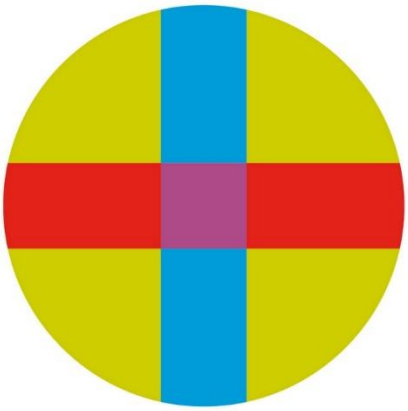
UNIVERSITY CEU - SAN PABLO

POLYTECHNIC SCHOOL

BIOMEDICAL ENGINEERING DEGREE



BACHELOR THESIS

Development of a tool for gait pattern characterization in a robotic walker for hip fracture rehabilitation

Author: Alfonso Rafael Gordon Cabello de los Cobos
Supervisor: Vanina Costa Cortez

June 2023

UNIVERSIDAD SAN PABLO-CEU
ESCUELA POLITÉCNICA SUPERIOR
División de Ingeniería

Datos del alumno

NOMBRE:

Datos del Trabajo

TÍTULO DEL PROYECTO:

Tribunal calificador

PRESIDENTE:	FDO.:

SECRETARIO:	FDO.:

VOCAL:	FDO.:

Reunido este tribunal el _____/_____/_____, acuerda otorgar al Trabajo Fin de Grado presentado por Don_____la calificación de _____.

ACKNOWLEDGMENTS

Gracias a Dios, todo en esta vida tiene un principio y un final por muy triste que resulte. Este trabajo marca el último hito de mí trayecto como estudiante, o por lo menos indica una parada larga en el camino, y querría aprovechar esta sección para dar gracias a todos aquellos que me han acompañado, sobre todo, durante este viaje de cuatro años pues es de bien nacido ser agradecido.

Primero, quiero agradecer a mi familia, en especial, a mi madre, a mi hermana y a mi padre, pues son ellos los que estaban en primera fila cuando el camino se empedraba, se embarraba, y no parecía que hubiese forma de continuar sin reventar. No me olvido de muchos de mis tíos y abuelos, tanto los que me ven físicamente como los que no, que siempre que han podido me han acabado echando una mano de una manera u otra, aunque he de admitir que le he cogido el brazo a más de uno.

Segundo, a mis amigos (tanto los que he hecho durante este camino, antes o después, como los que ya traía) y a mis primos, pues son ellos a los que llamaba cada vez que necesitaba desconectar, repostar y cambiar el punto de vista y que siempre me han respondido con "¿Dónde nos vemos?" sin preguntar nada más.

Por último, pero no menos importante, a los profesores que me han dado clase en este centro, pues todos me han dedicado parte de su tiempo, lo cual es sabido por todos, que vale oro. En especial me gustaría agradecer a Cristina Sánchez, Ana Rojo y Vanina Costa, que cuando mandaba un correo diciendo "ha pasado una cosa", o similares, me han respondido siempre con paciencia, una sonrisa y una velocidad pasmosa "No te preocupes que lo solucionamos en un momento".

A todos, gracias.

ABSTRACT

This work aims to develop a tool to identify and report gait parameters based on the range of hip joint motion obtained from robotic platforms. The tool relies on flexion and extension data obtained from the SWalker robotic platform, designed for rehabilitating patients with hip fractures, which are common in the elderly. The tool aims to provide useful information, beyond the range of motion, to physiotherapists responsible for directing rehabilitation sessions, helping them designing more specific sessions and monitor the patient's physical progress.

The platform SWalker measures the range of hip movement using potentiometers placed on a rigid structure at pelvis level. The tool designed on this project calculates the gait parameters: cadence, step length, stride length, step time, single support time, double support time, support time, stride time, walking speed, stride speed, and swing time.

RESUMEN

Este trabajo tiene como objetivo el desarrollo de una herramienta para identificar e informar sobre parámetros de marcha a partir del rango de la articulación de la cadera obtenido en una plataforma robótica. Dicha herramienta se basa en datos de flexión y extensión obtenidos de la plataforma robótica SWalker, diseñada para rehabilitar pacientes con fractura de cadera, comunes en los ancianos. La herramienta pretende aportar información útil, más allá del rango articular, a los fisioterapeutas que se hayan a cargo de dirigir las sesiones de rehabilitación, ayudando a diseñar sesiones más específicas y ver la evolución del estado físico del paciente.

La plataforma SWalker mide el rango de movimiento de la cadera mediante potenciómetros colocados en una estructura rígida a la altura de la pelvis. La herramienta desarrollada en este proyecto calcula los parámetros de la marcha: cadencia, longitud del paso, longitud de zancada, tiempo de paso, tiempo de soporte simple, tiempo de soporte doble, tiempo de apoyo, tiempo de zancada, velocidad de marcha, velocidad de zancada y tiempo de balanceo.

INDEX

FIGURE INDEX

TABLE INDEX

EQUATION INDEX

1 INTRODUCTION

1.1 Human gait

Gait can be described as using the lower limbs to move through a space at a determined speed. The gait cycle is the series of periodic events performed by each leg when one leg acts as a support source while the other one is advancing [1] and can be affected by numerous conditions such as sex, age, height, weight, or previous injuries like hip fracture. One complete cycle is achieved when two walking steps are made, one step per leg.

The gait cycle is divided into two main phases, swing and stance, which contain a total of seven subphases. Each leg goes through one of these phases per cycle. Depending on the legs supporting the weight there are also two different periods, the single support period when one leg is supporting all the body weight, and the double support period when both legs support the body weight.

1.1.1 Phases and parameters of the gait cycle

The gait cycle contains two mains, the swing, and the stance phase, which include other subphases phases [1] [2].

The stance phase starts with the initial contact of one leg and ends when that leg's toe stops contacting the floor. It is the phase at which the leg supports the body weight load and occupies 60% of the cycle. This phase includes the following subphases:

- Loading response (or heel-strike): starts with the initial contact of the foot on the floor and goes until the opposite toe takes off. During this phase, the foot lowers entirely to the ground due to ankle plantarflexion.
- Mid-stance: starts when the opposite toe takes off until the heel begins to lift. During this phase, the foot lowers entirely to the ground due to ankle plantarflexion.
- Terminal stance: begins when the heel rises until the other foot starts grounding. During this phase, the body weight moves in the direction of the walking.

- Pre-swing: begins with the opponent's leg initial contact until the toe takes off. During this phase, the body weight load changes from one leg to the opposite.

The swing phase starts with the toe taking off the ground and ends with the initial contact. This is the phase at which the leg is in the air and occupies 40% of the cycle. This phase includes the following subphases:

- Initial swing: starts when the foot is lifted from the floor until the foot is opposite to the stance foot. During this phase, the foot is lifted, and the limb advances due to hip and knee flexion.
- Mid-swing: starts when the foot is opposite to the stance foot until its forward and the tibia is aligned with the femur. During this phase, the foot is advanced from the weight line due to hip and knee flexion.
- Terminal swing: starts when the tibia is fully aligned with the femur until the foot lands. During this phase, the advancement of the body is completed.

During these phases, there are periods at which one or both legs support the body weight. These periods are called single support or double support, respectively. The double support periods occur between the initial contact of one leg and the toe off from the other (loading phase and pre-swing phase) and the single leg support occurs during the rest of the stance phase of that leg.

All these phases and stance periods are shown in Figure 1 [3].

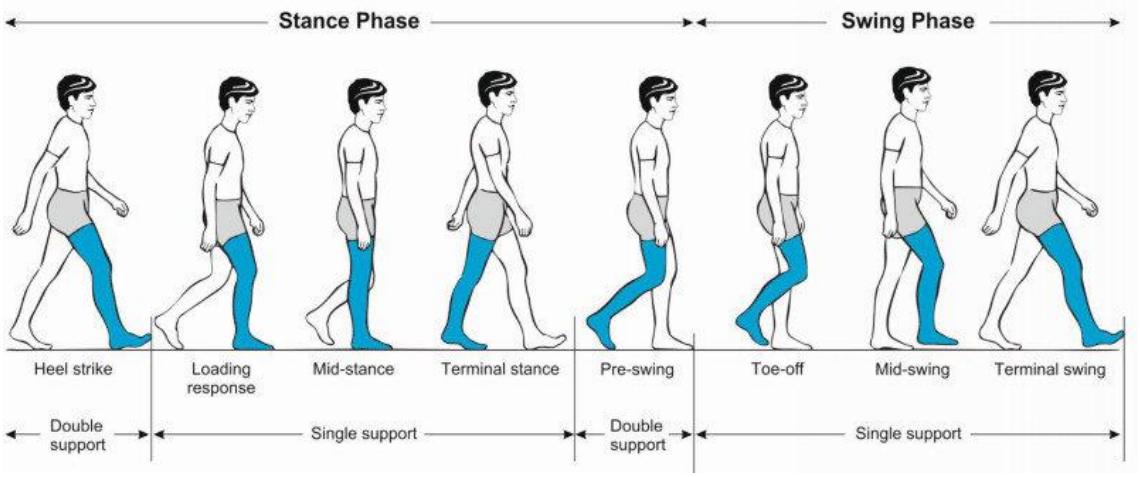


Figure 1: Representation of the gait cycle phases, subphases and periods.

There are some biomechanical parameters extracted during the walk that can give more information about the patient's gait quality by focusing on the physical properties [1]:

- Step length [cm]: distance from a heel to the opposite's footprint heel.
- Stride length [cm]: distance between the same heel at two consecutive steps.
- Step width [cm]: lateral distance from heel center to the line of progression. The line of progression is formed as the line formed by two consecutive footprints of the opposite foot.
- Cadence [steps/min]: also called step rate, is the number of steps per minute.
- Step time [s]: time from the initial contact of one foot to the initial contact of the other.
- Stride time [s]: time between two consecutive footfalls from the same foot.
- Stance time [s]: time between initial contact and foot take-off. This parameter can be normalized to stride time.
- Swing time [s]: time between foot take-off and initial contact. This parameter can be normalized to stride time.

- Single support time [s]: time between foot take-off and initial contact. This parameter can be normalized to stride time.
- Double support time [s]: sum of time during two periods of double support. This parameter can be normalized to stride time.
- Gait speed [cm/s]: distance walked divided by stride time.
- Stride speed [cm/s]: stride length divided by stride time.

1.1.2 Description of gait in the elderly and frequent pathologies

Age is a factor of influence in the gait cycle and its parameters. In a margin of 25 years, from 60 to 85, there is a drop of 67% in the elderly's parameters against the normal ones [4]. This is due to the age-related atrophy of motor and cortical regions and because of the atrophy, two consequences appear: the loss of muscle strength and the compensatory adaptations.

There is a decline in stride length, cadence, symmetry of the step, walking speed, knee flexion and ankle plantarflexion, and in the power of ankle, knee, and hip, also there is an increase on the stride width, step and stance times, and energy used. These changes indicate that older adults walk slower and try to maximize their stability [5].

Also, some multiple pathologies and injuries can affect the gait parameters and phases. Some pathologies are presented afterwards, including how they affect the gait cycle [3].

Musculoskeletal pathologies

This group includes non-neurological gait disorders in adults, osteoarthritis, and skeletal deformities are the most common ones. Disturbances are characterized by limited ROM, avoidance of weight-bearing and asymmetry or limping. The most common gaits in this group are:

- Antalgic gait: caused by painful conditions such as osteoarthritis, ankle sprains and stress fractures, patients appear to walk like a thorn in the sole by lifting and lowering the foot from the injured leg in a fixed angle.

- Coxalgic gait: caused by hip pain, patients shift the upper trunk towards the affected side to reduce forces exerted on the affected hip during the stance phase.
- Knee hyperextension gait: caused by quadriceps muscle weakness, patients perform initial contact with a flat foot also a high ankle plantar flexion and hip extension is used to advance the affected leg during the stance phase.

Brain disfunction pathologies

This group includes gait disorders related to brain diseases or dysfunctions. The most common gaits in this group are:

- Phobic gait: maximum variant of the typical cautious gait developed by elderly.
- Spastic gait disorders: knees slightly flexed and feet in plantar position decrease in step length and shortened weight-bearing phase on affected side.
- Ataxic gait disorders: characterized by abnormal lower limb posture, irregular leg movement and shortened step length.
- Frontal (or higher level) gait disorder: characterized by difficulties in initiating and maintaining gait, a broad base of support, short step length, shuffling or scuffling gait, reduced arm swing, and impaired balance and postural stability.
- Parkinsonian gait disorder: like frontal gait disorder, and can be mistaken sometimes, but is characterized by a stooped posture, reduced arm swing, short shuffling steps, and a tendency to freeze or get stuck while walking.
- Dystonic gait disorder: the abnormal posture of the foot involving inversion, plantar flexion, and tonic extension.

Neuromuscular pathologies

This group includes all the disorders that include severe peripheral paresis. The most common gaits in this group are:

- Waddling gait: caused by the weakness of the hip girdle and upper thigh muscles, patients bend the trunk towards the side which is in the stance phase since the hip on the swinging side drops with each step.
- Steppage gait: caused by paresis on the lifting foot, patient lifts legs higher than usual.
- Neurogenic and intermittent claudication: caused by peripheral arterial occlusive disease, the patient experience pain or cramps in the claves, feet, or thighs after walking over a distance.
- Lumbar spinal stenosis and neurogenic claudication: caused by the stenosis, the patient experiences deep muscular pain and neurological deficits.
- Myelopathic gait: caused by cervical spondylotic myelopathy, gait gets stiff and spastic (paraplegic and ataxic).

