ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF MECHANICAL AND INDUSTRIAL ENGINEERING

Classification Algorithm: Application to Orthotic Bracing

A thesis submitted for the partial fulfillments of MSc Degree in Mechanical Engineering, Mechanical Design specialization, in the School of Mechanical and Industrial Engineering, Addis Ababa Institute of Technology, Addis Ababa University, Ethiopia

Advisor: Dr. Demoz G.

October, 2016

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF MECHANICAL AND INDUSTRIAL ENGINEERING

Development and Performance Analysis of Gait Classification Algorithm : Application to Orthotic Bracing

By : Tewedage Sileshi

A thesis submitted for the partial fulfillments of Msc Degree in Mechanical Engineering, Mechanical Design specialization, in the School of Mechanical and Industrial Engineering, Addis Ababa Institute of Technology, Addis Ababa University, Ethiopia

Advisor : Dr. Demoz G.

October, 2016

Acknowledgment

First and foremost, I would like to thank God for his never ending support. Next, I would like to thank my advisor, Dr. Demoz G., who shaped my research structure and provided me with the kind instruction and comments. I would also like to gives thanks to my advisor and his colleagues in Minnesota University for providing me with the IMU experimental data. I am also gratified for James R. Gage Center for Gait and Motion Analysis at the Gillette Children's Hospital in St. Paul, Minnesota, for letting me use the Vicon data that was collected by them. I am also thankful to Addis Ababa University, institute of technology and School of Mechanical and Industrial engineering for the opportunity given to study and facilities provided to me. I profoundly would want to given by gratitude to my family and friends who give me love and support all through my life, and enable me to reach where are am today. Finally, I would like to give thanks to all who have helped in the preparation of the paper.

Abstract

Orthotic bracing are fitted to improve the quality of a deviating gait cycle, nevertheless recent studies show that gait cycle deviations are seen due to bracing [4, 36]. In order to correct the deviation, adjustments should be made in all stages of the process to which it requires evaluating the deviation. So the objective of this paper is to develop and analyze the performance of an algorithm for gait classification, to assess the deviation of gait cycle due to Orthotic bracing.

Acceleration data was taken as an input from Inertial Motion Sensor attached at the foot of a healthy subject. The algorithm was developed using Model based approach, by considering four machine learning classifiers. The performance of the algorithm was analyzed using the performance indicators.

The performance analysis illustrates that using categorical class type gives a relatively higher accuracy with a maximum accuracy of 86%. The performance related to the classifier on the other hand illustrates that Tree Bagger Classifier and Support Vector Machine classifier provide a higher accuracy with a balanced true rate. The analysis further indicate that the gait cycle of the braced side of an Ankle bracing is relatively expressed using the extracted features, it was expressed on average 58%. While the effect of the bracing on the unbraced side of the leg are shown more with Knee bracing.

Keywords: Gait classification, Feature extraction, Inertial motion sensor, Machine learning, Orthotic bracing, Performance analysis, Confusion matrix, hold out cross valdation.

Table of content

List of Tables

List of Figures

Chapter 1

Introduction

The manner of a person's movement, specifically during walking is called gait [17]. When the human gait becomes biomechanically inefficient it is described as abnormal or stated as a deviated gait [34]. One of the techniques used to correct these deviations is using Orthotic brace. Orthotic braces, or orthoses, are used to provide support to a weakened body part or joint. It is an external device with controlling forces to improve body alignment, improve function, immobilize the injured area, prevent or improve a deformity, protect a joint or limb, limit or reduce pain, and/or provide proprioceptive feedback [1]. Over 25 million people living in developing countries require the use of prosthetic and orthotic devices [2], among these over 800,000 are living in Ethiopia [3]. Furthermore, a replacement or adjustment is needed when the bones of the patients grow with time.

In order to have a proper fit of the brace a close working relationship between the brace and the patient is required [2]. Even though to create a proper fit of the braces are custom made after analyzing the patients problem, gait deviation is seen due to bracing [4]. In order to correct the deviation the parameter that needs to be corrected should be identified. This requires classifying the distinguishing factors of the normal gait and the pathological gait.

Recently, researches have used machine learning methods to classify data set, based on a set of features that could properly distinguish the data sets [5, 6]. If a classification with high accuracy is found based on the selected features, the distinguishing factor for the data set is found. In the case of human gait if a feature could be found that accurately distinguish normal gait from pathological gait, then deviation parameter could be taken from that feature set [7].

In addition, to the selected feature set a proper classification takes place using the acquired data. Traditionally gait data are acquired by semi-subjective methods, carried out by specialists who observe the quality of the patients gait; this is sometimes followed by survey in which patients are asked the quality of their gait [8]. This type analysis depends on the ability of the specialists; even with the most able one it is difficult to quantify the unique gait parameters that brought the deviation gait of the individual.

Advancement in technology has lead researchers to come up with sensor based method to measure gait; these sensors could be non-wearable or wearable. Non-wearable sensors require laboratories and can be expensive, which has lead researchers to focus on wearable sensors. Wearable sensors are, highly transportable, do not require stationary units, and, therefore, can be used outside laboratory conditions [8-10]. This makes them a good choice for monitoring patient's rehabilitation progress in everyday environment. One recently available technology for wearable sensor are inertial measurement units (IMU), which normally contain a triad of gyroscopes and accelerometers.

The IMU location on the human body influences the robustness and accuracy of the gait events identification [11]. As a general rule, the closer the IMU is to the point of impact, the higher are the chances of correctly detecting the Gait events [12]. The most intuitive solution would be to place the IMU on the foot.

Studies have been conducted to classify normal gait and pathological gait based on a data obtained from IMU sensor[6, 13, 14]. Although the studies show a promising accuracy based on

the features that are selected, that might not be the case for different pathological gaits. This is because gait deviations are different for each type of pathology.

1.1 Problem statement

A patient needs a proper designed orthotic brace for correcting the specific defect of the gait. In addition, proper recovery of walking ability requires a close working relationship between the designed orthotic and the gait motion [15]. Even though the braces are expected to help the patient in regaining this proper walking ability, studies show that gait deviation are seen due to the bracing [4]. In order to correct the deviation, adjustments should be made in all stages of the process including design, fitting and rehabilitation. This requires finding the parameters that created the deviation.

In order to find the deviating parameters, proper analysis of the both the normal gait and the deviated gait is required. It is also important to understand the similarities and differences between the two states of the gait. One way to understand this things is to classify the two data sets based on situated features. This means selecting features that can lead to the deviating parameters or gait cycle. If accurate classification (selected features) is accomplished, then the features can be taken for the identification of the deviating parameters.

A survey of the literature shows that classifying a data accurately requires to acquire data properly and to select the features carefully [5-7, 14]. One of the important qualities of the acquired data is its ability to properly analysis the gait. Next is its suitability for monitoring patient's rehabilitation progress in everyday environment. As introduced previously this

characteristics are all found in inertia measurement unit (IMU) sensors as proven by literature [8, 16].

Studies are conducted to classify normal gait and pathological gait based on a data obtained from IMU sensor and identifying distinguishing features [6, 13, 14]. This shows a promising way for other type of pathological gait distinguishing feature, since different pathologies show different type of deviation. In the work described in this thesis, a set of features are tested that are assumed to distinguish the gait deviation due to orthotic bracing to normal gait. The set of features are tested based on classification of the data sets.

The paper aim in answering the following research questions:

1. What fitting model is appropriate to eliminate the incorporated noise from a raw gait data; Without losing important gait cycle patterns ?
2. Which class type from the ones used for the research enhance the performance of the algorithm ?
3. Which classifier model from the ones used for the research is accurate in classifying the set of gait cycles ?
4. Which bracing from the ones used for the research is better depicted by the set of features ?

1.2 Objective of the study

1.2.1 General objective

The general objective of this research is to develop and analyze the performance of an algorithm for gait classification for orthotic bracing gait cycle.

1.2.2 Specific objective

To attain the general objective the following set of specific objectives are identified. These are;

- Fitting a suitable fitting model to eliminate the noise of the raw gait data
- Selecting features that could distinguish the different bracing
- Performance analysis depending on class type, type of classifier and type of bracing

1.3 Significance of the study

The outcome of this research will contribute to the body of knowledge in gait analysis in general and specifically in orthotic bracing studies. The selected model for curve fit would be used as a method of minimizing noise that is found from data acquisition from sensors. Taking the extracted feature set the research will aid orthotic bracing designers, and physical assistances, by pointing to a way that could lead to the improvement of the gait cycle of the patients. A related contribution of the study is the use of IMU sensor for gait analysis purpose. In addition, the developed algorithm could be used as a reference by other sectors. That is the finding related to the accuracy of the classifier and class type used will serve as path for different classifications.

1.4 Scope of the study

The focus of the research is to understand the basic deviation of orthotic braced gait to that of the normal gait. In order to see the effect of the Orthoses, without adding the effect of a specific abnormality, data was taken from a healthy subject. The experiment was conducted in Minnesota University because of better access to data collection equipments. The distinguishing features are selected based on a set of observational assumptions. The quality of the selected features is rated based only by the performance analysis of the classification algorithm. The research covers three type of orthotics bracing these are; Ankle bracing, Knee bracing and Ankle & Knee bracing.

1.5 Limitation of the study

Since classification of time series data set is recently developing, proper distinguish between different tools and there accuracy is yet to be studied. Therefore, the study only used basic machine learning classification algorithm. In addition, methods that could improve accuracy of the result are not investigated. The other limitation was the availability of related research on the gait deviation due to orthotic brace. This created a constraint for cross checking the accuracy of the result found on the study. Also, the sample was taken from one subject where the size of each trial shows a difference since the walking cover distance is different for each trail.

1.6 Thesis Organization

The organization of this thesis is as follows:

- Chapter 2 discusses basics of gait analysis and partitioning techniques. In addition, it summarizes recent literatures by stating its findings and gaps.
- Chapter 3 present the research methodology used in the study.
- Chapter 4 contains the detail explanation of the develop algorithm.
- Chapter 5 includes the performance evaluation of the classification algorithms based on a set of performance measurements. The effect of the type of class, the type of classifier and the type of bracing used on the performance is also evaluated.
- Chapter 6 gives a concise conclusion and indicates future research areas.

Chapter 2

Literature Review

2.1 Introduction

The manner of a person's movement, specifically during walking is called gait [17]. Human motion is very dynamic, meaning we are capable of changing direction and velocity abruptly [18], which makes the analysis difficult. The analysis has been conducted as early as the 19thcentury with the advent of motion camera systems [10]. This chapter first reviews the need for the gait analysis and the techniques used for analysis, section 2.2. The next section focuses on the basis for the gait analysis, which include basic gait deviations, focusing on the specific deviation studied in this thesis, section 2.3. In addition, it reviews researches on gait classification algorithms, by first reviewing techniques used to identify patterns in a time-series data set, section 2.4. The methods for evaluating the performance of these algorithm is also reviewed, section 2.4.3. Finally, the research gap and the focus of the thesis are noted in the conclusion.

2.2 Gait Analysis

Gait analysis is the systematic study of human walking. Gait analysis could be kinematic-based, kinetics-based or EMG-based. The kinematics of the human gait describes the movements of the major joints and components of the lower extremity in the human body. Gait kinetics focuses on

the study of forces and moments that result in the movement of human segments, in which the orientation of all the leg segments obtained from gait kinematics is often required. The EMG-based analysis of the human gait is used to detect and analyze muscle activity during human walking [10].

2.3.1 Application of gait analysis

There are a wide range of applications; it is an effective clinical tool for orthopedic treatment planning, outcome assessment, and longitudinal studies on maintenance and progress [19]. In orthopedics and rehabilitation, gait analysis is used to monitor the patient healing progress [10]. In order to evaluate the recovery status in patients after interventions or rehabilitation treatments the correct discrimination of gait phases is required [20].

Distinguishing between normal and pathological gait also require proper gait phase granularity [21]. In healthcare monitoring, gait analysis based on wearable sensors can also be applied in various occasions, such as in the detection of gait abnormalities, the assessment of recovery, fall risk estimation, and so on [10].

Research related to human walking has also been done by the medical community for purposes of prosthetic limb design, joint replacement technology and treatment of neuromuscular disorders [18]. In addition, analysis of skill level and locomotors performance of athletes or patients, Ambulatory measurement to monitor patients' daily activities, Clinical assessment for patients, Identification of different daily activities, requires gait analysis [9].

It is also applicable for personal navigation systems that can be used in general devices, to assist the blind or visual impaired, post-surgery monitoring, law enforcement and military personnel[18]. Even though, several studies have been conducted on gait analysis, much of the effort to date has been focused on healthy subjects [22]. This suggests that the medical sector needs more research targeting unhealthy subjects.

2.3.2 Techniques used for gait analysis

Different techniques have been used to characterize gait. The traditional scales used to analyze gait parameters in clinical conditions are semi-subjective. This is sometimes followed by a survey in which the patient is asked to give a subjective evaluation of the quality of his/her gait. The advantage of these methods is that they do not require special equipment and only need a trained specialist to carry out the test. However, the subjective nature of the evaluation affects the accuracy, exactitude, repeatability and reproducibility of the measurements [8]. Even though, the method is less expensive and still employed in most developing countries, the accuracy of the result depends on the ability of the specialist. Even with the most able specialist it is somewhat difficult to be sure with the result.

This leads to the need to develop an alternative method in aiding the clinical procedure. Progress in new technologies has given rise to devices and techniques which allow an objective evaluation of different gait parameters, resulting in more efficient measurement and providing specialists with a large amount of reliable information on patients' gaits. This reduces the error margin caused by subjective techniques[8].

These devices used can be classified as non-wearable (NWS) and wearable sensors[18][8]. Techniques such as threshold filtering which converts images into black and white, the pixel count to calculate the number of light or dark pixels, or background segmentation which simply removes the background of the image, are just some of the possible ways to gather data to measure the gait variables. This method has been widely studied in order to identify people by the way they walk [23].

Begonya and colleagues, conducted a literature review and conclude that although all gait analysis methods can be used for general analytical purposes, when higher accuracy is needed in the detection and analysis of more specific parameters, it is necessary to choose the adequate method. The approaches that allow simultaneous, in-depth analysis of a higher number of parameters are the NWS systems on a laboratory environment, and more specifically those which are based in a combination of several of the described techniques, such as marker or marker-less based image processing, EMG, inertial and floor sensors. However, the latest developments in WS allow cost-effective, non-intrusive methods which offer convenient solutions to specific analytical needs. WS systems make it possible to analyze data outside the laboratory and capture information about the human gait during the person's everyday activities[8].

A standard gait analysis method based on the multi-camera motion capture system and force platform with the capability of measuring ground-reaction forces was successfully developed and applied in a number of gait laboratories. However, this standard gait analysis requires specialized locomotion laboratories, expensive equipment, and lengthy set up and post-processing times. To eliminate this limitation, an alternative gait analysis method based on wearable sensor has begun

to be employed [10]. Since, the environment which the patient encounters affects the prosthetic/orthoses device proper fit. The testing environment must be similar with the patient's everyday environment [18]. Due to this the need for using wearable sensor for gait phase analysis for prosthetic/orthoses device fitting is high.

Wearable Sensors

Wearable sensors are motion sensors are worn or attached to various parts of the patient's body, such as the foot and waist. The movement signal recorded by these sensors can be used to perform the gait analysis [10]. Nowadays, wearable sensors are used for gait granularity; foot pressure insoles or foot switches, accelerometer, gyroscopes, inertial measurement unit (IMU) and electromyography (EMG) are used supplied to an algorithm for gait segmentation [16], which are described below:

1. Foot switches and Foot pressure

Nowadays, the golden standard for gait phase detection is footswitches and foot pressure insoles, since it directly detect the foot contact with the ground. In addition, it is low cost, requires simple signal conditioning and post processing, and it is gives accurate detection of phases [16]. However, the number of detected gait phases is limited and the sensor placement on pathological gait affects its accuracy [24].

2. Accelerometer

An accelerometer is a type of inertial sensor that can measure acceleration along its sensitive axis. The common operation principle of accelerometers is based on a mechanical sensing element that comprises a proof mass attached to a mechanical suspension system, with respect to a reference frame [10].An accelerometer basically uses the fundamentals of Newton's Laws of Motion, which say that the acceleration of a body is proportional to the net force acting on the body [8].By attaching these accelerometers to the feet or legs, the acceleration/velocity of the feet or legs in the gait can be determined to perform the gait analysis[25].

The basic criteria of accelerometer is the placement position. After reviewing four papers differing in targeted body segments and the sensor positioning, the sagittal acceleration of the foot was found to be the optimal choice to obtain the best results, regardless the given computational methodology. Moreover, the use of the resultant of the acceleration was a valid suggestion to eliminate the effect of an incorrect sensor placement on the body segment[16].

3. Gyroscope

In order to eliminate the influence of gravity in acceleration measurement and the effect of vibration during heel strike of the waveform angular velocity measurement using gyroscope is being employed. A gyroscope is an angular velocity sensor. The micro-machined gyroscope is based on the concept of measuring the Coriolis force, which is an apparent force proportional to the angular rate of rotation in a rotating reference frame. By detecting the linear motion from the Coriolis effort and performing an integration of the gyroscopic signal, the angular rate can be

obtained [10]. Gyroscope can be applied for the measurement of the motion and posture of the human segment in gait analysis by measuring the angular rate [26].

4. Electromyography (EMG)

Electromyography (EMG) was developed to perform an indirect measurement of muscle activity using surface or wire electrodes. These electrodes are a kind of sensor for EMG and can detect voltage potentials to provide information on the timing and intensity of muscle contraction [8, 10]. Generally, surface electrodes are used when only general information on muscle activity is required, whereas wire electrodes must be inserted into the designated muscle using a hypodermic needle to measure specific information on a particular muscle [27]. Even though, it is useful for identification of up to eight phases, which is the highest possible granularity in gait partitioning according to literatures [16], it is less used than the other wearable sensor since it is very complex in acquisition and post-processing [28].

5. Inertial motion sensor (IMU)

In the gait analysis, a gyroscope is usually combined with an accelerometer to construct a complete initial sensing system [10]. Although relative orientation could be estimated by integration of data from gyroscope, errors would accumulate by this method, which caused distortion and drift errors. Accelerometer can be used to compensate the drift of the gyroscope about the axes of the horizontal plane, while magnetic sensor which located orientation by earth's magnetic field was adopted to solve this drift problem about the vertical axis. However,

inside reinforced-concrete-covered buildings, the magnetic field on the earth was always perturbed[9].

An inertial measurement unit, or IMU, is comprised of a 3-axis accelerometer and a 3-axis gyroscope packaged together and can be used to measure angles, angular velocities, and angular accelerations of a single rigid body or when used in pairs between two rigid bodies in three-dimensional space[29].

To minimally alter the subject's gait, a single IMU is often attached at the waist level so that the impact of both feet could be detected [30]. A downside of this solution is the difficulty to implement a robust and accurate method for identifying Gait events, since in general, the closer the IMU is to the point of impact, the higher are the chances of correctly detecting the Gait events [12].

It can be taken from the review in this section that IMU(Inertial motion units) have the advantages over the others. Accelerometers which are one part of IMU, are widely used for gait analysis, since it induces less error. It is illustrated from the review that the positioning IMU close to the impact, that is the foot, gives higher accuracy. Based on this IMU measured acceleration data from the foot of the subject are used in this paper.

2.3 Basis of human gait analysis

2.3.1 Introduction of the Gait Cycle

Generally, human walking is a periodic movement of the body segments and includes repetitive motion. The gait cycle is defined as the time interval between two successive occurrences of one of the repetitive events of walking. Although any event could be chosen to define the gait cycle, it is generally convenient to use the instant at which one foot contacts the ground ("initial contact" or "heel strike"). If it is decided to start with initial contact of the right foot, as shown in Figure 2.1, then the cycle will continue until the right foot contacts the ground again. The left foot, of course, goes through exactly the same series of events as the right, but displaced in time by half a cycle [31].The Rancho los amigos analysis committee developed a generic terminology for the functional phases of these gaits [32], as shown in Figure 2.1.Different granularity of gait cycle have been proposed by different researches. Even though the granularity are different, the basis for all the granularities are the gait cycle stated above. Taborri and colleagues have summarized the gait cycles granularities given in the literature as shown in the Table 2.1.

Granularity	Gait phases							
Two phases	Stance					Swing		
Three phases	First Rocker		Second Rocker			Swing		
Four phases	Heel Strike			Flat Foot	Heel Off	Swing		
Five phases	Heel Strike		Flat Foot	Heel Off	Toe Off	Swing		
Six phases (a)	Initial Contact	Loading Response	Mid Stance	Terminal Stance	Pre Swing	Swing		
Six phases (b)	Loading Response		Mid Stance	Terminal Stance	Pre Swing	Swing 1		Swing 2
Seven phases	Loading Response		Mid Stance	Terminal Stance	Pre Swing	Initial Swing	Mid Swing	Terminal Swing
Eight phases	Initial Contact	Loading Response	Mid Stance	Terminal Stance	Pre Swing	Initial Swing	Mid Swing	Terminal Swing
Gait [%]	0				60			100

Table2-1 Nomenclature for different granularity of gait phases [32]

These seven events , stated by Rancho los amigos, subdivide the gait cycle into seven periods, four of which occur in the stance phase, when the foot is on the ground, and three in the swing phase, when the foot is moving forward through the air (Figure 2.1). The usual terminology is inadequate to describe some severely pathological gaits [31, 33].The stance phase, which is also called the ‘support phase’ or ‘contact phase’, lasts from Heel Strike to toe off. It is subdivided into:

1. Heel strike (Loading response)
2. Flat foot (Mid-stance)
3. Heel off (Terminal stance)
4. Toe off (Pre-swing)

The swing phase lasts from toe off to the next Heel Strike. It is subdivided into:

1. Initial swing
2. Mid-swing
3. Terminal swing.

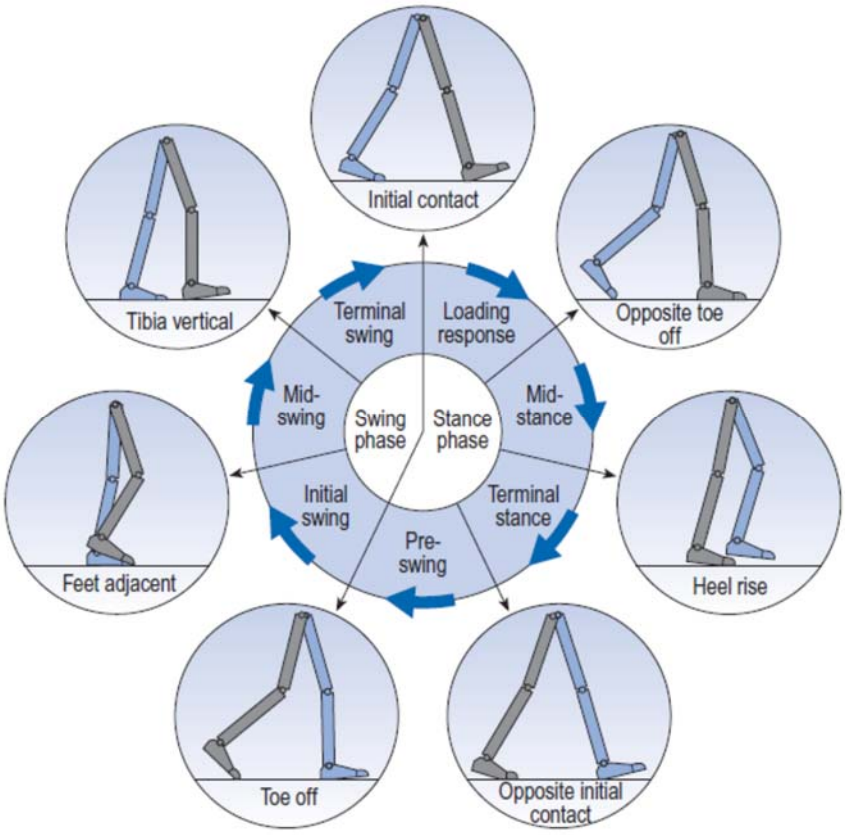


Figure 2-1 Positions of the legs during a single gait cycle by the right leg (gray) [31]

The following terms are used to identify major events during the gait cycle:

1. **Heel strike (Loading response):** This phase is the initial double-stance period. The phase begins with initial floor contact and continues until the other foot is lifted for swing.
2. **Flat foot (Mid-stance):** This phase is the first half of the single-limb support interval. It begins when the other foot is lifted and continues until body weight is aligned over the forefoot.

3. **Heel off (Terminal stance):** This phase completes the single-limb support. The stance begins with the heel rising and continues until the other foot strikes the ground, in which the heel rises and the limb advances over the forefoot rocker. Throughout this phase, body weight moves ahead of the forefoot.
4. **Toe off (Pre-swing):** This final phase of stance is the second double-stance interval in the gait cycle. It begins with the initial contact of the opposite limb and ends with the ipsilateral toe-off. The objective of this phase is to position the limb for swing.
5. **Initial swing:** This phase is approximately one-third of the swing period, beginning with a lift of the foot from the floor and ending when the swinging foot is opposite to the stance foot.
6. **Mid-swing:** This phase begins as the swinging limb is opposite to the stance limb and ends when the swinging limb is forward and the tibia is vertical.
7. **Terminal swing:** This final phase of swing begins with a vertical tibia and ends when the foot strikes the floor. Limb advancement is completed as the leg (shank) moves ahead of the thigh [10].

2.3.2 Gait cycle timing

Four divisions for functional gait assessment procedures are necessary to accumulate relevant information in the analysis of normal gait. These include:

- Weight Acceptance (initial contact and loading response)
- Stance (mid-stance and terminal stance)

- Forward Progression (terminal stance and pre-swing)
- Swing (initial swing, mid-swing and terminal swing)[1]

Figure 2.2shows the timings of initial contact and toe off for both feet during a little more than one gait cycle. Right initial contact occurs while the left foot is still on the ground and there is a period of double support (also known as 'double limb stance') between initial contact on the right and toe off on the left. During the swing phase on the left side, only the right foot is on the ground, giving a period of right single support (or 'single limb stance'), which ends with initial contact by the left foot. There is then another period of double support, until toe off on the right side. Left single support corresponds to the right swing phase and the cycle ends with the next initial contact on the right[31].

