio stops, the system prompts the clinician to enter the number of correct answers for later comparison with the stand cognitive trial.
- **Visualize user button:** Pressing this button generates a popup prompting the clinician to input the desired user identifier to visualize. The system needs to be stopped first and this user should already exist in the database. If the conditions check, the dynamic components are updated to the selected user data.
- **User progression button:** The user progression button generates a very simple PDF to '/Progression/Progression_UserID_{IDSessions}', filled with data comparing the users last trial with an average of the other past trials allowing for assessing gait

progression evaluation over time. The 'IDSessions' are the identifiers that should have been input in option 2: Progress Evaluation during the trials. An example of the generated PDF file can be visualized in Figure 4.33.

- **Delete trial button:** Delete trial button allows a deletion of the last processed trial. This is useful for when the clinician is that something went wrong with last set of trials.

- **Export/reset database button:** Pressing this button will prompt the clinician that the entire database will be exported to '/Past_Experiences/database_{Date.Today()}'. This is very useful when changing experiences, so that the clinician can automatically save all the data. Then, the systems prompts the user if it should delete all the data in the database. Responding 'yes' will delete all the data from the firebase database, preparing the system for a overall new run.

Progression_User45_Sessions_[43 46].pdf 1 / 1 100%

Progression Evaluation: User45
Comparing to Sessions: [43 46]

	Unit	Current45	$\bar{x}$_Past	σ_Past	mDTC [%]
ST - Detected Steps	Steps	14.0	15.0	0.0	N/A
ST - Step Duration	s	0.53	0.66	0.04	N/A
ST - Stride Duration	s	1.06	1.32	0.07	N/A
ST - Step Length	m	0.71	0.67	0.0	N/A
ST - Stride Length	m	1.43	1.33	0.0	N/A
ST - Walking Speed	m/s	1.43	1.19	0.04	N/A
ST - Step Regularity	N/A	0.79	0.74	0.02	N/A
ST - Stride Regularity	N/A	0.68	0.69	0.01	N/A
ST - Cadence	Steps/min	120.0	107.28	3.83	N/A
ST - Step Similarity	N/A	5.5	6.49	2.94	N/A
DT - Detected Steps	Steps	14.0	16.5	0.5	0.0%
DT - Step Duration	s	0.56	0.69	0.01	-5.66%
DT - Stride Duration	s	1.12	1.38	0.02	-5.66%
DT - Step Length	m	0.71	0.6	0.02	0.0%
DT - Stride Length	m	1.43	1.21	0.04	0.0%
DT - Walking Speed	m/s	1.43	1.0	0.08	0.0%
DT - Step Regularity	N/A	0.76	0.72	0.02	3.8%
DT - Stride Regularity	N/A	0.76	0.67	0.02	-11.76%
DT - Cadence	Steps/min	120.0	98.6	4.85	0.0%
DT - Step Similarity	N/A	4.65	6.63	2.92	15.45%
DT - Physical Affectance	N/A	328.01	365.9	15.73	NA%

Figure 4.33: Progress evaluation generated PDF. Comparing User45 with past sessions 43 and 46

When all the computations from the "Mainloop" are complete, the dynamic components will be updated with current user's results, as illustrated in Figure 4.34.

The clinician will now be able to inspect the bundle of generated charts, all the important features data and the results for both cognitive tests. It's also possible to mark the rows for easier comparison between both trials or different features, which can be seen in Figure 4.35.

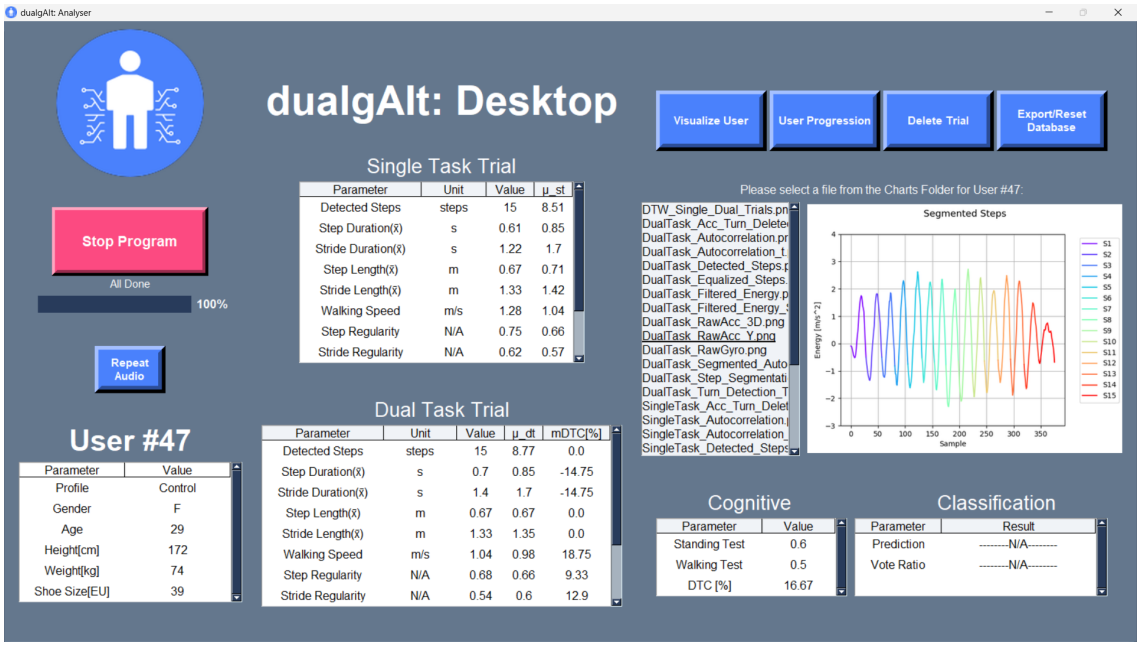

Figure 4.34: Process complete and table data is updated

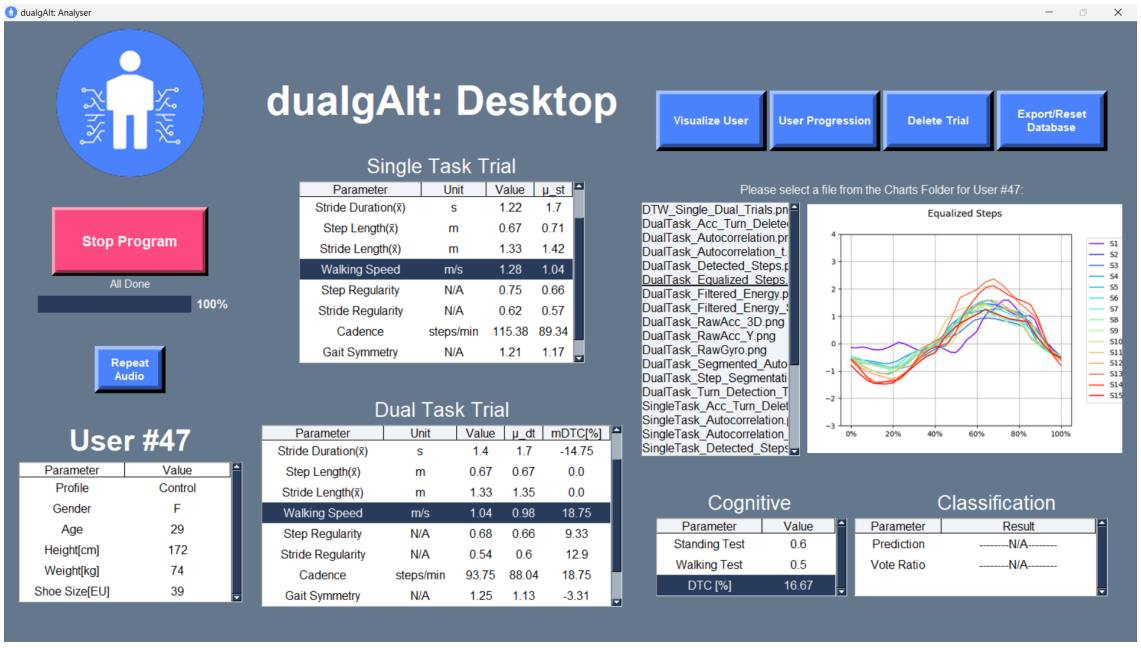

Figure 4.35: User can now select which generated chart to visualize and mark different rows

In the case that the user was set for being classified, the system will also present the classifier results once the processing is complete as depicted in Figure 4.36.