Hip fracture

Hip fracture injuries occur when the upper part of the femur, typically the femoral neck, breaks (Figure 2). They are most observed among elderly individuals, and the global incidence has been steadily increasing. In 2000, approximately 1.6 million cases were reported, projected to reach an estimated 4.5 million by 2050 due to population aging [6]. The mortality risk associated with hip fractures spans from the time of hospitalization (4.3%) to up to 12 months later (17.4%) [7].

Falls resulting in hip fractures have a notable impact on gait patterns. Fallers tend to exhibit slower walking speeds, shorter steps, an increased stance phase, and reduced swing phase. Among these factors, the variability of step time serves as a significant indicator [5].

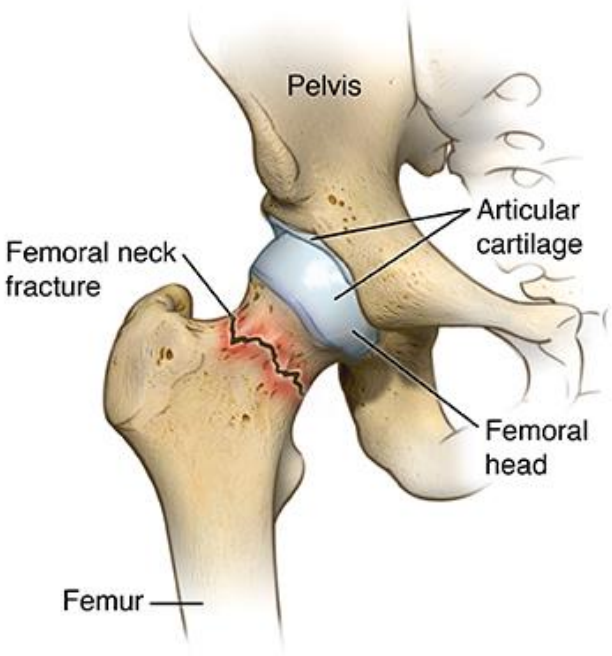


Figure 2: Hip fracture.

1.2 Range of motion

The Range of Motion (ROM) is the rotation made by a joint while moving. It is measured in degrees and compared to one of the anatomical planes. It allows to know if there is any kind of problem that is affecting the joint and it varies depending on the body part. The starting position before measuring is the anatomical position [8] [9]. There are two main types of ROM:

- Active: obtained by the subject unassisted. It allows to help focus physical examination. Provides information about willingness of move, coordination, muscle strength and joint ROM.
- Passive: obtained from the subject by the examiner. The subject is relaxed and plays no role in motion. Provides information about the integrity and extensibility of the joint.

Different facts influence the ROM measurement and takes the different anatomical planes (Figure 3), as reference. The anatomical planes divide the body in two parts, equal or not. The transverse plane divides the body in superior and inferior portions, the frontal plane divides the body into anterior and posterior sections, and the sagittal plane divides the body into right and left parts.

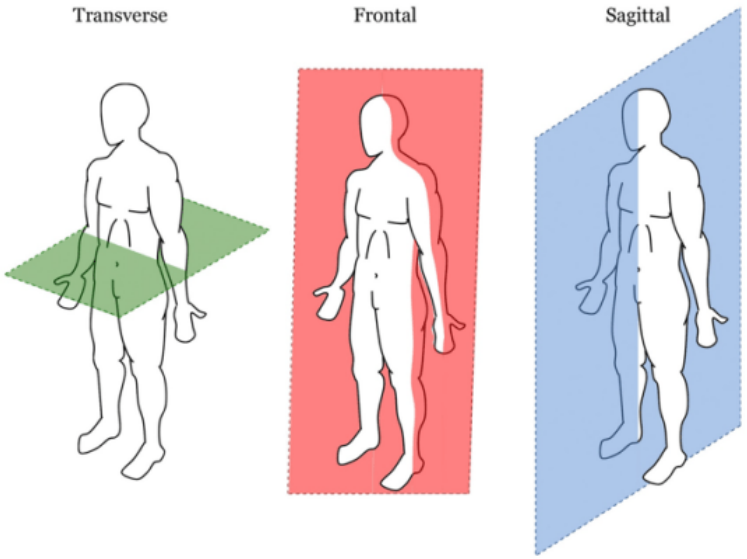


Figure 3: Anatomical planes on the human body at anatomical position.

1.2.1 Hip ROM

The hip (Figure 4) [10], links the lower extremity with the trunk and has three degrees of freedom being the center of the femoral head the axis of motion.

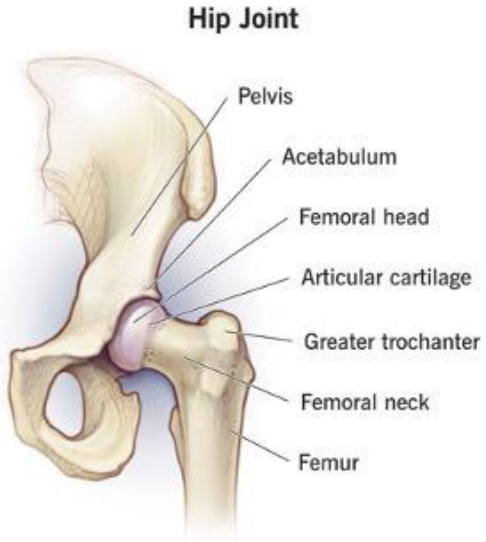


Figure 4: Anterior view of the left hip joint.

The movements carried out by the hip during the gait [9] on each anatomical plane are:

- Sagittal plane: flexion and extension
- Frontal plane: abduction and adduction
- Transverse plane: medial (or internal) rotation and lateral (or external) rotation.

These movements can be seen in Figure 5 [11]. On a normal gait, the hip mainly flexes and extends. Flexion occurs during the middle of the swing phase until the contact and extension occur from the beginning to the middle of the swing phase.

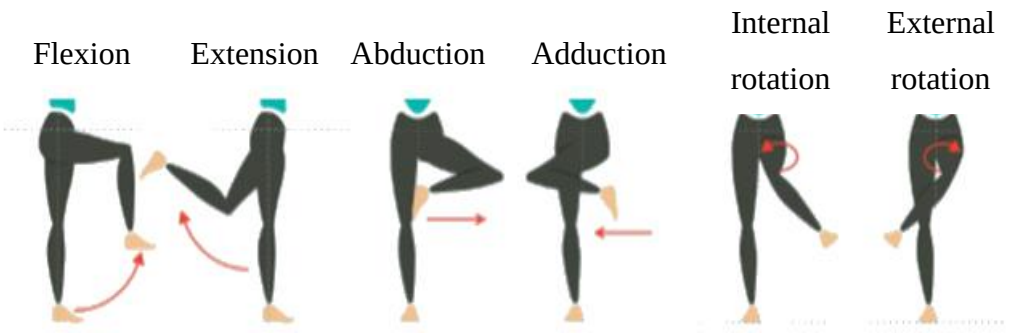


Figure 5: Hip movements.

The hip's mean ROM varies greatly among different age groups due to the problems that are related to aging like the loss of muscle strength or coordination. The variation on the ROM changes for each hip movement depending on the ages [9] can be seen in Table 1.

	25-39 years	40-59 years	60-74 years
	Mean (Standard Deviation)	Mean (Standard Deviation)	Mean (Standard Deviation)
Flexion	122 (12)	120 (14)	118 (13)
Extension	22 (8)	18 (7)	17 (8)
Abduction	44 (11)	42 (11)	39 (12)
Medial rotation	33 (7)	31 (8)	30 (7)
Lateral rotation	34 (8)	32 (8)	29 (9)

Table 1: Adult ROM values for hip movements, values in degrees.

1.2.2 ROM measuring methods

There are multiple ways to get the ROM from a patient, from analog to digital. Nowadays the most extended are goniometers, photogrammetry, inertial sensors, and potentiometers.

Goniometer

The goniometer (Figure 6) measures the angles created at the joints by the immediate proximal and distal bones [9]. This measure allows us to determine the joint position and functional movement.

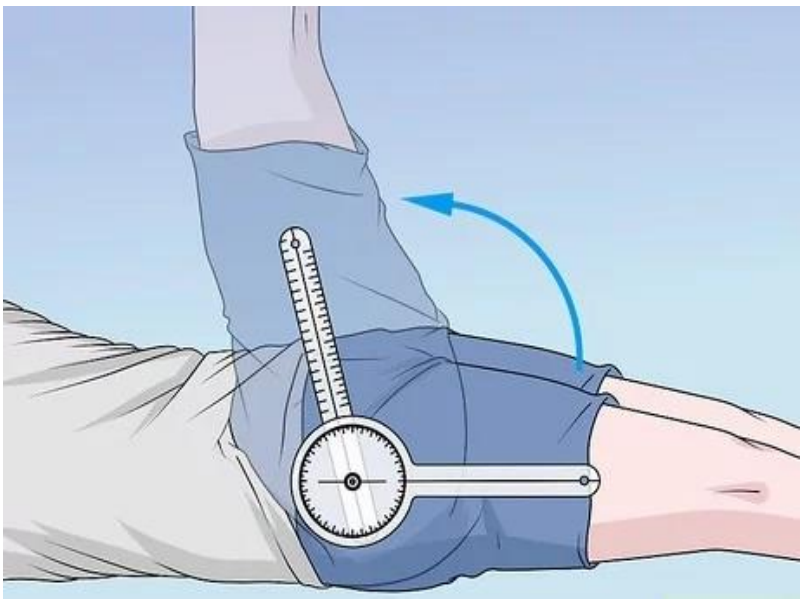


Figure 6: Goniometer measuring hip's flexion angle.

Although this technique allows to provide an objective measure of the joint ROM and can be used for the identification of joint limitations and asymmetries and for assessing the effectiveness of interventions, it requires specialized training and expertise and the accuracy can be influenced by pain, muscle spasm or similar [9].

Photogrammetry

During the past few years, motion capture has been used for detecting and digitally replicating the patient's movements. Video capture systems consist of placing reference points on the surface of the patient's body (Figure 7) and filming his movement using multiple cameras [12]. The program will reconstruct the movement using an artificial skeleton.

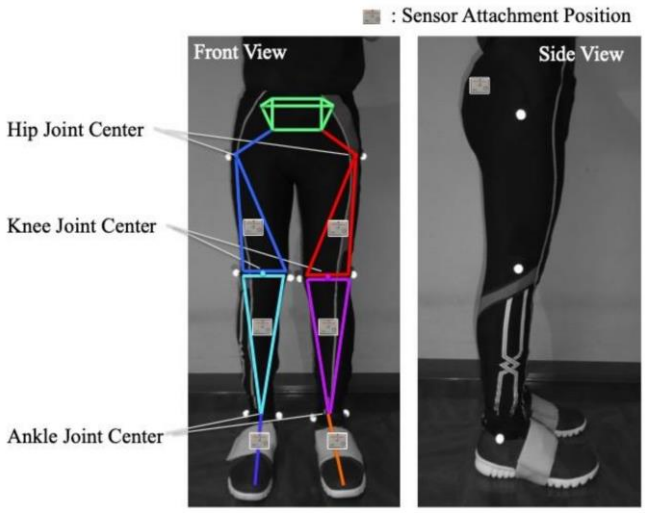


Figure 7: Markers used in photogrammetry.

The main advantages of this method are the visual feedback provided to patients and doctors, which also allows remote assessment and the objective and quantitative measurement it provides since the markers send the position and how it varies over time. This data allows to perform a complete biomechanical analysis of the patient's gait. Still, the main disadvantages are the costs required for the equipment, which usually include 8 to 10 cameras, the need of supervision while controlling the equipment and the space needed for this system to be installed.

Inertial sensors

Inertial sensors or inertial measurement units (IMUs) are devices composed of accelerometers and gyroscopes [13] that measure a body's specific force, angular rate, and sometimes the orientation of the body by detecting rotational movement of the three-axis (Figure 8).

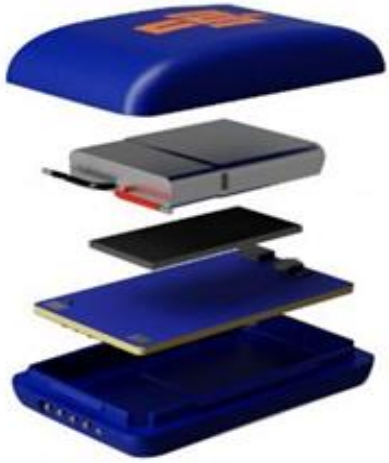


Figure 8: Inertial Motor Unit diagram.

On the one hand, these sensors need to train the patient for the correct use, also, they can be affected by placement. On the other hand, these sensors are portable and easy to use, making them useful for different things, and they provide both qualitative and quantitative data, which provides a better insight into the patient's gait.

Potentiometers

Potentiometers are composed of a variable resistor that measure the angle of the joint to which is attached [2]. When the central spindle is rotated due to the movement of one of the fixations respecting the other, the resistance variates and changes the electrical output. The following figure (Figure 9) [14] shows how potentiometers can be attached.