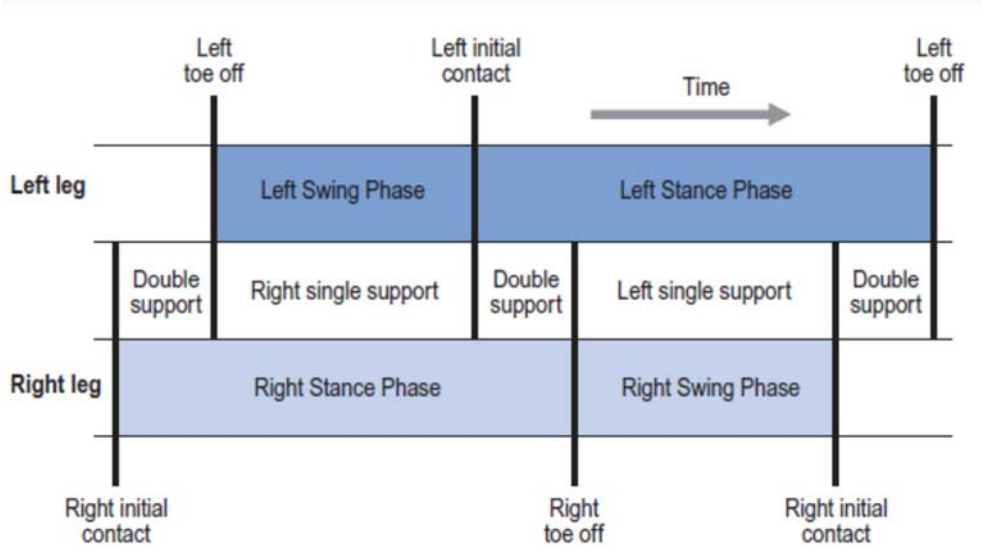


Figure 2-2 Timing of single and double support during a little more than one gait cycle, starting with right initial contact [1]

In each gait cycle, there are thus two periods of double support and two periods of single support. The stance phase usually lasts about 60% of the cycle, the swing phase about 40% and each period of double support about 10%. However, this varies with the speed of walking, the swing phase becoming proportionately longer and the stance phase and double support phases shorter, as the speed increases [1]

2.3.3 Gait deviation

When the human gait becomes biomechanically inefficient it is described as abnormal or stated as a deviated gait. Abnormality can have a number of causes, and manifest itself in many ways. Being able to describe the abnormality present in a gait is useful for those trying to correct it, and assessing the progress in rehabilitation process [34].

An important requisite for a portable system is the ability to provide kinematic information about gait. Kinematic data (angles, velocities and accelerations), can provide important insights into gait abnormalities. A low cost portable system based on accelerometers could, however, make quantitative gait analysis more clinically accessible [35].

Gait deviation is seen on the gait pattern from the sensor data that is taken from the individual[36], [37]. In the presence of abnormality of gait, the patterns show irregularity. Differencing this phases of gait could be one way detecting deviation of the gait [38]. One of the means to correct this deviation is the use of Orthotic bracing. According to ISPO (International society for Prosthetic and Orthotic), the lower limb orthotics have been classified according to the body segments and according to joints involved, as shown in 2.2.

Denomination	Body Segments involved	Main Lower-Limb Joints Involved
Foot Orthoses	Foot	-
Knee Orthoses	Thigh, Leg	Knee
Hip Orthoses	Pelvis, Thigh	Hip
Ankle Foot Orthoses	Leg, Foot	Ankle
Knee Ankle Foot Orthoses	Thigh, Leg, Foot	Knee, Ankle
Hip Knee Ankle Foot Orthoses	Pelvis, Thigh, Leg, Foot	Hip, Knee, Ankle
Thoracic Hip Knee Ankle Foot Orthoses	Trunk, Pelvis, Thigh, Leg, Foot	Hip, Knee, Ankle

Table 2-2: Classification of Lower-Limb Orthotics [58]

Alex and colleagues studied a new approach to study the asymmetry in gait . In their study theyexplained the effect of ankle and knee bracing in overall gait cycle. Based on their study, knee bracing affected the ankle and hip joints during both stance and swing; however, the greatest asymmetry in both joints occurred during the swing phase. it affects the knee during mid swing, but also increased ankle asymmetry during both terminal stance and mid-swing and hip asymmetry during mid-stance and mid-swing. The un-braced knee during knee bracing shows a small peak difference during initial swing. However, the braced knee had a much larger peak value during mid-swing [4].

Whereas, ankle-bracing created asymmetries at the ankle (terminal stance and initial swing) and hip (terminal stance), but not significant asymmetry at the knee. Ankle braced indicated that ankle bracing largely impacts the affected side of the body. Small asymmetries were present in the un-braced ankle during terminal stance, but the asymmetries at the braced ankle during terminal stance were much larger. Interestingly, the ankle bracing affects the behavior of the knees during swing, but not in an asymmetric manner[4].

As stated in the research, bracing also decreases the symmetry of paired joint range of motion, increased the subject's step width, and reduced the time subjects spent in single leg support on the affected leg. Also, knee bracing had a greater impact than ankle bracing on overall joint symmetry. Overall, knee bracing created a larger perturbation to gait than ankle bracing[4]. A similar finding was reported in[36].

2.4 Gait classification

2.4.1 Pattern recognition

Gait classification requires recognizing the pattern exhibited by the data that are going to be classified. Stated differently, gait classification is a form of pattern recognition. Pattern recognition is a machine's ability to understand the behavioral pattern of the data and distinguish between different pattern behaviors [39]. Gait is a time series data which varies with time. Time series data follow three approaches for classification or clustering, as stated in [40].

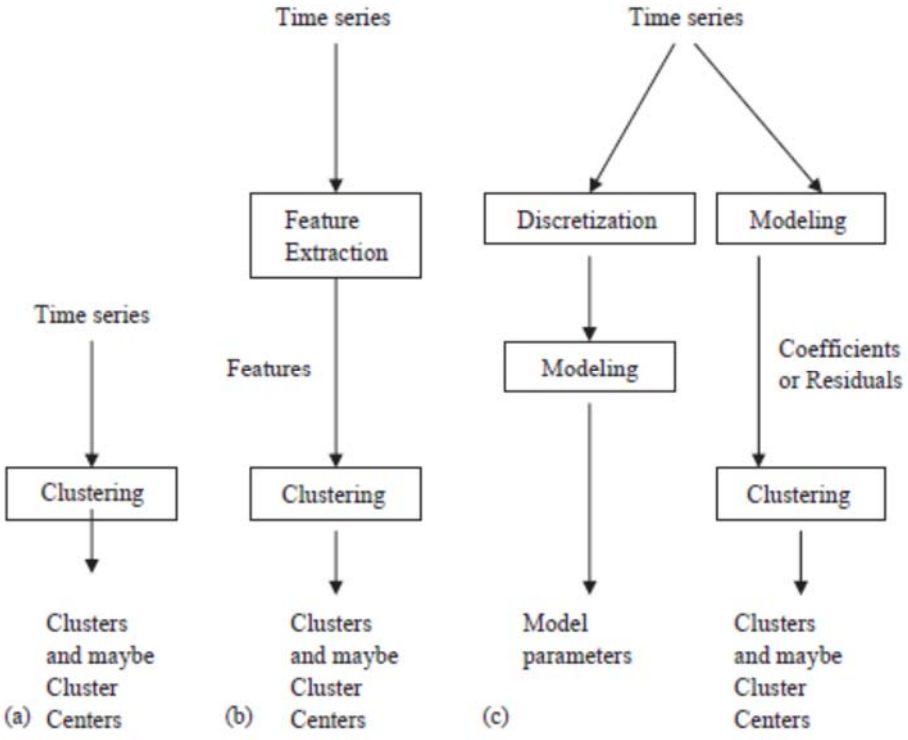


Figure 2-3 Three time series clustering approaches: (a) raw-data based (b) feature based (c) model based [40]

As shown in Figure 2.3, The former approach (a) usually works directly with raw time series data, thus called raw-data-based approach, in this approach the raw data is directly used without extracting any additional information from it. The second and third approach (b),(c) first converts a raw time series data either into a feature vector of lower dimension or a number of model parameters, and then applies a conventional clustering algorithm to the extracted feature vectors or model parameters, thus called feature- and model-based approach, respectively. Note that the left branch of model-based approach trained the model and used the model parameters for clustering without the need for another clustering algorithm[40].

The first approach incorporates the errors of the raw data into the classification which makes it the lesser type to use because of the error induced due to raw data. Feature based approach eliminate the error to some extent but some of the errors are still left on the selected feature. Whereas Model based approach minimize the error as much as possible and also it is an easy way to extract features. Due to this, the study described in this thesis uses model-based approach for developing the algorithm. This approaches follow a set of steps as shown in Figure 2.4, as stated in [5]

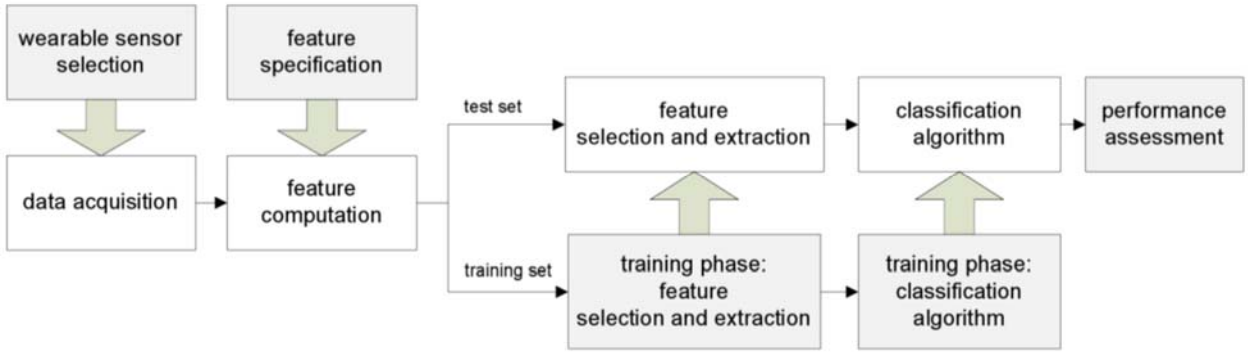


Figure 2-4 Conceptual scheme of a generic classification system with supervised learning.

The terms used in the approach are described as follows:

1. Wearable sensors and data acquisition

The first important aspect to be considered in building a system for automatic classification of human physical activity concerns the choice of sensors. Wearable sensors should be small and lightweight, in order to be fastened to the human body without compromising the user's comfort

and allowing her/him to perform under unrestrained conditions as much as possible[5]. This area is covered in section 2.2, as one of the techniques used for gait analysis.

2. Feature evaluation

A pattern recognition machine does not perform its classification tasks working directly on the raw sensor data, since the data has noise. Usually, the classification is pursued after that a data representation is built in terms of feature variables[5]. These feature variables can be found directly from the raw data set, like feature based approach, or can be found from the chosen model[6]. The choice of this features requires high information content for classification and depends on the specific type of classification and problem that are being handled [5].

3. Feature selection and extraction

When the dimension of the feature space is high, learning the parameters of a classifier becomes a difficult task, especially when the size of the training set is small (the curse of dimensionality). The available dataset is then divided into a training set and a test set. Since the achievable performance of a classification algorithm tends to critically depend on the dimension of the feature space, methods for reduction of dimensionality are oftentimes considered in developing the classifier[5]. These methods are based on two main approaches: feature selection and feature extraction [39].

The feature selection approach consists of detecting and discarding the features that are demonstrated to minimally help to cause a correct response by the classifier. The feature

extraction approach revolves around the idea that data representations can be constructed in subspaces with reduced dimension, while at the same time retaining, if not increasing, the discriminative capability of the new set of feature variables [39].

The fundamental prerequisite for the implementation of automatic methods for the classification of gait disorders is the possibility to identify selected gait features for which the intra-pathology variability is smaller than the inter-pathology variability [6].

4. Taxonomy of classifiers

A taxonomy of classifiers can be built according to different criteria. First, a distinction is between supervised and unsupervised classifiers. Supervised classification is when the input pattern is identified by predefined class (categories). Once the data is learned the test or actual data could be classified based on the trained model. Whereas unsupervised classification is when the pattern is assigned to a class which is exhibits a similar pattern behavior. Unsupervised classification is also called clustering since it clusters the pattern into N number of classes depending on the difference and similarities of the overall data set [39].

A few unsupervised approaches have been investigated for activity classification, such as hierarchical methods and self-organizing maps (SOM).This procedure is useful for exploring and investigating characteristics of the data set, however, an extra supervised layer is needed to achieve the complete classification task [13]. So, focus should be given to supervised classifiers.

Supervised classifiers can be further divided according to three main approaches: probabilistic, geometric, and template matching[39].In accordance to the rules of the probabilistic approach, a feature vector x is classified as belonging to the class which turns into the maximum value of the class-conditional PDFs p(x|Ci), i = 1, …, N. The class-conditional PDF denotes how likely it is for a feature vector to belong to a given class. An example of probabilistic classifiers is the optimal Bayesian classifier[5].

In the geometric approach the classification is performed based upon the construction of decision boundaries in the feature space that specify regions for each class. Decision boundaries are constructed during the training session via iterative procedures or geometrical considerations. Artificial Neural Networks (ANN), k-Nearest Neighbour (k-NN) classifiers, Nearest Mean (NM), and Support Vector Machines (SVM), are geometric approach classifiers. Another popular approach is the threshold-based classifier[5], which set a threshold value for the classifier to distinguish the classes.

The template matching approach is based on the concept of similarity between observed data and activity templates, either defined by the designer or obtained during the training session. The editing and condensing techniques, customarily applied to k-NN classifiers, can be useful for defining the template[5]. All of the classifier expect threshold approach are part of machine learning algorithm, which will be explained in detail in the next section including its applications for classifying a time-series data.

2.4.2 Classifier Models

2.4.2.1 Threshold

As stated in section 2.4.1, threshold algorithm sets a threshold value from the data set as means of classifying the data set. Darwin and colleague detected Heel strike and toe off events using combined algorithm of zero-crossing and heuristic. 16 healthy subjects were used on normal walking, wearing knee brace and wearing ankle brace. The algorithm showed a better identification time than the algorithm that used spatio-temporal threshold method [36].

These methods normally convey only temporal information about the signal and no information about how the subject's feet are moving through space is incorporated [13]. Also, the high inter-subject variability that can occur, especially in abnormal gait, can degrade significantly the performances of the classifier[5, 14]. Due to this reason the focus of this paper is on machine learning which is discussed on the next section.

2.4.2.2 Machine Learning

Machine learning teaches computers to learn from the data set and act accordingly. One of the application of these learning is on classifying a data set to N number of classes. A class in machine learning is the category of a specific data type. The machine learns the differences between the features of the given data set to classify it accordingly. It then extracts pattern on the basis of the selected classifiers. Different type of machine learning classifiers has been used as gait classification over the years, short description of these types are given below:

1. Naïve Bayes

It denotes a Bayesian network implemented as class Naïve Bayes. The Bayesian network represents a joint probability distribution over a set of categorical attributes. Numerical attributes are discretized. It comprise of a directed acyclic graph, conditional probability tables, and allows the computation of the (joint) posterior probability distribution of any subset of unobserved assignments of values to attributes, which makes it ideal for classification[41].

2. Support Vector Machine (SVM)

SVM is a geometric-based classifier that constructs boundaries maximizing the margins between the nearest features relative to two distinct classes It is in supervised learning models with associated learning algorithms that are used for data analysis and pattern recognition. It is a nonlinear classifier which is known to produce better classification result compared to other classification methods[6, 41].

3. K-Nearest Neighbor (KNN)

It groups the instances based on their similarity, where an object is classified by a majority of its neighbors. It is a type of Lazy learning algorithm where the function is only approximated locally and all computation is deferred until classification. The neighbors are selected from a set of objects for which the correct classification is known [41].

4. Artificial Neural Networks (ANN)

A neural network model which is the branch of artificial intelligence is generally referred to as artificial neural networks (ANNs). ANN teaches the system to execute task, instead of programming computational system to do definite tasks. To perform such tasks, Artificial Intelligence System (AI) is generated. It is a pragmatic model which can quickly and precisely find the patterns buried in data that replicate useful knowledge. One case of these AI models is neural networks. Neural network as a classifier is complex to design and to process[42].

5. Tree Bagger

It is an ensemble method, where when it is given a weak learning algorithm and a training set, it generates a number of training sets randomly including some samples from the initial training set, that are almost identical in size. The classification algorithm is trained on each of the training sets and generates a predicted sequence where the final function G can be obtained by voting[43].

6. Hidden Markov Model (HMM)

HMM is a double stochastic process in which the existence of a set of discrete states is assumed for a given system. The first stochastic process describes how the system may jump from one state to another (transition probability), under the hypothesis that the next state depends only on the state at the present time (Markov property). The state sequence is not observed—it is hidden to the observer, who has access to the emissions of each state only (in practical terms, the

emissions can be considered as the observed quantities such as sensor data that are used to infer the hidden structure of the modeled phenomenon). The second stochastic process yields the statistical description governing the emissions of each observed variable (emission probability), in terms of either discrete probabilities or probability density functions. Mostly HMM is used for clustering because it is good for summarizing signal characteristics[6].

Different researchers have used machine learning to classify different data types. Few researches have be conducted related to gait classification. In [44], Andrea Mannini and colleagues proposed a framework that detect 4 gait phases and that discriminate between walking and jogging, using supervised Hidden Markov Model. They used IMU sensors attached to the foot instep of the left foot, only data from the gyroscope was taken for the analysis. 6 healthy subjects were made to walk and jog on a treadmill for 2 minute with 5 walking speeds. The timing errors of gait events detected were compared to threshold based algorithm and the former method showed superiority. The extension of the method to pathologic gait has not been tested yet. Similar studies have been conducted based on Support Vector Machines (SVM)[45], Support Vector Machine (SVM) and Artificial Neural Network (ANN) [46], k-nearest neighbor [47], and Neural Network [48].

All of the above classification focus on healthy subjects, few studies have been made on classification of abnormal and normal gait. Classification of abnormal and normal gait aids in physical mobility and minimize the risk on effect of therapy and rehabilitation [6]. Even though no classification tests were conducted, Feature extraction of normal and pathological gait from kinematic data has been proposed by [7]. The paper extracts spatio-temporal features from the

kinematic data, which can be used by self-organized map to classify the normal and pathology gait. The extracted features show a promising result for use of classification of the gait, but extraction of the spatio-temporal data set is a long process which will elongate the algorithm processing time.

In [6], a machine learning based framework for classification of different pathological gait using IMU sensor mounted on the shank and waist was conducted. The framework is a combination of Hidden Markov Model and Support vector machine. The classification framework took three major steps (i) the classification process started from implementing class-specific HMMs, whose likelihoods of observing data given each model were evaluated; [49] the HMM likelihoods were included in a wider feature set, which was then classified using an SVM classifier; (iii) a majority voting (MV) classification post-processing was cascaded to summarize the results obtained in step ii. The proposed methodology was tested on three different population (17 Huntington's disease, 15 post stroke subjects and 10 healthy elderly controls) using cross validation approach.

In [50], the research compared several classifiers which uses Machine learning to automatically recognize five healthy controls and four patients with hemiplegia using kinematic data. In their study they found out that classification using machine learning could give an accuracy of more than 90%.

As seen from discussion all of the machine learning classifiers require further research. Most of the authors agree that SVM, Naive Bayes, KNN, Tree Bagger as a classifiers show a better

overall performance that any other. Neural Network is also an accurate classifier expect it is complex. Hidden Markov model is mostly used for clustering than classification. Based on that this study focuses on SVM, Naive Bayes, KNN, and Tree Bagger classifiers.

2.4.3 Performance analysis

2.4.3.1 Partitioning Method

The classification process starts by obtaining a data set (input–output data pairs) and dividing it into a training set and validation data set. Different methods are used to partition this training and test data, which are briefly discussed below:

1. k-Fold cross validation

A popular procedure for estimating the performance of a classification algorithm or comparing the performance between two classification algorithms on a data set is k-fold cross validation. This procedure randomly divides a data set into k disjoint folds with approximately equal size, and each fold is in turn used to test the model induced from the other k_1 folds by a classification algorithm. The performance of the classification algorithm is evaluated by the average of the k accuracies resulting from k-fold cross validation, and hence the level of averaging is assumed to be at fold[51].

2. Leave-one-out cross validation

Many studies adopt leave-one-out cross validation to evaluate the performance of a classification algorithm when the number of instances in a data set or the number of instances for a class value

is small. Since the randomness of dividing instances into for training and testing does not exist, the point estimate of accuracy for a given data set is constant[51].

3. Hold on cross validation

A hold out cross validation is similar to k-fold method expect it holds n percent of the data set rather than taken some k fold of it like k-fold. The data set was partitioned into training and testing data sets. It is recommended to use 70% of the data for training and 30% for testing. For performance analysis, the test data sets were used for assessment[52].

One problem of using cross-validation is that the evaluated performance is affected by the result of random division and different performances are achieved in every evaluation. To solve this problem, Gavrilov and colleagues used the average of a 100 time evaluation [53]. Based on the review this paper uses a hold out cross validation by taking 30% of the data set as a test set An average of 10 run is used to minimize the issue of random division.

2.4.3.2 Performance Indicators

Confusion matrix contains valuable information about actual and predicted classifications created by the classification model. Based on that it is used by most authors to calculate performance indicators [41, 51-56]

Studies conducted to evaluate the performance of a classification algorithm use the following indicators: Area Under the Curve (AUC), Overall accuracy (AC), Sensitivity (TPR), Specificity (TNR), Positive predictive value (PPV), Negative predictive value (NPV), F-measure[41, 52,

55]. Positive prediction is used to express the required class whereas negative prediction is the unrequired data set.

Some of the performance indicators are related to each other like the area under the curve (AUC) gives the sensitivity and the specificity of the measurement. F-measure gives the relationship between Sensitivity (TPR) &Positive predictive value (PPV) and Specificity (TNR) & Negative predictive value (NPV)[55]. Which means it can be used interchangeably. In addition to that Positive predictive value (PPV), Negative predictive value (NPV) evaluates the performance ability to predict positive or negative respectively. The accuracy of this values require the data set between the two classes to be proportional [41]. Based on that Accuracy, Sensitivity and Specificity are used as performance indicators in this study.

2.5 Conclusion

The literature shows that application of gait analysis is vast especially in medical applications where its use is ever growing. Gait analysis for medical propose requires evaluation of the patients continuously in the patients every day environment. In order to accomplish that the use of wearable sensors is the right choice as summarized by most literatures in section 2.2. Although, all wearable sensors have their own advantages and disadvantages, IMU (Inertial Motion Unit)have become sensors that are widely used. This is due to its low cost, miniaturized size, the fact that it easy to access, as stated in section 2.2.2. In addition, the closer the IMU sensor to the point of impact the higher the accuracy.

Mostly clinical application is figuring out the abnormality or normality of the gait that is being assessed, which requires first a method that could classify the two. Due to this gait classification has been an interesting research area as stated in section 2.4. Although most researches have focused on classification of gait phases, walking &jogging, etc., recently some researcher are being done on classification of abnormal and normal gait. Even those researches mostly focus on elderly, Stroke patients as stated on section 4. In this thesis the focus is mainly in understanding gait deviation due to wearing orthotic device and classifying it from the normal gait. This is because most of the literatures agree that knowing its deviation from the normal gait will aid in fitting the devices and design procedure of the device. Literatures mainly agree that machine learning is widely used for gait assessment, especially for gait classification, as stated in section 2.4.2. Based on the review SVM, Naive Bayes, KNN, and Tree Bagger classifiers are chosen as classifier for this study. As stated section 2.4.3 the performance of the algorithm is best evaluated using a hold-out cross validation. The core performance indicators to achieve that are accuracy, sensitivity and specificity.

Research area	Objectives	Finding and Conclusion	Reference
Gait analysis • Application to gait analysis • Techniques used for gait analysis	To understand the general concept of gait analysis To select the best technique by exploring different types	• Medical application requires further research • Wearable sensors are the best technique used to assess a patient • Accelerometer as part of IMU sensor are the most widely used wearable sensor • The closer the IMU sensor to the point of impact the higher the accuracy	[8-10], [12], [16], [18-30]
Basis of human gait analysis • Introduction of gait cycle	To understand the basic terminology of a gait cycle	• The general concept of gait cycle is stated	

• Gait cycle timing • Gait deviation	To distinguish between normal and deviated gait cycle To have an over view of the deviation due to Orthotic bracing	• Terminology related to gait cycle are explained • The basic deviations related to Orthotic bracing are summarized • Ankle bracing, Knee bracing and Ankle&Knee bracing are selected	[1], [4], [10], [31-38], [58]
Gait classification • Pattern recognition • Classifier model (tools) o Threshold o Machine learning	To investigate the basic approach used for pattern recognition To understand the basic step need for classification	• Model based approach is chosen since it minimizes the error and it is an easy way to extract features • Basic step which are used for the Research design are selected, see	

• Performance analysis	(pattern recognition) To select a fitting classifier model for the study To select a partitioning method To explore different type performance indicators	chapter 3 • SVM, Naive Bayes, KNN, and Tree Bagger classifiers are selected to be used as a classifier • A hold out cross validation was selected, taking 30% of the data as a test set. • The core performance indicators are Accuracy, Sensitivity and Specificity	[5-6], [13-14], [39-56]

Table 2-3 Summary of literature review findings

Chapter 3

Research Design and Methodology

This chapter presents the research design and methodology used in the thesis. The first section gives a general description of the research approach and the step used in the thesis. The second section explains the materials and methods used in the overall thesis process.

3.1 Research design

The main aim of the thesis is to develop an algorithm for classification of Normal, Ankle braced, Knee braced and Ankle&Knee braced gait. To develop the algorithm a supervised learning Machine learning algorithm is used. This approach is taken based on the consideration of the literature review as stated in section 2.4. The performance of the classification algorithm was then evaluated using performance indicators. Based on the performance evaluation the class type, the classifiers, and the bracings are analyzed. The analyzed results are to some extent cross checked with available related literatures. Based on the reviewed literatures the research frame work that has been used in the thesis is presented in Figure 3.1. The methods used for each section of the frame work is given in the next section.