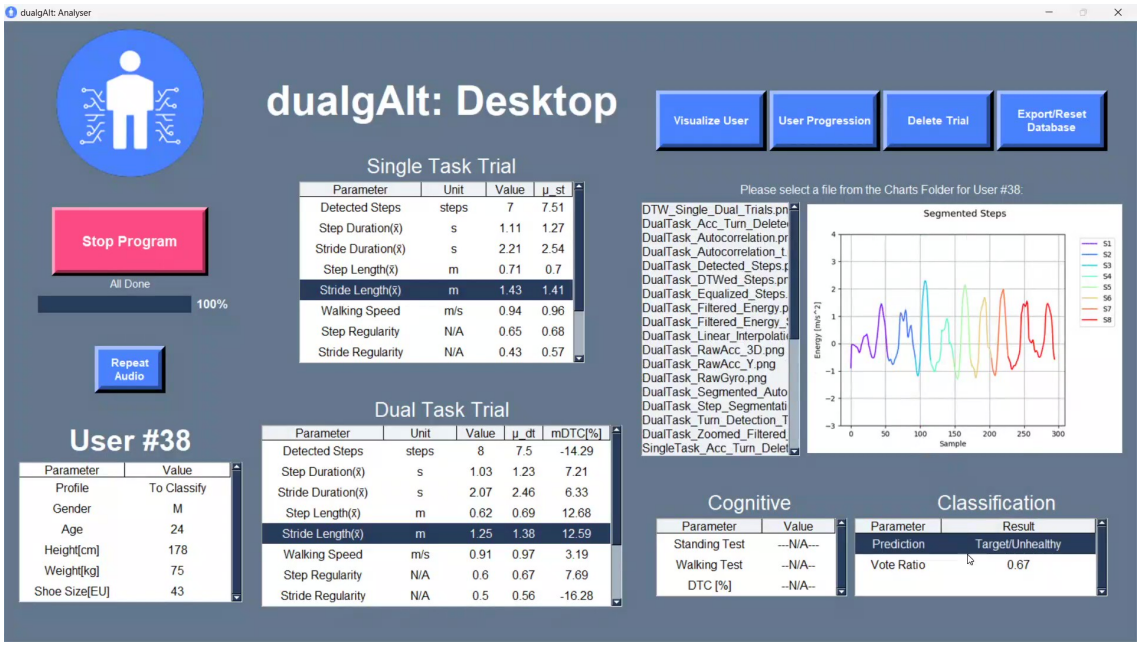

Figure 4.36: Classifier prediction and vote ratio

When this thread is completed, the number of processed users is incremented and the dynamic components of the GUI are updated to the current user. After the system implementation review, we will see some practical examples of application of the prototype. In the next chapter, Chapter 5, the system will be employed in two different experiments.

5

Evaluation and Results

In this Chapter, we present and discuss the evaluation of our system through two sets of experiments designed to both validate the system as a whole, its signal processing and step detection algorithms and the performance of the integrated machine learning classifiers. We begin by detailing the shared experimental methodology in Section 5.1.1, then focus on each experiment and its results in Sections 5.1.2 and 5.1.3. We conclude with a detailed discussion on the obtained results in Section 5.2.

5.1 Evaluation

5.1.1 Experimental Methodology

For both experiments the smartphone used was a *Xiaomi Redmi Note 11S* running Android 11. The smartphone was fixed to the user by using an elastic band attached to a smartphone case, which can be seen in Figure 5.1.

Figure 5.1: Elastic band and smartphone case used to fixed the smartphone

Upon filling the survey and taking the smartphone cognitive test, the smartphone is placed on the user's lower back, near the L2 to L5 vertebrae, tight enough not to move too much, as presented in Figure 5.2.

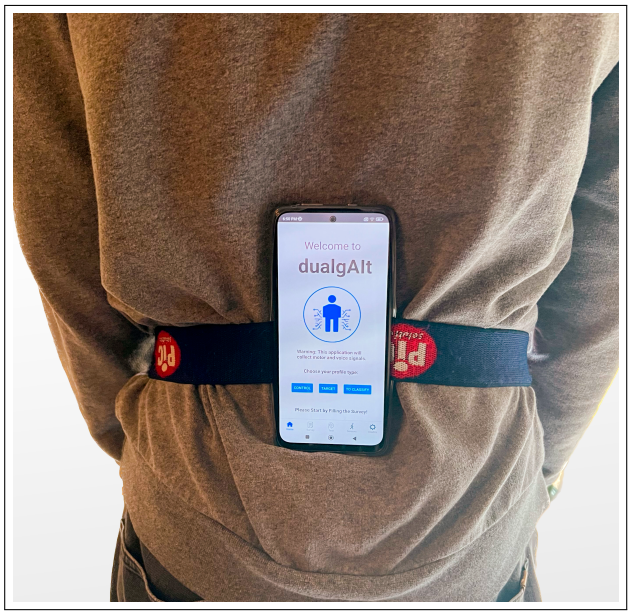

Figure 5.2: Smartphone fixed to the lower back using elastic band and case

All the trials using the developed system were performed under a self-selected comfortable speed. The 5 and 10 meters distances used were hand-measured using a measuring tape. Depending on the intention of the examination, the clinician starts by pressing the appropriate button: 'control' or 'target', if they intend to analyse the user's gait and save this data to train future models or 'toClassify' if there is a desire not only to assess the user's data but also get a class prediction, from the MV classifier, about this user's gait pattern.

5.1.1.1 Trial instructions

The experiment and instructions were inspired by the work of Bristow et al. [57]. The trial starts by the clinician then hands the phone to the user being tested saying:

> **Clinician → User:** " Remember you can stop the trials at any moment. Please start by filling the survey. Then you will perform a simple test that consists on repeatedly subtracting 7 from a number. You will start by subtracting 7 to the number that appears on the top of the cognitive test and then 7 to each number you answer until you are done or the timer is done. Switch to the test tab and press start whenever you are ready."

When the user is done with the single cognitive task test or the timer is over, we are set to start the walking trials. The phone is fixed to the user's lower back and before starting the single-task trial:

> **Clinician → User:** "Now, you will walk at a comfortable speed of your choice from this starting point to that finishing point. When you reach the finishing line, please make a turn a face the starting point. I'll say 3,2,1, go. Did you understand the task? If you did, please begin."

The clinician presses the "start single-task trial" button and waits. When the user is facing the starting point and the trial is over, the clinician presses the finish single-task trial button and asks the user to return to the starting point. Now, these are the instructions that for the user receives for the dual-task trial:

> **Clinician → User:** "Now, we will repeat the same task but we will try to add another task at the same time. While walking you will try to serial subtract seven to a certain number I will give to you. You will start by subtracting 7 to the number I hand you and keep subtracting 7 to that result until reaching the finish point. There is no minimum or maximum number of answers. Try your best performing both tasks. I'll provide you with the starting number and say 3,2,1, start. Did you understand the task? If you did, your number is 66, you can now start."

At the end of the trial, the clinician can choose to provide an identifier to the user. This is to maintain confidentiality and be able to evaluate progress in future sessions, if that's the case.

5.1.1.2 Scoring cognitive accuracy

When the user passed the finish line and is facing the starting point, the clinician presses the finish trial 2 button. The experiment is now over and all the necessary information will be available for the clinician in *dualgAIt: Desktop*. This includes the audio with the answers to the dual-task serial subtraction problem as well as the score to the standing cognitive serial subtraction problem.

For experiment A and experiment B, the walking path was defined as a straight line with the distance of 5 meters and 10, respectively. Users should walk normally reaching for the end-point, where they should turn and face the opposite side, the starting-point.

For experiment B, the experiment was ran with real volunteers, to which a voluntary consent participation form was handed. Those willing to participate in the trial and have their data studied and published, signed the agreement present in the Annex I. This form explains the purpose, the possible risks, the collected data and the voluntary nature of the trial. To be a participant, the document was signed when the user was willing participate in the study. In this experiment there was the objective of exploring which gait features are more affected by a simultaneous cognitive task while performing the walking task.

To do this, an experiment was designed using Serial Subtraction by 7 [57] problems. The first run, the standing cognitive test, is made via smartphone. Then, during the

dual-task trial, the user is prompted to make serial subtractions by 7 given a starting number, while walking. The selected number was 66.

To evaluate the user's answers we defined that the number of correct answers is counted only for the first 4 iterations of subtraction. Errors are not propagated as the only condition for a correct subtraction, is the actual subtraction with the last number that the user verbalized. The number 4 was selected given that in our pool, the average number of answers during the dual-task trial was 4, so we decided to use this number as a reference.

The application is started and the survey is filled. The trial always runs in the same order. The phone is placed in the lower back and the start trial button is pressed.

5.1.2 Experiment A: Motor Impairment Simulation

The control group for this experiment representing a healthy gait was realized barefoot while the target group, representing an impaired or "unhealthy" gait, will be simulated by a boot on the right foot, causing a disturbance to the user's gait. This impairment simulation or difference in height in one of the sides should be enough to enforce a less stable gait profile. All the tests for this experiment were run by the same user even thought different kinds of walk (e.g. speed, stability) were tested for both groups. This represented the first set of tests for the system in order to test it's performance. This experiment was ran with a 5 meters covered distance.

In order to test the classification module, two groups were simulated. The negative here is considered to have an 'Unimpaired Gait'. On the other end, the target group of this experiment is considered to be 'Impaired Gait'. The system's covered distance is calibrated to 5 meters. The system was ran 30 times (2 trials for each) - 15 times with 'Control' and 15 times with 'Target', without the usage of the cognitive test module, with the purpose of testing the classification module. After extracting the feature vectors from each run, the classifiers were optimized by a grid search and we ran 10 trials, pre-labeled and enforcing similar conditions using the system in the 'To Classify' mode. We will start by comparing the generated charts for two trial examples. The images in the left will be figuring the 'Impaired Gait' group which was simulated by the boot. In the right side, the 'Unimpaired Gait' group will be depicted. In Figure 5.3, we can see the step segmentation for both cases.

From this Figure 5.3 it's possible to see that the unimpaired gait trials usually detect less steps because there are more missed detections. In this case, there was a clear missed detection in the impaired gait case while the unimpaired gait has a clear detection. The obtained stride length was of 1.25 meters for the impaired case while it's counterpart obtained an average length of 1.11 meters. As for stride duration, the left side obtained 2.4 seconds while the right side attained 1.95 seconds. There is also a difference pattern between the steps given by the foot wearing the boot and the ones without. In Figure 5.4, we can visualize the equalized steps for each case.