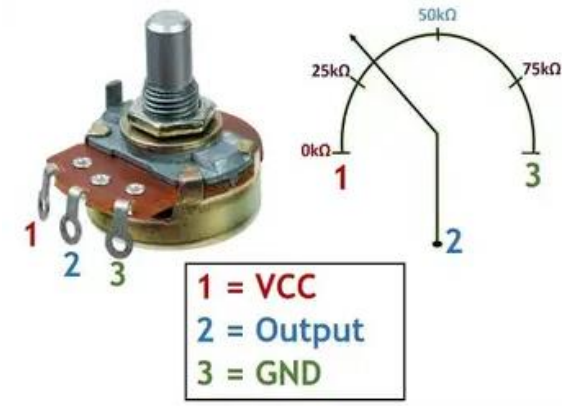


Figure 9: Potentiometer.

The main advantages are the high accuracy that they provide, they are noninvasive and portable. On the other hand, the measurement is only taken on one axis, so the information on the rest has a limited range. They might not be suitable for some joints, and the measurement is only taken in one axis, so the information on the rest will be missing unless multiple are used (one per axis).

1.3 Gait rehabilitation

Rehabilitation when talking about hip fractures, includes a wide range of practices and techniques to recuperate an almost correct gait pattern [15]. Some of these techniques are:

- Clinical pathway: describes routine interventions for a group of patients with similar needs, including expected outcomes at each step.
- Early supported discharge: intensive rehabilitation program in patient's own homes rather than in the hospital, reducing risks of complications, hospital resources costs and improving quality of life.
- Interdisciplinary care: a group of healthcare professionals from different areas assessing the patient during rehabilitation. Since different specialists form the group, the rehabilitation program will be adjusted to the patient's unique needs improving the outcomes.
- Occupational therapy: teaching patients' new ways to perform daily activities for them to be able to perform such activities without external help.
- Exercise: the most known way to perform exercises for gaining muscle mass and strength in the parts where there were lost. In the case of this project would be in the hip and the main method would be by walking using some kind of support like parallel bars, crutch, or a walking stick.

Due to the technological advances during the past decade, also some technical devices were developed and implemented in the rehabilitation sessions, like the SWalker [16], a robotic platform that can support part of the patients weight during the rehabilitation and program a setup speed for the patient to be walking at.

1.3.1 Gait rehabilitation exoskeletons

The technological advancements allow the creation of several types of prosthesis and orthoses that show promising results concerning their usefulness in rehabilitation sessions.

Several types of rehabilitation platforms depend on the joint and pathology to be rehabilitated. When talking about the purpose of rehabilitation, we can divide the exoskeletons into three main types [17]:

- Joint rehabilitation: they are mainly designed for hip, knee, and ankle rehabilitation. These exoskeletons take measures from the patient, using the sensors like IMUs or potentiometers, to evaluate the biomechanics and physiological responses of the patient and improve their therapy.
- Gait rehabilitation: they help the patients to improve their abnormal gait by giving power and in some cases sustaining a part of the patient's weight. Some clinical protocols use EMG for recording the patient's metrics, but also EEG has been used to improve human-computer interactions.
- Daily life assistive: these exoskeletons are not designed properly for rehabilitation but for aiding patients that have lost control over most of their lower limbs to complete different everyday activities. These patients include affected by spinal cord injury, stroke, or brain trauma.

The sensors used are usually located in different parts of the exoskeleton and a program can process the different signals from each sensor and extract the different gait parameters.

Most of these devices take the information on the sagittal plane [18]. As it was mentioned before (Section 1.2), there are three main planes that divide the human body and by using this plane we can get most of the gait information form the patient on an easy way.

Another reported problem is the integration between the human body and the exoskeleton due to the rigid design adopted by most of them. These designs might be

uncomfortable for the patient while getting on or using it. Some commercially available exoskeletons for rehabilitation of the lower limb are LOKOMAT [19] and HAL [20], both sowed in Figure 10.

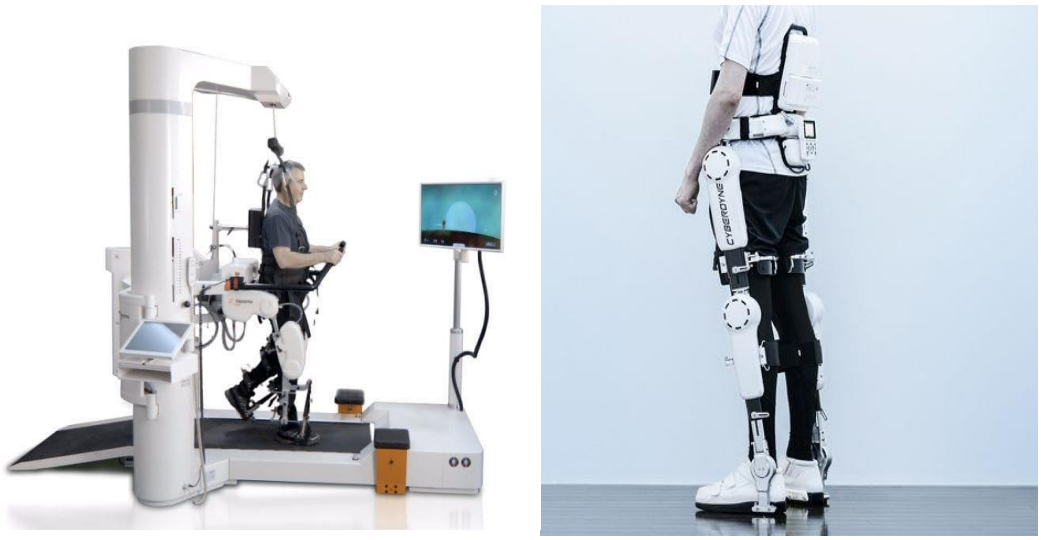


Figure 10: LOKOMAT (right) and HAD (left) exoskeletons.

In general, the biggest problem with these devices are their sizes, the need of supervision by the designers and the use of walking aids [21]. Also, another problem arises when talking to the therapists, which is the user experience. Most of the therapists have a basic knowledge on computers and how to manage those systems, this is the reason the Graphical User Interfaces (GUIs) and other tools are created. These tools help the therapists by giving an easier way to use the exoskeleton and a simplification of the results, which simplifies the work of analysis of the signals.

The detection and processing of the ROM signal and the gait parameters is highly relevant for the physiotherapists since they represent the mobility status of the patient. As the patient gets better it will show a higher ROM signal and the number of steps will increase along with the swing phase duration. By giving this data to the physiotherapist, it would be able to modify the rehabilitation therapy based on quantitative data to get a better outcome.

2 PROYECT FRAMEWORK

The design of the robotic platform is a collaboration led by Albertia, a group of nursing homes in Spain, and Werium Assistive Solutions. The SWalker [16] is a robotic platform designed for gait rehabilitation on elderlies who have suffered of hip fracture to promote an early recovery for the elderly to get a higher level of independency and quality of life at the lowest cost possible.

This device can be divided into three modules (Figure 11):

- Traction: includes the electronic control and the drive system that impulses the patient.
- Weight support: hydraulic system that sustains a part or all the patient's weight to reduce the weight supported by the hip.
- Patient's support elements: parallel bars and a trunk harness that help the patients.

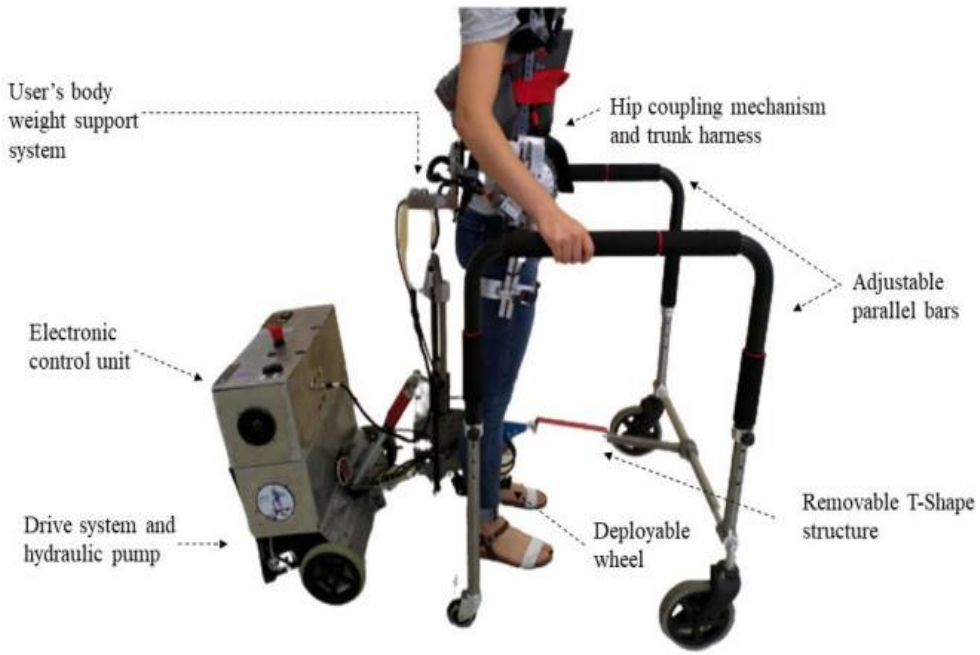


Figure 11: SWalker robotic platform.

Sensors built within the SWalker gather data on physiological and mechanical characteristics. The patient's weight relief and walking speed are the two most important

therapy parameters that are configured using this data. Potentiometers also enable the collection and visualization of the patient's hip ROM.

The system can reach up to 0.4 meters per second and it is designed to support patients that have a height up to 180 cm and up to 90 kilograms weight. The speed can be adjusted at three constant velocities of 0.058, 0.225 and 0.4 m/s, also the weight support can be configurated for supporting a percentage of the patient’s weight.

The rehabilitation with this device works by adjusting the weight supported and the speed of the traction system. Initially all the patient’s weight will be fully supported, and the rehabilitation sessions will be performed at different speed. As the patient gains muscle strength and recovers, the speed will increase, and the weight supported by the device will decrease until the patient is able to support its own full weight and walk at the maximum speed.

The device is controlled by a web interface that allows to configure and save the session’s ROM recorded with the potentiometers located in the harness at the heigh of the hip. There are only two potentiometers, one per leg, which are the ones in charge of measuring the hip ROM.

The calibration of the potentiometers is usually done before the recording of the rehabilitation session. The therapist asks the patient to stand up straighten up and calibrates the sensors through the web interface. The values registered by the potentiometer are stored in an Arduino board and sent to the application, installed on a tablet, which controls the SWalker. The files from each session can be exported after the recording of the session as a Microsoft Excel file (.xlsx). The .xlsx file extension was created by Microsoft as an open format [22].

3 PROJECT OBJECTIVES

The main objective of this project is to create a tool that calculates the main gait parameters from the hip ROM extracted during rehabilitation sessions on a robotic platform. We are going to use Python as programming language since it has multiple specialized libraries that are going to be helpful, such as NumPy or SciPy. Python is an interpreted language meaning that scripts can be executed on any platform without prior compilation and has an extensive user community which can be helpful while looking for information in case needed.

The main steps for developing this project will be:

- Gathering information on the gait cycle, gait parameters, devices used for patient measurements during rehabilitation and rehabilitation methods.
- Understanding of the parts and operation of the robotic rehabilitation device, the measuring instrument of this project. One of the key points in this phase is the study of the ROM signal acquired by the device.
- Programming of the gait biomechanical parameter extraction tool.
- Testing phase of the tool with different real ROM signals.

4 TOOL DEVELOPMENT

The programming language used for signal processing will be Python and the files analyzed will be an original file from an elderly patient, with pathological gait, and a file from a young healthy patient, with a normal gait. The calculated parameters will be step length, stride length, cadence, step time, stride time, stance time, swing time, single support time, double support time, gait speed and stride speed.

4.1 Acquisition of the ROM signals

The SWalker use two potentiometers, one per leg, located on a hip coupling mechanism (Figure 12) [16] for recording the hip's ROM at each leg. The potentiometers can change their resistance while rotated and that resistance change allows to capture the amplitude of the ROM signal for each leg.

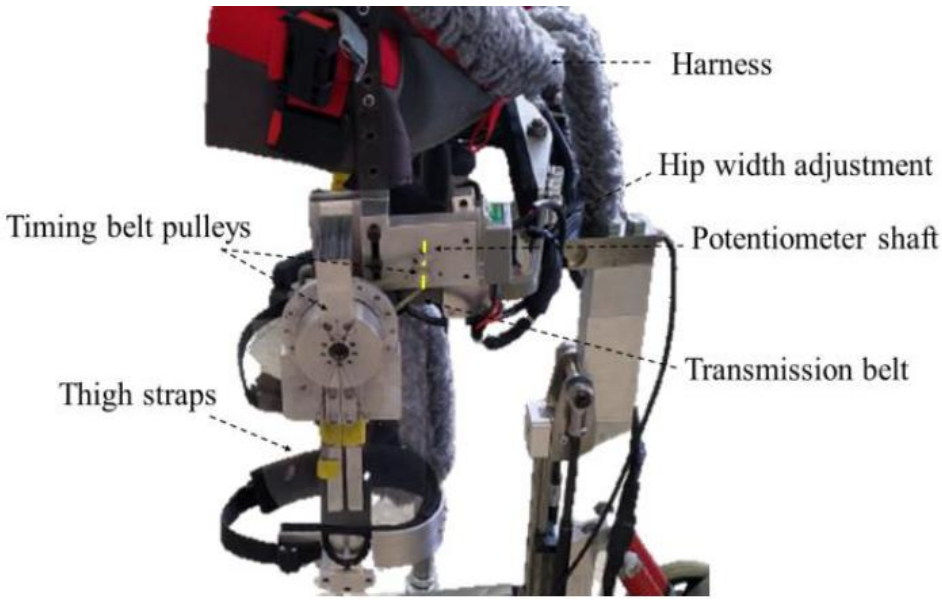


Figure 12: SWalker's potentiometer.