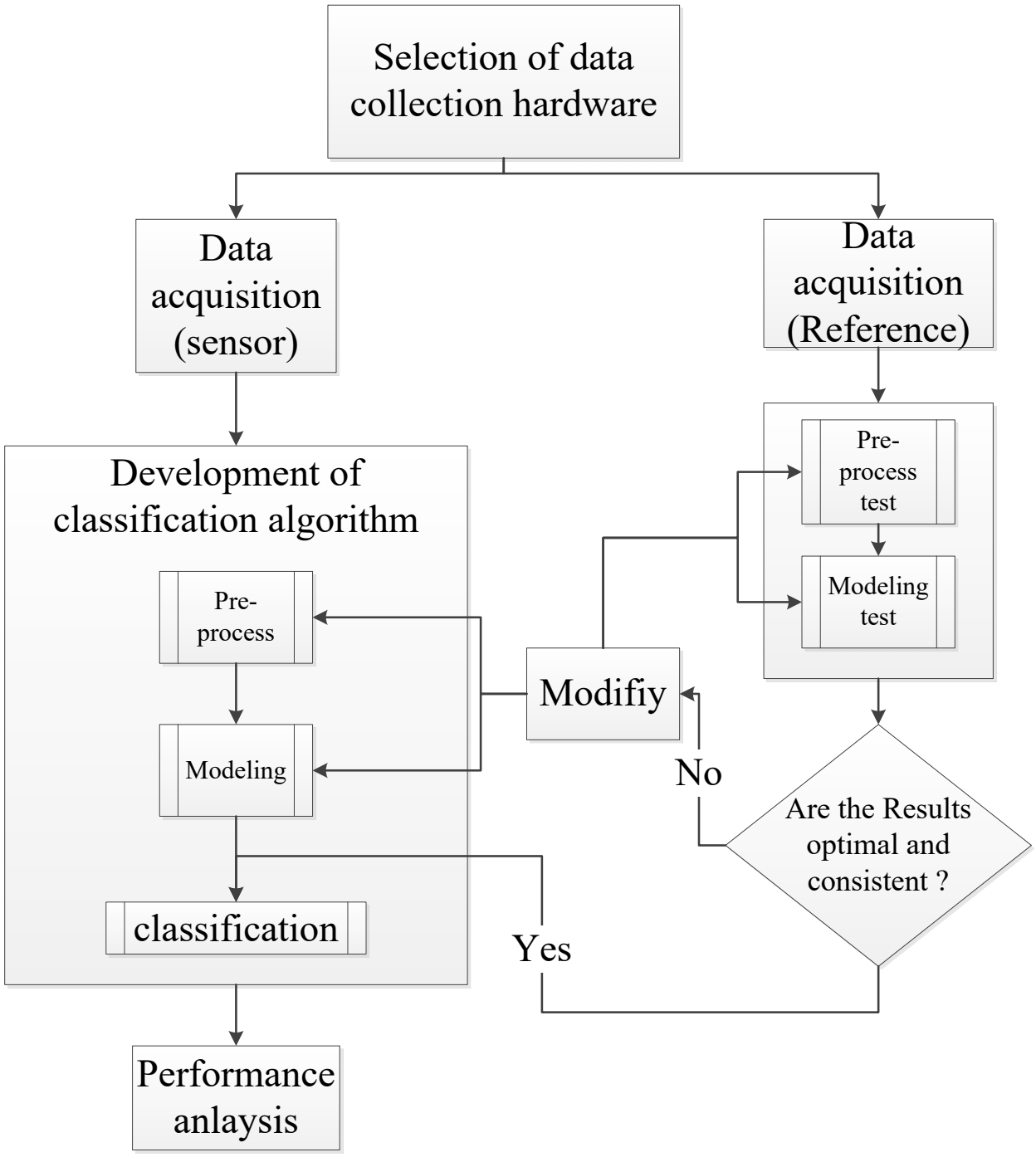


Figure 3-1 Research frame work

3.2 Research Methodology (Methods & Material)

3.2.1 Data collection hardware

As depicted in the research frame work the first stage of the research is to select a data collection hardware. Two data collection hardware were selected, the first one is a wearable sensor to be used for the overall classification. IMU wearable sensor was chosen for this paper, this is based on finding of the literature review, as stated in section 2.2.2. The second data collection hardware was needed to cross refer the first two stages of the algorithm process, as depicted on the research frame work. Vicon video was chosen as a reference based on its repeatability on most literatures as discussed in section 2.2.2. Also because it is used in laboratory analysis in medical sectors.

A standard Inertial Measurement Unit consists of three accelerometers and three gyroscopes, with the measurement axis of the accelerometers and the gyroscopes aligned was used. Each of these sets of accelerometer/gyroscope pairs is placed perpendicular to the remaining two sets in an axis set that follows a right handed convention; this allows for measuring of specific acceleration and angular rate in three independent directions as shown in Figure 3.2. The IMU Parameters Used in the paper is shown in Table 3.1.

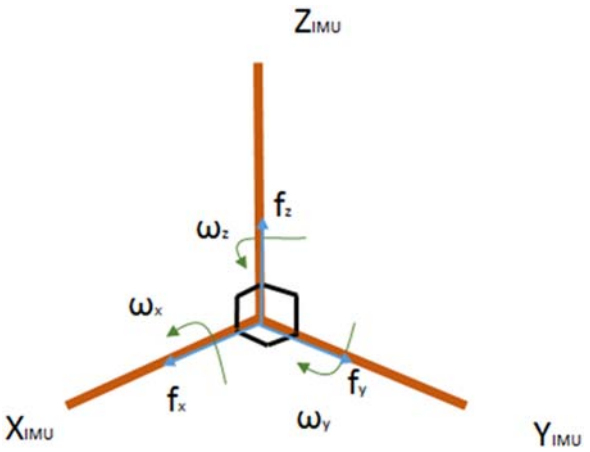


Figure 3-2 The nominal axes for IMU Measurements

IMU Parameter Description	Value
Correlation Time Constant of Gyro(τ_g)	300 (sec)
Stdv of Gyro Markov Bias ($\sigma_{\mu g}$)	0.3 (deg/s)
Stdv of Gyro Wide Band Noise(σ_{wg})	0.1 (deg/s)
Correlation Time Constant of Acc (τ_a)	500 (sec)
Stdv of Acc Markov Bias ($\sigma_{\mu a}$)	0.01 g (m/s/s)
Stdv of Acc Wide Band Noise (σ_{wa})	0.005 g (m/s/s)

Table 3-1 IMU Parameters Used in the thesis

A Vicon Motion capture data was acquired using a Vicon 512 motion capture system operating at 60 Hz (Oxford Metrics, Oxford, UK). There were 12 cameras, mounted approximately 2 m high encircling the 10 m long ×5 m wide ×2.5 m tall capture volume. The full description is given in the Vicon manual [57].

3.2.2 Data acquisition

Data was acquired from the IMU sensor by attaching the sensor on an elastic band and worn on the foot of the subject. The sensors was attached on both the right and the left foot, as shown in Figure 3.3. The study uses one healthy subject who is asked to reproduce the abnormality of the gait by wearing the orthotic brace. One subject was only taking the theory that the gait pattern of all normal gait is similar, as stated in the literature review. This can also been seen from the Vicon data which was collected from five different subjects, see appendix D. A healthy subjects was used because it is know that the only thing that is affecting the subjects gait is the braces. If we used unhealthy subjects we would not know whether or not what we were Identifying is the effect of the brace or the gait anomaly. The orthotic brace was worn only on the right foot, that is to see the effect of bracing on both the braced foot and the other unbraced foot.

The subject was asked to walk around for a while wearing a comfortable shoes around the hall way. Subject walked at self selected speed wearing his own shoes, but a constant speed was tried to be maintained. Data was collected for three trial for each conditions. whereas, normal gait condition data was collected from the subject both before wearing the brace and after removing the brace. Each gait cycle of the acceleration data set is taken as one sample size in the classification process. The accelerometer data was only used for the study, based on the literature review, see section 2.2.2. The IMU sensor data was acquired in Minnesota University, twin cite, USA. The Vicon data was taken only from a normal subject, and it was collected from five subjects. The acquisition process is found in the Vicon motion capture manual [57]. The Vicon

experiments were held at the James R. Gage Center for Gait and Motion Analysis at the Gillette Children's Hospital in St. Paul, Minnesota, USA.

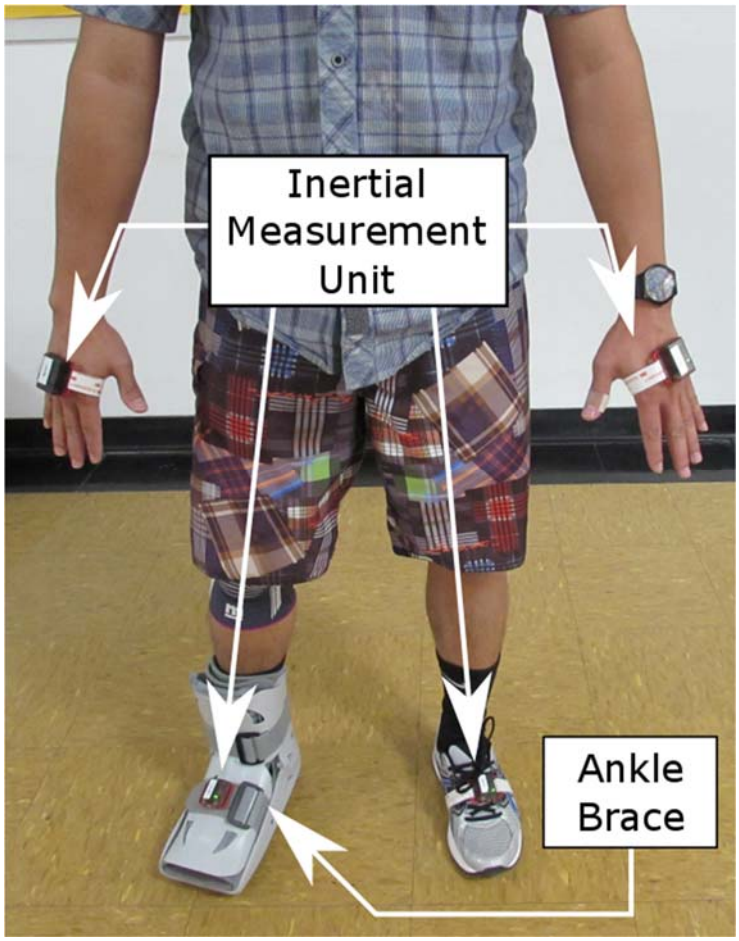


Figure 3-3Demonstration of data acquisition from IMU sensor

3.2.3 Development of classification algorithm

The developed algorithm includes a pre-processing, a curve modeling, a feature extraction and a classifier modeling stages as explained in chapter 4 The algorithm was developed using Mat lab software. This software was selected because of its ease of use and accessibility. The algorithm

uses model based approach, which is shown in Figure 2.4. This approach is chosen because classification of raw data is both tedious and error induced. Feature based approach also involves higher error due to the noise that are incorporated, as explained in the literature review in section 2.4.1.

The unfitted discretized gait cycle patterns was cross checked with both the Vicon and the IMU fitted gait cycle. Based on that an 8th order Gaussian model was chosen for fitting the raw sensor gait cycle, further explanation is given in section 4.2. The coefficients of the Gaussian fit were taken as a features for classification. This is based on analyzing the induced differences between the data sets, it is further explained in section 4.3.

Naive Bayes, Tree bagger, KNN (K-Nearest Neighbor), and Support Vector Machine (SVM) are selected as classifiers tools. classifiers were chosen based on the suggestion of several literatures, as shown in section 2.4.2.2.The classification of the left and the right foot are classified separately, since there is a distinction between the braced and unbraced side.

As stated in the research design the class type, the classifiers, and the bracings are evaluated. To accomplish this different type of algorithms were required. The first three algorithms implement overall classification. The first one classifies all the data set as multi-class classifier, as seen in appendix A1. Only Naive Bayes and Tree bagger classifiers are used, since the other classifiers only work for binary classification. The second and the third approach distinguish the data into sequential binary classification. This is to include other classifier type and to eliminate error induced by multi-class classification, as recommended by literatures. The second approach takes

each specific data verse the others and combines the result, as seen in appendix A2. The third approach takes a step wise binary classification, this is to reduce the error due to the combination.

The rest of the algorithms classify the sensor data categorically. That is each class type verses another class type is classified separately. The aim of this type of classification is to see the effect of the extracted feature on each type of the classes, see appendix A3. The detail explanation of the developed algorithm and the procedure used is given in chapter 4. The sample of the developed algorithms are found in appendix A. The rest of the developed algorithms are found in the CD that is included with this paper.

3.2.4 Performance analysis

To evaluate the performance of the proposed algorithm approach, a cross-validation was carried out. The data set was partitioned by holding out a test set for evaluation. A hold out cross-validation is recommended for better reliability of test results as stated in the literature review. Based on the recommendation of researchers 30% of the data set was hold out, see section 2.4.3.1.

The accuracy of the algorithm was calculated by taking the average of ten consecutive run of the Mat lab program. An average was taken to minimize the error that is included in training different partition data set with each run. In addition to that the accuracy run was taken to see the reputability of the classification.

Based on the literatures reviewed in section 2.4.3.2, performance indicators are selected for analysis. Confusion matrix has been used to synthesis the performance indicators. Even though there are five unique performance indicators as reviewed in section 2.4.3.2, only three are used in this study. The evaluation parameters used are; sensitivity, specificity and accuracy, as explained in the review.

Chapter 4

Development of the Classification Algorithm

In this chapter a classification algorithm for automatic identification of pathological gaits based on foot mounted IMU data was implemented. The classification algorithm was defined through four major steps: (1) The raw sensor data was pre-processed to discriticize the gait cycle from the overall covered distance; (2) Each specific gait cycle was fitted using 8th order Gaussian equation; (3) Distinguishing features were selected by comparing Gaussian curve fits between the normal gait data and each pathological gait data; (4) The data sets in step (3) were Classified using the chosen classifiers. The overall flow chart for the classification algorithm is shown in Figure 4.1. Based on the method explained in section 3.3.5, we have a total of 18 classification algorithms. The bases of all algorithms are similar and only small changes are made, see appendix A and the provided CD. The major aspect of the algorithm is explained in detail in subsequent section of this chapter.

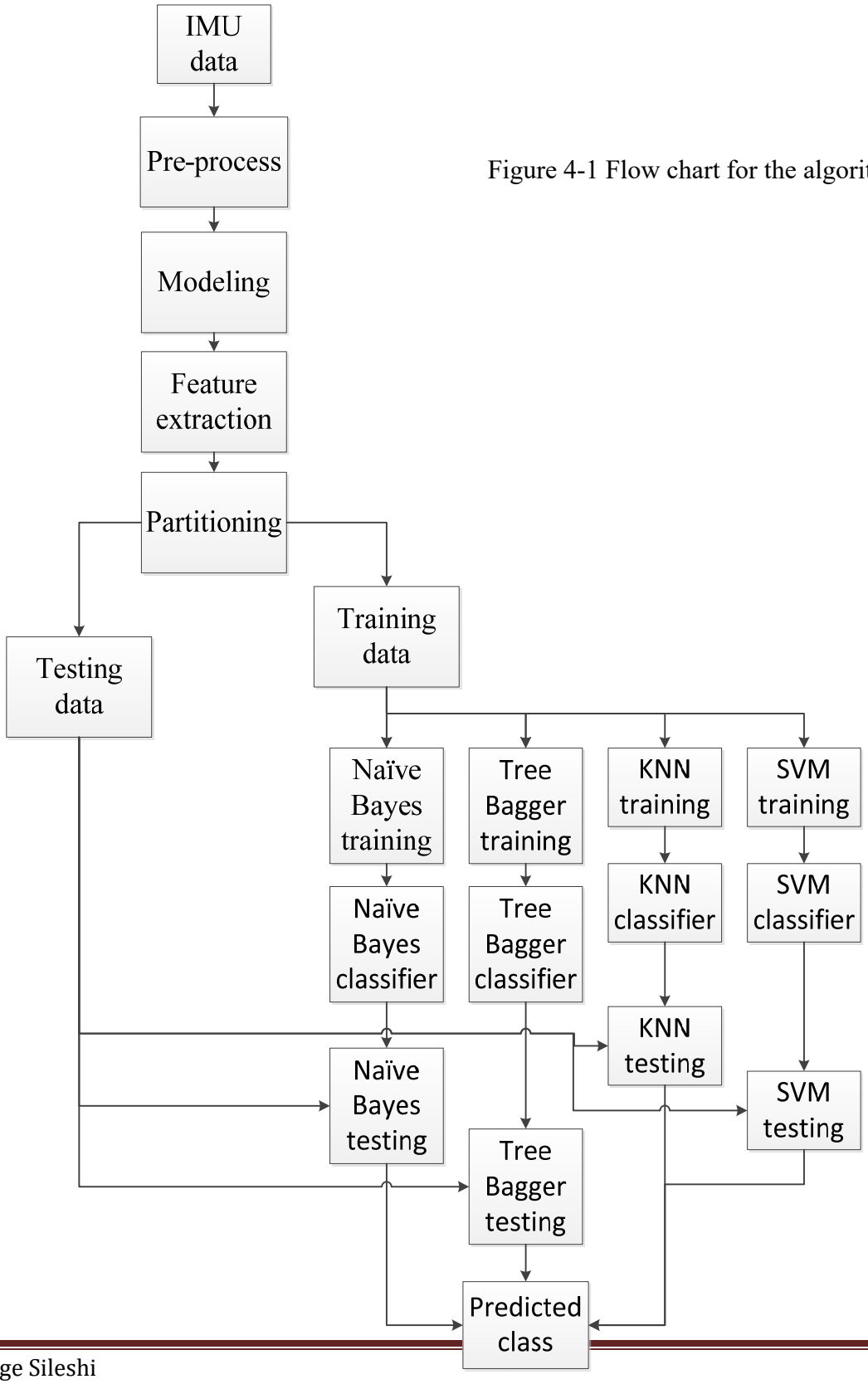


Figure 4-1 Flow chart for the algorithm

4.1 Pre-processing

As shown in Figure 4.1, the data acquired from the IMU sensor pass through pre-process. In this phase of the algorithm a single gait cycle was extracted from the total acceleration plot. So, that each gait cycle could be used as one sample size, as stated in section 3.2.2. From the basics of gait cycles in the literature review in section 2.3.1, the gait cycle starts and ends with a heel strike. So, finding the heel strike is the basis for separating each gait cycles.

From basic of dynamics acceleration at the heel strike leads to a large acceleration spike. Also, between the two large spikes there are two smaller ones corresponding to the start of heel-off and swing phases. This could be seen from the total acceleration plot shown in Figure 4.2. The figure shows resultant acceleration plot at the ankle from a normal gait data using Vicon motion capture. Vicon was used as a reference for pre-processing and modeling phases of the algorithm as explained in section 3.2.3. This is also seen from the total resultant acceleration plot using IMU data sensor shown in Figure 4.3.

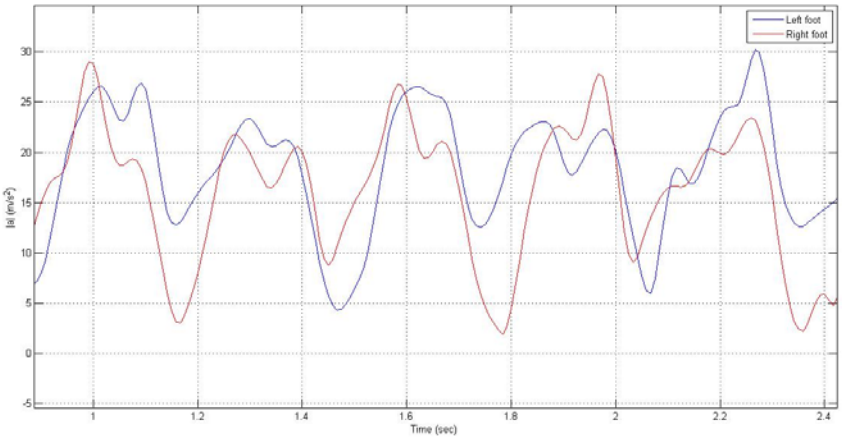


Figure 4-2 Resultant acceleration of Vicon motion capture

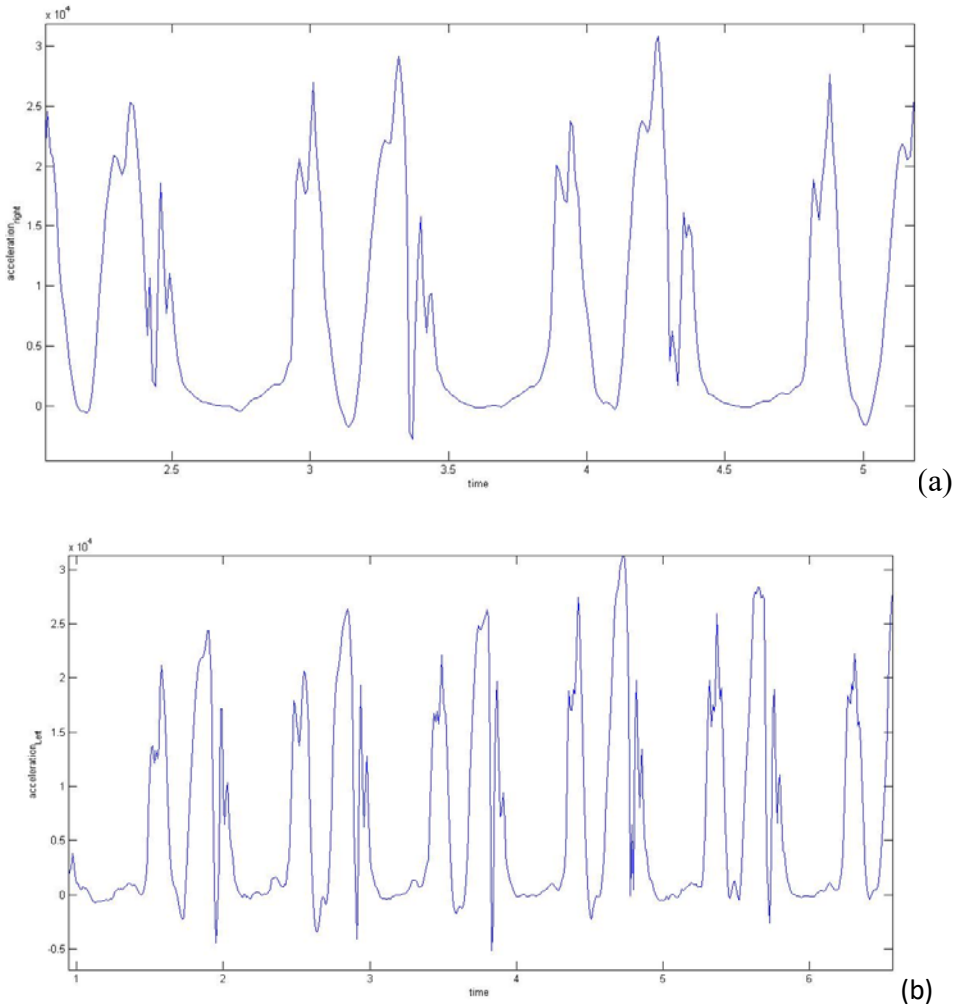


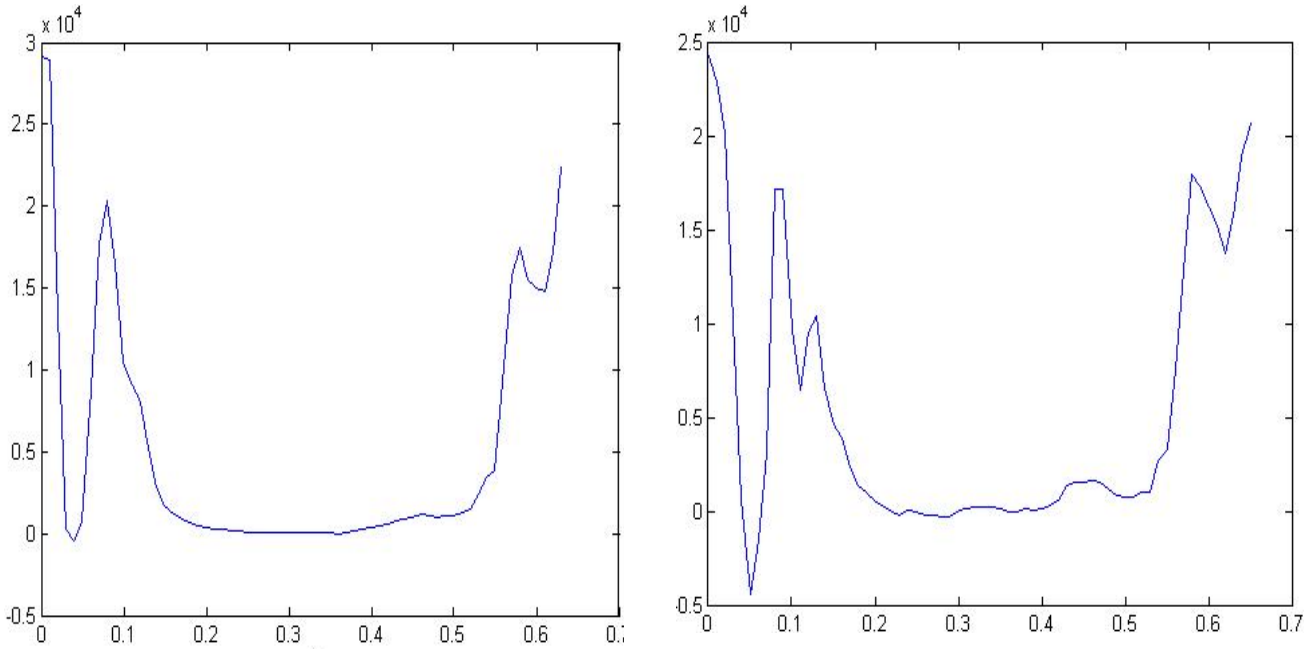
Figure 4-3 Resultant acceleration of IMU sensor (a) Right foot (b) Left foot

When Vicon and IMU's acceleration plot was compared it shows a slight difference. This is due to the position from which the acceleration was synthesized at. The Vicon acceleration plot shows resultant acceleration at the ankle, where as the IMU acceleration plot shows resultant acceleration at the foot, as explained in section 3.2.2. The general steps followed in discriticize the gait cycle are:

1. The specified four acceleration peaks per gait cycle are identified and based on the four peaks the points surrounding the heel strike are found. See appendix A1.