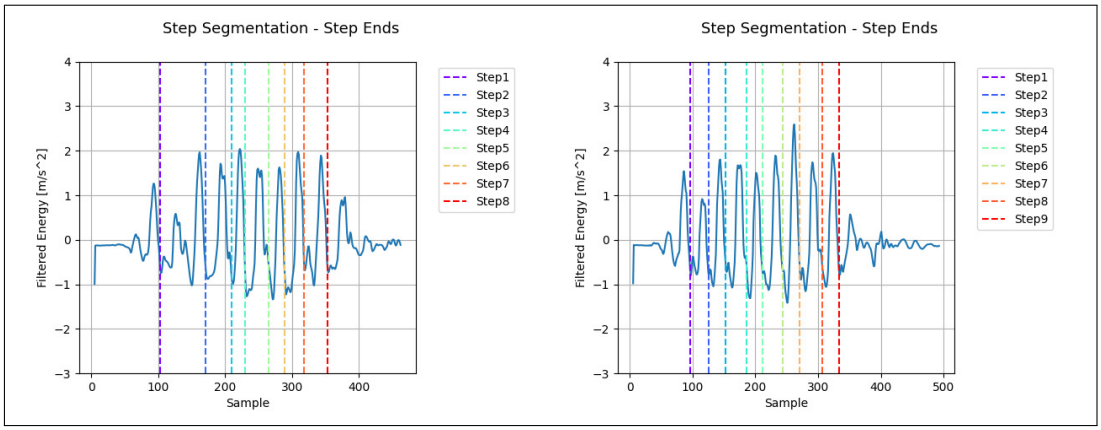

Figure 5.3: Step segmentation for impaired gait (left) and unimpaired gait (right)

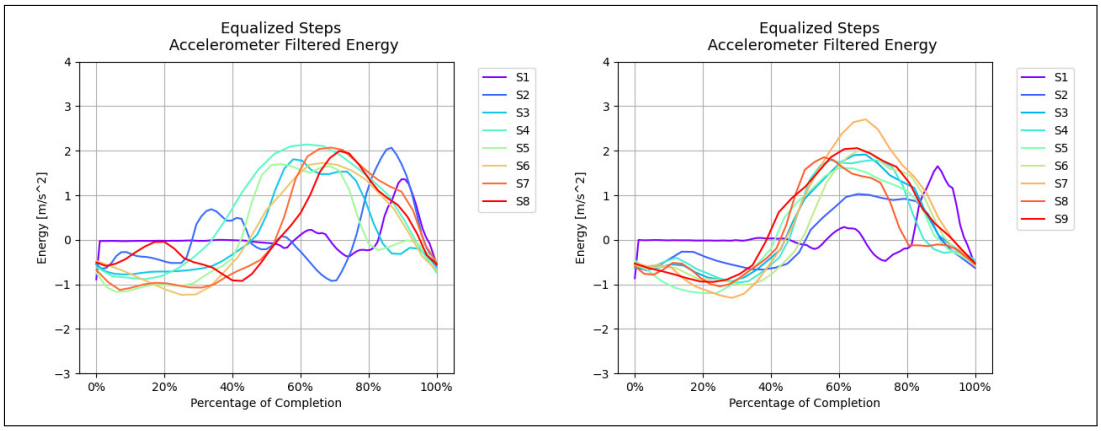

Figure 5.4: Equalized detected steps for impaired gait (left) and unimpaired gait (right)

By computing the DTW between all the obtained steps (except the first and last steps), we obtain values of 0.82 for the impaired case and 1.33 for the unimpaired one. In Figure 5.5, we can see the results for autocorrelation analysis in both cases.

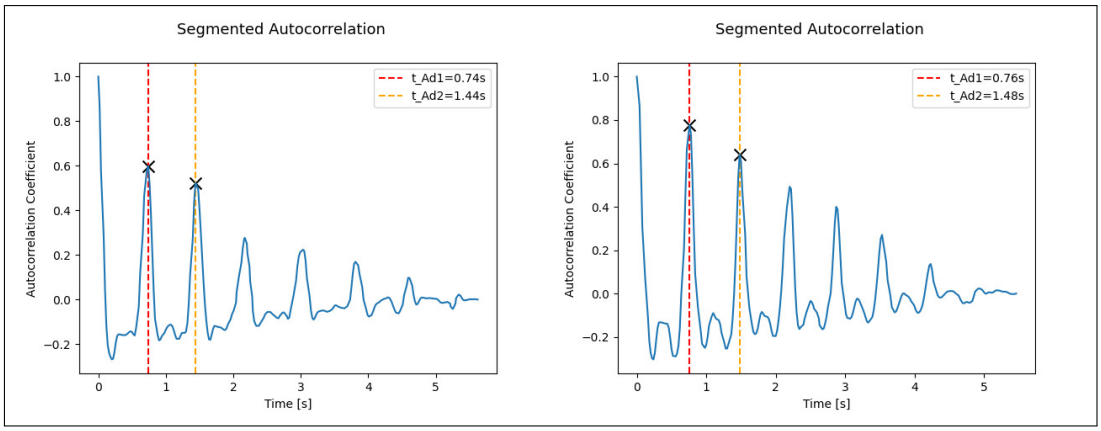

Figure 5.5: Autocorrelation analysis for impaired gait (left) and unimpaired gait (right)

When it comes to symmetry and regularity, for the step regularity and stride regularity, the impaired case obtained 0.6 and 0.52 respectively, while it's counterpart attained values

of 0.78 and 0.64 for the same features. The resulting computations for the step symmetry result in 1.14 for the left case and 1.21 for the right case. When it comes to the cadence and walking speed, the impaired case obtained 81.08 steps/min and 0.84 m/s, respectively. The unimpaired case obtained 78.95 steps/min and 0.73 m/s.

These 10 classification runs were a combination of 'Unimpaired Gait' (n=5) and 'Impaired Gait' (n=5). The feature vectors from these pre-labeled "users" were sent for prediction, with the intention of testing if the system would be able to predict each user's original class. For this test we did not consider the last step as a misdetection as our system implementation uses the rotation at the end of the trial as protocol. This means the last step will usually have lower acceleration values, making it undetectable to the step segmentation algorithm at times. Because this isn't a normal step and in the context of this work, this step isn't really useful for analysis overall as it doesn't help identifying the pattern in the user's steps. The first 10 entries in the system for 'Unimpaired Gait' trials were manually verified to count the missed detections. These entries were collected while performing a covered distance of 5 meters. This was done by counting the spikes in the generated charts. for Results for the test are present in Table 5.1.

Test#	1^{st} **run**	2^{nd} **run**
1	1	0
2	0	0
3	0	1
4	1	0
5	0	0
6	0	1
7	2	1
8	1	1
9	0	1
10	0	0

Table 5.1: Results: Missed step detections for 'unimpaired gait'

Averaging the values for the 10*2 trials, we conclude that the average number of missed step detections for the 'unimpaired gait' is of 0.5 ± 0.6 steps. The same process was repeated for 'Impaired Gait' trials resulting in the Table 5.2.

Test#	1^{st} **run**	2^{nd} **run**
1	2	1
2	1	1
3	2	1
4	3	1
5	1	3
6	3	0
7	0	2
8	3	2
9	2	3
10	2	2

Table 5.2: Results: Missed step detections for 'impaired gait'

After inspecting the table and computing the average, we conclude that the average number of missed step detections for the 'Impaired Gait' is of 1.85 ± 0.99 steps.

When the first 'toClassify' user is processed, it sets a flag that will force the system to train the models. These models are trained using the obtained configuration for each of the ML models and training the machine learning models that will be used for classification. This array is sent to the MV classifier for prediction of which class this user's gait should belong to. The predicted labels for each of the 10 users set for classification can be seen in Table 5.3, alongside their respective prediction vote scores. The MV voting score goes from 0 to a maximum of 1, representing the number of votes that our implemented classifiers voted as positive or 'impaired gait' group.

	Real Label	Predicted Label	MV Vote Score
1	Unimpaired Gait	Unimpaired Gait	0
2	Unimpaired Gait	Unimpaired Gait	0
3	Unimpaired Gait	Unimpaired Gait	0
4	Unimpaired Gait	Unimpaired Gait	0
5	Unimpaired Gait	Unimpaired Gait	0
6	Impaired Gait	Impaired Gait	0.67
7	Impaired Gait	Unimpaired Gait	0.0
8	Impaired Gait	Impaired Gait	0.67
9	Impaired Gait	Impaired Gait	1.0
10	Impaired Gait	Impaired Gait	0.67

Table 5.3: Majority of votes predictions and predictions scores

In order to evaluate the resulting classifier performance, the confusion matrix was constructed as seen in Table 5.4.

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive 4	False Negative 1	P′
	n′	False Positive 0	True Negative 5	N′
	total	P	N	

Table 5.4: Majority of votes classifier confusion matrix

Then, the most important performance metrics for a binary classifier such as ours were calculated using the metrics presented in Subsection 2.3.3.3. These results are displayed in Table 5.5.

Accuracy	Precision	Recall	F1-Score
0.90	1.00	0.80	0.89

Table 5.5: Majority of votes classifier performance metrics

Overall, the majority of votes classifier obtained a very good accuracy of 90%, predicting correctly 9 out of the 10 labels. This will be reviewed in more detail in Subsection 5.2.1.