The resultant ROM signal from the session recorded with the SWalker (Figure 13) shows the variance of the hip's angle in the sagittal plane across the time, being each maximum peak when the leg is in the stance phase and each minimum peak when the leg is in the swing phase. As shown in Figure 13, if the patient has some kind of gait pathology that leads to tremor or unsteady gait, some noise will appear during the recording of the joint range, which will form multiple peaks in the signal.

Figure 13: Hip ROM signal of one leg.

For this project, two different signals were used while developing and testing the tool, the first one was a signal from an elderly patient, a pathological gait, the second one was from a young healthy patient, a normal gait. The tool was developed using the pathological gait signal since it will be the future target signals to be analyzed. The normal gait signal was used for improving the algorithms for the parameters' calculations.

In subjects with impaired gait quality, as is the case with the elderly's signal, it is common to have some records before the patient is upright and the whole system is calibrated. This is often due to poor understanding of the instructions by the patient. It will be noted below that the file used for this project includes this additional set of records. A new file will be generated, excluding these values, and then evaluated to see if they could have an impact on the final analysis if they are not removed.

4.2 General overview

The tool will first read the files at an indicated directory and save them into a dictionary. From the dictionary it will take the values from the time stamps and the left and right hips' ROM for calculating the parameters.

The parameters will be calculated for each file, saved as keys in the dictionary, by the following order:

1. Cadence [steps/min]
2. Step length [cm]
3. Stride length [cm]
4. Step time [s]
5. Single support time [s]
6. Double support time [s]
7. Stance time [s]
8. Stride time [s]
9. Gait speed [cm/s]
10. Stride speed [cm/s]
11. Swing time [s]

After the parameters for each file are calculated and printed on the console, they will be saved on a CSV and a XLSX documents to facilitate visualization and analysis by the clinical team. This flow can be seen on the Figure 14. Right before calculating the parameters, a function called *Peak Detection* will detect and save the indexes at which each hill and valley from the ROM signals is. This is performed for accessing to those points easily on every function they are needed since they indicate the maximum ROM of the stance and swing phases as it will be explained later.

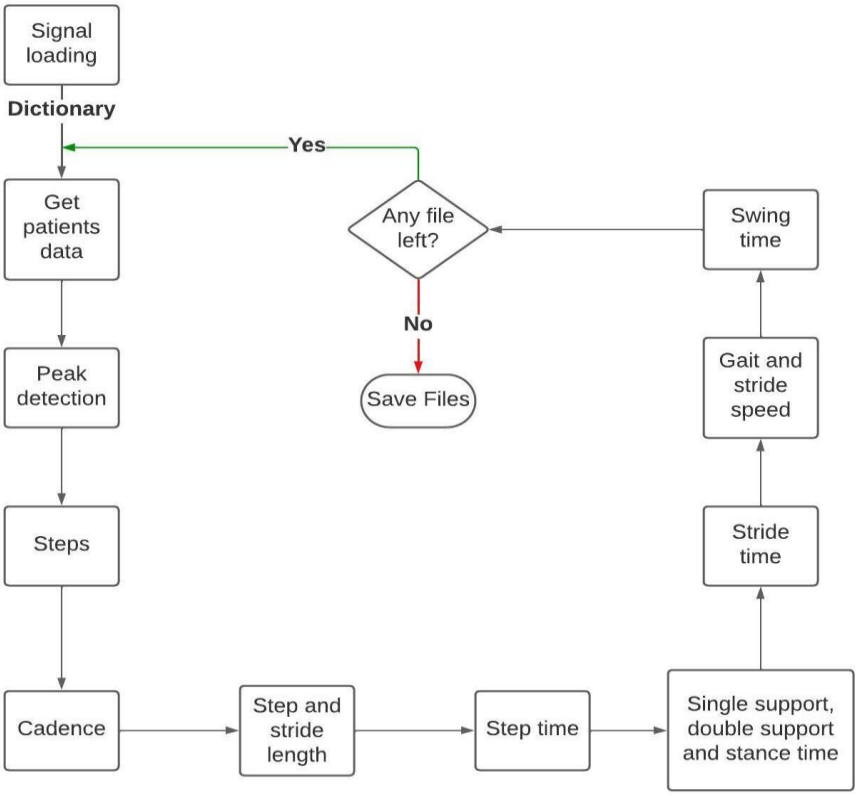


Figure 14: General overview flow from the tool.

The following sections explain the tasks performed by each programmed function. The complete developed code can be found in the annex of the document.

4.3 Signal integration

First, the function the user will need to introduce the path of the directory at which the files containing the sessions, .xlsx, are located. Care should be taken to ensure that all the sessions that are wanted to be analyzed are in the same directory and no other .xlsx files.

After introducing the directory, the user is asked to provide the operating system at which the program is working. This is done because each operating system uses a different format for indicating paths. Also, a data frame is created for future use when saving the results into the files.

Finally, we create a dictionary. The key to each entry is the name of the file excluding the .xlsx and each entry will contain all the columns from the file. From here on, all the functions will be executed inside a loop that will go through each key stored in the dictionary so each function will be executed once per key in the dictionary.

4.4 Getting patient's data

After loading all the patients and their data in the dictionary, the next step is to access and saving into variables. This function, *getDataPatient*, takes a key from the dictionary, which are the names of the files at which the data of the sessions were stored, as an argument and returns three arrays, each one containing the time register from that session, the left and the right hip's ROM stored values at that session. A simplification of the information of the dictionary can be seen in Figure 15. In orange, the data that will be used for the parameters' calculations.

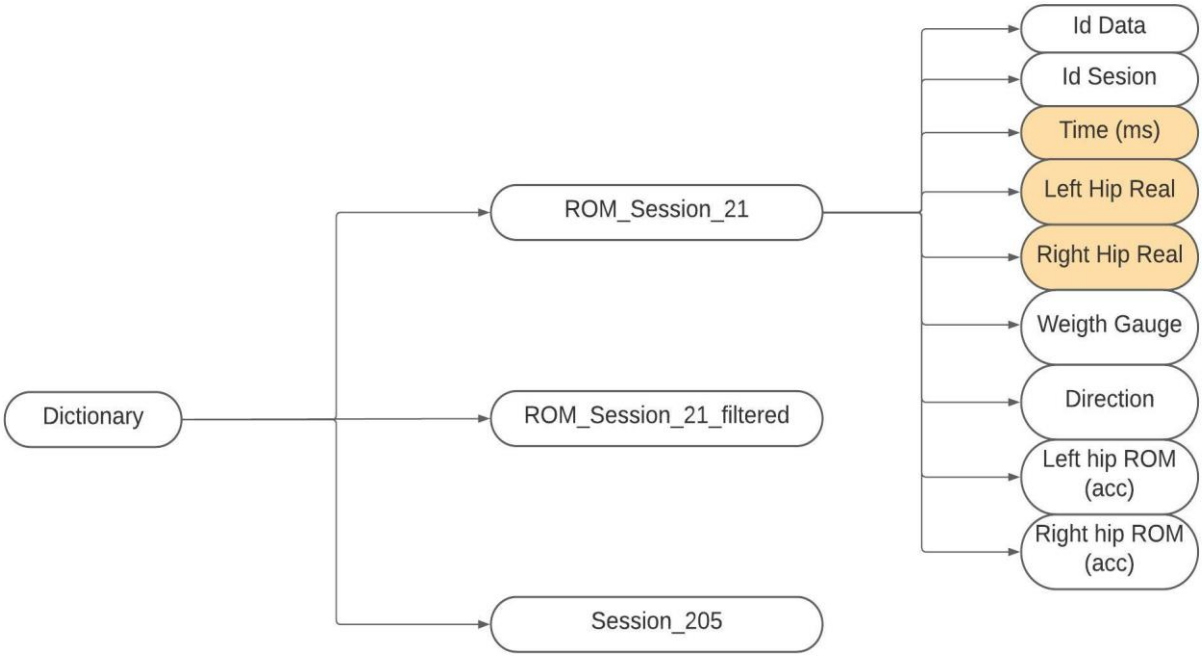


Figure 15: Simplification from the dictionary made from the files.

4.5 Peak detection

This function, *peak_detection*, takes the ROM data from one leg as an input and allows to locate the indexes at which the signal has a hill and a valley, corresponding to the peak moment at each phase, swing, and stance, returning an array with those indexes. It also takes the kay and a string that indicates to which leg it is referring while plotting the signals.

First the function *find_peaks* (from scipy.signal) is applied to the signal, locating the indexes of the hill values, and storing them in an array called *maxPeaks*. Since there is no function that can detect the peaks of the valleys, the signal is inverted by multiplying by -1 and *find_peaks* function is again applied but storing the values in a different array, minPeaks.

Then both arrays, *maxPeaks* and *minPeaks*, are combined and ordered in one only array, called *peakOrderSequence*, by looking the values and comparing them. This final array will store all the peaks of the sequence in real order of appearance.

After the peaks are ordered, a filter is applied. As it has been said previously, some noise might appear in the signal if the patient tripped, has some kind of gait pathology that leads to tremor or unsteady gait or due to a failure in the sensor that must be filtered. This noise can be seen as multiple peaks in the signal between the swing and stance phases or at the maximum point of such phases. The filter first calculates the mean value of the original leg signal, then creates a threshold of ± 2 units. The mean value is the reference since the pathological gait signals show a deviation above the zero which should be the refence point. For every value in the *peakOrderSequence*, it checks if the values are inside the threshold, if not then it checks if it is a hill and from that it adds the peaks to a new array, *realPeakSequence*. This last array is the one returned.

Also, this function plots both signals, the original one and the simplified one against the time that has passed, for this purpose, the time array is modified to show the real milliseconds. The results of the process over the signal of the young patient can be seen in Figure 16. The “Real ROM”, Figure 16a, shows the original signal. The “Mixed ROM”, Figure 16b, shows the peaks detected being the ones in orange the hills representing the maximum of the swing phase and the green ones the valleys representing

the stance phase. The "Simplified ROM", Figure 16c, is the result of the function. The orange line showed in Figure 16a, and Figure 16c is the mean ROM value.

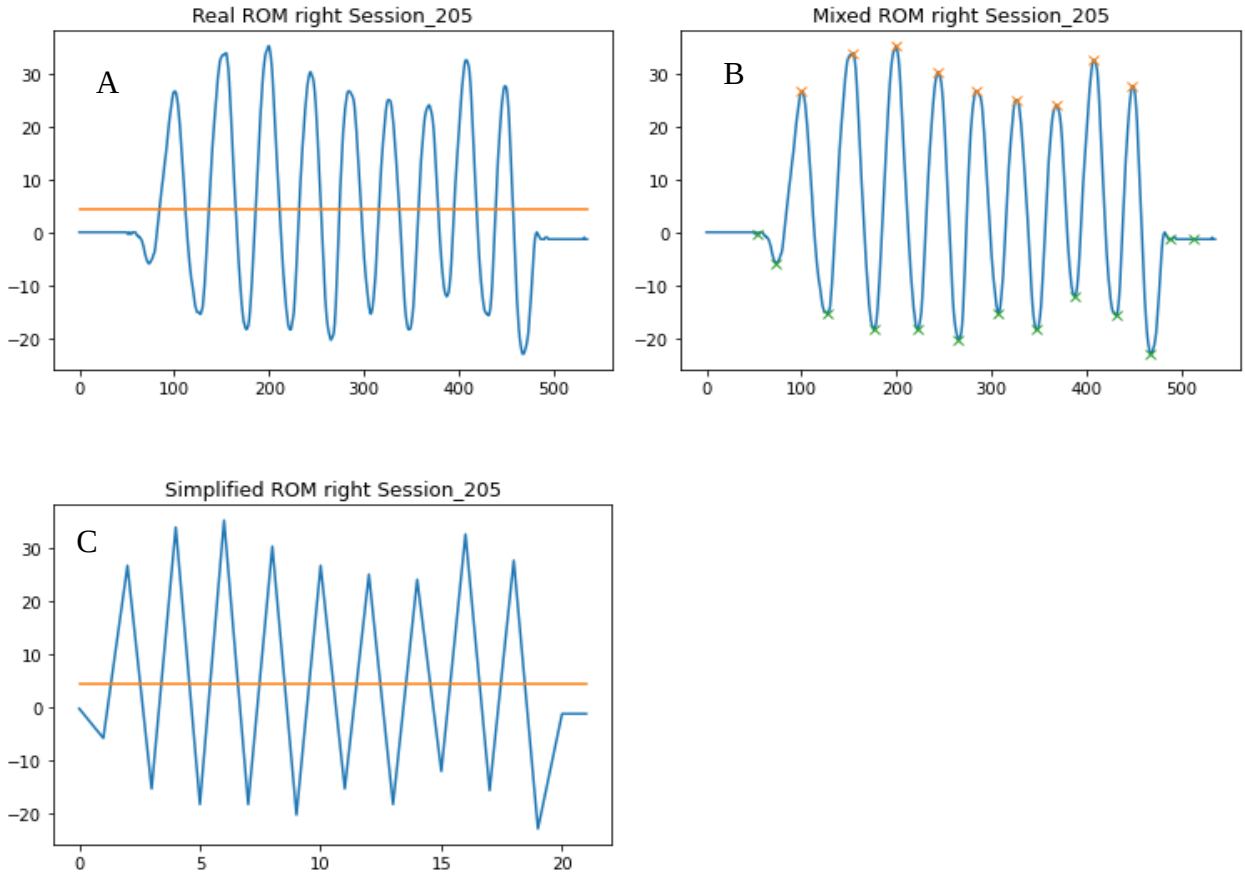


Figure 16: Plotting of the signals while running the peak detection function.