2. Based on the points found in step 1 above, the two large peaks will be located which are the heel strikes corresponding to the start and end of a step. See appendix A1.

3. Once all heel strike in the data are found, the raw data set is separated into steps taken by the subject. This is seen in Figure 4.4, when the gait cycle start with a large acceleration spike. See appendix A1.



(a)

(b)

Figure 4-4 Gait cycle from IMU sensor for (a) Right foot (b) Left foot

4. Some of the separated data at step (3) present a fewer number of data sets. This is came from the error that is induced from the separated algorithm and the noise from the sensor. In order to remove this data from the sample size all separated data set with points less than 24 are removed. This cutoff is chosen because for the gait cycle to be fitted by Gaussian equation requires a minimum of 24 points (8th order and 3 coefficient each). See appendix A.

5. Next data sets corresponding to incomplete steps are removed. They are identified from the fact that the number of acceleration peaks in these steps is less than what is expected. It is seen from Figure 4.5 that the depicted gait cycle fails to have the required number of peaks. In order to eliminate this data's, all data sets with peaks less than four are removed. That is because as seen from Figure 4.3, the gait cycle is expected to have four peaks. See appendix A.

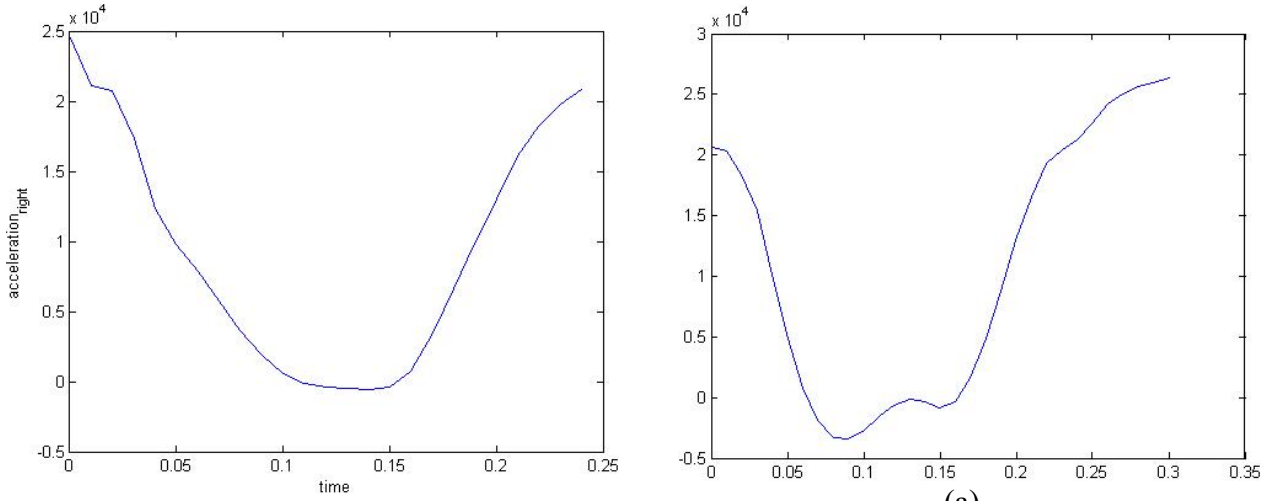


Figure 4-5 Unfinished gait cycle from IMU sensor for (a) Right foot (b) Left foot

4.2 Fitting model

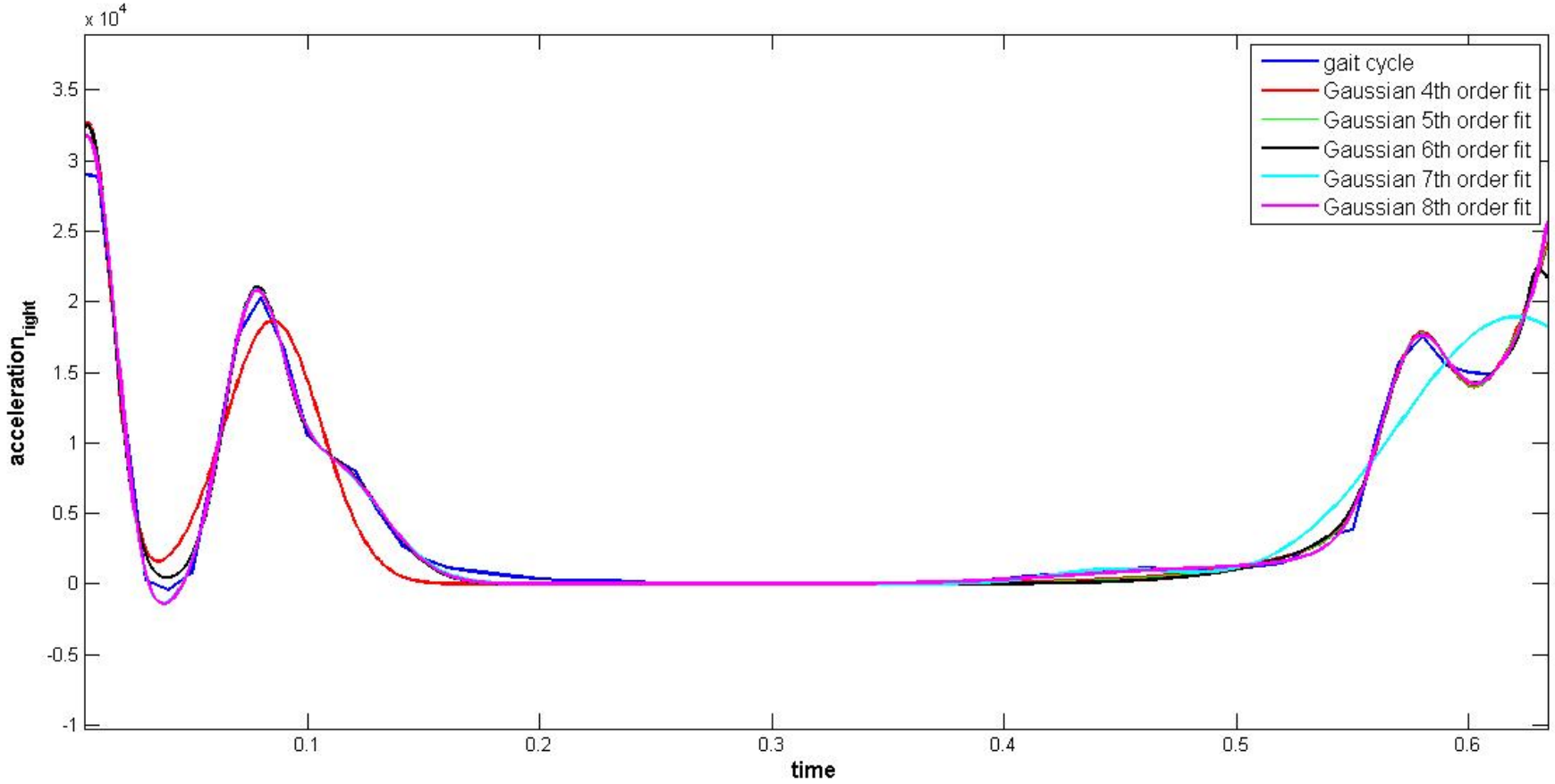
As shown in the flow chart of the algorithm after the raw data was pre-processed it was curve fitted. Curve fitting was need to eliminate individual gait cycle differences and difference from one trial to the next. Also, to eliminate some of the noises from the sensor data as explained in section 3.2.3. In order to come up with a best curve fit model, fitting of the Vicon sample data using different methods are checked by plotting each fit verses the discretized gait cycle. Gaussian, polynomial and Fourier models are cross checked, as shown in appendix B1. As seen from the figures a relatively proper depicted patterns of the gait cycle is illustrated in Gaussian curve fit. In addition to the Vicon gait cycle fit, gait cycle from IMU sensors for all conditions was checked for the above stated curve fitting models, as shown in appendix B2. Results from this figures also show Gaussian model as a proper fitting for the gait cycles.

Even though Gaussian fits from 4th till 8th order show relatively similar confirmation, an 8th order Gaussian fit was chosen. This is because only 8th order fit properly fits the data set without losing it proper patter , as shown in Figure 4.6. The figure shows a 4th-8th order Gaussian fitted IMU normal gait cycle plus the gait cycle before fitting for both the right and left foot. This is to cross check which order fits the pattern of the gait cycle properly. It is illustrated in the figure that most of the orders fail to depict all the pattern of the gait cycles as that of the 8th order. So Gaussian 8th order was used to fitted all of the gait cycles. Curve fitting tool box in Mat lab (cftool) was used and the generated code is shown in appendix A. Sample fitted gait cycles for each data type for both the right and left foot is shown in appendix C.

4.3 Feature extraction

Once each gait cycle was fitted properly next stage as depicted in the flow chart was to extract feature which could express the gait cycle. As stated in the reviewed literatures in section 2.4.1, extraction of feature requires first understanding the deviation at hand. In order to achieve that observation was made of each type of pathologies relative to the normal data set and relative to each other, as shown in Figure 4.7 & 4.8. It can be observed from the figure that, basic deviation of the gait cycles was on the height of peak, position of the peak, and the number of peak. This is deviation is also explained in [4, 36], as reviewed in section 2.3.3.

Derived from the two findings it is assumed that the coefficients of the Gaussian fit are enough to differentiate the data set. Since Gaussian coefficients incorporate the height of the curve's peak (a) , the position of the center of the peak (b) and the standard deviation, that controls the width of the bell (c). The extracted features are then added to one cell called X. The size of X differs for each type of algorithm as explained in section 3.2.3. The overall feature extraction is shown in appendix A.


x 10^4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
acceleration_right
0.1
0.2
0.3
0.4
0.5
0.6
time
gait cycle
Gaussian 4th order fit
Gaussian 5th order fit
Gaussian 6th order fit
Gaussian 7th order fit
Gaussian 8th order fit

(a)

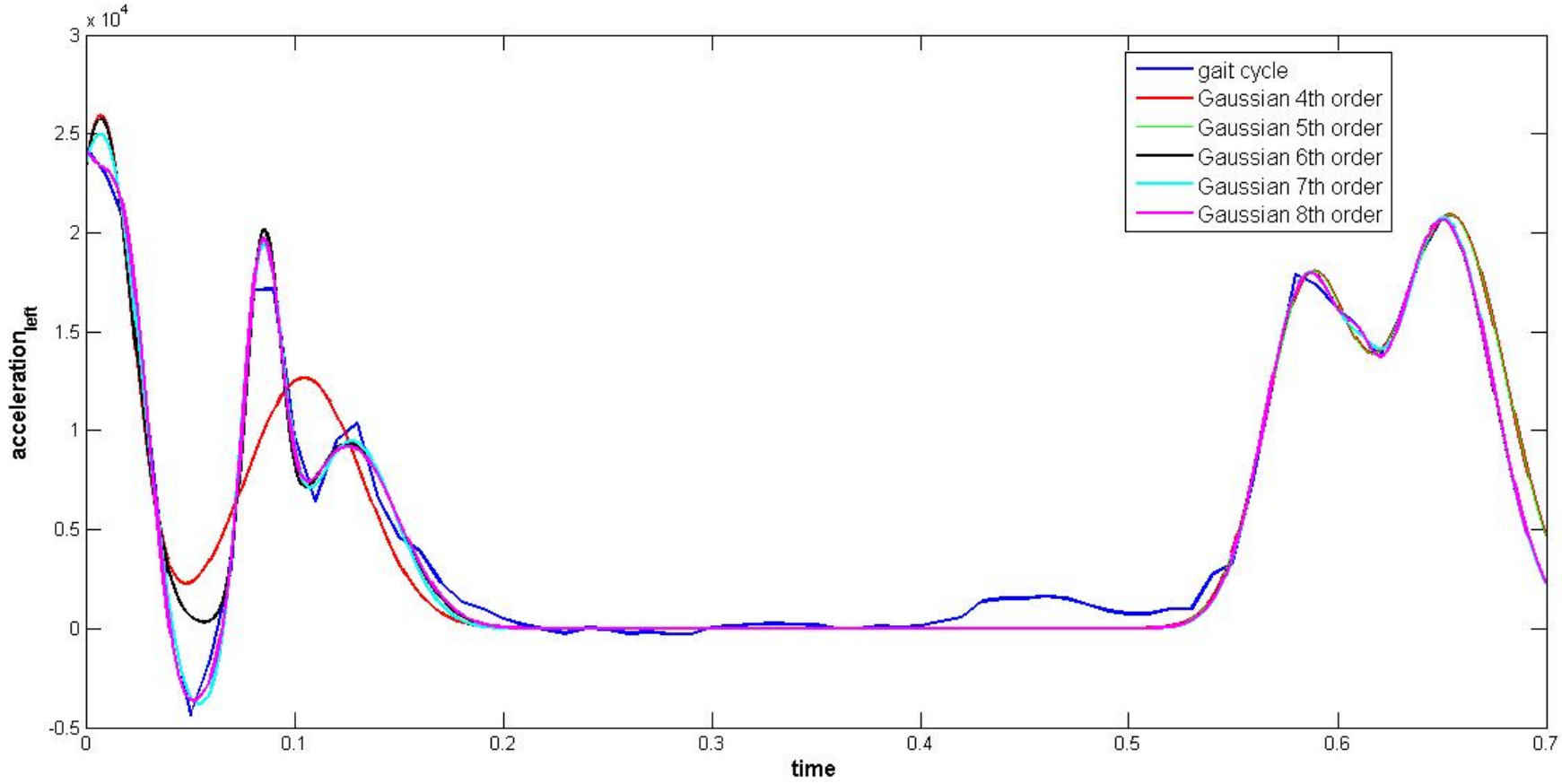
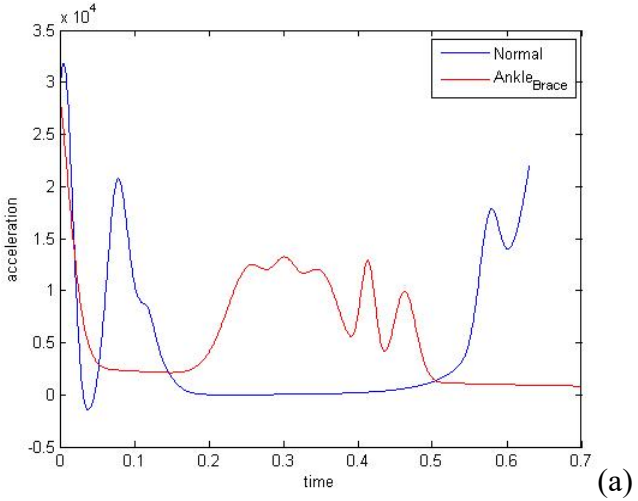
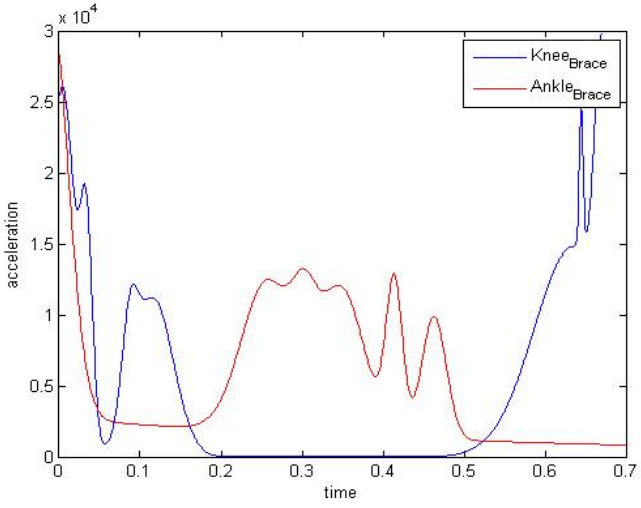


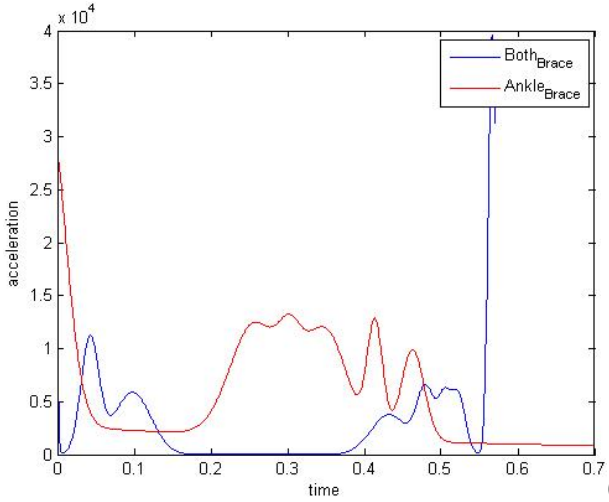
Figure 4-6 Gaussian fits for gait cycle at different orders for the (a) Right foot (b) Left foot


x 10^4
Normal
Ankle_Brace
acceleration
time

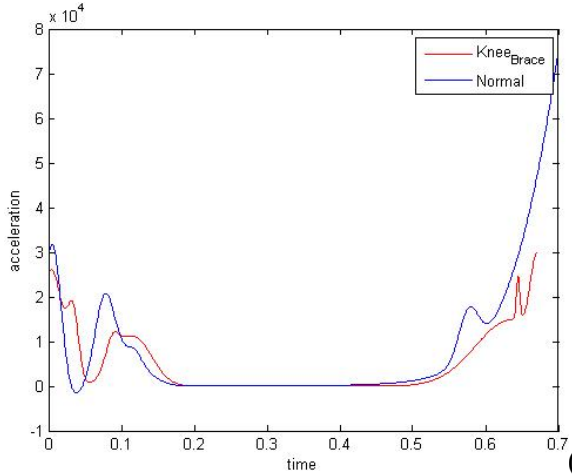
(a)


x 10^4
Knee_Brace
Ankle_Brace
acceleration
time

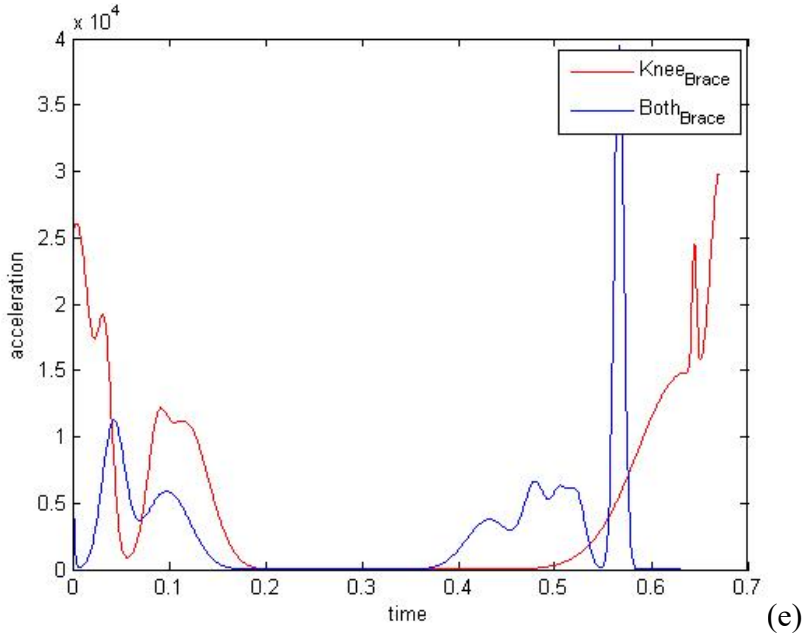
(b)


x 10^4
Both_Brace
Ankle_Brace
acceleration
time

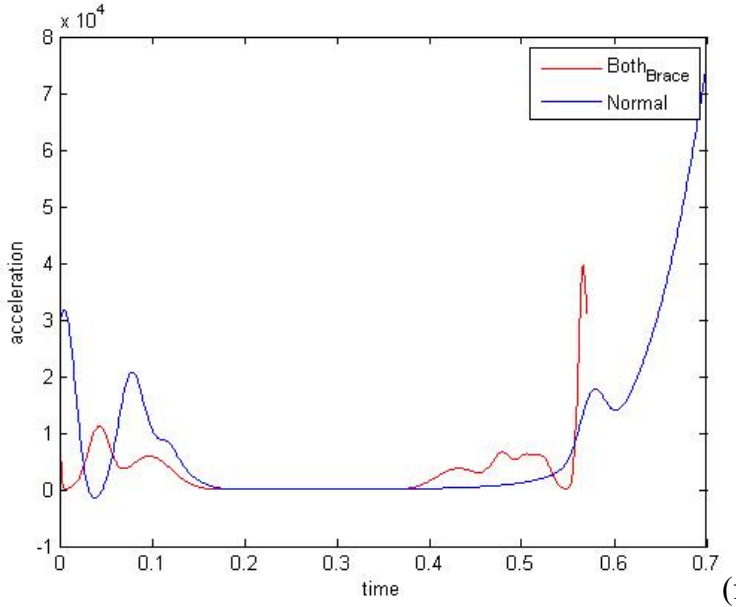
(c)


x 10^4
Knee_Brace
Normal
acceleration
time

(d)

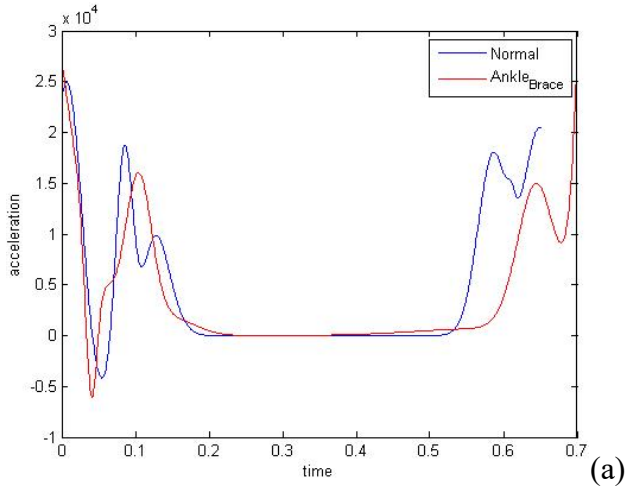


(e)

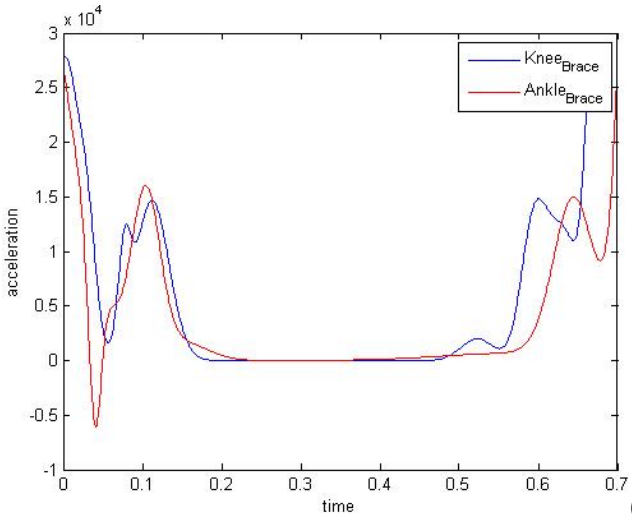


(f

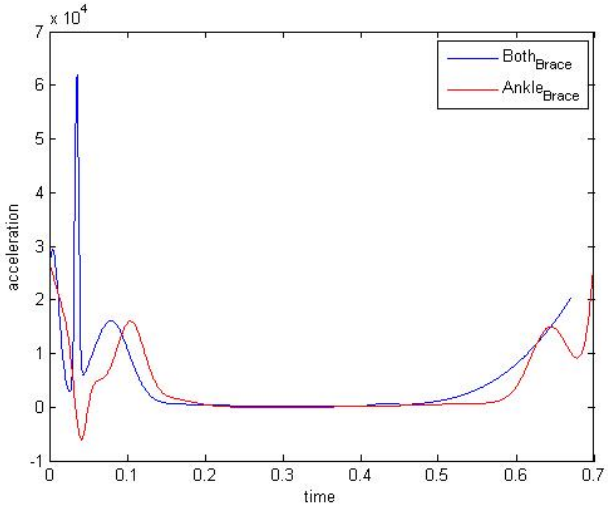
Figure 4-7 Acceleration gait cycle of Right foot (a) Ankle Brace in relation to Normal (b) Ankle Brace in relation to Knee Brace (c) Ankle Brace in relation to Ankle & Knee Brace (d) Knee Brace in relation to Normal (e) Knee Brace in relation to Ankle & Knee Brace (f) Ankle & Knee Brace in relation to Normal gait


x 10^4
Normal
Ankle_Brace
acceleration
time

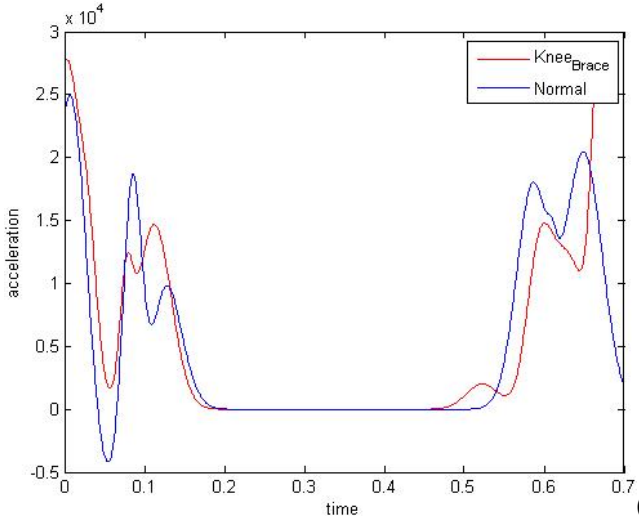
(a)


x 10^4
Knee_Brace
Ankle_Brace
acceleration
time

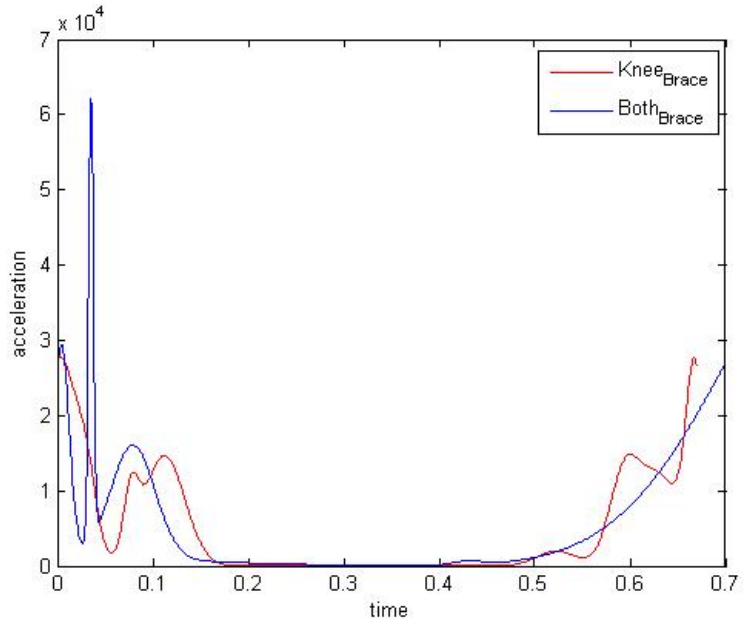
(b)


x 10^4
Both_Brace
Ankle_Brace
acceleration
time

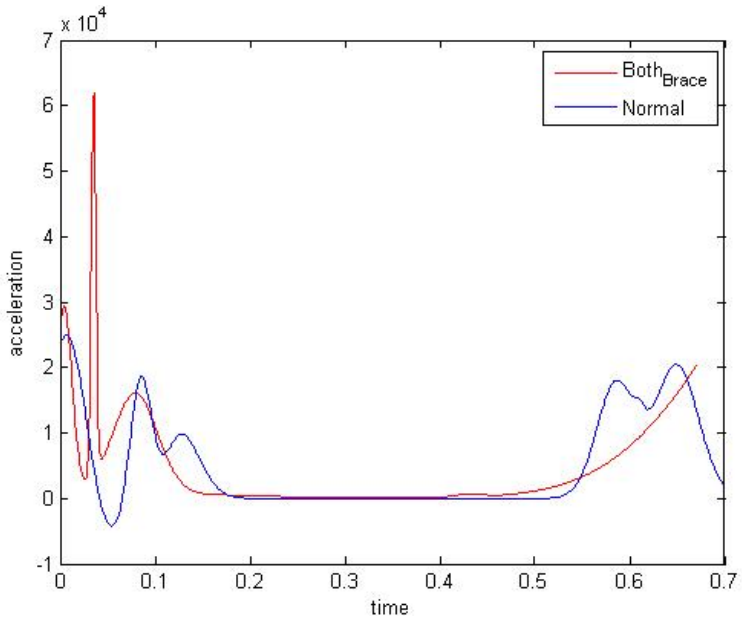
(c)


x 10^4
Knee_Brace
Normal
acceleration
time

(d)



(e)



(f)

Figure 4-8 Acceleration gait cycle of Left foot (a) Ankle Brace in relation to Normal (b) Ankle Brace in relation to Knee Brace (c) Ankle Brace in relation to Ankle & Knee Brace (d) Knee Brace in relation to Normal (e) Knee Brace in relation to Ankle & Knee Brace (f) Ankle & Knee Brace in relation to Normal gait

4.4 Classification

Once distinguishing features are selected the next step was to model the data set as a classifiers. In order to achieve this things three major steps are defined:

1. Assigning respective classes for all data set, which are the true responses of the acquired data. The assigned classes are added to one cell called Y, see appendix A. The size of Y also differs for each type of algorithm, similar to the X data.