5.1.3 Experiment B: Simultaneous Cognitive Task Interference

For experiment B, a group of 6 voluntaries participated in dual-task trials. The average age of the population of study is of 26.33 ± 1.63 years. These users did not declare previous gait-related health conditions. Before the trials started, the users are tested on the standing cognitive test, performing the serial subtraction by 7 problem in the smartphone application. The first trial, was the single-task trial, in which the users walked at a self-selected comfortable speed for a 10 meters straight path. Then, on the dual-task trial, the users are yet again exposed to the simultaneous cognitive task of subtracting 7 in a serial way, which they should do while walking. The data was then extracted by trial and the average values for both the single-task trial and the dual-task trial are displayed and compared in order to perceive which motor features suffered significant alterations and which did not. How much a user was affected cognitively is also computed by using the Equation 2.1 and these results are displayed in the Table 5.6.

Table 5.6: Cognitive test results for experiment B

	Standing Score	Dual-Task Score	Cognitive DTC
Participant 1	1.0	0.5	50.0%
Participant 2	1.0	0.75	25.0%
Participant 3	1.0	1.0	0.0 %
Participant 4	1.0	1.0	0.0 %
Participant 5	1.0	1.0	0.0 %
Participant 6	0.6	0.5	16.67%
$\bar{X}$	0.93	0.79	15.28 %
σ	±0.16	±0.25	±20.01%

Analysis of the results for a participant with cDTC > 0% : We will start by analysing one of the users with cDTC > 0 %by comparing the generated charts for the single-task trial (left) and the dual-task trial (right). In Figure 5.6, we can see the step segmentation for both cases.

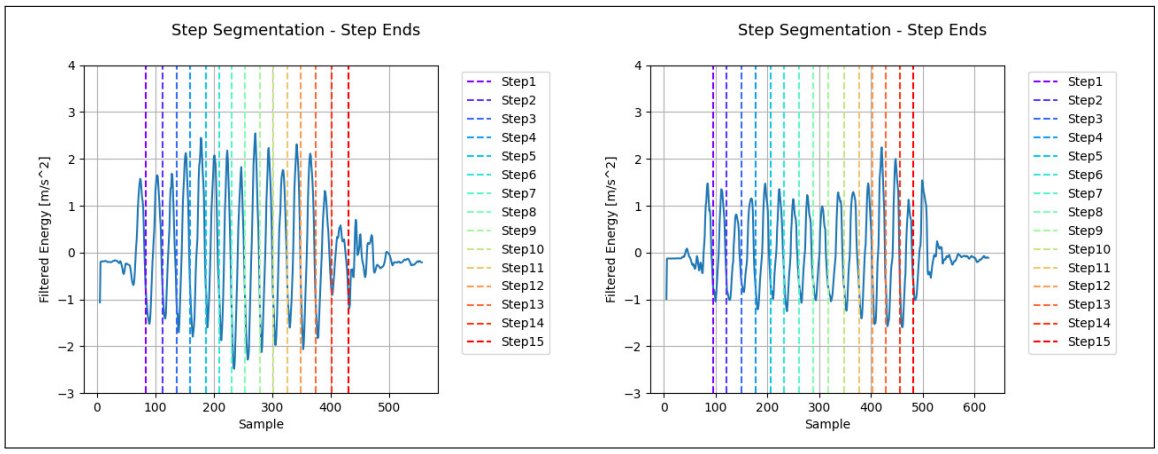

Figure 5.6: Step segmentation for the single-task (left) and dual-task trials (right)

The obtained stride length was of 1.33 meters for both trials. As for stride duration, the single-task obtained 1.48 seconds while it's counterpart attained 1.67 seconds. The equalized steps for each trial are depicted in Figure 5.7.

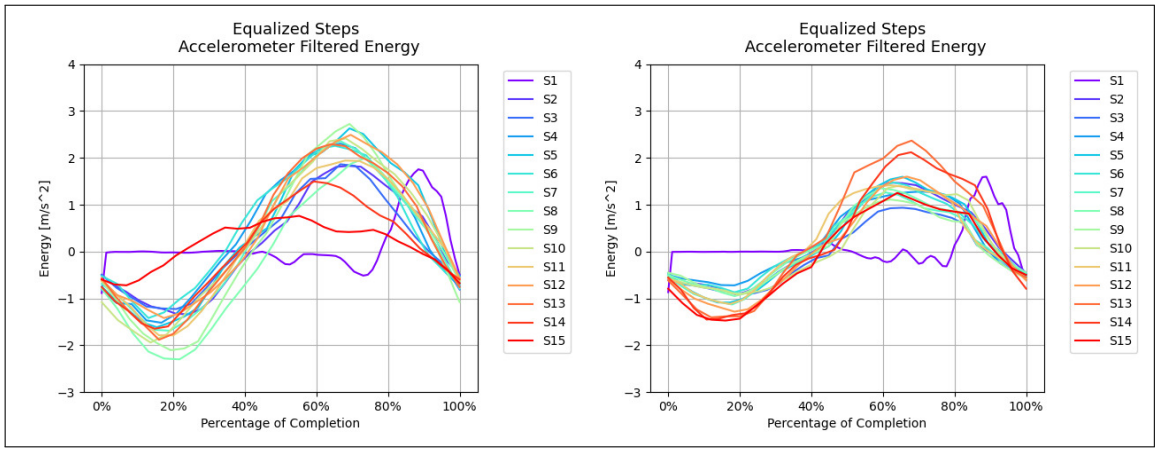

Figure 5.7: Equalized detected steps for the single-task (left) and dual-task trials (right

By computing the DTW between all the obtained steps, not accounting the first and last steps, we obtain values of 1.12 for the first case and 1.0 for the second trial, for the obtained distance metric between all the user's steps. In Figure 5.8, we can see the results for autocorrelation analysis in both cases.

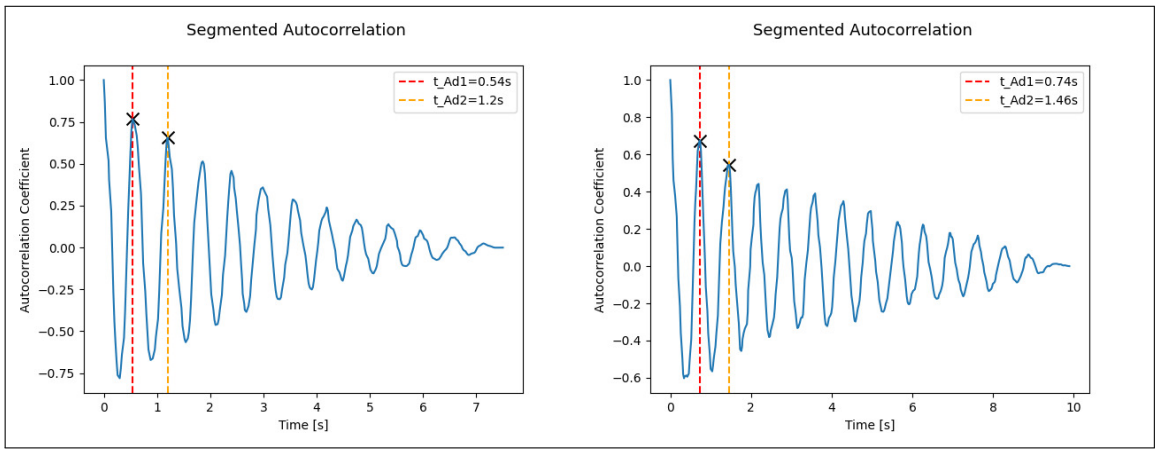

Figure 5.8: Autocorrelation analysis for the single-task (left) and dual-task trials (right)

When it comes to symmetry and regularity, for the step regularity and stride regularity, the single-task obtained 0.77 and 0.66 respectively. The dual-task trial attained values of 0.67 and 0.54 for the same features. The resulting computations for the step symmetry result in for the left case was of 1.17 and 1.24 for the right case. The results for cadence and walking speed for the single-task were of 111.11 steps/min and 1.23 m/s, respectively. For the trial with cognitive interference, we have obtained values of 81.08 steps/min and 0.9 m/s.

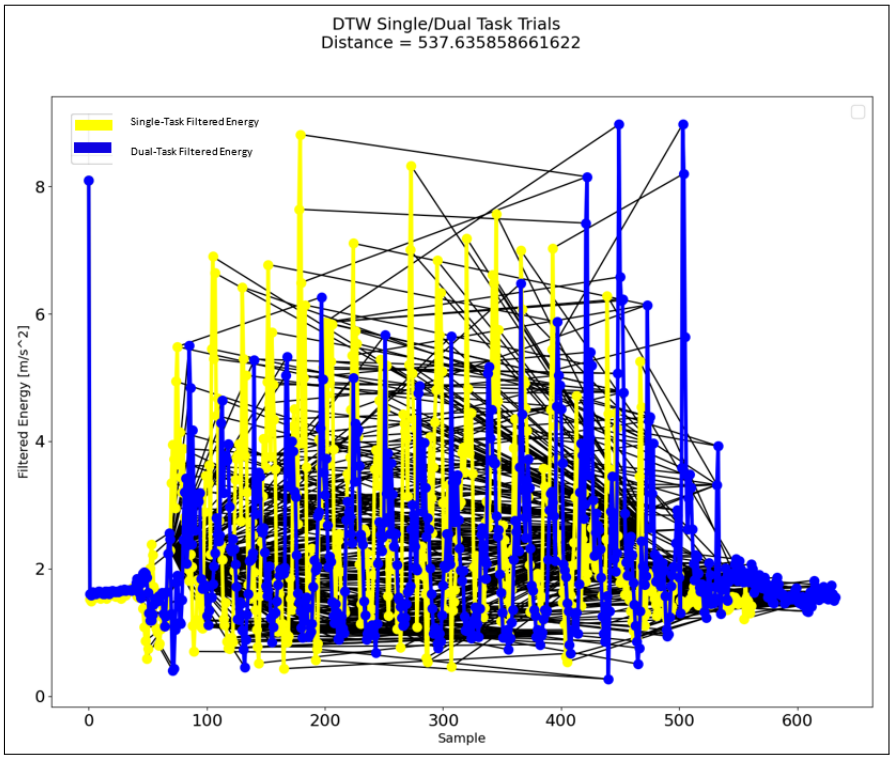

Figure 5.9: Obtained DTW distance between the single-task (left) and dual-task trials energies (right)

This participant obtained a DTW distance between the trials energies of 537.64.