4.6 Functions for gait parameters' calculation

Step calculation

This function, *calculateSteps*, takes the simplified sequences from both legs. These simplified sequences are the ones returned in *peak_detection* that contains the indexes of hills and valleys of the original function. Since we have the hills and valleys which represent the peak moment of the stance and swing phase (Figure 17), it is only needed to divide the length of the array with the indexes by 2 to group the hills and valleys. Additionally, this function adds the steps made by each leg to get the total of steps performed during the therapy. The function returns the total steps made and the steps made by each leg.

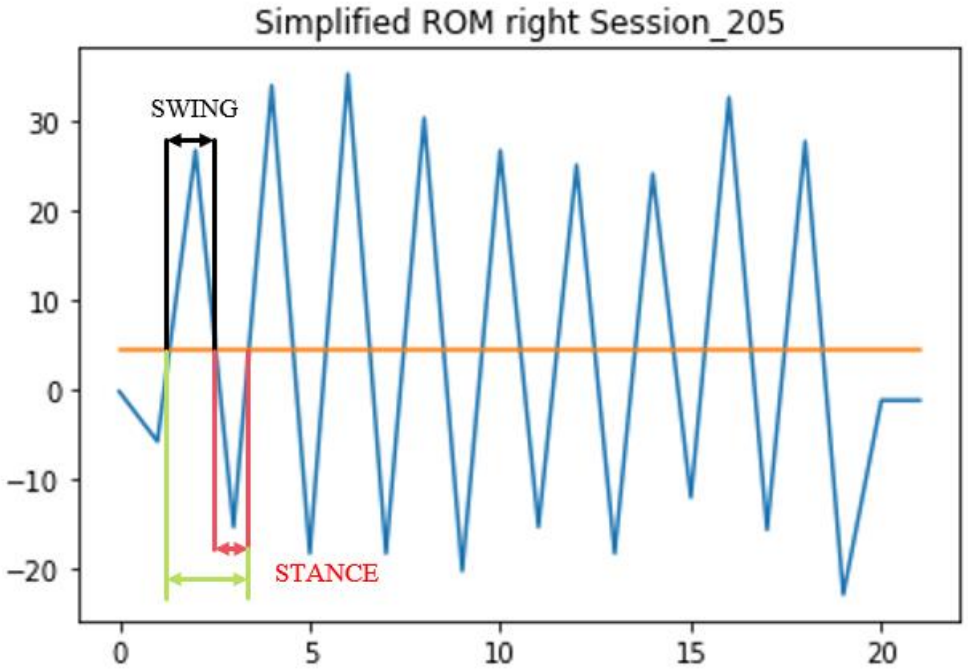


Figure 17: Simplified ROM indicating stance and swing phases.

Cadence calculation

This function, *calculateCadence*, takes again the total number of steps and the time record from the session. Since the cadence is the number of steps performed by minute, we calculate the time difference between the first and last element of the time array, which is the total time the session has last, and we divide that time difference in milliseconds by 1000 to get the total in seconds. Then we divide the steps by the time difference and multiply by 60 to change from steps per seconds to steps per minutes. This last value is the one returned. The whole operation can be seen in Equation 1 where Δt indicates the time difference, the duration of the session, t_f is the last time value stored and t_0 is the first time value stored.

$$\Delta \boldsymbol{t} = (\boldsymbol{t_f} - \boldsymbol{t_0})[\boldsymbol{ms}] * \frac{\mathbf{1}\,[\boldsymbol{s}]}{\mathbf{1000}\,[\boldsymbol{ms}]} \rightarrow \boldsymbol{Cadence} = \frac{\boldsymbol{Steps}}{\Delta \boldsymbol{t}} * \frac{\mathbf{60}\,[\boldsymbol{s}]}{\mathbf{1}\,[\boldsymbol{min}]}$$

Equation 1: Cadence calculation.

Step and stride length calculation

This function, *calculateStepAndStrideLength*, takes the total of steps performed by the patient as input and asks the therapist for the distance that the patient has walked through during the session. For calculating the average step length, the distance is divided by the number of steps.

Since the stride length is the distance between two consecutive steps from the same foot it can be calculated by multiplying the average step length by two. This is because these two steps from the same foot include a step from the other foot as can be seen in Figure 18.

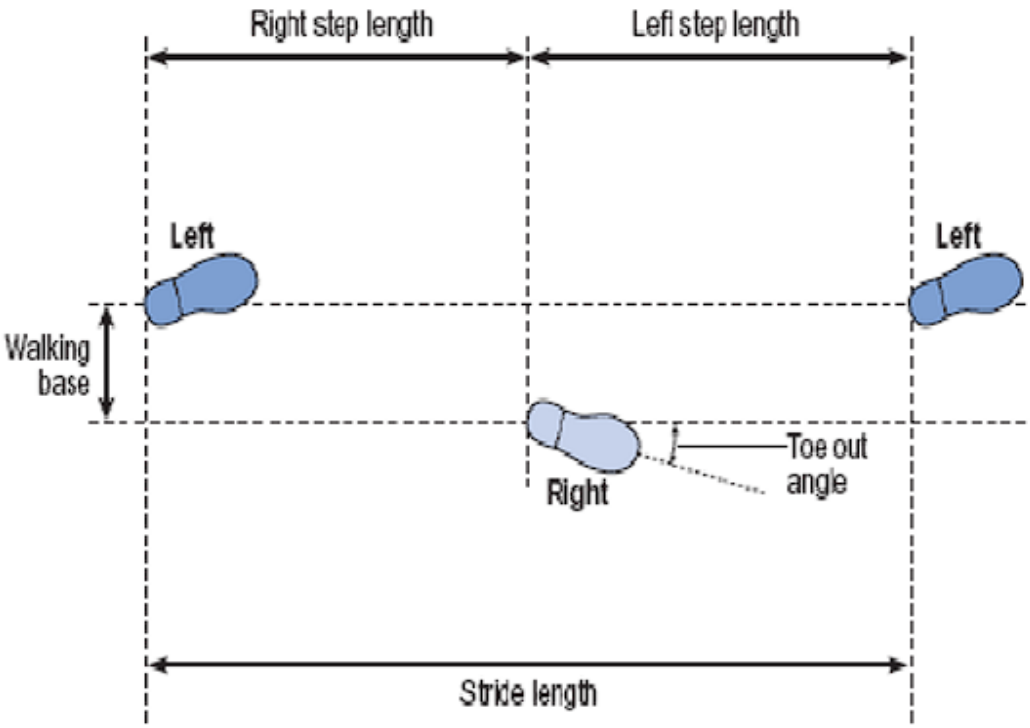


Figure 18: Step and Stride length representation.

Step time calculation

This function, *calculateStepTime*, takes the simplified sequences from both legs and the time sequence. This is done by going through the simplified arrays, independently, getting the time difference between steps, storing such time difference in two different arrays, one per leg, and returning the average step time for each leg. For differentiating the moment at which a step occurs we check if the variable that indicates the position on the simplified sequence is even, if it is the case then we subtract the time at which the peak occur from the time at which the next peak occur so the time difference is calculated every two peaks. The workflow of this function for one leg can be seen in Figure 19.

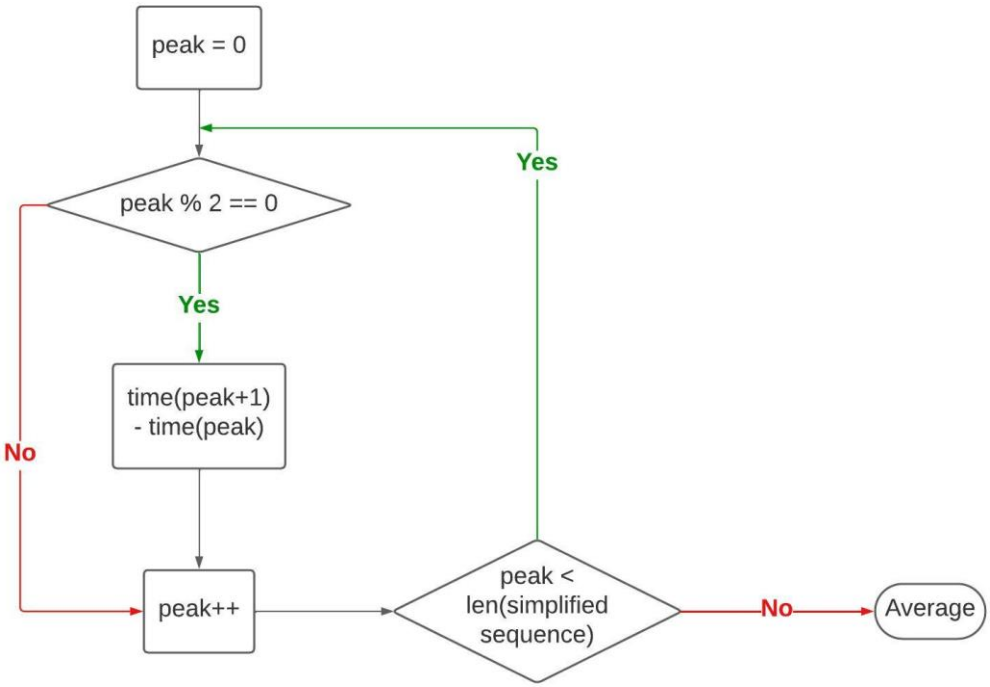


Figure 19: Step time calculation workflow.

Stance, single support, and double support times

The Double Support Period (DSP) is represented on the ROM when the ROM of both legs are getting approximately the same value and the Single Support Period (SSP) is when the ROM signals are getting different as can be seen Figure 20.

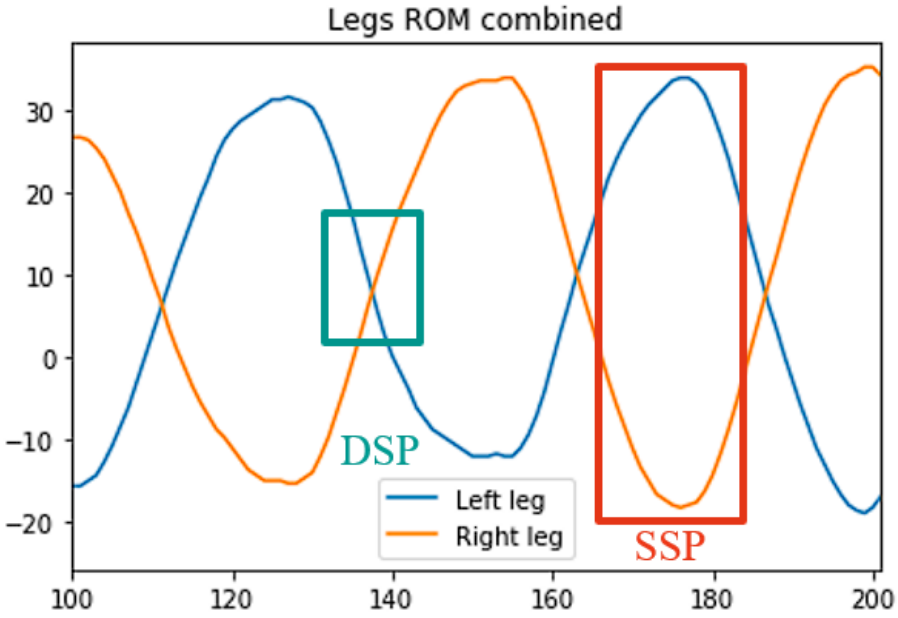


Figure 20: Single and Double Support ROM representation

This function, *calculateSingleStanceAndDoubleSupportTimes*, takes the times and the complete record of both legs. First it sums up the absolute values of both legs to get the total degrees of difference in the hip. It is made this way for avoiding the cases at which the subtraction of the original values results on zero just because the value of the swinging leg and the supporting leg are opposites.

After the sum is performed, it is needed to store the indexes at which the legs sum is less than 10 degrees, when the patient was on double support, and the indexes at which the sum is greater than 10 degrees, the single support period, on two different arrays.

Each array is runed over and the time difference, converted to seconds, of each period is saved in two new arrays. At last, the average single support and double support times are calculated and stored in two variables that will be returned.

The stance time, the time between initial contact and foot take off, is calculated based on the averages of the single and double support times. Since it is the time between the initial contact and the takeoff, it involves two double support times and one single support time.