2. The feature cell (X) and the class cell (Y) are partitioned. It is used to define test set and training set for validating a statistical model using cross validation. A hold out type partitioning is done, which divides the observation (n) into training set and hold out set (test set). It will leave p (left out percentage) * n (observation) observations for the test set. As stated in section 3.2.3, 30% of the data set are left for test set. See appendix A.

3. The partitioned data sets (X_Train & Y_Train) are then modeled using Naive Bayes, Tree Bagger, K-Nearest Neighbor Algorithm (KNN), and Support Vector Machine (SVM) classifiers. This classifiers were chosen based on the suggestion of several literatures, as illustrated in section 3.2.3. See appendix A. Multi-class classification on the other hand only uses Naive Bayes and Tree Bagger classifiers, since the other only work for binary classification as stated in section 3.2.3. See appendix A1.

Chapter 5

Performance Analysis

In this chapter, the classification algorithms performance is evaluated, based on the selected evaluation parameters. The main objective of performance evaluation is to increase the effectiveness, efficiency and adaptability of Orthotic bracing so as to add more value to the patients walking ability. One of the key determining factor for the classification algorithm's performance is data set characteristics. Researches divide the data set characteristics in three categories [54]. The focus of this paper is on assessing one of the categories of the data set characteristics, data set Content. It is the characteristics about the value of features and target class in the data set. In addition, the effect of the classifier used is evaluated.

To evaluate the performance of the proposed classification approach, a cross-validation was carried out, as stated in section 3.2.4. The test data (X_Test) was classified using the developed classifier in chapter 4, as presented by the flow chart in Figure 4.1. The predicted result (Y_'name of the classifier') is then cross checked with the true class of the test data (Y_Test) for evaluation, see appendix A.

5.1 Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It is used to show the relationships between outcomes and predicted classes. Effectiveness of a classification

model is determined by the measure of correct and incorrect classification in each possible value of the variable being classified [41].

It contains information about actual and predicted classifications done by a classification system. The data found in the confusion matrix and the format of a matrix is shown in Table 5.1. The values in the matrix have the following meaning:

- TC_1 represents the correct predictions that a given instance is class 1,
- FC_1 represents the incorrect predictions that a given instance is class 1.
- TC_2 represents the correct predictions that a given instance is class 2,
- FC_2 represents the incorrect predictions that a given instance is class 2.

		Predicted	
		Class 1	Class 2
Actual	Class 1	T C_1	F C_2
	Class 2	F C_1	T C_2

Table 5-1 Confusion matrix format

In order to develop the confusion matrix the partitioned test data set is used. That is, the test data (X_Test) is predicted (Y_(classifier name) and the accuracy is cross checked with the true class of the test data (Y_Test), see appendix A. The developed confusion matrix is shown in Table (5.2-5.3), it is the average result of 10 consecutive algorithm runs, as stated in section 3.2.4.

Naïve Bayes					Tree Bagger				
	Normal	Knee_Brace	Ankle_Brace	Both_Brace		Normal	Knee_Brace	Ankle_Brace	Both_Brace
Normal	10	3	4	2	Normal	30	4	2	1
Knee_Brace	4	2	1	1	Knee_Brace	13	3	2	0
Ankle_Brace	3	2	1	1	Ankle_Brace	6	2	9	4
Both_Brace	3	1	0	1	Both_Brace	4	1	5	9

(a)

Naïve Bayes					Tree Bagger				
	Normal	Knee_Brace	Ankle_Brace	Both_Brace		Normal	Knee_Brace	Ankle_Brace	Both_Brace
Normal	10	5	10	2	Normal	24	1	5	0
Knee_Brace	3	2	6	1	Knee_Brace	5	2	5	0
Ankle_Brace	2	2	2	1	Ankle_Brace	4	1	2	0
Both_Brace	2	1	5	1	Both_Brace	1	2	4	4
KNN					SVM				
	Normal	Knee_Brace	Ankle_Brace	Both_Brace		Normal	Knee_Brace	Ankle_Brace	Both_Brace
Normal	24	2	5	0	Normal	27	2	6	0
Knee_Brace	6	1	4	1	Knee_Brace	8	4	3	1
Ankle_Brace	6	1	1	0	Ankle_Brace	8	1	2	0
Both_Brace	5	1	2	0	Both_Brace	2	3	3	2

(b)

Naïve Bayes			Tree Bagger		
	Normal	Abnormal		Normal	Abnormal
Normal	11	8	Normal	21	18
Abnormal	7	10	Abnormal	11	46
KNN			SVM		
	Normal	Abnormal		Normal	Abnormal
Normal	15	23	Normal	26	12
Abnormal	13	44	Abnormal	19	38
Naïve Bayes			Tree Bagger		
	Indiviual_Brace	Both_Brace		Indiviual_Brace	Both_Brace
Indiviual_Brace	9	3	Indiviual_Brace	36	4
Both_Brace	2	1	Both_Brace	12	6
KNN			SVM		
	Indiviual_Brace	Both_Brace		Indiviual_Brace	Both_Brace
Indiviual_Brace	32	7	Indiviual_Brace	26	13
Both_Brace	15	3	Both_Brace	8	10
Naïve Bayes			Tree Bagger		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	4	3	Knee_Brace	13	6
Ankle_Brace	3	2	Ankle_Brace	4	16
KNN			SVM		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	14	5	Knee_Brace	11	7
Ankle_Brace	11	10	Ankle_Brace	5	15

(c)

Naïve Bayes			Tree Bagger		
	Normal	Ankle_Brace		Normal	Ankle_Brace
Normal	14	5	Normal	35	3
Ankle_Brace	3	2	Ankle_Brace	7	13
KNN			SVM		
	Normal	Ankle_Brace		Normal	Ankle_Brace
Normal	37	2	Normal	32	6
Ankle_Brace	14	6	Ankle_Brace	9	11

(d)

Naïve Bayes			Tree Bagger		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	4	3	Knee_Brace	13	6
Ankle_Brace	3	2	Ankle_Brace	4	16
KNN			SVM		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	14	5	Knee_Brace	11	7
Ankle_Brace	11	10	Ankle_Brace	5	15

(e)

Naïve Bayes			Tree Bagger		
	Ankle_Brace	Both_Brace		Ankle_Brace	Both_Brace
Ankle_Brace	2	3	Ankle_Brace	14	6
Both_Brace	0	3	Both_Brace	8	10
KNN			SVM		
	Ankle_Brace	Both_Brace		Ankle_Brace	Both_Brace
Ankle_Brace	13	8	Ankle_Brace	10	10
Both_Brace	12	6	Both_Brace	8	10

(f)

Naïve Bayes			Tree Bagger		
	Normal	Knee_Brace		Normal	Knee_Brace
Normal	12	7	Normal	33	6
Knee_Brace	4	3	Knee_Brace	13	5
KNN			SVM		
	Normal	Ankle_Brace		Normal	Knee_Brace
Normal	36	3	Normal	24	14
Knee_Brace	17	2	Knee_Brace	10	9

(g)

Naïve Bayes			Tree Bagger		
	Knee_Brace	Both_Brace		Knee_Brace	Both_Brace
Knee_Brace	5	2	Knee_Brace	14	5
Both_Brace	2	2	Both_Brace	3	15
KNN			SVM		
	Ankle_Brace	Both_Brace		Knee_Brace	Both_Brace
Knee_Brace	8	11	Knee_Brace	13	6
Both_Brace	8	11	Both_Brace	3	16

(h)

Naïve Bayes			Tree Bagger		
	Normal	Both_Brace		Normal	Both_Brace
Normal	14	5	Normal	36	2
Both_Brace	3	1	Both_Brace	6	13
KNN			SVM		
	Normal	Both_Brace		Normal	Both_Brace
Normal	35	3	Normal	30	8
Both_Brace	12	6	Both_Brace	3	15

Table 5-2 Right foot Confusion matrix for (a) Multi-class (b) Binary combination (c) Binary step (d) Ankle Vs Normal (e) Ankle Vs Knee (f) Ankle Vs Both brace (g) Knee Vs Normal (h) Knee Vs Both brace (I) Both brace Vs Normal

Naïve Bayes					Tree Bagger				
	Normal	Knee_Brace	Ankle_Brace	Both_Brace		Normal	Knee_Brace	Ankle_Brace	Both_Brace
Normal	10	3	5	2	Normal	29	5	5	4
Knee_Brace	3	3	2	1	Knee_Brace	10	5	4	3
Ankle_Brace	3	2	2	2	Ankle_Brace	9	3	6	5
Both_Brace	3	3	3	2	Both_Brace	11	4	4	3

(a)

Naïve Bayes					Tree Bagger				
	Normal	Knee_Brace	Ankle_Brace	Both_Brace		Normal	Knee_Brace	Ankle_Brace	Both_Brace
Normal	12	10	16	2	Normal	22	6	13	1
Knee_Brace	3	3	9	1	Knee_Brace	4	3	10	0
Ankle_Brace	2	4	3	0	Ankle_Brace	3	3	3	0
Both_Brace	3	5	7	1	Both_Brace	4	4	7	1
KNN					SVM				
	Normal	Knee_Brace	Ankle_Brace	Both_Brace		Normal	Knee_Brace	Ankle_Brace	Both_Brace
Normal	11	10	18	0	Normal	30	5	4	2
Knee_Brace	6	1	9	0	Knee_Brace	8	5	3	1
Ankle_Brace	2	4	3	0	Ankle_Brace	5	1	3	1
Both_Brace	5	4	7	0	Both_Brace	7	5	4	1

(b)

Naïve Bayes			Tree Bagger		
	Normal	Abnormal		Normal	Abnormal
Normal	12	10	Normal	18	26
Abnormal	12	19	Abnormal	12	55
KNN			SVM		
	Normal	Abnormal		Normal	Abnormal
Normal	6	37	Normal	25	18
Abnormal	9	58	Abnormal	25	42
Naïve Bayes			Tree Bagger		
	Indiviual_Brace	Both_Brace		Indiviual_Brace	Both_Brace
Indiviual_Brace	13	5	Indiviual_Brace	38	8
Both_Brace	6	3	Both_Brace	18	4
KNN			SVM		
	Indiviual_Brace	Both_Brace		Indiviual_Brace	Both_Brace
Indiviual_Brace	39	7	Indiviual_Brace	28	18
Both_Brace	19	2	Both_Brace	15	6
Naïve Bayes			Tree Bagger		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	6	4	Knee_Brace	10	12
Ankle_Brace	5	4	Ankle_Brace	8	15
KNN			SVM		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	12	10	Knee_Brace	13	9
Ankle_Brace	13	10	Ankle_Brace	11	12

(c)

Naïve Bayes			Tree Bagger		
	Normal	Ankle_Brace		Normal	Ankle_Brace
Normal	14	8	Normal	35	8
Ankle_Brace	4	4	Ankle_Brace	14	10
KNN			SVM		
	Normal	Ankle_Brace		Normal	Ankle_Brace
Normal	37	6	Normal	33	11
Ankle_Brace	18	5	Ankle_Brace	12	11

(d)

Naïve Bayes			Tree Bagger		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	6	4	Knee_Brace	10	12
Ankle_Brace	5	4	Ankle_Brace	8	15
KNN			SVM		
	Knee_Brace	Ankle_Brace		Knee_Brace	Ankle_Brace
Knee_Brace	12	10	Knee_Brace	13	9
Ankle_Brace	13	10	Ankle_Brace	11	12

(e)

Naïve Bayes			Tree Bagger		
	Ankle_Brace	Both_Brace		Ankle_Brace	Both_Brace
Ankle_Brace	4	5	Ankle_Brace	13	10
Both_Brace	4	5	Both_Brace	12	9
KNN			SVM		
	Ankle_Brace	Both_Brace		Ankle_Brace	Both_Brace
Ankle_Brace	11	12	Ankle_Brace	13	10
Both_Brace	9	12	Both_Brace	10	11

(f)

Naïve Bayes			Tree Bagger		
	Normal	Knee_Brace		Normal	Knee_Brace
Normal	11	6	Normal	36	8
Knee_Brace	4	5	Knee_Brace	12	11
KNN			SVM		
	Normal	Ankle_Brace		Normal	Knee_Brace
Normal	37	7	Normal	33	11
Knee_Brace	17	6	Knee_Brace	10	12

(g)

Naïve Bayes			Tree Bagger		
	Knee_Brace	Both_Brace		Knee_Brace	Both_Brace
Knee_Brace	5	3	Knee_Brace	13	10
Both_Brace	5	4	Both_Brace	10	11
KNN			SVM		
	Ankle_Brace	Both_Brace		Knee_Brace	Both_Brace
Knee_Brace	8	14	Knee_Brace	11	11
Both_Brace	8	14	Both_Brace	14	8

(h)

Naïve Bayes			Tree Bagger		
	Normal	Both_Brace		Normal	Both_Brace
Normal	12	8	Normal	34	9
Both_Brace	4	4	Both_Brace	14	7
KNN			SVM		
	Normal	Both_Brace		Normal	Both_Brace
Normal	35	8	Normal	29	14
Both_Brace	18	3	Both_Brace	11	10

Table 5-3 Left foot Confusion matrix for (a) Multi-class (b) Binary combination (c) Binary step (d) Ankle Vs Normal (e) Ankle Vs Knee (f) Ankle Vs Both brace (g) Knee Vs Normal (h) Knee Vs Both brace (I) Both brace Vs Normal

5.2 Performance indicators

The performance of each classifier was evaluated by using performance indicators such as sensitivity (TPR), specificity (TNR), and accuracy. In order to calculate the indicators a confusion matrix is developed for each algorithm.

5.2.1 Sensitivity and Specificity

Sensitivity and specificity are statistical measures of the performance of a classification test. Sensitivity is a measure of the proportion of true positives from the overall actual positive data set. It is the ratio of the True Positive (the number of correctly predicted cases) to the sum of the True Positive and the False Negative [41, 55].

Specificity is a measure of the proportion of true negative from the overall actual negative data set. It is the ratio of the True Negative (the number of correctly predicted cases) to the sum of the True Negative and the False Positive [41, 55].

Both sensitivity and specificity measure the ability of the classifier to predict the classes from the total actual class. Which means it is the measure of true rate of a specific class. Since in this paper all the bracings are required as positive classes rather than measuring the specificity and sensitivity separately, the true rate with respect to a given data set is calculated. The following expression is used for calculating the true rate:

$$Truerate = \frac{TC}{(TC + FC)}$$

The value of both the true rate is shown in Table (5.4 - 5.5), for all algorithms.

Classifier	Multi-class	Binary combination	Binary step	Ankle Vs Normal	Knee Vs Normal	Both Vs Normal
			Step 1			
Naïve Bayes	54%	26%	57%	72%	62%	75%
Tree Bagger	80%	64%	54%	93%	85%	94%
KNN		63%	39%	96%	93%	92%
SVM		71%	68%	85%	64%	80%

(a)

Classifier	Multi-class	Binary combination	Binary step	Ankle Vs Normal	Knee Vs Ankle	Both Vs Ankle
			Step 3			
Naïve Bayes	14%	20%	38%	37%	38%	39%
Tree Bagger	42%	16%	79%	63%	79%	68%
KNN		11%	49%	32%	49%	63%
SVM		15%	74%	57%	74%	51%

(b)

Classifier	Multi-class	Binary combination	Binary step	Knee Vs Normal	Knee Vs Ankle	Both Vs Knee
			Step 3			
Naïve Bayes	24%	12%	53%	39%	53%	68%
Tree Bagger	15%	10%	68%	29%	68%	75%
KNN		8%	73%	8%	73%	42%
SVM		24%	62%	48%	62%	69%

(c) `

Classifier	Multi-class	Binary combination	Binary step Step 2	Both Vs Normal	Both Vs Ankle	Both Vs Knee
Naïve Bayes	21%	4%	35%	28%	88%	56%
Tree Bagger	46%	25%	32%	69%	54%	82%
KNN		3%	14%	32%	34%	58%
SVM		16%	57%	81%	56%	86%

Table 5-4 Right foot True rate for (a) Normal gait cycle (b) Ankle brace gait cycle (c) Knee brace gait cycle (d) Ankle and Knee brace gait cycle

Classifier	Multi-class	Binary combination	Binary step Step 1	Ankle Vs Normal	Knee Vs Normal	Both Vs Normal
Naïve Bayes	50%	27%	55%	65%	63%	62%
Tree Bagger	66%	50%	41%	81%	82%	78%
KNN		25%	14%	85%	85%	81%
SVM		69%	57%	76%	74%	68%

(a)

Classifier	Multi-class	Binary combination	Binary step Step 3	Ankle Vs Normal	Knee Vs Ankle	Both Vs Ankle
Naïve Bayes	25%	26%	46%	49%	46%	49%
Tree Bagger	24%	30%	65%	41%	65%	58%
KNN		27%	44%	22%	44%	50%
SVM		27%	51%	49%	51%	55%

(b)

Classifier	Multi-class	Binary combination	Binary step Step 3	Knee Vs Normal	Knee Vs Ankle	Both Vs Knee
Naïve Bayes	35%	18%	60%	52%	60%	60%
Tree Bagger	24%	14%	44%	48%	44%	57%
KNN		8%	55%	25%	55%	36%
SVM		27%	57%	54%	57%	49%

(c)

Classifier	Multi-class	Binary combination	Binary step Step 2	Both Vs Normal	Both Vs Ankle	Both Vs Knee
Naïve Bayes	17%	6%	33%	50%	55%	44%
Tree Bagger	14%	3%	19%	32%	44%	52%
KNN		0%	11%	13%	58%	64%
SVM		6%	81%	46%	54%	38%

Table 5-5 Left foot True rate for (a) Normal gait cycle (b) Ankle brace gait cycle (c) Knee brace gait cycle (d) Ankle and Knee brace gait cycle

5.2.2 Accuracy

Accuracy is known as the percentage of records that is correctly predicted by the model. It is the ratio of correctly predicted cases to the total number of cases [52]. The expression used is:

$$Accuracy = \frac{Total\ TC}{Total\ TC + Total\ FC}$$

The accuracy of each algorithm is shown in Table (5.6).

Classifier	Multi-class	Binary combination	Binary step				Ankle Vs Normal	Ankle Vs Knee	Ankle Vs Both	Knee Vs Normal	Knee Vs Both	Both Vs Normal
			Step 1	Step 2	Step 3	Overall						
Naïve Bayes	38%	19%	57%	68%	47%	57%	65%	47%	59%	56%	63%	66%
Tree Bagger	53%	37%	70%	72%	74%	72%	83%	74%	61%	66%	79%	86%
KNN		31%	62%	61%	61%	61%	74%	61%	49%	65%	50%	73%
SVM		41%	67%	64%	68%	67%	75%	68%	53%	58%	78%	80%

(a)

Classifier	Multi-class	Binary combination	Binary step				Ankle Vs Normal	Ankle Vs Knee	Ankle Vs Both	Knee Vs Normal	Knee Vs Both	Both Vs Normal
			Step 1	Step 2	Step 3	Overall						
Naïve Bayes	36%	19%	58%	59%	54%	57%	61%	54%	52%	59%	52%	58%
Tree Bagger	38%	28%	66%	62%	55%	61%	67%	55%	51%	71%	55%	63%
KNN		15%	58%	62%	50%	57%	63%	50%	54%	65%	50%	59%
SVM		37%	61%	51%	54%	55%	66%	54%	55%	67%	43%	61%

(b)

Table 5-6 Accuracy of (a) Right foot (b) Left foot for all algorithm

5.3 Performance analysis

5.3.1 Type of class

One of the factors which affect the evaluation of performance is the type of class used. Whether the class value is binary, multiple, or categorical may influence the performance of the classifier algorithms. The discussion of this section focus on assessing the effect of the type class used on our data set.

Four type of classes are used for the discussion, the first two algorithms uses all the class types (responses). The distinguish between the two is that the first classifies uses all verse all classification method, whereas the second one uses one verse all classification method. All verse all classification considers all the bracing type and the normal gait cycle as one data set. Whereas the one verse all classifier considers each bracing and normal gait cycle one by one and combine the prediction result.

The third algorithm applies a step wise classification. All bracing considered as Abnormal gait cycle are classified with the normal gait cycle. The abnormal data is further classified as Individual brace and Both brace. The Individual brace is then classified as Ankle and Knee brace. The final algorithm set uses categorical classification, which considers classification of each type of data verses another data set. The aim of this type of classification is to see the effect of the extracted feature on each type of the data set.