Analysis of obtained averaged features for all participants: These features were separated into two tables seen in Tables 5.7 and 5.8.

Table 5.7: Averaged features results for experiment B - Part I

	Number of Steps		Mean Stride Time		Mean Stride Length		Walking Speed		Step Regularity	
	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ
Single-Task	15.67	±1.51	1.46	±0.2	1.28	±0.12	1.12	±0.07	0.77	±0.02
Dual-Task	16.83	±1.33	1.63	±0.37	1.19	±0.09	1.00	±0.08	0.72	±0.07

Table 5.8: Averaged features results for experiment B - Part II

	Stride Regularity		Step Symmetry		Cadence		Step Similarity		Trial Distance	
	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ
Single-Task	0.71	±0.03	1.08	±0.05	104.78	±7.25	1.13	±0.36	N/A	N/A
Dual-Task	0.65	±0.09	1.12	±0.09	101.03	±12.14	0.99	±0.31	464.77	±94.04

The mDTC for each extracted feature comparable between trials was then computed by using the Equation 2.2. These results may be seen in Table 5.9.

Table 5.9: Experiment B: obtained motor dual-task cost for each feature

Feature	Motor DTC
Number of Steps	-7.4 %
Mean Stride Time	-11.64 %
Mean Stride Length	7.03 %
Walking Speed	10.71 %
Step Regularity	6.49 %
Stride Regularity	8.45%
Step Symmetry	-3.7%
Cadence	3.58%
Step Similarity	12.39%

5.2 Discussion

In this Section we will be discussing the results obtained in Sections 5.1.2 and 5.1.3 in Sections 5.2.1 and 5.2.2 respectively.

5.2.1 Experiment A

By inspecting the obtained values for the example run presented in experiment A, we can start by noting that unlike what was assumed, the impaired users did not take smaller steps and consequently smaller strides according to the obtained data given that the 'Impaired Gait' user obtained 1.25 meters for the stride length while the 'Unimpaired Group' obtained an average of 1.11 meters. We don't think this is legitimate but rather the effects of the missed detections that end up biasing the results for stride length. With regard to the stride time, the impaired user had an increase of 23.08% when compared to the counterpart. This is expected because by being impaired the user is forced to take more time executing the steps to avoid any falls. When it comes to step similarity, the impaired case had their combination of steps with a smaller DTW distance then the unimpaired case, registering values of 0.82 and 1.33, respectively.

The obtained values for cadence and walking speed were both higher for the unimpaired case which although wasn't expected could be realistic since this set of tests was more subjective in the sense that multiple speeds and patterns were tested out and we could be dealing with a case when this is true. Analysing the symmetry and regularity of these cases, we can see that for step regularity, the impaired case was a lot more irregular in their steps with a decrease of approximately 23.08%, when comparing with the unimpaired case, which makes total sense in the current context. For the stride regularity, the impaired group also presented lesser regularity with a decrease of 18.75% when comparing to the counterpart. We think that the obtained values for the step symmetry don't reflect the truth because it's value is directly obtained from the extracted step and stride regularity. Because it is computed by the quotient of the step and stride regularities, the unimpaired case ended up having higher values for this metric, which by definition, means they had a less symmetrical gait given that perfect symmetry is represented by values close to 1.

As figured in Table 5.1, the average number of missed for the 'Unimpaired Gait' class is fairly low with an average of 0.5 missed steps, among the reviewed trials. On the other end, the average number of missed for 'impaired gait' was higher than initially expected, with an average of 1.85 missed steps which can be inspected in Table 5.2. This means that the system will most likely miss, at least, three times more steps while detecting an 'impaired gait' when compared to the counterpart. We haven't observed overestimation of steps but rather an underestimation of it. We believe the cause to this may be the relatively low cadence that was used during the 5 meters trials given the limited space we had available to conduct this experiment, in "laboratory-like" conditions at the time this experiment was ran. Walking with an impairment, in this case simulated by a boot, also constraints the movement. The foot shuffling caused by the impairment could be leading to non-detected steps, making the system perceive it as a longer step instead of two different steps. This will also affect other measures that depend directly on it such as the stride length and stride time.

As previously presented in Table 5.5, our classifier achieved a precision score of 1.0 indicating that it does not produce false positive classifications. This property is important as we want to avoid misdiagnosing someone as positive when that is not the case. On the other hand, our system attained a recall score of 0.80 revealing a tendency to provide false negative recommendations. Ideally, in this clinical context, we would rather have higher recall over precision, as false negatives may have larger consequences. The F1 score indicates good overall performance obtaining a value of 0.89. This reveals balance between the classifier's precision and recall. Even though the results for the classification module demonstrated good performance, the models might be overfitted given our small data set size and lack of participant variability.

5.2.2 Experiment B

From Table 5.6, we can observe the obtained cDTC values, in percentage, for each user. The average score of standing cognitive test was of 0.93 ± 0.16, which represents the users maximum focus in this context. The average ratio of the dual-task cognitive performance was of 0.79 ± 0.25, which reveals that the users suffered from the interference of the superposition of the walking task, when compared to the standing score. The average for the total cDTC, in percentage, for all users was of 15.28 ± 0.25%, showing that on average the users have lower ability to dual-task when compared to the single-task performances.

The results of analysis for the example participant with cDTC > 0% showcased that the dual-task trial filtered energy has lower averages on average when comparing to the the single-task trial. The stride length remained constant while the stride duration increased to 1.67 seconds from 1.48 seconds representing an increase of 12.84% when compared to the single-task trial stride duration. This is in accordance with the revised literature that refers a decrease of around 11 % on the stride time while dual-tasking.

The obtained DTW distance between the selected steps decreased by 10.71% in the

second trial meaning that the steps have gotten more similar. This should be related with the fact the strides were slower and therefore had a more repetitive pattern.

Regularity and symmetry wise, we expected that the values of these metrics would reveal a lesser symmetrical and rhythmic pattern because the user's attention is divided between both tasks. This was confirmed for both cases, since the step regularity dropped from 0.77 to 0.67 and the stride regularity lowered from 0.66 to 0.54. The resulting step symmetry values showcased an increase from 1.17 to 1.24, which is interpreted as becoming more asymmetrical between right and left foot. This is because according to the reviewed literature, the farthest from the value of 1.0, the less symmetrical.

The cadence revealed a decrease of 27.02% from the value obtained in the single-task trial. The walking speed followed this pattern with a decrease of 26.83%, revealing that the user decreased the speed and cadence in order to perform both tasks.

This participant also obtained a DTW between trials energies of 537.64 which is a very high value the usual range of values users have obtained. This might reflect the difficulties that the user experienced while performing both tasks at the same time.

From Tables 5.7 and 5.8, we can gather some conclusions regarding the obtained averaged values for the experiment with the values reviewed. To evaluate the system's ability to compute these metrics, we start by comparing the single-task trials with some reference values from the reviewed literature. The average values for the mean stride time were a little higher then the expected of 1.05 seconds[14]. The mean stride length returned values in the expected range of 1.20 meters[38]. The walking speed of 1.12 m/s fits perfectly with our reference value of 1.15 m/s[38]. The observed cadence fitted perfectly with the reference value of 105 steps per min [51].

For step similarity, we have obtained an average of 1.13 for the metric that represents the distance between the detected steps which is out of the expected range given our reference value was of 0.17[15] for a normal gait pattern. When it comes to regularity and symmetry, the obtained values for the step regularity and stride regularity were of 0.77 and 0.71, respectively. The value for the step regularity is in accordance with the ranges of our values which are between 0.74[15] to 0.89[16]. For the stride regularity, the obtained value was a little lower then expected, obtaining 0.71 when our reference values are of 0.75[15] and 0.91[16]. For the step symmetry we were expecting values around 0.98[16] (for "perfect gait") and we have obtained 1.08, which should be due to the fact the computation of this metric depends directly on the value of stride regularity which was slightly lower then expected, increasing the value for this metric.

From Table 5.9, we can see the values, in percentage, for the mDTC for each feature. These values represent the cost of the secondary task over the primary task, walking. It's visible that the most affected features were the step similarity (+12.39%), the mean stride time (−11.64%) and the walking speed (+10.71%). This reveals that the users reduced their walking speed which consistent with having strides that take more time to execute. We suspect that the fact that the steps have become more similar while DT conditions as to do with the fact that they were walking slower in a repetitive way so that they could

have more time to think about the cognitive test.

On average, the participants appear to take more steps that were smaller in length. Cadence presented minimal change (+3.58%). When it comes to the study of symmetry and regularity, users seem to be more irregular in both their step (+6.49%) and stride (+8.45%) regularities while dual-tasking, which makes sense given that their attention is divided. The values for step symmetry also presented a relatively low variation (−3.7%). In the context of symmetry gait analysis, this means that the participants got less symmetrical while under the DT paradigm as the closeness to 1.0 represents perfect symmetry between right and left feet, revealing that participants become not only less regular but also less synchronized while under DT conditions.