Stride time calculation

The function *calculateStrideTime* calculates the average stride time, the time between two consecutive footfalls from the same foot, for both legs and the average using the simplified sequences, the time record, and the original ROM values at each leg.

First the average of the ROM value of both legs is calculated and then used for comparing with the values on the peaks, if the values is lower than the average values (meaning is a stance phase maximum), then the time for that peak is stored in a new array, *strideTimes*, one per leg. Then the time difference between all the values stored in *strideTimes* is calculated and the average stride time for each leg is calculated using such time differences and will be returned.

At last, the average stride time between both legs is computed and returned also.

Gait and stride speed calculation

In this case, the function *calculateGaitStrideSpeed* uses the stride time, the stride length, and the walked distance during the session, which was previously provided on the step and stride length calculation, for calculating these parameters.

First the distance and stride length are changed from meters to centimeters. The gait speed is the distance walked divided by the stride time and the stride speed is the stride distance divided also by the stride time.

Swing time

This function *calculateSwingTime* takes the step length and the gait speed as inputs and simply divides the length by the speed to get the swing time.

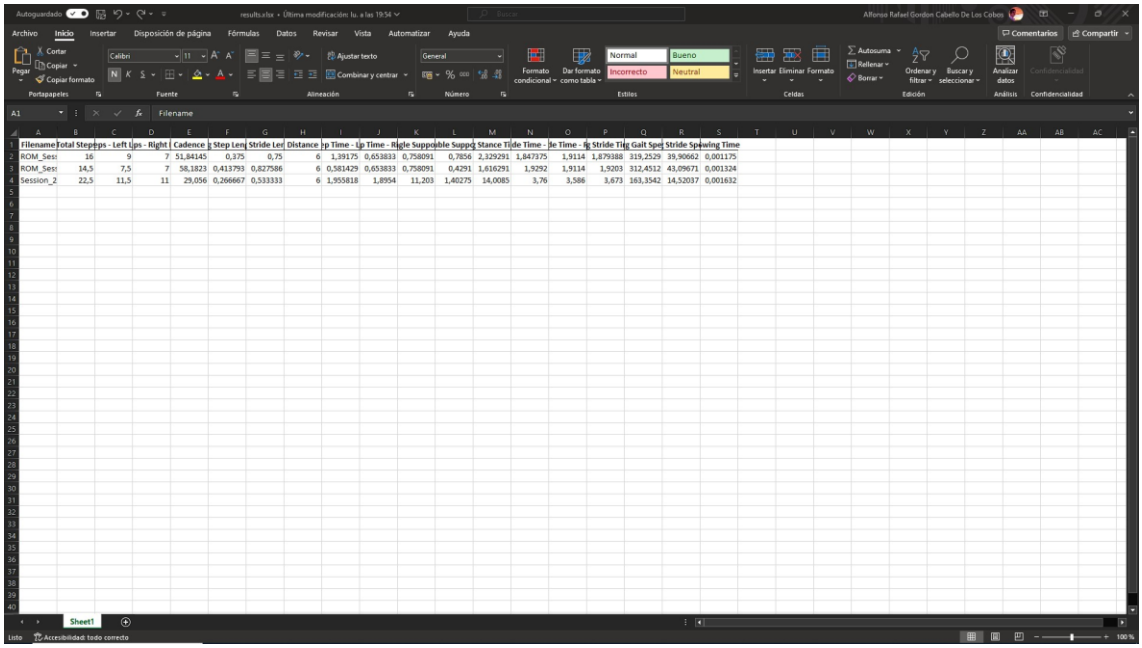
4.7 Saving and showing the results

After each iteration, when the results for each file are obtained a dictionary is created for associating the results of that file with a header indicating which parameter corresponds to which value. This dictionary will be appended to the data frame created at the beginning.

After all files are processed and appended to the data frame, the data frame is converted to a CSV and a XLSX file, so the physiologists have an easier way to see the data instead of having to look only at the console which have those results printed. The resulting files will be saved in a new directory named RESULTS with a name introduced by the user. The structure of the dictionary in the code and the result of a running example are showed in Figure 21 and Figure 22.

```
results = {
    'Filename': key,
    'Total Steps': totalSteps,
    'Steps - Left Leg': stepsL,
    'Steps - Right Leg': stepsR,
    'Cadence': cadence,
    'Avg Step Length': averageStepLength,
    'Avg Stride Length': averageStrideLength,
    'Distance': distance,
    'Avg Step Time - Left Leg': timesL,
    'Avg Step Time - Right Leg': timesR,
    'Avg Single Support Time': averageSst,
    'Avg Double Support Time': averageDst,
    'Avg Stance Time': averageStance,
    'Avg Stride Time - Left Leg': averageSTL,
    'Avg Stride Time - Right Leg': averageSTR,
    'Avg Stride Time': totalAvgST,
    'Avg Gait Speed': gaitSpeed,
    'Avg Stride Speed': strideSpeed,
    'Swing Time': swingTime
}
data_frame = data_frame.append(results, ignore_index=True)
```

Figure 21: Dictionary for giving format to the results file.



Filename	Total Steps	Steps - Left	Steps - Right	Cadence	Step Len	Stride Len	Distance	Stance Time - Left	Stance Time - Right	Stance Time - Double Support	Swing Time - Left	Swing Time - Right	Stride Time - Left	Gait Speed	Stride Speed	Swing Time
ROM_Sess	16	9	7	51,84145	0,375	0,75	6	1,39175	0,653833	0,758091	0,7856	2,329291	1,847375	1,9114	1,879388	319,2529	39,90662	0,001175
ROM_Sess	14,5	7,5	7	58,1823	0,413793	0,827586	6	0,581429	0,653833	0,758091	0,4291	1,616291	1,5292	1,9114	1,9203	312,4512	43,09671	0,001324
Session_2	22,5	11,5	11	29,056	0,266667	0,533333	6	1,955818	1,8954	11,203	1,40275	14,0085	3,76	3,586	3,673	163,3542	14,52037	0,001632

Figure 22: Final form of the results .xlsx format file.

5 RESULTS

The script was used on three different files, two from an elderly patient, referred as "Patient with pathological gait", the original and a new file with some noise data from the calibration deleted, and the other one from a healthy young patient, referred as "Patient with normal gait", that performed ten steps. The results are summed up in Table 2. The distance walked for both sessions was set to be six meters since it is the distance walked by both patients. On the side of the table the different gait parameters are located and each "Calculated" column represents a processed file. The "Real value" column is used as a reference point for the parameters' calculation. Unfortunately, only the steps performed by the patients at each leg were available.

	Patient with pathological gait			Patient with normal gait	
	Calculated Filtered	Calculated	Real value	Calculated	Real value
Total steps	14.5	16	14	22.5	20
Steps left leg	7.5	9	7	11.5	10
Steps right leg	7	7	7	11	10
Cadence [steps/min]	58.18	51.84		29.06	
Step length [cm]	0.41	0.38		0.27	
Stride length [cm]	0.83	0.75		0.53	
Step time left leg [s]	0.58	1.39		1.96	
Step time right leg [s]	0.65	0.65		1.9	
Single support time [s]	0.76	0.76		11.20	

Double support time [s]	0.43	0.79		1.40	
Stance time [s]	1.62	2.33		14.01	
Stride time left leg [s]	1.93	1.85		3.76	
Stride time right leg [s]	1.91	1.91		3.59	
Gait speed [cm/s]	312.45	319.25		163.35	
Stride speed [cm/s]	43.1	39.91		14.52	
Swing time [s]	0.0013	0.0012		0.0016	

Table 2: Results from execution using 15 meters as distance walked.

As it can be seen in the table above, the step counter approximated the steps with an error between 0.5, when comparing to the modified version of the pathological gait, and 2.5 steps, when comparing to the normal gait, depending on the signal. Also, the modified version of the elderly file, on the column "Calculated Filtered", shows an improvement on the steps calculated for the left leg. The modification on the steps leads to an increase in the parameters of the cadence, step length, stride length, step time on the left leg, gait speed and on the stride speed, and a decrease on the double support time, stance time and stride times for both legs. Some of these changes are very subtle like the increase on the step and stride length by a few millimeters while other parameters experience a great change like the gait or the stride speed that increments by almost ten units.

An interesting outcome is how the deletion of the noise from the calibration data of the first file is shown in the plots. The Figure 23 shows the original data and circled in red is the noise from the calibration and position of the patient. The Figure 24 shows the corrected file on the left leg, the one that changed the most.

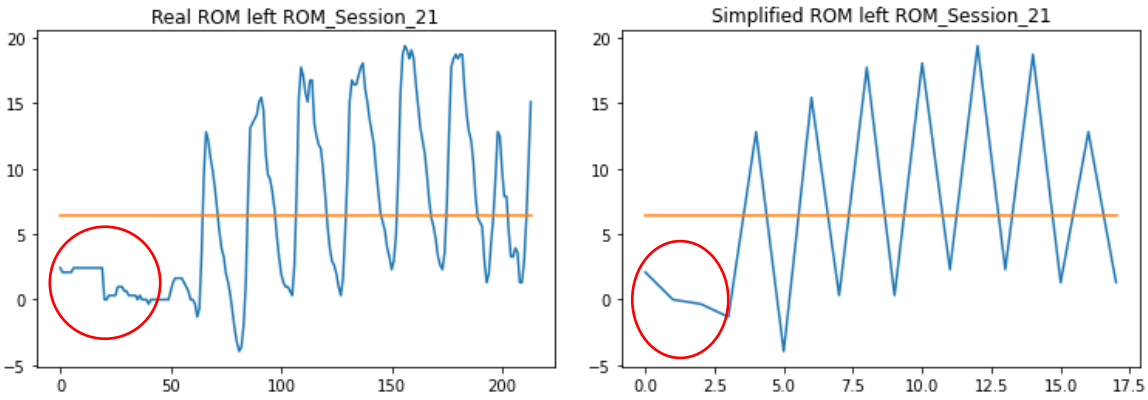


Figure 23: Original left leg draws from elderly patient's ROM.

As it can be seen, the simplified left leg sequence from the corrected file doesn't include the hills and the valley that are inside the red circle on the original file, which are responsible of adding those two more steps to the leg.

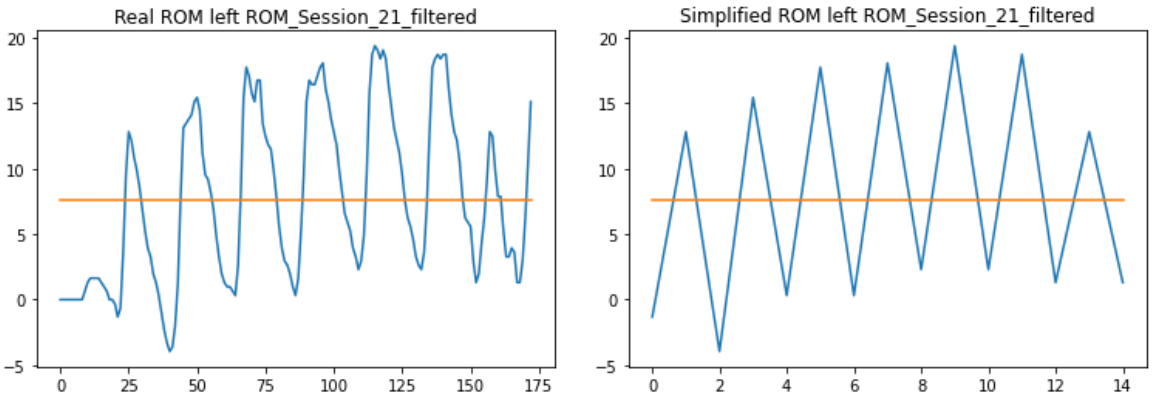


Figure 24: Left leg draws from filtered elderly patient's ROM.

6 DISCUSSION

In this section the results and some variations will be discussed among with the issues that this project had during its development.

This tool can correctly approximate the number of steps and most of the parameters except of the stride time and the parameters affected by it, whose calculation must be improved to be more precise but is a step in the right direction.

The main issue for this project were the lack of signals and the quality of the signal for the pathological gait patient. The lack of signals is due to a problem while recruiting patients for the study that was supposed to start on March 2023. The issue on the signal's quality is due to the drift that occurs when the patient is not correctly straight. During the sessions, due to the tiredness of the patient, it might sit on the harness instead of maintaining a straight posture, when this happens, the potentiometers start recording higher values than the ones supposed to be. This effect can be seen when comparing the signal from the elderly patient and the one of the young healthy one on Figure 25. The orange line indicates the 0.