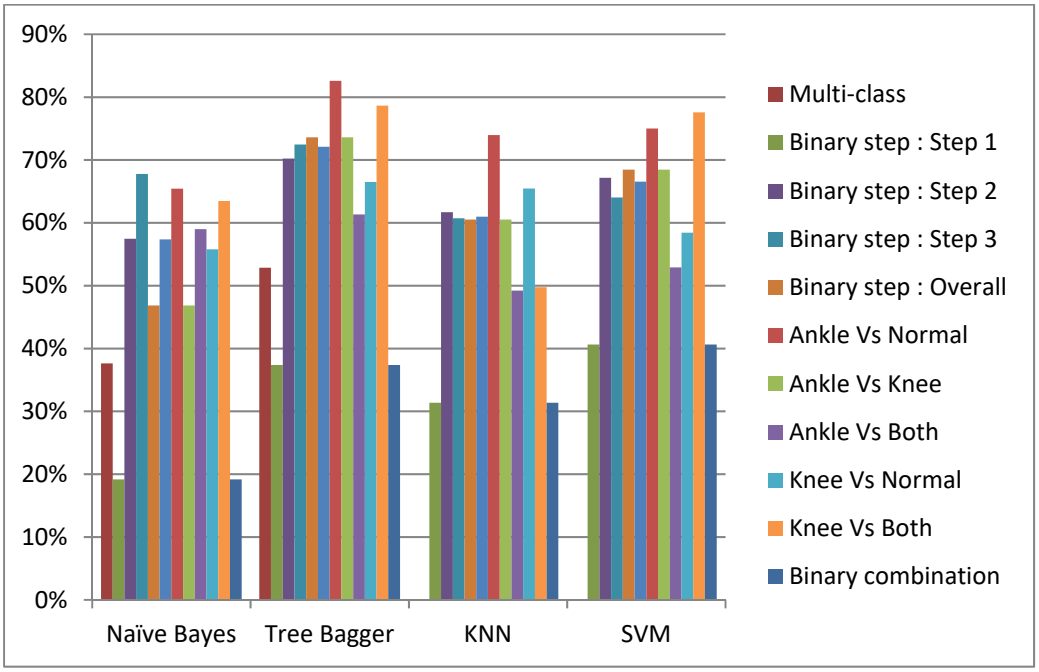


Figure 5-1 Accuracy of all classification algorithms for the right foot

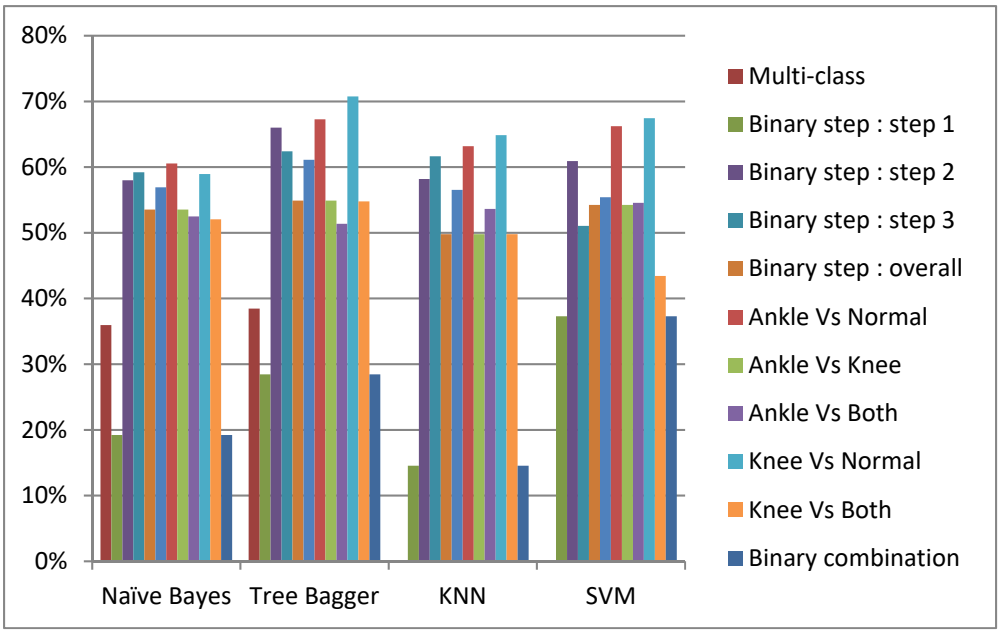


Figure 5-2 Accuracy of all classification algorithms for the left foot

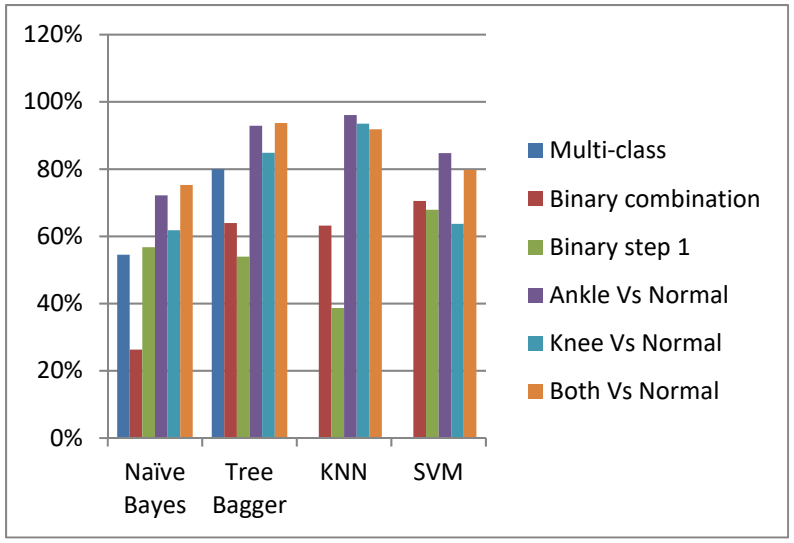
The accuracy of the multi-class classification algorithm shows a maximum of 53% for the right foot and a maximum of 38% for the left foot, as shown in Figure 5.1 and 5.2. Lower accuracy of the left foot is seen since the bracing is suited in the right foot and only traces from it is shown in the left foot. Comparing this with the binary combination classification algorithm it shows a small increment. The accuracy of the binary combination has a maximum of 41% for the right foot and 37% for the left foot. This small difference is the result of the error induced by the combiner. This implies that if the error from the combiner can be removed or minimized, using either of the two classification algorithms will give as similar accuracy result.

The use of a step wise binary classification increases the accuracy of the overall classification up to 72% for the right foot and 61% for the left foot. This implies that almost 30% of the data set are misclassified. As compared to a classification algorithm done by Mannini and colleagues [6], this shows a huge performance gap. This performance gap could be because of the chosen features sets. The features set might not be perfect for distinguishing all of the data sets. A way to minimize this and increase the accuracy is by applying a categorical classification. This is seen by most of the categorical classification accuracy shown in Figure 5.1-5.2. Highest accuracy of the right foot is seen in Both brace verses Normal gait cycle classification, having a value of 86%. Whereas the left foot higher accuracy value is 71%, which is the classification of Knee brace to normal gait cycle data set. The implication of the specific bracing accuracy effect is discussed in detail in section 5.2.3.

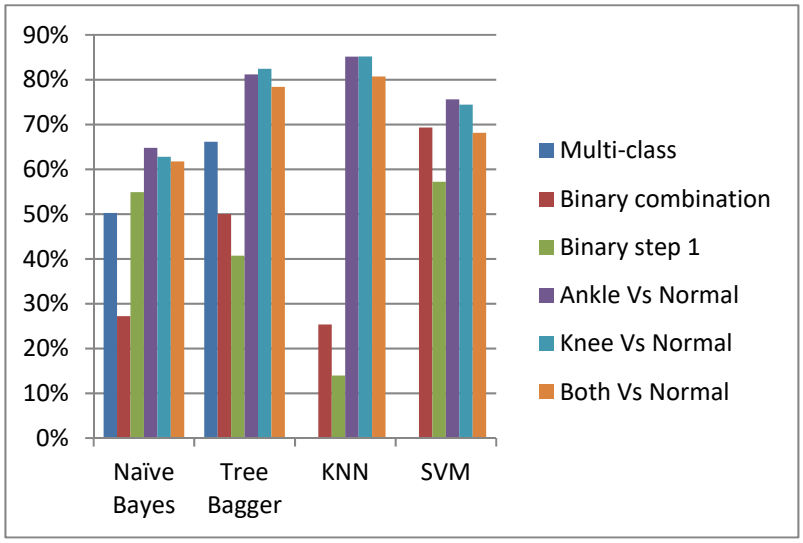
In addition to the overall accuracy the performance of an algorithm is evaluated based on the true rate, as stated in section 5.1.2. Which shows the rate at which the algorithm detects a certain type of data set. According to the plot of the true rate for each data type, shown in Figure 5.3 (a) and (b), Normal gait cycle shows a high rate of overall accurate classification. The categorical classification methods shows a true rate of above 90% for the right foot Normal gait cycle. Next to that Multi-class classification shows a 80 % true rate, whereas Binary combination and Binary step classification shows a relative low rate. The rate of the left foot Normal gait cycle also follows a similar pattern with a lower value, except Binary combination shows a higher true rate than that of Multi-class classification. The difference between the two methods is very small, which adds to the previous discussion on the similarity of the value gain from both the methods.

True rate of the Ankle_Brace shown in Figure 5.3 (c) and (d), illustrates that the overall value is on average 50%. The Binary step 3 classification which is similar to Ankle brace verse Knee brace classification shows a relatively higher rate of classification for both the right and left foot, with maximum value of 79% and 65% respectively. The Knee brace true rate shown in Figure 5.3 (e) and (f), also illustrates a similar pattern with a maximum value of 73% and 60% for the right and left foot respectively.

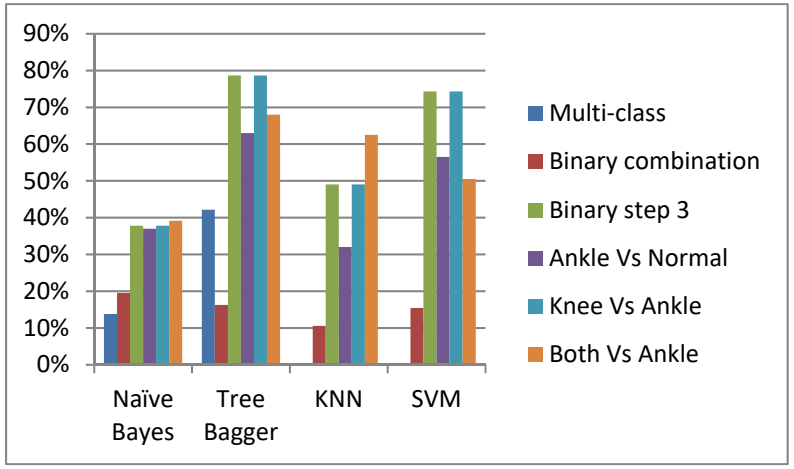
Ankle & Knee brace shows a relatively lower overall true rate, as shown in Figure 5.3 (g) and (h). A relatively higher rate is seen in Binary step and categorical classification. This is because when it comes to an algorithm with a relatively lower accuracy the true rate is a tradeoff between different data types. So, most of the algorithms show a high value of classification to one of the data types.


120%
100%
80%
60%
40%
20%
0%
Naïve Bayes
Tree Bagger
KNN
SVM
Multi-class
Binary combination
Binary step 1
Ankle Vs Normal
Knee Vs Normal
Both Vs Normal

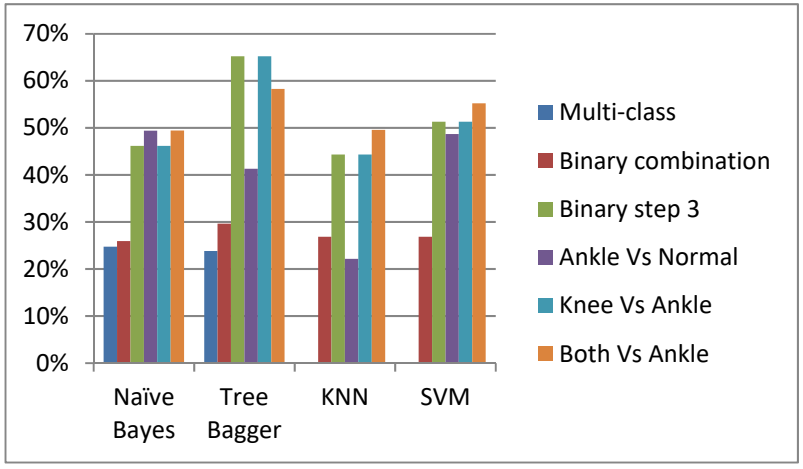
(a)


90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Naïve Bayes
Tree Bagger
KNN
SVM
Multi-class
Binary combination
Binary step 1
Ankle Vs Normal
Knee Vs Normal
Both Vs Normal

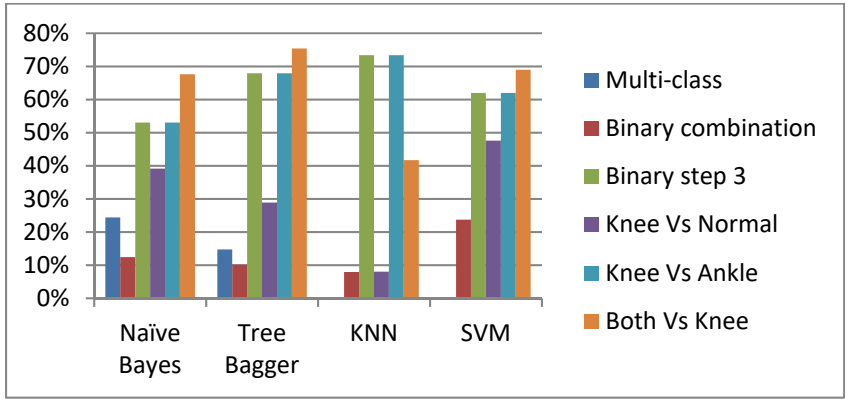
(b)


90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Naïve Bayes
Tree Bagger
KNN
SVM
Multi-class
Binary combination
Binary step 3
Ankle Vs Normal
Knee Vs Ankle
Both Vs Ankle

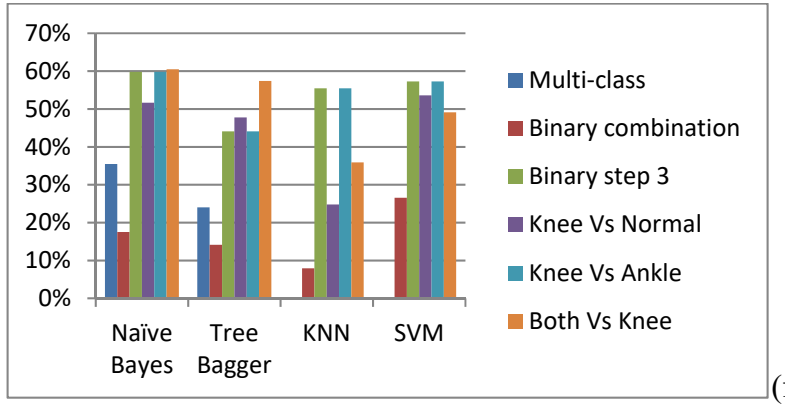
(c)


70%
60%
50%
40%
30%
20%
10%
0%
Naïve Bayes
Tree Bagger
KNN
SVM
Multi-class
Binary combination
Binary step 3
Ankle Vs Normal
Knee Vs Ankle
Both Vs Ankle

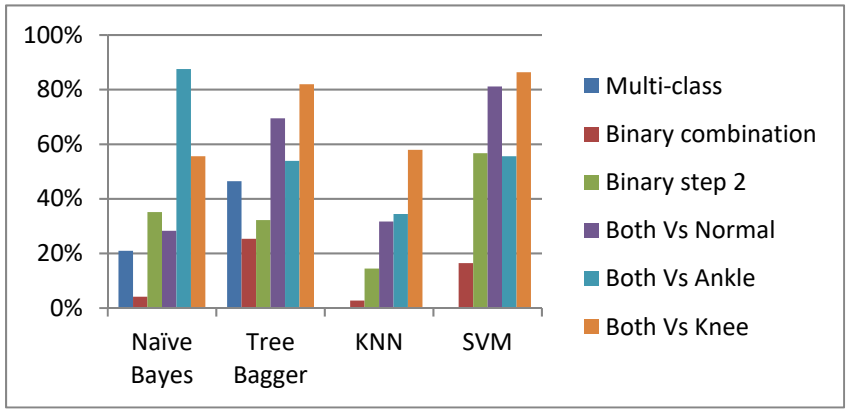
(d)



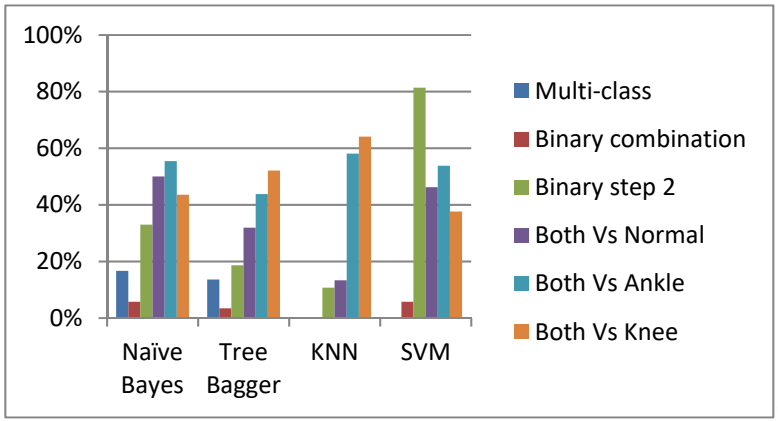
(e)



(f)



(g)



(h)

Figure 5-3 True rate (a) Right foot Normal gait cycle, (b) Left foot Normal gait cycle, (c) Right foot Ankle_Brace gait cycle, (d) Left foot Ankle_Brace gait cycle, (e) Right foot Knee_Brace gait cycle, (f) Left foot Knee_Brace gait cycle, (g) Right foot Ankle & Knee_Brace gait cycle, (h) Left foot Ankle & Knee_Brace gait cycle

Based on the overall discussion of the section categorical classification shows a higher value of accuracy and true rate. Binary step classification gives a relatively higher accuracy with a relatively lower true rate as compared to the categorical classification. One advantage of the Binary step classification over the categorical classification is that it takes all the data types as one data set, rather than classifying each data type separately like that of categorical classification. Since the two classification methods show a relatively higher performance evaluation, most of the discussion from here forward will focus on the two methods.

5.3.2 Type of classifier

In this section the effect of the type classifier used on the accuracy of classification is discussed, by evaluating the performance of each classifier. As stated in section 4.4, Naive Bayes, Tree Bagger, KNN (K-Nearest Neighbor), SVM (Support Vector Machine) classifiers are used. Generally the type of feature set used affects the performance accuracy, so higher accurate classification is yield using highly accurate feature set. Although that is the case, comparison between different classifiers yield a similar effect if a similar feature set is used for the comparison. Based on this each classifier is analyzed based on Binary step classification and Categorical classification.

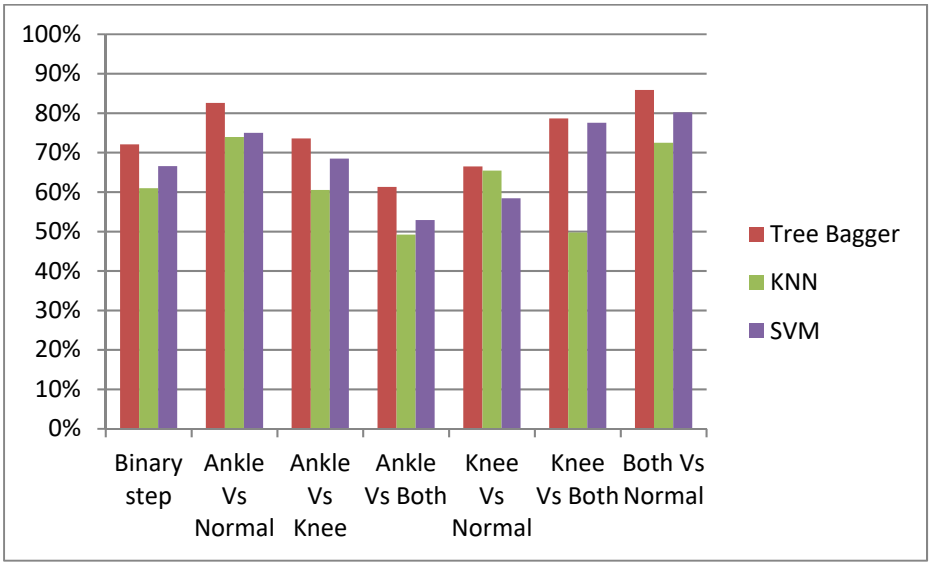


Figure 5-4 Accuracy of all Binary step and Categorical algorithms for the right foot based on type of classifier

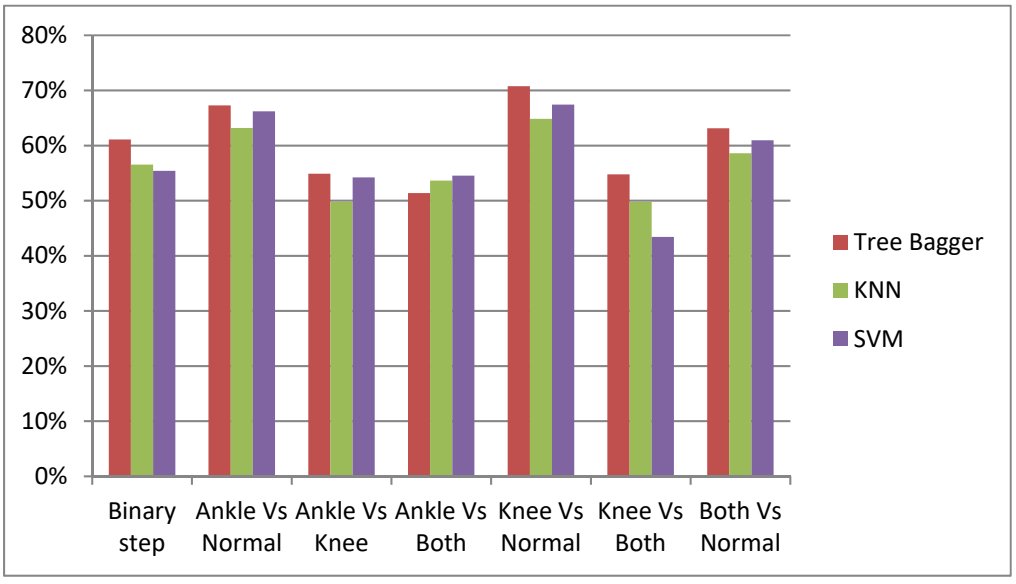


Figure 5-5 Accuracy of all Binary step and Categorical algorithms for the left foot based on type of classifier

The accuracy of the four classifiers is presented in Figure 5.4-5.5. It illustrates that a highest classification accuracy is found using Tree Bagger classifier for both right and left foot, having an average accuracy of 74% and 60% respectively. SVM and KNN classifiers show a similar average accuracy of 68% and 62% respectively for the right foot and 57% for both classifiers for the left foot. Naive Bayes classifier shows a minimum average accuracy of 59% for the right foot and 56% for the left foot. In addition to the low accuracy Naive Bayes classifier considers data sets with negative standard deviation as missing data sets, which excludes some data sets. Therefore, the discussion of the classification performance analysis here forth excludes Naive Bayes classifier.

As stated in section 5.1.2, in addition to the accuracy the performance of classifier also depends on the true rate. Mostly classifiers make a true rate tradeoff between data sets. Ideally a classifier with 100% true rate for each data type is desired. Although that is hard to get, a balanced true rate is preferred.

True rate of the normal gait cycle presents averagely a higher value based on Tree Bagger classifier, as shown in Figure 5.2 (a) and (b). The percentage of KNN classifier is also high except on Binary step classification. SVM classifier shows a closer true rate to that of the Tree Bagger classifier on the left foot. The true rate of the Ankle brace gait cycle also imitates a similar path with a lower value, as presented in Figure 5.2 (c) and (d). SVM classifier presents a higher value than that of KNN classifier.

SVM classifier illustrates a relatively higher true rate as compared to the rest of the classifiers for both Knee brace and Ankle&Knee brace gait cycle, as seen in Figure 5.2 (e-h). Tree Bagger shows an almost similar result for Knee brace gait cycle and a relatively lesser value is presented for Ankle&Knee brace gait cycle. Whereas KNN classifier presents relatively the lowest true rate for both Knee brace and Ankle&Knee brace gait cycles.

Based on the overall discussion of this section Tree Bagger classifier offers a relatively higher accuracy and relatively balanced true rate. SVM also presents a closer balanced true rate and accuracy to that of Tree Bagger classifier. Since the Tree Bagger classifiers show a relatively higher performance evaluation, most of the discussion from here forward will focus on Tree Bagger classifier.

5.3.3 Type of bracing

The aim of classification is to distinguish the pathologies from the normal gait cycle. In that case the specific behavior of the pathology determines the accuracy of the classification. In order to analyze this effect, classification of each specific pathology with each other and the normal gait cycle has been developed.

5.2.3.1 Ankle brace

The true rate of Ankle brace measures the ability of the classifier to distinguish Ankle brace gait cycle from the other gait cycles. This depends on deviation of the Ankle brace acceleration parameters, which are used as feature for classification. In this sub-section Ankle brace gait cycle deviation in relation to the other gait cycles based on the acceleration parameters is discussed.

The true rate result shown in Figure 5.3 (c) illustrates that as compared to the right foot Normal gait cycle 63% of the data set are classified accurately using Tree Bagger classifier. An increase of the rate is seen as compared to the right foot Knee brace gait cycle and Ankle&Knee brace gait cycle, with a value of 79% and 68% respectively. This implies that the Ankle brace gait cycle shows a relatively higher similarity to that of Normal gait cycle and a respectively lower similarity to Ankle&Knee brace gait cycle and Knee brace gait cycle in that order. This is also can be seen form a multi-class classification confusion matrix in Table 5.2(a). As seen in the table most of the misclassification of the Ankle brace data is to Normal data, next to that to Ankle&Knee brace data and Knee brace data respectively.

The result from true rate implies that right foot Ankle brace gait cycle shows a similarity of 37% with the Normal gait cycle. This contradicts with the acceleration plot of the two gait cycle show in Figure 4.7(a), which illustrates the deviation of the two gait cycles. Acceleration plot of the Ankle brace and Ankle&Knee brace in Figure 4.7(b) also shows a more larger deviation than 68% which is found using the classification algorithm. Similarly the plot of Ankle brace and Ankle&Knee brace in Figure 4.7(c) presents the deviation of the two gait cycles more than 79%

found from the classifier. This is also stated on literatures related to deviation due to bracing [4, 36]. This implies that the extracted feature set for the right foot Ankle brace gait cycle shows a certain gap in expressing the full identity of the gait cycle.

As compared to the right foot, the true rate of the left foot Ankle brace gait cycle shows a decrease in value. This implies that the effect of Ankle brace on the unbraced foot is lower than the braced foot, similar to the founding on [4]. Like that of the right foot the left foot Ankle bracing also shows a similarity to Normal, Ankle&Knee brace and Knee brace gait cycle in that order. This is also seen in Multi-class classification confusion matrix Table 5.3(a). That is the misclassification of the Ankle brace toward Normal, Ankle&Knee brace and Knee brace gait cycle in that order.

The result of the true rate, shown in Figure 5.3(d), implies that left foot Ankle Brace gait cycle shows a similarity of 59%, 42% and 35% with Normal, Ankle&Knee brace and Knee brace gait cycle respectively. Which slightly contradicts to the acceleration plot seen in Figure 4.8(a,b&c). Based on the plot the gait cycles show a larger difference than what we get from the classifier. This is also stated on literatures related to deviation due to bracing [4]. This implies that the extracted feature set for the left foot Ankle brace gait cycle shows a certain gap in expressing the full identity of the gait cycle as that of the right foot.

Even if the extracted features for the Ankle bracing shows a certain gap, the average result of the true rate claim that the extracted features to some extent express the Ankle brace gait cycle. The right and left foot of the Ankle brace has been expressed on average 58% and 47% respectively.