As it becomes more difficult to keep a steady pattern while simultaneous performing another task we also expected that the variability of the gait parameters, in this case represented by the standard deviation, would increase while dual-tasking, which is in agreement with the data. Except for the number of steps, the mean stride length and the step similarity, all other DT features presented higher variability then their ST counterparts.

It would be interesting to investigate which dual-task trial features vary with the change of cDTC. To do that, we studied the correlation between each of these features and the percentage of cDTC each participant experienced. To assess the statistical significant of the relationships between these features, the p-value was computed. These computations, in Figure 5.10 were made using Pearson correlation.

Table 5.10: Experiment B: correlation and p-value between cognitive dual-task cost and motor features

Feature	Correlation Coefficient	p-value
Number of Steps	0.43	0.40
Mean Stride Time	0.06	0.91
Mean Stride Length	-0.39	0.44
Walking Speed	-0.34	0.51
Step Regularity	0.14	0.79
Stride Regularity	0.09	0.87
Cadence	0.05	0.92
Gait Symmetry	0.03	0.95
Step Similarity	-0.12	0.82
Trial Distance	0.86	0.03

Finally, in order to find evidence that our theory that the DTW between the single-task trial and dual-task trial energy signals, computed from the accelerometer signals, is directly related to how cognitively interfered the person was from the performance of the simultaneous secondary task, we started by inspecting the obtained average values of trial distance. We detected a large discrepancy in the average values of trial distance for the users with 0 % cDTC, which were participants 3,4 and 5, and the remaining participants who obtained values of cDTC surpassing 0%. Upon investigation, we found that the resulting averages for each these two groups were of 394.02 ± 51.19 for the cDTC equal to 0

case and of 535.51 ± 66.90 for the second case. This represents an increase in the computed distance of the trials of 35.91% between the two groups, which is very significant.

Then we decided to study if there was actual correlation between these two variables as seen in Figure 5.10.

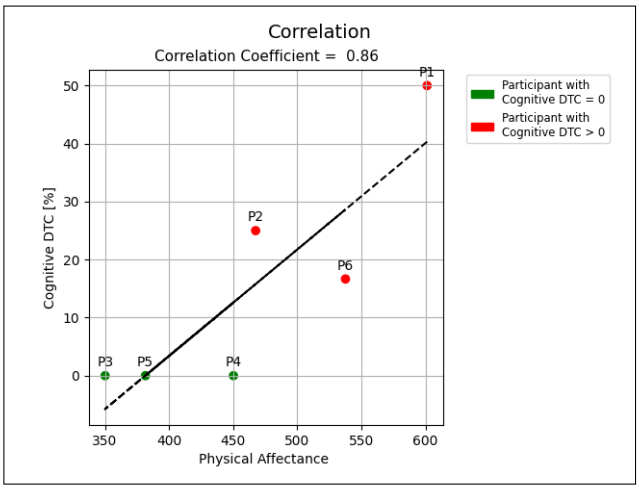

Figure 5.10: Experiment B: correlation between trial distance and cDTC

We obtained 0.86 for the Pearson correlation coefficient between the variables indicating a positive and strong correlation between these. This means there might be a solid linear relationship between these variables. The obtained p-value for this relationship was of 0.03. These results suggest that trial distance, or how different the two trials were, could be a good way to predict the cDTC and vice-versa. A linear regression was made from the participants points that is given by the Equation 5.1.

$$cDTC(TrialDistance) = 0.18 * TrialDistance - 69.99 \tag{5.1}$$

Could be a very useful way to predict the percentage of cDTC without a need to score the cognitive tests and make computations for it. Even though there is high correlation and the p-value is below the conventional significance level of $\alpha = 0.05$, further investigation would be necessary to verify it.

6

Conclusions and Future Work

As we approach the conclusion of this thesis, we revisit the crucial takeaways and drawbacks of our work in Section 6.1. Then, in Section 6.2, we discuss practical advice to future researchers, potential enhancements to our system and possible directions for future research.

6.1 Conclusions

We set out to address the difficult process of detecting gait alterations in patients possibly suffering from muscoskeletal or neurological diseases. Existing solutions require expensive equipment and experienced clinicians, and thus our goal was to develop a framework which would drastically democratize this process, while offering decision support to experts. We hypothesized that modern smartphones in particular, due to their affordability, ubiquity and array of typical sensors, would be a solid platform for the development of such a framework.

Our literature survey revealed that dual-task is a promising gait alteration detection method, which would also translate well to a smartphone setting and thus we chose to implement it to demonstrate our system. Our system consists of a *smartphone application* for raw data collection, a *desktop application* for data processing, classification and analysis and a *real-time database* to connect the previous two. The desktop application extracts important gait features, such as stride time and length or step similarity and symmetry, from the raw data, visualizes them in a helpful dashboard and additionally presents a diagnosis suggestion created using machine learning classifiers trained on the aforementioned features. Through an evaluation of the system, we found it to work successfully as a data collection tool, and to be quite usable and effective at detecting simulated gait alterations. Despite the many external factors that could affect measurements, the signal processing algorithms we implemented were able to successfully extract important features from the data. For impairment detection, step regularity and stride regularity were the most reliable features at differentiating between impaired and unimpaired users, as they presented very distinct ranges of values. This may be because these two metrics do not rely on step

detection as an input, and thus do not accumulate errors from missed detections (see Tables 5.1 & 5.2) , such as is the case with stride length. We found that the features with the highest mDTC in the serial subtraction task were step similarity, mean stride time and walking speed, meaning that if we wish to quantify the motor interference, caused by the simultaneous cognitive task, these should be used Our feature trial distance, needs more investigation before strong statements, but has potential to be very useful at predicting cDTC.

In the process of development, we replicated several prior works on gait features and presented a small-scale reproduction study of the effectiveness of each feature. Trial Distance, a conceptually simple feature that we proposed based on DTW between single- and dual-task trials energy signals, was found to have the highest correlation (0.86) with cDTC, with a p-value of 0.03.

With *dualgAIt*, we have shown that it is possible to greatly increase the accessibility and lower the cost of clinical diagnosis through the use of a device that most people have access to. Additionally, our replications provide important insight into which gait features are most effective when working with the sensors offered by most smartphones. The high correlation of Trial Distance could represent an easy way to predict a user's cognitive DTC without the need to compute multiple gait features. This system is also useful for assessing gait changes over time, which is especially useful in cases such as rehabilitation and physical therapy. This system could allow a clinician to run different experiments in a fast centralized way without the need for external integration or importing the data manually.

On the other hand, it is important to consider the limitations of both the developed system, as well as, the studies we conducted. While the system meets our requirements, it could be improved by providing the dual-task cognitive test instructions directly from the smartphone device. This would avoid the need for the clinician to handle a script and deliver it effectively. Similarly, the patient's responses would ideally be automatically recorded by the application, removing the clinician from this step of the process, and thus reducing the amount of subjectivity and interpretation in the data collection. Additionally, the application was was limited to Android smartphones, which leaves out a significant portion of the market share of users which use Apple devices. Regarding the studies conducted and conclusions drawn, these must be taken with a grain of salt, due to the small sample size used and the lack of diversity in tested smartphone models. Finally, no neurologically or muscoskeletally affected individuals were tested in our case studies. Instead, such alterations were simulated through the use of a boot on a single foot, inducing an imbalance in the gait. Sourcing a larger, more representative set of individuals was unfortunately difficult to arrange for logistical reasons and within timeframe of this dissertation.

6.2 Future Work

We highlight three kinds of future work. First, we discuss important enhancements that could be applied to our system, *dualgAIt*. Then, we discuss practical recommendations and tips for future researchers in the field. Finally, we finish by proposing future research directions that became apparent throughout the development of this thesis.

6.2.1 Enhancements & Improvements

Our framework could be enhanced in several ways which we shall discuss. First, in a clinical setting, false negatives are more critical than false positives. Thus, classifiers should be tuned to trade-off some precision for a bit more recall. Security is important when handling patient data. A possible solution would be handing the clinician responsible for the trials the option of running the experiments in local mode only directly within the system. The system should also be made parameterizable by the clinician, allowing it to be better tuned for different diseases, room conditions (e.g. size) or patient characteristics (e.g. gender or weight) and balancing feature relevance for classifiers purposes. These enhancements are critical in order for these approaches to graduate from academia to real-world clinical and commercial settings. Finally, larger scale experiments must be undertaken to truly validate the system.

6.2.2 Recommendations

Throughout the development of this system we learned some practical lessons which we now offer to future researchers. In low-end smartphones, a hardware gyroscope does not typically exist, and instead a virtual one is implemented through *sensor fusion*, by combining other sensors. This is done to save costs, as the general use-case is to simply support screen rotation, but leads to bad performance and resolution. Thus, one should not rely on this sensor for the extraction of gait features. In our work, we use it only to detect the end of the trial, when a user turns. We also found simple classifiers, namely Decision Tree, to be more explainable, as one can directly inspect the decisions taken at each point in the tree toward a final outcome. This is an important factor when communicating with patients. Thus, we recommend focusing on improving the accuracy of this classifier over exploring others. Finally, the serial subtraction task does not offer proper accountability to users who simply answer more times within a certain time period. What this means is that one can "trick" the task by saying many numbers. A factor could be designed to balance the different number of answers during the dual-task trial. Alternatively, standardizing a different secondary activity for the dual-task setting trials or running multiple types of cognitive tests with the same user in order to compare the different affections may yield better results.