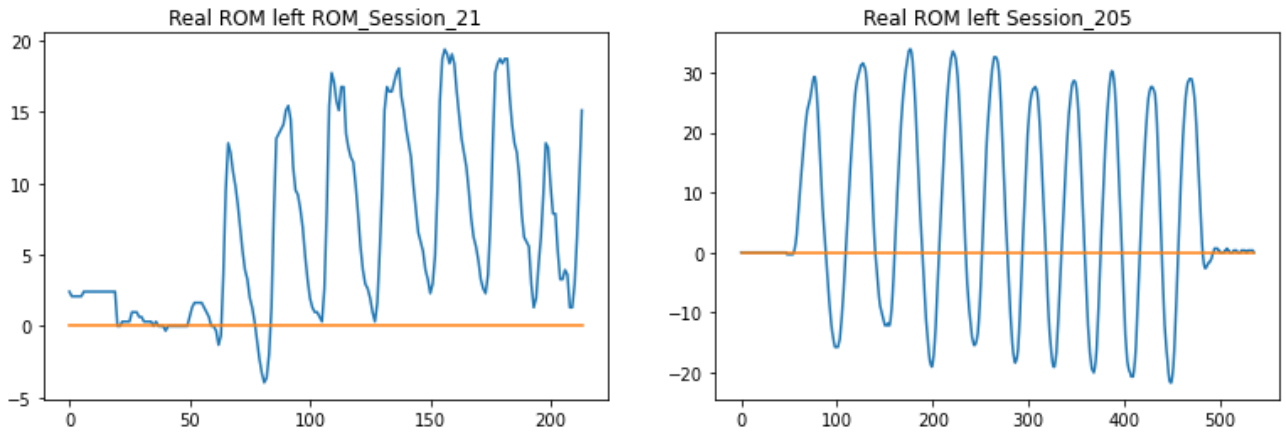


Figure 25: Differences on the recording of the signals.

The filter for deleting the extra hills and valleys in the middle of a step could be modified to work around 0 instead to work around the mean value but for this to work, the recording should start after the complete calibration and the patient must have remained straight for avoiding the mentioned effect on the signal. This should allow to perfectly calculate the steps of the patient on that session.

In case the distance is modified, the parameters affected would be the step and stride length, the gait and stride speed and the swing time. The Figure 26 shows how these parameters change among time using different files, the distances used went from 1 meter to 20 meters. This figure also shows how the swing time varies so slightly that is imperceptible.

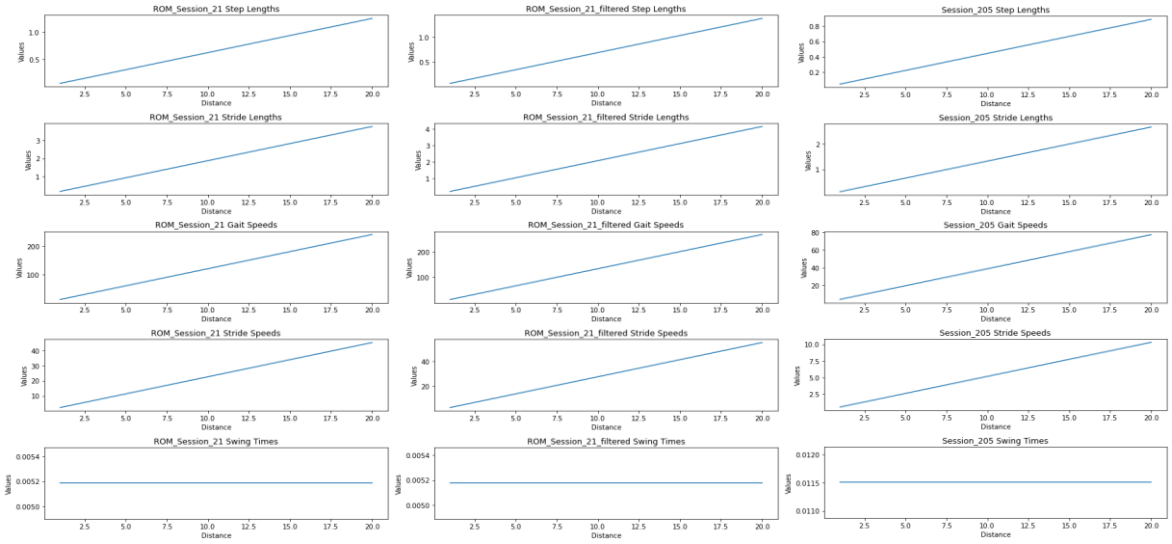


Figure 26: Changes of parameters over distance walked.

The program uses the distance walked by the patient for some parameters. In the actual version, the distance walked must be inserted manually but there are other options for using it. One of the options is to save it in a column inside the .xlsx files so it can be accessed like the rest of the information of the session, this way is the most precise but involves calculating or to measure such distance precisely. Another option is to calculate the distance based on the speed at which the SWalker is configured for the session based on the time the session lasted, this option also needs to store or know the speed of the session.

Another issue of the project is the inability of calculating the step width. This parameter is usually measured individually by using multiple wearable sensors [23], motion capture [24], computer vision [25] or by directly using different prints of the feet. An option for the calculation could be to use wearable sensors located along the leg [23] and approximate it based on the step length, but it is not viable in this project since we are no using IMU's.

7 CONCLUSIONS

This project was designed to generate a tool that could calculate the gait parameters from a ROM signal to provide a better assistance to the physiotherapists in charge of the hip fracture rehabilitation, one of the mayor health problems on elderlies. It can be concluded that the script is able to calculate most of the gait parameters with a decent accuracy.

Although this script seems to work well, there are some future works to be done to improve it. One thing would be to test it using more samples of ROM session, since this project only had two main files, a larger number of tests could reveal bugs or issues in the code that are not seen due to the lack of testing. Some updates could be how the distance walked is determined or a change in the filter of middle steps so the step detection can be increased.

Now the script can be used for the gait analysis of elderly patients using the SWalker robotic platform to check if they are improving their physical conditions but in a very rudimentary way. After the mentioned future work, the accuracy of the project will increase greatly, providing a better assistance to the physiotherapist which will result in a better and faster rehabilitation for the patient.

8 REFERENCES

[1] A. Kharb, V. Saini, Y. K. Jain and S. Dhiman, "A review of gait cycle and its parameters," *IJCEM,* vol. Vol. 13, pp. 78-83, July 2011.

[2] M. W. Whittle, Gait analysis an introduction, Fourth ed., Philadelphia, PA: ELSEVIER, 2007.

[3] W. Pirker and R. Katzenschlager, "Gait disorders in adults and the elderly," *The Central European Journal of Medicine,* 21 October 2016.

[4] A. Aboutorabi, M. Arazpour, M. Bahramizadeh, S. W. Hutchins and R. Fadayevatan, "The effect of aging on gait parameters in able-bodied older subjects: a literature review," *Aging Clin Exp Res,* 8 July 2015.

[5] M.-S. Kwon, Y.-R. Kwon, Y.-S. Park and J.-W. Kim, "Comparison of gait patterns in elderly fallers," *Technology and Health Care,* vol. Vol. 26, pp. S472-S436, 2018.

[6] R. Wiklund, A. Toots, M. Conradsson, B. Olofsson, H. Holmberg, E. Rosendahl, Y. Gustafson and H. Littbrand, "Risk factors for hip fracture in very old people:," *Springer: Osteoporosis International,* no. 27, pp. 923-931, 4 November 2015.

[7] J. Magaziner, E. M. Simonsick, T. M. Kashner, J. R. Hebel and J. E. Kenzora, "Survival Experience of Aged Hip Fracture Patients," *American Journal of Public Health,* pp. 274-278, March 1989.

[8] Physiopedia, "Range of Motion: Physiopedia," 2 March 2023. [Online]. Available: https://www.physio-pedia.com/Range_of_Motion.

[9] C. C. Norkin and D. J. White, Measure of Joint Motion: A guide to Goniometry, Fourth ed., Philadelphia, PA: F.A. Davis Company, 2009.

[10] Cleveland Clinic, "Hip Joint," 30 January 2023. [Online]. Available: https://my.clevelandclinic.org/health/body/24675-hip-joint.

[11] R. Delahunty, "Anatomy 101 - The hips," 29 May 2022. [Online]. Available: https://www.yogaru.ie/pause/anatomy-101-the-hips.

[12] M. Magone, E. Marinelli, G. Santilli, N. Finanore, F. Agostini, V. Santilli, A. Bernetti, M. Paoloni and S. Zaami, "Gait analysis advancements: rehabilitation value and new perspectives from forensic application," *European Review for Medical and Pharmacological Sciences*, pp. 3-12, 2023.

[13] M. Ramteke, "IMU Sensor – Working & Its Applications," 1 August 2022. [Online]. Available: https://www.semiconductorforu.com/imu-sensor-working-its-applications/.

[14] C. Büttner, T. L. Milani and F. Sichting, "Integrating a Potentiometer into a Knee Brace Shows High Potential for Continuous Knee Motion Monitoring," *MDPI*, vol. 21, no. 6, pp. 2150-2163, 19 March 2021.

[15] A. M. Chudyk, J. W. Jutai, R. J. Petrella and M. Speechley, "Systematic Review of Hip Fracture Rehabilitation Practices in the Elderly," *ELSEVIER: Archives of Physiscal Medicine and Rehabilitation*, vol. 90, no. 2, pp. 246-262, February 2009.

[16] V. Costa, O. Ramírez, L. Perea, A. Velásquez, A. Otero, E. Rocon and R. Raya, "Development and clinincal validation of a rehabilitation platform for hip fracture in elderly population".

[17] J. Zhou, S. Yang and Q. Xue, "Lower limb rehabilitation exoskeleton robot: A review," *Advances in Mechanical Engineering,* vol. 13, no. 4, pp. 1-17, 2021.

[18] D. Shi, W. Zhang, W. Zhang and X. Ding, "A review on lower limb rehabilitation exoskeleton robots," *Chinese Journal of Mechanical Engineering,* vol. 32, no. 74, pp. 1-11, 2019.

[19] MOVARD, "Productos: Lokomat," [Online]. Available: https://www.movard.es/productos/lokomat/.

[20] Cyberdyne, "What is HAL?," [Online]. Available: https://www.cyberdyne.jp/english/products/HAL/.

[21] A. Rodríguez-Fernandez, J. Lobo-Prat and J. M. Font-Llagunes, "Systematic review on wearable lower-limb exoskeletons for gait training in neuromuscular impairments," *Journal of NeuroEngineering and Rehabilitation,* 2021.

[22] B. Gavin, "How-To-Geek," 31 10 2022. [Online]. Available: https://www.howtogeek.com/392333/what-is-an-xlsx-file-and-how-do-i-open-one/.

[23] S. Díaz, S. Disdier and M. A. Labrador, "Step Length and Step Width Estimation using," *IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON),* pp. 997-1001, 2018.

[24] U. Rosenblum, G. Zeilig, Y. Bahat and L. Ks, "Novel methodology for assessing total recovery time in response to unexpected perturbations while walking," *Public Library of Science ONE,* vol. 15, no. 6, 3 June 2020.

[25] K. D. Ng, S. Mehdizadeh, A. Laboni, A. Mansfield, A. Flint and B. Taati, "Measuring Gait Variables Using Computer Vision to Assess Mobility and Fall Risk in Older Adults With Dementia," *IEEE Journal of Translational Engineering in Health and Medicine,* vol. 8, 28 May 2020.