This means correction of the specified features will to some extent decrease deviation of gait cycle due to Ankle bracing.

5.2.3.2 Knee brace

The true rate of Knee brace measures the ability of the classifier to distinguish Knee brace gait cycle from the other gait cycles. This depends on deviation of the Knee brace acceleration parameters, which are used as feature for classification. In this sub-section Knee brace gait cycle deviation in relation to the other gait cycles based on the acceleration parameters is discussed.

Result from the true rate in Figure 5.3 (e) and (f) illustrates that as compared to the Normal gait cycle, it shows 29% and 48% deviation for the right and left foot respectively. This implies that Knee brace shows more effect on unbraced than the braced foot. In addition to that, the effect of the Knee bracing on the unbraced leg is larger as compared to the effect of the Ankle bracing. The deviation of Knee brace from the Ankle brace shows a increase in the right foot from the Normal gait cycle. Whereas the left foot shows a slight decrease, with value of 68% and 44% respectively.

The deviation of the Knee brace gait cycle as compared to the Ankle&Knee brace gait cycle shows an increase in contrast to Normal and Ankle brace gait cycle. The true rate of the right foot implies that right foot Knee brace shows a similarity of 71%, 32% and 25% to that of Normal, Ankle brace and Ankle&Knee brace gait cycle respectively. This could also be seen from the acceleration plot of the gait cycles in Figure 4.7(b,d&e).

In contrasts to the right foot, the left foot shows that the Knee brace gait cycle to be more similar to the Ankle brace than the Normal gait cycle. The deviation of the left foot Knee brace gait cycle to that of the Ankle brace and Ankle&Knee brace gait cycle, illustrates a decrease as compared to the right foot. This is illustrated in acceleration plot of the gait cycles in Figure 4.8(b,d&e). This implicates that Ankle bracing and Ankle&Knee bracing effect on unbraced side is less than that of Knee bracing. This a similar finding to that of [4], as reviewed in section 2.3.3.

Even though the extracted features for the Knee brace show a certain gap, the average result of the true rate claim that the extracted features to some extent express the gait cycle. The right and left foot of the Knee brace has been expressed on average 44% and 39% respectively. Which shows a decrease as compared to Ankle brace gait cycle. This means correction of the specified features will to some extent decrease deviation of gait cycle due to Knee bracing.

5.2.3.3 Ankle and Knee_Brace

The true rate of Ankle&Knee brace measures the ability of the classifier to distinguish Ankle&Knee brace gait cycle from the other gait cycles. This depends on deviation of the Ankle&Knee brace acceleration parameters, which are used as feature for classification. In this sub-section Ankle&Knee brace gait cycle deviation in relation to the other gait cycles based on the acceleration parameters is discussed.

As stated in the previous sections 5.2.3.1 & 5.2.3.2, the deviation of Ankle&Knee brace gait cycle to that of Knee brace is relatively high. True rate of Ankle&Knee brace also reveals that as

shown in Figure 5.3(g)& (h). It is also indicated from the acceleration plot of the two gait cycles, in Figure 4.7&4.8(e). The implication of the confusion matrix in Table 5.2-5.3 (a) is also similar. That is Knee brace data having the lowest misclassification to Ankle&Knee brace data set.

True rate of Ankle&Knee brace gait cycle with respect to Ankle brace shows more deviation in the left foot than that of the right foot. This is because Ankle&Knee brace gait cycle is more similar to Normal gait cycle on the unbraced leg than that of the braced leg. This is also indicated from the acceleration plot of the two gait cycles, in Figure 4.7&4.8(f). This is also indicated in the confusion matrix in Table 5.2-5.3 (a).

Indication of the result form the true rate show that right foot Ankle&Knee brace gait cycle is similar to Ankle brace gait cycle. This contradicts to the from the acceleration plot of the two gait cycles, in Figure 4.7(c). This suggests that the extracted feature sets show a gap in expressing Ankle&Knee brace gait cycle.

Although the extracted features for the Ankle&Knee brace show a certain gap, the average result of the true rate claim that the extracted features to some extent express the gait cycle. The right and left foot of the Ankle&Knee brace has been expressed on average 52% and 27% respectively. Which shows a increase in right foot as compared to gait cycle Knee brace and a decrease as compared to Ankle brace. Whereas the left foot shows a decrease as compared to both Ankle brace and Knee brace gait cycles. This means correction of the specified features will to some extent decrease deviation of gait cycle due to Ankle&Knee bracing.

Chapter 6

Conclusion and Considerations for Future Research

6.1 Conclusion

Gait classification algorithm was develop to distinguish different type of Orthotic bracing and normal gait cycle. The developed algorithm uses acceleration raw data as an inputs from IMU sensors. The acceleration data was taken from the foot of a healthy subject who is asked to reproduce the abnormality of the gait by wearing the orthotic brace on the right foot. The raw data input was pre-processed to discriticize the gait cycle from the overall covered distance. The separated acceleration was curve fitted, which was selected by checking the fitting of Vicon motion capture sample data and IMU sensor data using different models. Based on the findings Gaussian 8th order was preferred and used for the development of the algorithm.

Features which describe each gait cycle was then chosen by studying the deviation in relation to each other and the normal gait cycle. Based on the study coefficients of the Gaussian curve fit was taken as features for the algorithm development. The performance analysis depending on the bracing type illustrates that, the gait cycle for the braced side (Right foot) due to Ankle bracing relatively more expressed by the extracted feature set, it is express on average 58%. The least expressed is the gait cycle from the unbraced side (Left foot) due to Ankle&Knee bracing , it is express on average 27%. The unbraced side is most affected from Knee bracing than Ankle and Ankle&Knee bracing. It can be taken from the finding that small correction to the acceleration

extracted features will to some extent minimize the deviation of the gait cycle due to different bracings.

The performance analysis for different class types revel that categorical classification shows higher accuracy, with a maximum of 86%, than binary step, with a maximum of 72%, binary combination, with a maximum of 41%, and multi class type classifications, with a maximum of 53%. The performance analysis related to the classifier considers binary step classification and categorical classification to evaluate each classifier. Based on the evaluation Tree Bagger classifier shows relatively higher accuracy with a relatively balanced true rate. Support Vector Machine (SVM) also presents a closer balanced true rate and accuracy to that of Tree Bagger.

6.2 Considerations for Future Research

As presented the performance of the developed algorithm illustrates that the feature extracted is a way toward improving the design, fitting and rehabilitation of Orthotic bracing. In order to advance the findings to achieve the long term-goal, future research should focus on five different points:

1. **Assessing different wearable sensors for better performance result;** This is required to testing different features from different wearable sensors to see which type performs well. Also a combination of sensors could be taken into consideration to get the advantage of different wearable sensors. Different position data's from the same wearable sensors should be seen.

2. **Evaluating different extracted features;** Extracted features from different wearable sensors, from spatial and temporal parameters, time domains, frequency domains, etc.. should be considered. This is to find the best feature that could express the deviation of each type of Orthotic bracing.

3. **Improving algorithm complexity;** The developed algorithm could be further improved by considering different classifiers or a combination of classifiers. In addition to the considering different inputs from points (1) and (2).

4. **Testing the result on different subjects;** The developed algorithm was only cross-validated using data's from the same source as the training data. The algorithm should further be tested by taken data from different subjects, as well as data's from different pathologies with Orthotic bracing.

5. **Application of findings on improving Orthotic bracing;** The findings about the extracted features should be used to improve the Orthotic bracing. This could be accomplished by assessing a way to use the features in designing or fitting or rehabilitation process of the Orthotic bracing. The improved Orthotic should then be evaluated.

Reference

1. Fish, D.J. and J.-P. Nielsen, *Clinical Assessment of Human Gait.* JPO: Journal of Prosthetics and Orthotics, 1993. **5**(2): p. 39.
2. Strait, E., *Prosthetics in developing countries.* Prosthetic Resident, 2006: p. 1-40.
3. affiars, M.o.l.a.s., *National physical rehabilitation strategy* F.D.R.o.E. (FDRE), Editor. 2011.
4. Shorter, K.A., et al., *A new approach to detecting asymmetries in gait.* Clinical Biomechanics, 2008. **23**(4): p. 459-467.
5. Mannini, A. and A.M. Sabatini, *Machine learning methods for classifying human physical activity from on-body accelerometers.* Sensors, 2010. **10**(2): p. 1154-1175.
6. Mannini, A., et al., *A Machine Learning Framework for Gait Classification Using Inertial Sensors: Application to Elderly, Post-Stroke and Huntington's Disease Patients.* Sensors, 2016. **16**(1): p. 134.
7. Lakany, H., *Extracting a diagnostic gait signature.* Pattern recognition, 2008. **41**(5): p. 1627-1637.
8. Muro-de-la-Herran, A., B. Garcia-Zapirain, and A. Mendez-Zorrilla, *Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications.* Sensors, 2014. **14**(2): p. 3362-3394.

9. Fong, D.T.-P. and Y.-Y. Chan, *The use of wearable inertial motion sensors in human lower limb biomechanics studies: a systematic review.* Sensors, 2010. **10**(12): p. 11556-11565.

10. Tao, W., et al., *Gait analysis using wearable sensors.* Sensors, 2012. **12**(2): p. 2255-2283.

11. Trojaniello, D., A. Cereatti, and U. Della Croce, *Accuracy, sensitivity and robustness of five different methods for the estimation of gait temporal parameters using a single inertial sensor mounted on the lower trunk.* Gait & posture, 2014. **40**(4): p. 487-492.

12. Alvarez, J.C., et al., *Pedestrian navigation based on a waist-worn inertial sensor.* Sensors, 2012. **12**(8): p. 10536-10549.

13. Anna, A.S. and N. Wickström, *A symbol-based approach to gait analysis from acceleration signals: Identification and detection of gait events and a new measure of gait symmetry.* Information Technology in Biomedicine, IEEE Transactions on, 2010. **14**(5): p. 1180-1187.

14. Mannini, A., et al. *Hidden Markov model-based strategy for gait segmentation using inertial sensors: Application to elderly, hemiparetic patients and Huntington's disease patients*. in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2015. IEEE.

15. Kapti, A.O. and G. Muhurcu, *Wearable acceleration sensor application in unilateral trans-tibial amputation prostheses.* Biocybernetics and Biomedical Engineering, 2014. **34**(1): p. 53-62.

16. Taborri, J., et al., *Gait Partitioning Methods: A Systematic Review.* Sensors, 2016. **16**(1): p. 66.

17. Chellappa, R., A. Veeraraghavan, and N. Ramanathan, *Gait biometrics, overview.* Encyclopedia of Biometrics, 2015: p. 783-789.

18. Matthews, C., et al. *In-situ step size estimation using a kinetic model of human gait*. in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010)*. 2001.

19. Shull, P.B., et al., *Quantified self and human movement: A review on the clinical impact of wearable sensing and feedback for gait analysis and intervention.* Gait & posture, 2014. **40**(1): p. 11-19.

20. Fraccaro, P., et al., *Real-world gyroscope-based gait event detection and gait feature extraction.* 2014.

21. Abaid, N., et al., *Gait detection in children with and without hemiplegia using single-axis wearable gyroscopes.* PloS one, 2013. **8**(9): p. e73152.

22. Smith, B.A., *Determination of Normal or Abnormal Gait Using a Two-Dimensional Video Camera.* 2007.

23. Pratheepan, Y., J. Condell, and G. Prasad. *The use of dynamic and static characteristics of gait for individual identification.* in *Machine Vision and Image Processing Conference, 2009. IMVIP'09. 13th International*. 2009. IEEE.

24. Aminian, K., et al., *Spatio-temporal parameters of gait measured by an ambulatory system using miniature gyroscopes.* Journal of biomechanics, 2002. **35**(5): p. 689-699.

25. Zeng, H. and Y. Zhao, *Sensing movement: Microsensors for body motion measurement.* Sensors, 2011. **11**(1): p. 638-660.

26. Ayrulu-Erdem, B. and B. Barshan, *Leg motion classification with artificial neural networks using wavelet-based features of gyroscope signals.* Sensors, 2011. **11**(2): p. 1721-1743.

27. Davis, R.B., *Clinical gait analysis.* Engineering in Medicine and Biology Magazine, IEEE, 1988. **7**(3): p. 35-40.

28. Hof, A., et al., *Speed dependence of averaged EMG profiles in walking.* Gait & posture, 2002. **16**(1): p. 78-86.

29. Mayagoitia, R.E., A.V. Nene, and P.H. Veltink, *Accelerometer and rate gyroscope measurement of kinematics: an inexpensive alternative to optical motion analysis systems.* Journal of biomechanics, 2002. **35**(4): p. 537-542.

30. Moe-Nilssen, R. and J.L. Helbostad, *Estimation of gait cycle characteristics by trunk accelerometry.* Journal of biomechanics, 2004. **37**(1): p. 121-126.

31. Whittle, M.W., *Gait analysis: an introduction*. 2014: Butterworth-Heinemann.

32. Perry, J., *Observational gait analysis handbook.* The Pathokinesiology Service and The Physical Therapy Department of Rancho Los Amigos Medical Center, 1989.

33. Whittle, M., *Gait Analysis: An introduction. 2007.* Heidi Harrison: p. 47-100.

34. Beckett, N., *Classification of Gait Abnormality.* 1998.

35. Jasiewicz, J.M., et al., *Gait event detection using linear accelerometers or angular velocity transducers in able-bodied and spinal-cord injured individuals.* Gait & Posture, 2006. **24**(4): p. 502-509.

36. Gouwanda, D. and A.A. Gopalai, *A robust real-time gait event detection using wireless gyroscope and its application on normal and altered gaits.* Medical engineering & physics, 2015. **37**(2): p. 219-225.

37. Sant'Anna, A., *A symbolic approach to human motion analysis using inertial sensors: framework and gait analysis study.* 2012.

38. Senanayake, C.M. and S.A. Senanayake, *Computational intelligent gait-phase detection system to identify pathological gait.* IEEE Transactions on Information Technology in Biomedicine, 2010. **14**(5): p. 1173-1179.

39. Jain, A.K., R.P.W. Duin, and J. Mao, *Statistical pattern recognition: A review.* IEEE Transactions on pattern analysis and machine intelligence, 2000. **22**(1): p. 4-37.

40. Liao, T.W., *Clustering of time series data—a survey.* Pattern recognition, 2005. **38**(11): p. 1857-1874.

41. Gatuha, G. and T. Jiang. *Evaluating Diagnostic Performance of Machine Learning Algorithms on Breast Cancer*. in *International Conference on Intelligent Science and Big Data Engineering*. 2015. Springer.

42. Saravanan, K. and S. Sasithra, *Review on classification based on artificial neural networks.* International J. of Ambient Systems and Applications, 2014. **2**: p. 11-8.

43. Breiman, L., *Bagging predictors.* Machine learning, 1996. **24**(2): p. 123-140.

44. Mannini, A. and A.M. Sabatini, *Gait phase detection and discrimination between walking–jogging activities using hidden Markov models applied to foot motion data from a gyroscope.* Gait & posture, 2012. **36**(4): p. 657-661.

45. Shibuya, N., et al. *A real-time fall detection system using a wearable gait analysis sensor and a support vector machine (svm) classifier*. in *Mobile Computing and Ubiquitous Networking (ICMU), 2015 Eighth International Conference on*. 2015. IEEE.

46. Nukala, B.T., et al., *An efficient and robust fall detection system using wireless gait analysis sensor with artificial neural network (ANN) and support vector machine (SVM) algorithms.* Open Journal of Applied Biosensor, 2015. **3**(04): p. 29.

47. Choi, S., et al. *Biometric gait recognition based on wireless acceleration sensor using k-nearest neighbor classification*. in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. 2014. IEEE.

48. Cinza-González, A., et al., *A non-supervised classification neural network reveals temporal patterns of kinematic strategies in children's gait cycle.* Gait & Posture, 2015. **42**: p. S44.

49. Rabiner, L.R. and B.-H. Juang, *An introduction to hidden Markov models.* ASSP Magazine, IEEE, 1986. **3**(1): p. 4-16.

50. Pogorelc, B., Z. Bosnić, and M. Gams, *Automatic recognition of gait-related health problems in the elderly using machine learning.* Multimedia Tools and Applications, 2012. **58**(2): p. 333-354.

51. Wong, T.-T., *Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation.* Pattern Recognition, 2015. **48**(9): p. 2839-2846.

52. Delen, D., et al., *A comparative analysis of machine learning systems for measuring the impact of knowledge management practices.* Decision Support Systems, 2013. **54**(2): p. 1150-1160.

53. Gavrilov, S., et al., *Feature Analysis and Classification of Particle Data from Two-Dimensional Video Disdrometer.* Advances in Remote Sensing, 2015. **4**(01): p. 1.

54. Kwon, O. and J.M. Sim, *Effects of data set features on the performances of classification algorithms.* Expert Systems with Applications, 2013. **40**(5): p. 1847-1857.

55. Azar, A.T. and S.A. El-Said, *Performance analysis of support vector machines classifiers in breast cancer mammography recognition.* Neural Computing and Applications, 2014. **24**(5): p. 1163-1177.

56. Patil, T.R. and S. Sherekar, *Performance analysis of Naive Bayes and J48 classification algorithm for data classification.* International Journal of Computer Science and Applications, 2013. **6**(2): p. 256-261.

57. Woolard, A., *Vicon 512*, O.M. Ltd, Editor. 1999: Oxford.

58. Interational Society for Prosthetic and Orthtoic

Appendix A : MATLAB code for the algorithm

This section includes the code for multi-class classification, binary combination classification and a sample for categorical classification for one of the legs. The others are not included since the code is similar to the once that are. The complete set of the algorithmsare included in the provided CD.

A1: Multi-class classification for the left foot

```
%%   Packages data from the collected from IMU

%   Clear up workspace

closeall;
clearall;
clc;

%   Load up data

loadprocessed_imu_data

%   Extracts each gait cycle from normal data and
%   stuffs them into a structure.
%   Define structure for housing data

gait_data = struct('subject',[], 'trial', [],.....
'right_foot',struct('t', [], 'accel',[]));

a1 = struct('trial',[],'complete',[]); a2 = struct('trial',[],'complete',[]);
a3 = struct('trial',[],'complete',[]); a4 = struct('trial',[],'complete',[]);
a5 = struct('trial',[],'complete',[]); a6 = struct('trial',[],'complete',[]);
a7 = struct('trial',[],'complete',[]); a8 = struct('trial',[],'complete',[]);
b1 = struct('trial',[],'complete',[]); b2 = struct('trial',[],'complete',[]);
b3 = struct('trial',[],'complete',[]); b4 = struct('trial',[],'complete',[]);
b5 = struct('trial',[],'complete',[]); b6 = struct('trial',[],'complete',[]);
b7 = struct('trial',[],'complete',[]); b8 = struct('trial',[],'complete',[]);
c1 = struct('trial',[],'complete',[]); c2 = struct('trial',[],'complete',[]);
c3 = struct('trial',[],'complete',[]); c4 = struct('trial',[],'complete',[]);
c5 = struct('trial',[],'complete',[]); c6 = struct('trial',[],'complete',[]);
c7 = struct('trial',[],'complete',[]); c8 = struct('trial',[],'complete',[]);
```

```
[o,Data_size] = size(accel_data);

for k=1:Data_size;

%   Stuff known fields into gait structure

gait_data(k).subject = accel_data(1,k).subject;
gait_data(k).trial = accel_data(1,k).trial;
idx_left=accel_data(1,k).LANK_MAG;

%   Find index of heel strikes for left foot.  Leverage the fact that
%   the gait cycle starts and ends with heel strike leading to large
%   acceleration spikes.  Between the two large spikes there are two
%   smaller ones corresponding to the start of heel-off and swing.

    [~,idx_lp] = findpeaks(idx_left);
    left_cut_1 = mean(idx_left(idx_lp));
    idx_cut_1 = idx_left> left_cut_1;        %  Find all four peaks will be
above average
    left_cut_2 = mean(idx_left(idx_cut_1));
    idx_cut_2 = find(idx_left> left_cut_2);        %  Find points surrounding
heel strikes
    [~,idx] = findpeaks(idx_left(idx_cut_2));    %  Find heel strikes
idx_lhs = idx_cut_2(idx);

%   Stuff data for each cycle into a gait_data structure
%   right foot

%   Stuff data for each cycle into a gait_data structure
%   left foot

num_gait_cycles = length(idx_lhs);
for m = 1:(num_gait_cycles-1)
m_lower = idx_lhs(m);
m_upper = idx_lhs(m + 1);

t_plot = accel_data(1,k).t(m_lower:m_upper) - accel_data(1,k).t(m_lower);
mag_plot = idx_left(m_lower:m_upper);

gait_data(k).left_foot(m).t = t_plot;
gait_data(k).left_foot(m).accel = mag_plot;

% Processing finished cycles only

        [T,S] = size(gait_data(k).left_foot(m).accel);
        Peaks = findpeaks(gait_data(k).left_foot(m).accel);
        [Q,P] = size(Peaks);
if S > 24;
if P > 4;
% gaussian fit to each gait cycle
```

```
            [xData, yData] =
prepareCurveData(gait_data(k).left_foot(m).t,....
gait_data(k).left_foot(m).accel );

% Set up fittype and options.
ft = fittype( 'gauss8' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -
Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0];
opts.StartPoint = [25304.0846037163 0 0.0138089761463768 20574.9756149364
0.609999999999957 0.0240799643087144 19697.9331861504 0.0199999999999818
0.0123971874863801 18576.0098094693 0.109999999999957 0.00978562351870957
14870.6628491009 0.0399999999999636 0.0142068835652844 11052.9580286452
0.139999999999986 0.0122707734005006 10532.7090398849 0.0699999999999932
0.0167500967218961 5502.143072838 0.159999999999968 0.017599409350113];
opts.Upper =
[InfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInf];

% Fit model to data.
            [fitresult, gof] = fit(xData, yData, ft, opts );

% Gaussian coffecients for each steps

a1(k).trial(m) = fitresult.a1;
a2(k).trial(m) = fitresult.a2;
a3(k).trial(m) = fitresult.a3;
a4(k).trial(m) = fitresult.a4;
a5(k).trial(m) = fitresult.a5;
a6(k).trial(m) = fitresult.a6;
a7(k).trial(m) = fitresult.a7;
a8(k).trial(m) = fitresult.a8;
b1(k).trial(m) = fitresult.b1;
b2(k).trial(m) = fitresult.b2;
b3(k).trial(m) = fitresult.b3;
b4(k).trial(m) = fitresult.b4;
b5(k).trial(m) = fitresult.b5;
b6(k).trial(m) = fitresult.b6;
b7(k).trial(m) = fitresult.b7;
b8(k).trial(m) = fitresult.b8;
c1(k).trial(m) = fitresult.c1;
c2(k).trial(m) = fitresult.c2;
c3(k).trial(m) = fitresult.c3;
c4(k).trial(m) = fitresult.c4;
c5(k).trial(m) = fitresult.c5;
c6(k).trial(m) = fitresult.c6;
c7(k).trial(m) = fitresult.c7;
c8(k).trial(m) = fitresult.c8;
else
end
else
```

```
end
end

end

savegait_data.matgait_data

%% preparing the cofficient for features in the classification algorithm
% cutting parts of the data that is not fitted, since the cycle is not
% completed
% finding non zero points of cofficient a1 since a1 is always a non-zero
% number

for k=1:Data_size;

cut = find(a1(k).trial);
a1(k).complete = rot90(a1(k).trial[19]);
a2(k).complete = rot90(a2(k).trial[19]);
a3(k).complete = rot90(a3(k).trial[19]);
a4(k).complete = rot90(a4(k).trial[19]);
a5(k).complete = rot90(a5(k).trial[19]);
a6(k).complete = rot90(a6(k).trial[19]);
a7(k).complete = rot90(a7(k).trial[19]);
a8(k).complete = rot90(a8(k).trial[19]);
b1(k).complete = rot90(b1(k).trial[19]);
b2(k).complete = rot90(b2(k).trial[19]);
b3(k).complete = rot90(b3(k).trial[19]);
b4(k).complete = rot90(b4(k).trial[19]);
b5(k).complete = rot90(b5(k).trial[19]);
b6(k).complete = rot90(b6(k).trial[19]);
b7(k).complete = rot90(b7(k).trial[19]);
b8(k).complete = rot90(b8(k).trial[19]);
c1(k).complete = rot90(c1(k).trial[19]);
c2(k).complete = rot90(c2(k).trial[19]);
c3(k).complete = rot90(c3(k).trial[19]);
c4(k).complete = rot90(c4(k).trial[19]);
c5(k).complete = rot90(c5(k).trial[19]);
c6(k).complete = rot90(c6(k).trial[19]);
c7(k).complete = rot90(c7(k).trial[19]);
c8(k).complete = rot90(c8(k).trial[19]);

end
%% putting all the features from the different trials into one features
% with there respective feature type
% creating a matrix for each features and inserting the feature data
% initial conditions

n=1; t=0;
for n =1:Data_size;
if t==0;
        t = t + size(a1(n).complete);
```

```
        t = t(1,1);
        feature_a1(1:t,1) = [a1(n).complete];
        feature_a2(1:t,1) = [a2(n).complete];
        feature_a3(1:t,1) = [a3(n).complete];
        feature_a4(1:t,1) = [a4(n).complete];
        feature_a5(1:t,1) = [a5(n).complete];
        feature_a6(1:t,1) = [a6(n).complete];
        feature_a7(1:t,1) = [a7(n).complete];
        feature_a8(1:t,1) = [a8(n).complete];
        feature_b1(1:t,1) = [b1(n).complete];
        feature_b2(1:t,1) = [b2(n).complete];
        feature_b3(1:t,1) = [b3(n).complete];
        feature_b4(1:t,1) = [b4(n).complete];
        feature_b5(1:t,1) = [b5(n).complete];
        feature_b6(1:t,1) = [b6(n).complete];
        feature_b7(1:t,1) = [b7(n).complete];
        feature_b8(1:t,1) = [b8(n).complete];
        feature_c1(1:t,1) = [c1(n).complete];
        feature_c2(1:t,1) = [c2(n).complete];
        feature_c3(1:t,1) = [c3(n).complete];
        feature_c4(1:t,1) = [c4(n).complete];
        feature_c5(1:t,1) = [c5(n).complete];
        feature_c6(1:t,1) = [c6(n).complete];
        feature_c7(1:t,1) = [c7(n).complete];
        feature_c8(1:t,1) = [c8(n).complete];
        n = n+1;
        p = t+1;
elseif t==p;
        t = t + size(a1(n).complete);
        t = t(1,1)-1;
        feature_a1(p:t,1) = [a1(n).complete];
        feature_a2(p:t,1) = [a2(n).complete];
        feature_a3(p:t,1) = [a3(n).complete];
        feature_a4(p:t,1) = [a4(n).complete];
        feature_a5(p:t,1) = [a5(n).complete];
        feature_a6(p:t,1) = [a6(n).complete];
        feature_a7(p:t,1) = [a7(n).complete];
        feature_a8(p:t,1) = [a8(n).complete];
        feature_b1(p:t,1) = [b1(n).complete];
        feature_b2(p:t,1) = [b2(n).complete];
        feature_b3(p:t,1) = [b3(n).complete];
        feature_b4(p:t,1) = [b4(n).complete];
        feature_b5(p:t,1) = [b5(n).complete];
        feature_b6(p:t,1) = [b6(n).complete];
        feature_b7(p:t,1) = [b7(n).complete];
        feature_b8(p:t,1) = [b8(n).complete];
        feature_c1(p:t,1) = [c1(n).complete];
        feature_c2(p:t,1) = [c2(n).complete];
        feature_c3(p:t,1) = [c3(n).complete];
        feature_c4(p:t,1) = [c4(n).complete];
        feature_c5(p:t,1) = [c5(n).complete];
        feature_c6(p:t,1) = [c6(n).complete];
```

```
        feature_c7(p:t,1) = [c7(n).complete];
        feature_c8(p:t,1) = [c8(n).complete];
        n = n+1;
        p = t+1;
end
    t=p;
end

%% bringing all features in one data set for ease of modeling
% all 4 classes all together

X = [feature_a1,feature_a2,feature_a3,feature_a4,feature_a5,feature_a6,....
    feature_a7,feature_a8,feature_b1,feature_b2,feature_b3,feature_b4,.....
    feature_b5,feature_b6,feature_b7,feature_b8,feature_c1,feature_c2,.....
    feature_c3,feature_c4,feature_c5,feature_c6,feature_c7,feature_c8];

% response data for the each cycle or row data
% that is the class of the data
% second by separating the both Ankle_Brace & Knee_Brace and the others

% initial conditions

n=1; t=0;
while n<=Data_size;
if t==0;
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(1:t,1) = {'Normal'};
        p = t+1;
elseif (2<=n)&&(n<=3);
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(p:t,1) = {'Normal'};
        p = t+1;
elseif (4<=n)&&(n<=6);
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(p:t,1) = {'Knee_Brace'};
        p = t+1;
elseif (7<=n)&&(n<=9);
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(p:t,1) = {'Ankle_Brace'};
        p = t+1;
elseif (10<=n)&&(n<=12);
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(p:t,1) = {'Both'};
        p = t+1;
elseif (13<=n)&&(n<=15);
        t = t + size(a1(n).complete);
```

```
        t = t(1,1);
Y(p:t,1) = {'Normal'};
        p = t+1;
end
    n = n+1;

end

%% partitioning data for training and test (for validation)

Z = cvpartition(Y,'holdout',0.3);

% training data

X_Train = X(training(Z,1),:);
Y_Train = Y(training(Z,1));

% test data

X_Test = X(test(Z,1),:);
Y_Test = Y(test(Z,1));

%% using Naive Bayes model for classifiction

X_Naive =
NaiveBayes.fit(X_Train,Y_Train,'Distribution','kernel','Prior','uniform');

% predict the test data based on the model

[Y_Naive] = predict(X_Naive,X_Test);

% Evaluating accuracy
% confusion matrix

[conf_Naive,order] = confusionmat(Y_Test,Y_Naive);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_Naive = 1-trace(conf_Naive)/sum(conf_Naive(:));

%% using TreeBagger

X_TreeBagger = TreeBagger(22,X_Train,Y_Train,'oobpred','on');
plot(oobError(X_TreeBagger))
xlabel('number of grown trees')
ylabel('out-of-bag classifcation error')

% predict the test data based on the model
```

```
Y_TreeBagger = predict(X_TreeBagger,X_Test);

% Ensemble predictions for out-of-bag observations.

Ensemble_TreeBagger = oobPredict(X_TreeBagger);

% Evaluating accuracy
% misclassfication probability

ERR_TreeBagger = error(X_TreeBagger,X_Train,Y_Train);

% confusion matrix

[conf_TreeBagger,order] = confusionmat(Y_Test,Y_TreeBagger);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_TreeBagger = 1-trace(conf_TreeBagger)/sum(conf_TreeBagger(:));
```