6.2.3 Future Research Directions

Finally, we suggest a few future research directions. The detection of gait cycles is easiest for unaffected individuals, as they have a periodic cycle. Methods to better detect impaired gait cycles are needed as these deviate from the normal template. Furthermore, given that we have modules for isolating and comparing all the equalized steps, it would be interesting to explore template-based approaches. These steps could be used to build a step template for each user that could be used for multiple purposes. A potential direction is to adjust the detection sliding window for each user's cadence. To achieve this, an initial trial could be ran before the experiments to detect the user's cadence and adapt the sliding window to it. Some work was produced towards automating the dual-task trials, yet unfinished. Better techniques for the preparation of the voice signal (e.g. noise-reduction and audibility checks) should be investigated. More information could be extracted from the voice signal and be used as features for gait classification on a dual-trial approach.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[2] *WHO - What are neurological disorders*. https://www.who.int/news-room/q-a-detail/what-are-neurological-disorders. Accessed: 2020-11-30 (cit. on p. 1).

[3] P. M. Brooks. "The burden of musculoskeletal disease—a global perspective". In: *Clinical rheumatology* 25 (2006), pp. 778–781 (cit. on p. 1).

[4] U. Ahmed et al. "Biomarkers of early stage osteoarthritis, rheumatoid arthritis and musculoskeletal health". In: *Scientific reports* 5.1 (2015), pp. 1–7 (cit. on p. 1).

[5] T. T. Htike et al. "Peripheral biomarkers for early detection of Alzheimer's and Parkinson's diseases". In: *Molecular neurobiology* 56 (2019), pp. 2256–2277 (cit. on p. 1).

[6] Y. Moon et al. "Gait variability in people with neurological disorders: A systematic review and meta-analysis". In: *Human movement science* 47 (2016), pp. 197–208 (cit. on p. 1).

[7] E. Rovini et al. "Wearable Sensors for Prodromal Motor Assessment of Parkinson's Disease using Supervised Learning". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2019), pp. 4318–4321. ISSN: 1557170X. DOI: 10.1109/EMBC.2019.8856804 (cit. on pp. 1, 16, 27–29).

[8] E. Stone and M. Skubic. "Evaluation of an inexpensive depth camera for in-home gait assessment". In: *JAISE* 3 (2011-01), pp. 349–361. DOI: 10.3233/AIS-2011-0124 (cit. on pp. 1, 9).

[9] D. I. Stoia and M. Toth-Tascau. "Comparison of treadmill-based and overground gait analysis". In: *IFMBE Proceedings* 36 (2011), pp. 368–371. ISSN: 16800737. DOI: 10.1007/978-3-642-22586-4_77 (cit. on pp. 1, 9).

[10] C. 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 (cit. on pp. 1, 28).

[11] C Goetz *et al*. "Movement Disorder Society-Sponsored Revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results". In: *Movement Disorders* 23.15 (2008), pp. 2129–2170. ISSN: 08853185. DOI: 10.1002/mds.22340 (cit. on pp. 1, 7).

[12] E. Warmerdam et al. "Long-term unsupervised mobility assessment in movement disorders". In: *The Lancet Neurology* 19 (2020), pp. 462–470. ISSN: 14744465. DOI: 10.1016/S1474-4422(19)30397-7. URL: http://dx.doi.org/10.1016/S1474-4422(19)30397-7 (cit. on pp. 1, 9, 10).

[13] M. Grossi. "A sensor-centric survey on the development of smartphone measurement and sensing systems". In: *Measurement* 135 (2019), pp. 572–592 (cit. on p. 2).

[14] B. Manor et al. "Smartphone App–Based Assessment of Gait During Normal and Dual-Task Walking: Demonstration of Validity and Reliability". In: *JMIR Mhealth Uhealth* 6.1 (2018-01), e36. ISSN: 2291-5222. DOI: 10.2196/mhealth.8815. URL: http://www.ncbi.nlm.nih.gov/pubmed/29382625 (cit. on pp. 2, 11, 15, 18, 20, 49, 50, 85).

[15] A. A. Perez and M. A. Labrador. "A Smartphone-Based System for Clinical Gait Assessment". In: *2016 IEEE International Conference on Smart Computing, SMARTCOMP 2016* (2016). DOI: 10.1109/SMARTCOMP.2016.7501675 (cit. on pp. 2, 6–8, 16–21, 23, 24, 51, 52, 54–56, 58, 59, 85).

[16] R. Moe-Nilssen and J. L. Helbostad. "Estimation of gait cycle characteristics by trunk accelerometry". In: *Journal of Biomechanics* 37.1 (2004), pp. 121–126. ISSN: 00219290. DOI: 10.1016/S0021-9290(03)00233-1 (cit. on pp. 2, 8, 21, 59, 85).

[17] A. Leland et al. "The Role of Dual Tasking in the Assessment of Gait, Cognition and Community Reintegration of Veterans with Mild Traumatic Brain Injury". In: *Materia Socio Medica* 29 (2017-12), p. 251. DOI: 10.5455/msm.2017.29.251-256 (cit. on pp. 2, 11).

[18] A. Burkov. *The Hundred-Page Machine Learning Book*. 2019 (cit. on pp. 2, 26, 27).

[19] V. L. Feigin et al. "Global, regional, and national burden of neurological disorders during 1990–2015: a systematic analysis for the Global Burden of Disease Study 2015". In: *The Lancet Neurology* 16.11 (2017), pp. 877–897 (cit. on p. 2).

[20] E. Dorsey et al. "The emerging evidence of the Parkinson pandemic". In: *Journal of Parkinson's disease* 8.s1 (2018), S3–S8 (cit. on p. 2).

[21] D. Arnott and G. Pervan. "A Critical Analysis of Decision Support Systems Research Revisited: The Rise of Design Science". In: *Journal of Information Technology* 29 (2014-12). DOI: 10.1057/jit.2014.16 (cit. on p. 5).

[22] C. Brandaş. "Contributions to Conception , Design and Development of Decision Support Systems Phd Candidate :" in: June (2007) (cit. on pp. 5, 6).

[23] M. W. L. Moreira et al. "A Comprehensive Review on Smart Decision Support Systems for Health Care". In: *IEEE Systems Journal* 13.3 (2019), pp. 3536–3545. DOI: 10.1109/JSYST.2018.2890121 (cit. on p. 5).

[24] E. B. Sloane and R. J. Silva. "Chapter 83 - Artificial intelligence in medical devices and clinical decision support systems". In: (2020). Ed. by E. Iadanza, pp. 556–568. DOI: https://doi.org/10.1016/B978-0-12-813467-2.00084-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128134672000845 (cit. on p. 6).

[25] J. C. Schlachetzki et al. "Wearable sensors objectively measure gait parameters in Parkinson's disease". In: *PLoS ONE* 12.10 (2017), pp. 1–18. ISSN: 19326203. DOI: 10.1371/journal.pone.0183989 (cit. on pp. 7, 9, 17, 25).

[26] M. M. Hoehn and M. D. Yahr. "Parkinsonism : onset , progression , and mortality". In: 17.May (1967). DOI: 10.1212/WNL.17.5.427 (cit. on p. 7).

[27] W. Pirker and R. Katzenschlager. "Gait disorders in adults and the elderly: A clinical guide". In: *Wiener klinische Wochenschrift* 129 (2016-10). DOI: 10.1007/s00508-016-1096-4 (cit. on p. 8).

[28] W. Tao et al. "Gait analysis using wearable sensors". In: *Sensors* 12.2 (2012), pp. 2255–2283. ISSN: 14248220. DOI: 10.3390/s120202255 (cit. on p. 9).

[29] M. O. Derawi et al. "Unobtrusive user-authentication on mobile phones using biometric gait recognition". In: *Proceedings - 2010 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2010* (2010), pp. 306–311. DOI: 10.1109/IIHMSP.2010.83 (cit. on pp. 9, 23, 24, 58).

[30] M. Gabel et al. "Full body gait analysis with Kinect". In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (2012), pp. 1964–1967. ISSN: 1557170X. DOI: 10.1109/EMBC.2012.6346340 (cit. on p. 9).

[31] S. Nishiguchi et al. "Reliability and validity of gait analysis by android-based smartphone". In: *Telemedicine and e-Health* 18.4 (2012), pp. 292–296. ISSN: 15305627. DOI: 10.1089/tmj.2011.0132 (cit. on pp. 9, 12).

[32] D. Gouwanda and S. M. Arosha Senanayake. "Identifying gait asymmetry using gyroscopes-A cross-correlation and Normalized Symmetry Index approach". In: *Journal of Biomechanics* 44.5 (2011), pp. 972–978. ISSN: 00219290. DOI: 10.1016/j.jbiomech.2010.12.013. URL: http://dx.doi.org/10.1016/j.jbiomech.2010.12.013 (cit. on pp. 9, 21).

[33] T. B. De Freitas et al. "The effects of dual task gait and balance training in Parkinson's disease: a systematic review". In: *Physiotherapy Theory and Practice* 36.10 (2020), pp. 1088–1096. ISSN: 15325040. DOI: 10.1080/09593985.2018.1551455. URL: https://doi.org/10.1080/09593985.2018.1551455 (cit. on p. 10).

[34] B. Postigo-Alonso et al. "Cognitive-motor interference during gait in patients with Multiple Sclerosis: a mixed methods Systematic Review". In: *Neuroscience & Biobehavioral Reviews* 94 (2018-09), pp. 126–148. DOI: 10.1016/j.neubiorev.2018.08.016 (cit. on p. 11).