9 ANNEX

```
# -*- coding: utf-8 -*-
"""
Analysis of biomechanical gait patterns in elderly subjects using a
robotic walker for hip fracture rehabilitation.

This script takes an .xlsx that stores the information from a ROM
signal
and calculates the gait parameters from it.

@author: Alfonso Rafael Gordon Cabello de los Cobos
"""

# Imported libraries
import glob

import pandas as pd
import os

import numpy as np

from scipy.signal import find_peaks

import matplotlib.pyplot as plt

"""
This function checks the data in the column from a patient, which is
the ROM from one of the legs, and return an array storing the index at
which maximums and minimums where detected.
"""
def peak_detection(leg, legId, key):
    #First we find all the peaks of the function
    maxPeaks, _ = find_peaks(leg, height=np.mean(leg), distance=10)

    #Since the find_peaks function can't detect the peaks
corresponding to the
    #minumuns
    #   the columns must be wsiped for finding them
    columnaGirada = leg * (-1)
    minPeaks, _ = find_peaks(columnaGirada,
height=np.mean(columnaGirada),distance=10)

    #The array will store the times of maximuns and minimuns in order
    peakOrderSequence = []
    actualMax = 0
    actualMin = 0
```

```
    actualCol = 0


    for actualCol in np.arange(len(leg)):
        if (len(peakOrderSequence) <= (len(maxPeaks) +
len(minPeaks))):

            if(actualCol == maxPeaks[actualMax]):
                peakOrderSequence.append(actualCol)
                if(actualMax < len(maxPeaks) - 1):
                    actualMax += 1

            if(actualCol == minPeaks[actualMin]):
                peakOrderSequence.append(actualCol)
                if(actualMin < len(minPeaks) - 1):
                    actualMin += 1

    #Some peaks might be errors while getting the ROM signal
    #   that are detected by find_peaks so they must be filtered by
getting
    #   only the peaks that are above the average
    media = np.mean(leg)
    upperBound = media + 2
    lowerBound = media - 2
    realMax = False
    realPeakSequence = []
  #     saco el primero
    for picos in np.arange(len(peakOrderSequence)):
        if (leg[peakOrderSequence[picos]] > upperBound or
leg[peakOrderSequence[picos]] < lowerBound):
            if(leg[peakOrderSequence[picos]] > media):
               realMax = True
            cummulatedPeaks = []
            cummulatedPeaks.append(peakOrderSequence[picos])

            for picosPosteriores in np.arange(picos + 1,
len(peakOrderSequence)):
                if((leg[peakOrderSequence[picosPosteriores]] > media)
== realMax):

cummulatedPeaks.append(peakOrderSequence[picosPosteriores])
                else: break

            if(realMax == True):
                realPeakSequence.append(max(cummulatedPeaks))
            else: realPeakSequence.append(min(cummulatedPeaks))
            realMax = False
```

```
    x_axis = np.arange(len(leg))
    x_axis1 = np.arange(len(realPeakSequence))
    plt.figure()
    plt.title("Real ROM " + legId + key)
    plt.plot(x_axis, leg)
    y = [media] * len(x_axis)
    plt.plot(x_axis, y)

    plt.figure()
    plt.title("Simplified ROM " + legId + key)
    plt.plot(x_axis1, leg[realPeakSequence])
    y = [media] * len(x_axis1)
    plt.plot(x_axis1, y)

    return realPeakSequence



"""
This function gets all the ROM values from a patient stored in the
dictionary

"""
def getDataPatient(key):
    leftLeg = dictionaryRom.get(key).iloc[:,3].values
    rightLeg = dictionaryRom.get(key).iloc[:,4].values
    times = dictionaryRom.get(key).iloc[:,2].values
    return leftLeg, rightLeg, times



"""
This function calculates the number of steps a patient had performed.
Since the functions SecuenciaPicos and segundaSecuenciaPicos store in
an array
at which moment are the Max and Min values of the hip ROM we can
calculate
the number of steps just by dividing the length of the final array by
2 since
each pair of Max and Min corresponds to an step.

"""
def calculateSteps(gaitSimplifiedSequenceLeft,
gaitSimplifiedSequenceRight):
    stepsL = len(gaitSimplifiedSequenceLeft) / 2
    stepsR = len(gaitSimplifiedSequenceRight) / 2
```

```
    totalSteps = stepsL + stepsR

    return totalSteps, stepsL, stepsR


"""
This function calculates the average cadence, number of steps made by
minute,
by getting the number of steps and dividing it by the time of the
session

"""
def calculateCadence(steps, times):
    cadence = steps / ((times[-1] - times[0]) / 1000) #[steps/s]
    finalCadence = cadence * 60 #The time is recorded in seconds and
the candence
                                # is usually calculated in
steps/minute.
    return finalCadence #[steps/min]

"""
This function calculates the average step length by dividing the
distance walked, which is asked to the therapist, by the number of
steps
made by the patient.
Also the average stride length is calculated by the sum of 3 steps
since is
the distance between two consecutive steps of the same foot.

"""
def calculateStepAndStrideLength(steps):
    while True:
        distance_txt = input("Insert the walked distance in
meters:\n") #[m]

        if distance_txt.isdigit() and int(distance_txt) > 0:
            distance = int(distance_txt)
            break
    averageStepLength = distance / steps
    averageStrideLength = averageStepLength * 2
    return averageStepLength, averageStrideLength, distance

"""
This function calculates the times between each step. Since the data
stored in
gaitSimplifiedSequence are the time of peak of the stance phase
followed by
```

```
the peak of the swing phase, the we must only take into acount the
stance peaks
for calculating the time.

"""
def calculateStepTime(gaitSimplifiedSequenceLeft,
gaitSimplifiedSequenceRight, time):
    timesL = []
    timesR = []
    for i in range(len(gaitSimplifiedSequenceLeft) - 2):
        if i < len(gaitSimplifiedSequenceLeft) - 2 and i % 2 == 0:
            timesL.append((time[gaitSimplifiedSequenceLeft[i + 1]] -
time[gaitSimplifiedSequenceLeft[i]]) / 1000) #ms to s

    for i in range(len(gaitSimplifiedSequenceRight) - 2):
        if i < len(gaitSimplifiedSequenceRight) - 2 and i % 2 == 0:
            timesR.append((time[gaitSimplifiedSequenceRight[i + 1]] -
time[gaitSimplifiedSequenceRight[i]]) / 1000)

    averageTimesL = sum(timesL)/len(timesL)
    averageTimesR = sum(timesR)/len(timesR)

    return averageTimesL, averageTimesR

"""
This function calculates the single and double support times using
the patient's whole data. First we calculate the total degrees by
summing the
absolute values from the left and right legs. Then we get the indexes
at wich
this sum is less or is higher or equal to 10° for classifying into two
different
list. The stance time contains a full Single support and two double
support periods

"""
def calculateSingleStanceAndDoubleSupportTimes(leftLeg, rightLeg,
times):
    sst = []
    dst = []
    averageSst = averageDst = 0

    legDifference = abs(leftLeg) + abs(rightLeg)

    indexListDst = [i for i in range(len(legDifference)) if
legDifference[i] < 10]
```

```
    indexListSst = [i for i in range(len(legDifference)) if
legDifference[i] >= 10]

    start = None
    end = None

    for i in range(len(indexListDst)-1):
        if indexListDst[i]+1 == indexListDst[i+1]:
            if start is None:
                start = i
            end = i+1
        elif start is not None:
            dst.append((times[indexListDst[end]] -
times[indexListDst[start]]) / 1000)
            start = None
            end = None

    start = None #sometimes the variables dont reset after finishing
with the calculation of the DST
    end = None

    for j in range(len(indexListSst)-1):
        if indexListSst[j]+1 == indexListSst[j+1]:
            if start is None:
                start = j
            end = j+1
        elif start is not None:
            sst.append((times[indexListSst[end]] -
times[indexListSst[start]]) / 1000)
            start = None
            end = None

    averageSst = sum(sst)/len(sst)
    averageDst = sum(dst)/len(dst)
    averageStance = averageDst * 2 + averageSst


    return averageSst, averageDst, averageStance

"""
This function calculates the stride time.

"""
def calculateStrideTime(gaitSimplifiedSequenceLeft,
gaitSimplifiedSequenceRight, leftLeg, rightLeg, times):
    averageSTL = averageSTR = 0
    strideTimesLeft = []
```

```
    strideTimesRight = []
    avgRomL = sum(leftLeg) / len(leftLeg)
    avgRomR = sum(rightLeg) / len(rightLeg)

    for i in range(len(gaitSimplifiedSequenceLeft) - 1):

        if leftLeg[gaitSimplifiedSequenceLeft[i]] < avgRomL:

strideTimesLeft.append((times[int(gaitSimplifiedSequenceLeft[i])] -
times[0]) / 1000)

    for i in range(len(gaitSimplifiedSequenceRight)- 1):
        if rightLeg[gaitSimplifiedSequenceRight[i]] < avgRomR:

strideTimesRight.append((times[int(gaitSimplifiedSequenceRight[i])]  -
times[0]) / 1000)

    differenceStrideTimesLeft = []
    differenceStrideTimesRight = []

    for j in range(len(strideTimesLeft) - 2):
        differenceStrideTimesLeft.append(strideTimesLeft[j + 1] -
strideTimesLeft[j])

    for j in range(len(strideTimesRight) - 2):
        differenceStrideTimesRight.append(strideTimesRight[j + 1] -
strideTimesRight[j])

    averageSTL = sum(differenceStrideTimesLeft) /
len(differenceStrideTimesLeft)
    averageSTR = sum(differenceStrideTimesRight) /
len(differenceStrideTimesRight)
    totalAvgST = (averageSTL + averageSTR)/2

    return averageSTL, averageSTR, totalAvgST

"""
This function calculates the stride and gait speed. It takes the
Stride time,
length and distance, which are already calculated in previous
functions, and
changes the units to cm before performing the calculations.

"""
def calculateGaitStrideSpeed(stride_time, stride_length, distance):
    newDistance = distance * 100 #[m] to [cm]
    newSL = stride_length * 100
```

```
    gaitSpeed = newDistance / stride_time
    strideSpeed = newSL / stride_time
    return gaitSpeed, strideSpeed


"""
This function calculates the swing time. It can be calculated by
dividing the
step length by the speed since time = space/speed

"""
def calculateSwingTime(stepLength, gaitSpeed):
    swingTime = stepLength / gaitSpeed
    return swingTime


#-------------------------------------------------------------------
------------------------------

# Preparing the data access
# Asking for the directory for extracting the ROM signals
path = input("Where are the ROM signals stored?\n")

# Takes every xlsx on the folder, each xlsx is a ROM signal

so = input("Which OS are you using?\n1. Windows\n2. Linux\n")
if so == 1:
    files = glob.glob(path + "\*.xlsx")
else:
    files = glob.glob(path + "/*.xlsx")

dictionaryRom = {}

data_frame = pd.DataFrame()

# Storing the data of each ROM file into the dictionary,
#   the key is only the filename and the result is the dataset.
for filename in files:
    dictionaryRom[filename[(len(path) + 1):(len(filename) - 5)]] =
pd.read_excel(filename)
    #filename[(len(path) + 1):(len(filename) - 5)] isolates only the
name of
    #   the file (excluding directory and file sufix)
    #There are multiple columns per patient. Column 3 stores the time
sequence, 4 stores the Left Hip ROM,
    #   Column 5 stores Right Hip ROM and Column 6 stores the Weight
```

```
#Calculates the gait parameters for every patient on each leg
for key in dictionaryRom.keys():
    print("-----------------" + key + "-----------------\n")
    leftLeg, rightLeg, times = getDataPatient(key)
    simplifiedLeftLeg = peak_detection(leftLeg, "left ", key)
    simplifiedRightLeg = peak_detection(rightLeg, "right ", key)

    plt.figure()
    plt.title("Legs ROM combined")
    plt.plot(leftLeg, label = "Left leg")
    plt.plot(rightLeg, label = "Right leg")
    plt.legend()
    plt.xlim(100, 201)

    totalSteps, stepsL, stepsR = calculateSteps(simplifiedLeftLeg,
simplifiedRightLeg)
    print("Total steps: " + str(totalSteps) + "\n" + "Total left leg
steps: " + str(stepsL) + "\n" + "Total right leg steps: " +
str(stepsR) + "\n" )

    cadence = calculateCadence(totalSteps, times)
    print("Cadence [steps/min]: " + str(cadence) + "\n")

    averageStepLength, averageStrideLength, distance =
calculateStepAndStrideLength(totalSteps)
    print("Average Step Length [cm]: " + str(averageStepLength) + "\n"
+ "Average Stride Length [cm]: " + str(averageStrideLength) + "\n" +
"Distance [m]: " + str(distance) + "\n" )

    timesL, timesR = calculateStepTime(simplifiedLeftLeg,
simplifiedRightLeg, times)
    print("Average Step time left leg [s]: " + str(timesL) + "\n" +
"Average Step time right leg [s]: " + str(timesR) + "\n")

    averageSst, averageDst, averageStance =
calculateSingleStanceAndDoubleSupportTimes(leftLeg, rightLeg, times)
    print("Average Single Support time [s]: " + str(averageSst) + "\n"
+ "Average Double Support time [s]: " + str(averageDst) + "\n" +
"Average Stance time [s]: " + str(averageStance) + "\n" )

    averageSTL, averageSTR, totalAvgST =
calculateStrideTime(simplifiedLeftLeg, simplifiedRightLeg, leftLeg,
rightLeg, times)
    print("Average Stride time left leg [s]: " + str(averageSTL) +
"\n" + "Average Stride time right leg [s]: " + str(averageSTR) + "\n"
+ "Average Stride time [s]: " + str(totalAvgST) + "\n")
```

```
    gaitSpeed, strideSpeed = calculateGaitStrideSpeed(totalAvgST,
averageStrideLength, distance)
    print("Gait speed [cm/s]: " + str(gaitSpeed) + "\n" + "Stride
speed [cm/s]: " + str(strideSpeed) + "\n")

    swingTime = calculateSwingTime(averageStepLength, gaitSpeed)
    print("Swing time [s]: " + str(swingTime) + "\n")

    results = {
        'Filename': key,
        'Total Steps': totalSteps,
        'Steps - Left Leg': stepsL,
        'Steps - Right Leg': stepsR,
        'Cadence': cadence,
        'Avg Step Length': averageStepLength,
        'Avg Stride Length': averageStrideLength,
        'Distance': distance,
        'Avg Step Time - Left Leg': timesL,
        'Avg Step Time - Right Leg': timesR,
        'Avg Single Support Time': averageSst,
        'Avg Double Support Time': averageDst,
        'Avg Stance Time': averageStance,
        'Avg Stride Time - Left Leg': averageSTL,
        'Avg Stride Time - Right Leg': averageSTR,
        'Avg Stride Time': totalAvgST,
        'Avg Gait Speed': gaitSpeed,
        'Avg Stride Speed': strideSpeed,
        'Swing Time': swingTime
    }
    data_frame = data_frame.append(results, ignore_index=True)


# Create the full path to the new directory
new_directory_path = os.path.join(path, "RESULTS")

# Create the new directory if it doesn't already exist
if not os.path.exists(new_directory_path):
    os.makedirs(new_directory_path)

#Ask for the file name to the user and creating a CSV file with that
name
file_name = input("Name of the file for storing the results: ")
result_file = file_name + '.csv'

# Save results to CSV file
csv_file = os.path.join(new_directory_path, result_file)
```

```
data_frame.to_csv(csv_file, index=False)

#Changing file tipe to XLSX
result_file = file_name + '.xlsx'

# Save results to XLSX file
xlsx_file = os.path.join(new_directory_path, result_file)
data_frame.to_excel(xlsx_file, index=False)

print("A CSV and XLSX file was generated into the directory " +
new_directory_path)

"""
TODO
Remaining:
    •  Step width [cm]: Lateral distance from heel center to line of
progression. Line of progression is formed as the line formed by two
consecutive footprints of the opposite foot.

"""
```