[35] C. Leone et al. "Comparing 16 Different Dualâ€"Tasking Paradigms in Individuals With Multiple Sclerosis and Healthy Controls: Working Memory Tasks Indicate Cognitiveâ€"Motor Interference". In: *Frontiers in Neurology* 11 (2020). ISSN: 1664-2295. DOI: 10.3389/fneur.2020.00918. URL: https://www.frontiersin.org/articles/10.3389/fneur.2020.00918 (cit. on p. 11).

[36] *Kooistra, J. Newzoo's 2018 Global Mobile Market Report: Insights into the World's 3 Billion Smartphone Users. Newzoo. 11 September 2018.* https://newzoo.com/insights/articles/newzoos-2018-global-mobile-market-report-insights-into-the-worlds-3-billion-smartphone-users/. Accessed: 2020-11-30 (cit. on p. 12).

[37] L. Brognara et al. "Assessing Gait in Parkinson's Disease Using Wearable Motion Sensors: A Systematic Review". In: *Diseases* 7.1 (2019), p. 18. ISSN: 2079-9721. DOI: 10.3390/diseases7010018 (cit. on pp. 12, 13, 24, 25).

[38] R. Bouça-Machado et al. "Gait Kinematic Parameters in Parkinson's Disease: A Systematic Review". In: *Journal of Parkinson's Disease* 10.3 (2020), pp. 843–853. ISSN: 1877718X. DOI: 10.3233/JPD-201969 (cit. on pp. 12–14, 24, 25, 85).

[39] S. Majumder and M. J. Deen. "Smartphone sensors for health monitoring and diagnosis". In: *Sensors (Switzerland)* 19.9 (2019), pp. 1–45. ISSN: 14248220. DOI: 10.3390/s19092164 (cit. on p. 12).

[40] Q. Zou et al. "Deep Learning-Based Gait Recognition Using Smartphones in the Wild". In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3197–3212. DOI: 10.1109/TIFS.2020.2985628 (cit. on p. 13).

[41] M. Gonzalez Rodriguez et al. "Multi sensor system for pedestrian tracking and activity recognition in indoor environments". In: *International Journal of Ad Hoc and Ubiquitous Computing* 23.1/2 (2016), p. 3. ISSN: 1743-8225. DOI: 10.1504/ijahuc.2016.10000202 (cit. on pp. 13, 49).

[42] C. Höchsmann et al. "Validity of activity trackers, smartphones, and phone applications to measure steps in various walking conditions". In: *Scandinavian Journal of Medicine and Science in Sports* 28 (2018-07), pp. 1818–1827. DOI: 10.1111/sms.13074 (cit. on p. 14).

[43] F. Proessl et al. "Good agreement between smart device and inertial sensor-based gait parameters during a 6-min walk". In: *Gait and Posture* 64.March (2018), pp. 63–67. ISSN: 18792219. DOI: 10.1016/j.gaitpost.2018.05.030 (cit. on pp. 14, 25).

[44] M. Furrer et al. "Validation of a smartphone-based measurement tool for the quantification of level walking". In: *Gait and Posture* 42.3 (2015), pp. 289–294. ISSN: 18792219. DOI: 10.1016/j.gaitpost.2015.06.003. URL: http://dx.doi.org/10.1016/j.gaitpost.2015.06.003 (cit. on p. 14).

[45] P. Silsupadol, K. Teja, and V. Lugade. "Reliability and validity of a smartphone-based assessment of gait parameters across walking speed and smartphone locations: Body, bag, belt, hand, and pocket". In: *Gait and Posture* 58.September (2017), pp. 516–522. ISSN: 18792219. DOI: 10.1016/j.gaitpost.2017.09.030. URL: http://dx.doi.org/10.1016/j.gaitpost.2017.09.030 (cit. on pp. 15, 16, 25).

[46] C. Nickel. "Accelerometer-based Biometric Gait Recognition for Authentication on Smartphones". In: (2012) (cit. on pp. 15–17, 25, 26, 28, 29).

[47] S. W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. USA: California Technical Publishing, 1997. ISBN: 0966017633 (cit. on pp. 16, 17).

[48] J. J. Marron et al. "Multi sensor system for pedestrian tracking and activity recognition in indoor environments". In: *Int. J. Ad Hoc Ubiquitous Comput.* 23 (2016), pp. 3–23. URL: https://api.semanticscholar.org/CorpusID:430281 (cit. on pp. 16, 51).

[49] J. Winkler, J. Klucken, and B. Eskofier. "Subsequence dynamic time warping as a method for robust step segmentation using gyroscope signals of daily life activities". In: (2013), pp. 6744–6747. DOI: 10.1109/EMBC.2013.6611104 (cit. on p. 17).

[50] J. J. Marron et al. "Multi sensor system for pedestrian tracking and activity recognition in indoor environments". In: *Int. J. Ad Hoc Ubiquitous Comput.* 23 (2016), pp. 3–23. URL: https://api.semanticscholar.org/CorpusID:430281 (cit. on p. 17).

[51] C. C. Yang et al. "Real-time gait cycle parameter recognition using a wearable accelerometry system". In: *Sensors* 11.8 (2011), pp. 7314–7326. ISSN: 14248220. DOI: 10.3390/s110807314 (cit. on pp. 20–22, 85).

[52] S. Majumder, T. Mondal, and M. J. Deen. "Wearable sensors for remote health monitoring". In: *Sensors (Switzerland)* 17.1 (2017). ISSN: 14248220. DOI: 10.3390/s17010130 (cit. on p. 24).

[53] S. Arora et al. "High accuracy discrimination of Parkinson's disease participants from healthy controls using smartphones". In: (2014), pp. 3641–3644. DOI: 10.1109/ICASSP.2014.6854280 (cit. on pp. 25, 28, 29).

[54] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597 (cit. on pp. 26–28).

[55] N. Ş. Köktaş and R. P. Duin. "Statistical analysis of gait data to assist clinical decision making". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5853 LNCS (2010), pp. 61–68. ISSN: 03029743. DOI: 10.1007/978-3-642-11769-5_6 (cit. on p. 27).

[56] A. Tharwat et al. "Linear discriminant analysis: A detailed tutorial". In: *AI Communications* 30.2 (2017), pp. 169–190. ISSN: 09217126. DOI: 10.3233/AIC-170729 (cit. on p. 28).

[57] T. Bristow et al. "Standardization and adult norms for the sequential subtracting tasks of serial 3's and 7's". In: *Applied Neuropsychology: Adult* 23 (2016-05), pp. 1–9. DOI: 10.1080/23279095.2016.1179504 (cit. on pp. 73, 74).

I

Participation Form

Gait Dual-Task Study - Participation Form

This study will be conducted using a project that was developed by Ricardo F. Silvestre for the partial fulfillment of this dissertation in the Integrated Master's Program in Electrical and Computer Engineering, under the supervision of Professor Fernando Ferreira – **AI-ENABLED ASSESSMENT OF GAIT PATTERNS VIA SMARTPHONE**. The system, named dualgAIt, will retrieve data from the smartphone during two trials: a single-task trial and a dual-task trial. We are looking for volunteers to test the system. This data will be used to study the interference of a cognitive task on each subject's gait pattern.

Procedures: A phone will be attached to your lower back. During the single-task trial you will be asked to walk for approximately 10 meters and turn left when you reach the finish line. During the dual-task trial you will be asked to repeat the same procedure while performing serial subtraction by 7 on a given starting number. This trial will also generate a voice recording that contains your answers.

Possible Risks: Walking always carries the risk of falling. The dual-task trial could further reduce your attention, increasing the risk of falling. Please walk at a comfortable and safe speed during the trials.

Collected Data: The application will collect accelerometer and gyroscope data during the trials. It will also collect general information in the form of a survey. The system won't collect the user's name, address or any Identification numbers and the collected data won't be used to identify the subject in any way. The system will also run a cognitive test in the form of serial subtraction by 7 given an initial number. The voice recording will also be stored for analysis. All the data will be sent to a database and will be retrieved for analysis. The average values of the obtained spacio-metrical features may also be used to train machine learning models.

Even after the trials begin you are free to stop at any time.

Please mark (✓) the options you agree with:

□ I agree to voluntarily participate in this study.

□ I've read this document and understand its contents.

□ I accept that my generated data may be used for study, use and publication of future academic work.

_____	_____	____/____ /_____
Participant's Signature	Research Student	Date

II

Image License Information

https://creativecommons.org/licenses/by/4.0/

- Document Figure 2.2 by J. C. Schlachetzki et al.

  (https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0183989)

Citation: Schlachetzki JCM, Barth J, Marxreiter F, Gossler J, Kohl Z, Reinfelder S, et al. (2017) Wearable sensors objectively measure gait parameters in Parkinson's disease. PLoS ONE 12(10): e0183989. https://doi.org/10.1371/journal.pone.0183989

Editor: Mathias Toft, Oslo Universitetssykehus, NORWAY

Received: February 20, 2017; **Accepted:** August 15, 2017; **Published:** October 11, 2017

- Document Figure 2.3 by W. Pierker and R. Katzenschlager (https://www.researchgate.net/publication/309362425_Gait_disorders_in_adults_and_the_elderly_A_clinical_guide)

Gait disorders in adults and the elderly: A clinical guide

October 2016 · Wiener klinische Wochenschrift 129(3)
DOI:10.1007/s00508-016-1096-4
License · CC BY 4.0

Authors:

Walter Pirker
Klinik Ottakring

Regina Katzenschlager
Wiener Krankenanstaltenverbund

- Document Figure 2.8 by B. Manor et al. (https://mhealth.jmir.org/2018/1/e36/)