A2: Binary combination classification for the right foot

```
%%   Packages data from the collected from IMU
%   Clear up workspace

closeall;
clearall;
clc;

%   Load up data

loadprocessed_imu_data

%   Extracts each gait cycle from normal data and
%   stuffs them into a structure.
%   Define structure for housing data

gait_data = struct('subject',[], 'trial', [],.....
'right_foot',struct('t', [], 'accel',[]));
```

```
a1 = struct('trial',[],'complete',[]); a2 = struct('trial',[],'complete',[]);
a3 = struct('trial',[],'complete',[]); a4 = struct('trial',[],'complete',[]);
a5 = struct('trial',[],'complete',[]); a6 = struct('trial',[],'complete',[]);
a7 = struct('trial',[],'complete',[]); a8 = struct('trial',[],'complete',[]);
b1 = struct('trial',[],'complete',[]); b2 = struct('trial',[],'complete',[]);
b3 = struct('trial',[],'complete',[]); b4 = struct('trial',[],'complete',[]);
b5 = struct('trial',[],'complete',[]); b6 = struct('trial',[],'complete',[]);
b7 = struct('trial',[],'complete',[]); b8 = struct('trial',[],'complete',[]);
c1 = struct('trial',[],'complete',[]); c2 = struct('trial',[],'complete',[]);
c3 = struct('trial',[],'complete',[]); c4 = struct('trial',[],'complete',[]);
c5 = struct('trial',[],'complete',[]); c6 = struct('trial',[],'complete',[]);
c7 = struct('trial',[],'complete',[]); c8 = struct('trial',[],'complete',[]);

[o,Data_size] = size(accel_data);

for k=1:Data_size

%   Stuff known fields into gait structure

gait_data(k).subject = accel_data(1,k).subject;
gait_data(k).trial = accel_data(1,k).trial;
idx_left=accel_data(1,k).LANK_MAG;
idx_right=accel_data(1,k).RANK_MAG;

%   Find index of heel strikes for right foot.  Leverage the fact that
%   the gait cycle starts and ends with heel strike leading to large
%   acceleration spikes.  Between the two large spikes there are two
%   smaller ones corresponding to the start of heel-off and swing.

    [~,idx_rp] = findpeaks(idx_right);
    right_cut_1 = mean(idx_right(idx_rp));
    idx_cut_1 = find(idx_right> right_cut_1);        %  Find all four peaks
will be above average
    right_cut_2 = mean(idx_right(idx_cut_1));
    idx_cut_2 = find(idx_right> right_cut_2);        %  Find points
surrounding heel strikes
    [temp,idx] = findpeaks(idx_right(idx_cut_2));    %  Find heel strikes
idx_rhs = idx_cut_2(idx);

%   Stuff data for each cycle into a gait_data structure
%   right foot

num_gait_cycles = length(idx_rhs);
for m = 1:(num_gait_cycles-1)
m_lower = idx_rhs(m);
m_upper = idx_rhs(m + 1);

t_plot = accel_data(1,k).t(m_lower:m_upper) - accel_data(1,k).t(m_lower);
mag_plot = idx_right(m_lower:m_upper);
```

```
gait_data(k).right_foot(m).t = t_plot;
gait_data(k).right_foot(m).accel = mag_plot;

% Processing finished cycles only

        [T,S] = size(gait_data(k).right_foot(m).accel);
        Peaks = findpeaks(gait_data(k).right_foot(m).accel);
        [Q,P] = size(Peaks);
if S > 24;
if P > 4;
% gaussian fit to each gait cycle
            [xData, yData] =
prepareCurveData(gait_data(k).right_foot(m).t,....
gait_data(k).right_foot(m).accel );

% Set up fittype and options.
ft = fittype( 'gauss8' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -
Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0];
opts.StartPoint = [25304.0846037163 0 0.0138089761463768 20574.9756149364
0.609999999999957 0.0240799643087144 19697.9331861504 0.0199999999999818
0.0123971874863801 18576.0098094693 0.109999999999957 0.00978562351870957
14870.6628491009 0.0399999999999636 0.0142068835652844 11052.9580286452
0.139999999999986 0.0122707734005006 10532.7090398849 0.0699999999999932
0.0167500967218961 5502.143072838 0.159999999999968 0.017599409350113];
opts.Upper =
[InfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInf];

% Fit model to data.
            [fitresult, gof] = fit(xData, yData, ft, opts );

% Gaussian coffecients for each steps

a1(k).trial(m) = fitresult.a1;
a2(k).trial(m) = fitresult.a2;
a3(k).trial(m) = fitresult.a3;
a4(k).trial(m) = fitresult.a4;
a5(k).trial(m) = fitresult.a5;
a6(k).trial(m) = fitresult.a6;
a7(k).trial(m) = fitresult.a7;
a8(k).trial(m) = fitresult.a8;
b1(k).trial(m) = fitresult.b1;
b2(k).trial(m) = fitresult.b2;
b3(k).trial(m) = fitresult.b3;
b4(k).trial(m) = fitresult.b4;
b5(k).trial(m) = fitresult.b5;
b6(k).trial(m) = fitresult.b6;
b7(k).trial(m) = fitresult.b7;
b8(k).trial(m) = fitresult.b8;
```

```
c1(k).trial(m) = fitresult.c1;
c2(k).trial(m) = fitresult.c2;
c3(k).trial(m) = fitresult.c3;
c4(k).trial(m) = fitresult.c4;
c5(k).trial(m) = fitresult.c5;
c6(k).trial(m) = fitresult.c6;
c7(k).trial(m) = fitresult.c7;
c8(k).trial(m) = fitresult.c8;
else
end
else
end
end

end

savegait_data.matgait_data

%% preparing the cofficient for features in the classification algorithm
% cutting parts of the data that is not fitted, since the cycle is not
% completed
% finding non zero points of cofficient a1 since a1 is always a non-zero
% number

for k=1:Data_size

cut = find(a1(k).trial);
a1(k).complete = rot90(a1(k).trial[19]);
a2(k).complete = rot90(a2(k).trial[19]);
a3(k).complete = rot90(a3(k).trial[19]);
a4(k).complete = rot90(a4(k).trial[19]);
a5(k).complete = rot90(a5(k).trial[19]);
a6(k).complete = rot90(a6(k).trial[19]);
a7(k).complete = rot90(a7(k).trial[19]);
a8(k).complete = rot90(a8(k).trial[19]);
b1(k).complete = rot90(b1(k).trial[19]);
b2(k).complete = rot90(b2(k).trial[19]);
b3(k).complete = rot90(b3(k).trial[19]);
b4(k).complete = rot90(b4(k).trial[19]);
b5(k).complete = rot90(b5(k).trial[19]);
b6(k).complete = rot90(b6(k).trial[19]);
b7(k).complete = rot90(b7(k).trial[19]);
b8(k).complete = rot90(b8(k).trial[19]);
c1(k).complete = rot90(c1(k).trial[19]);
c2(k).complete = rot90(c2(k).trial[19]);
c3(k).complete = rot90(c3(k).trial[19]);
c4(k).complete = rot90(c4(k).trial[19]);
c5(k).complete = rot90(c5(k).trial[19]);
c6(k).complete = rot90(c6(k).trial[19]);
c7(k).complete = rot90(c7(k).trial[19]);
c8(k).complete = rot90(c8(k).trial[19]);
```

```
end

%% bringing all features in one data set for ease of modeling
% the data taken as indiviual binary classification

X = [feature_a1,feature_a2,feature_a3,feature_a4,feature_a5,feature_a6,....
    feature_a7,feature_a8,feature_b1,feature_b2,feature_b3,feature_b4,.....
    feature_b5,feature_b6,feature_b7,feature_b8,feature_c1,feature_c2,.....
    feature_c3,feature_c4,feature_c5,feature_c6,feature_c7,feature_c8];

% response data for the each cycle or row data
% that is the class of the data
% frist by separating the normal and the others

% initial conditions

n=1; t=0;
while n<=Data_size;
if t==0;
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_1(1:t,1) = {'Normal'};
        p = t+1;
elseif (2<=n)&&(n<=3);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_1(p:t,1) = {'Normal'};
        p = t+1;
elseif (4<=n)&&(n<=6);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_1(p:t,1) = {'Other'};
        p = t+1;
elseif (7<=n)&&(n<=9);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_1(p:t,1) = {'Other'};
        p = t+1;
elseif (10<=n)&&(n<=12);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_1(p:t,1) = {'Other'};
        p = t+1;
elseif (13<=n)&&(n<=15);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_1(p:t,1) = {'Normal'};
        p = t+1;
end
    n = n+1;
```

```
end

%% partitioning data for training and test (for validation)

Z_1 = cvpartition(Y_1,'holdout',0.3);

% training data

X_Train_1 = X(training(Z_1,1),:);
Y_Train_1 = Y_1(training(Z_1,1));

% test data

X_Test_1 = X(test(Z_1,1),:);
Y_Test_1 = Y_1(test(Z_1,1));

%% using Naive Bayes model for classifiction

X_Naive =
NaiveBayes.fit(X_Train_1,Y_Train_1,'Distribution','kernel','Prior','uniform')
;

% predict the test data based on the model

[Y_Naive_1] = predict(X_Naive,X_Test_1);

% Evaluating accuracy
% confusion matrix

[conf_Naive_1,order] = confusionmat(Y_Test_1,Y_Naive_1);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_Naive_1 = 1-trace(conf_Naive_1)/sum(conf_Naive_1(:));

%% using TreeBagger

X_TreeBagger = TreeBagger(22,X_Train_1,Y_Train_1,'oobpred','on');
plot(oobError(X_TreeBagger))
xlabel('number of grown trees')
ylabel('out-of-bag classifcation error')

% predict the test data based on the model

Y_TreeBagger_1 = predict(X_TreeBagger,X_Test_1);

% Ensemble predictions for out-of-bag observations.
```

```
Ensemble_TreeBagger_1 = oobPredict(X_TreeBagger);

% Evaluating accuracy
% misclassfication probability

ERR_TreeBagger_1 = error(X_TreeBagger,X_Train_1,Y_Train_1);

% confusion matrix

[conf_TreeBagger_1,order] = confusionmat(Y_Test_1,Y_TreeBagger_1);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_TreeBagger_1 = 1-trace(conf_TreeBagger_1)/sum(conf_TreeBagger_1(:));

%% using Nearest Neighbor
X_KNN = ClassificationKNN.fit(X_Train_1,Y_Train_1,'NumNeighbors',16);

% predict the test data based on the model

[Y_KNN_1] = predict(X_KNN,X_Test_1);

% Evaluating accuracy
% resubstitution loss; fraction of misclassification from prediction

RLoos_KNN_1 = resubLoss(X_KNN);

% cross validation error

CVmdl_KNN = crossval(X_KNN);
KLoss_KNN_1 = kfoldLoss(CVmdl_KNN);

% loss Classification error

Loss_KNN_1 = loss(X_KNN,X_Test_1,Y_Test_1);

% confusion matrix

[conf_KNN_1,order_KNN] = confusionmat(Y_Test_1,Y_KNN_1);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_KNN_1 = 1-trace(conf_KNN_1)/sum(conf_KNN_1(:));

%% using SVM (suppor vector machine)

X_SVM = svmtrain(X_Train_1,Y_Train_1,'Kernel_Function','rbf');
```

```
% predict the test data based on the model

Y_SVM_1 = svmclassify(X_SVM,X_Test_1);

% Evaluating accuracy
% confusion matrix

[conf_SVM_1,order_SVM] = confusionmat(Y_Test_1,Y_SVM_1);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_SVM_1 = 1-trace(conf_SVM_1)/sum(conf_SVM_1(:));

%% bringing all features in one data set for ease of modeling

X = [feature_a1,feature_a2,feature_a3,feature_a4,feature_a5,feature_a6,....
    feature_a7,feature_a8,feature_b1,feature_b2,feature_b3,feature_b4,.....
    feature_b5,feature_b6,feature_b7,feature_b8,feature_c1,feature_c2,.....
    feature_c3,feature_c4,feature_c5,feature_c6,feature_c7,feature_c8];

% response data for the each cycle or row data
% that is the class of the data
% second by separating the Ankle_Brace and the others

% initial conditions

n=1; t=0;
while n<=Data_size;
if t==0;
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_2(1:t,1) = {'Other'};
        p = t+1;
elseif (2<=n)&&(n<=3);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_2(p:t,1) = {'Other'};
        p = t+1;
elseif (4<=n)&&(n<=6);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_2(p:t,1) = {'Ankle_Brace'};
        p = t+1;
elseif (7<=n)&&(n<=9);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_2(p:t,1) = {'Other'};
        p = t+1;
elseif (10<=n)&&(n<=12);
```

```
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_2(p:t,1) = {'Other'};
        p = t+1;
elseif (13<=n)&&(n<=15);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_2(p:t,1) = {'Other'};
        p = t+1;
end
    n = n+1;

end

%% partitioning data for training and test (for validation)

Z_2 = cvpartition(Y_2,'holdout',0.3);

% training data

X_Train_2 = X(training(Z_2,1),:);
Y_Train_2 = Y_2(training(Z_2,1));

% test data

X_Test_2 = X(test(Z_2,1),:);
Y_Test_2 = Y_2(test(Z_2,1));

%% using Naive Bayes model for classifiction

X_Naive =
NaiveBayes.fit(X_Train_2,Y_Train_2,'Distribution','kernel','Prior','uniform')
;

% predict the test data based on the model

[Y_Naive_2] = predict(X_Naive,X_Test_2);

% Evaluating accuracy
% confusion matrix

[conf_Naive_2,order] = confusionmat(Y_Test_2,Y_Naive_2);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_Naive_2 = 1-trace(conf_Naive_2)/sum(conf_Naive_2(:));

%% using TreeBagger
```

```
X_TreeBagger = TreeBagger(22,X_Train_2,Y_Train_2,'oobpred','on');
plot(oobError(X_TreeBagger))
xlabel('number of grown trees')
ylabel('out-of-bag classifcation error')

% predict the test data based on the model

Y_TreeBagger_2 = predict(X_TreeBagger,X_Test_2);

% Ensemble predictions for out-of-bag observations.

Ensemble_TreeBagger_2 = oobPredict(X_TreeBagger);

% Evaluating accuracy
% misclassfication probability

ERR_TreeBagger_2 = error(X_TreeBagger,X_Train_2,Y_Train_2);

% confusion matrix

[conf_TreeBagger_2,order] = confusionmat(Y_Test_2,Y_TreeBagger_2);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_TreeBagger_2 = 1-trace(conf_TreeBagger_2)/sum(conf_TreeBagger_2(:));

%% using Nearest Neighbor
X_KNN = ClassificationKNN.fit(X_Train_2,Y_Train_2,'NumNeighbors',16);

% predict the test data based on the model

[Y_KNN_2] = predict(X_KNN,X_Test_2);

% Evaluating accuracy
% resubstitution loss; fraction of misclassification from prediction

RLoos_KNN_2 = resubLoss(X_KNN);

% cross validation error

CVmdl_KNN = crossval(X_KNN);
KLoss_KNN_2 = kfoldLoss(CVmdl_KNN);

% loss Classification error

Loss_KNN_2 = loss(X_KNN,X_Test_2,Y_Test_2);

% confusion matrix
```

```
[conf_KNN_2,order_KNN] = confusionmat(Y_Test_2,Y_KNN_2);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_KNN_2 = 1-trace(conf_KNN_2)/sum(conf_KNN_2(:));

%% using SVM (suppor vector machine)

X_SVM = svmtrain(X_Train_2,Y_Train_2,'Kernel_Function','rbf');

% predict the test data based on the model

Y_SVM_2 = svmclassify(X_SVM,X_Test_2);

% Evaluating accuracy
% confusion matrix

[conf_SVM_2,order_SVM] = confusionmat(Y_Test_2,Y_SVM_2);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_SVM_2 = 1-trace(conf_SVM_2)/sum(conf_SVM_2(:));

%% bringing all features in one data set for ease of modeling

X = [feature_a1,feature_a2,feature_a3,feature_a4,feature_a5,feature_a6,....
    feature_a7,feature_a8,feature_b1,feature_b2,feature_b3,feature_b4,.....
    feature_b5,feature_b6,feature_b7,feature_b8,feature_c1,feature_c2,.....
    feature_c3,feature_c4,feature_c5,feature_c6,feature_c7,feature_c8];

% response data for the each cycle or row data
% that is the class of the data
% second by separating the Knee_Brace and the others

% initial conditions

n=1; t=0;
while n<=Data_size;
if t==0;
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_3(1:t,1) = {'Other'};
        p = t+1;
elseif (2<=n)&&(n<=3);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_3(p:t,1) = {'Other'};
        p = t+1;
```

```
elseif (4<=n)&&(n<=6);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_3(p:t,1) = {'Other'};
        p = t+1;
elseif (7<=n)&&(n<=9);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_3(p:t,1) = {'Knee_Brace'};
        p = t+1;
elseif (10<=n)&&(n<=12);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_3(p:t,1) = {'Other'};
        p = t+1;
elseif (13<=n)&&(n<=15);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_3(p:t,1) = {'Other'};
        p = t+1;
end
    n = n+1;

end

%% partitioning data for training and test (for validation)

Z_3 = cvpartition(Y_3,'holdout',0.3);

% training data

X_Train_3 = X(training(Z_3,1),:);
Y_Train_3 = Y_3(training(Z_3,1));

% test data

X_Test_3 = X(test(Z_3,1),:);
Y_Test_3 = Y_3(test(Z_3,1));

%% using Naive Bayes model for classifiction

X_Naive =
NaiveBayes.fit(X_Train_3,Y_Train_3,'Distribution','kernel','Prior','uniform')
;

% predict the test data based on the model

[Y_Naive_3] = predict(X_Naive,X_Test_3);

% Evaluating accuracy
```

```
% confusion matrix

[conf_Naive_3,order] = confusionmat(Y_Test_3,Y_Naive_3);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_Naive_3 = 1-trace(conf_Naive_3)/sum(conf_Naive_3(:));

%% using TreeBagger

X_TreeBagger = TreeBagger(22,X_Train_3,Y_Train_3,'oobpred','on');
plot(oobError(X_TreeBagger))
xlabel('number of grown trees')
ylabel('out-of-bag classifcation error')

% predict the test data based on the model

Y_TreeBagger_3 = predict(X_TreeBagger,X_Test_3);

% Ensemble predictions for out-of-bag observations.

Ensemble_TreeBagger_3 = oobPredict(X_TreeBagger);

% Evaluating accuracy
% misclassfication probability

ERR_TreeBagger_3 = error(X_TreeBagger,X_Train_3,Y_Train_3);

% confusion matrix

[conf_TreeBagger_3,order] = confusionmat(Y_Test_3,Y_TreeBagger_3);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_TreeBagger_3 = 1-trace(conf_TreeBagger_3)/sum(conf_TreeBagger_3(:));

%% using Nearest Neighbor
X_KNN = ClassificationKNN.fit(X_Train_3,Y_Train_3,'NumNeighbors',16);

% predict the test data based on the model

[Y_KNN_3] = predict(X_KNN,X_Test_3);

% Evaluating accuracy
% resubstitution loss; fraction of misclassification from prediction

RLoos_KNN_3 = resubLoss(X_KNN);
```

```
% cross validation error

CVmdl_KNN = crossval(X_KNN);
KLoss_KNN_3 = kfoldLoss(CVmdl_KNN);

% loss Classification error

Loss_KNN_3 = loss(X_KNN,X_Test_3,Y_Test_3);

% confusion matrix

[conf_KNN_3,order_KNN] = confusionmat(Y_Test_3,Y_KNN_3);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_KNN_3 = 1-trace(conf_KNN_3)/sum(conf_KNN_3(:));

%% using SVM (suppor vector machine)

X_SVM = svmtrain(X_Train_3,Y_Train_3,'Kernel_Function','rbf');

% predict the test data based on the model

Y_SVM_3 = svmclassify(X_SVM,X_Test_3);

% Evaluating accuracy
% confusion matrix

[conf_SVM_3,order_SVM] = confusionmat(Y_Test_3,Y_SVM_3);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_SVM_3 = 1-trace(conf_SVM_3)/sum(conf_SVM_3(:));

%% bringing all features in one data set for ease of modeling

X = [feature_a1,feature_a2,feature_a3,feature_a4,feature_a5,feature_a6,....
    feature_a7,feature_a8,feature_b1,feature_b2,feature_b3,feature_b4,.....
    feature_b5,feature_b6,feature_b7,feature_b8,feature_c1,feature_c2,.....
    feature_c3,feature_c4,feature_c5,feature_c6,feature_c7,feature_c8];

% response data for the each cycle or row data
% that is the class of the data
% second by separating the both Ankle_Brace & Knee_Brace and the others

% initial conditions

n=1; t=0;
```

```
while n<=Data_size;
if t==0;
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_4(1:t,1) = {'Other'};
        p = t+1;
elseif (2<=n)&&(n<=3);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_4(p:t,1) = {'Other'};
        p = t+1;
elseif (4<=n)&&(n<=6);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_4(p:t,1) = {'Other'};
        p = t+1;
elseif (7<=n)&&(n<=9);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_4(p:t,1) = {'Other'};
        p = t+1;
elseif (10<=n)&&(n<=12);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_4(p:t,1) = {'Both'};
        p = t+1;
elseif (13<=n)&&(n<=15);
        t = t + size(a1(n).complete);
        t = t(1,1);
        Y_4(p:t,1) = {'Other'};
        p = t+1;
end
    n = n+1;

end

%% partitioning data for training and test (for validation)

Z_4 = cvpartition(Y_4,'holdout',0.3);

% training data

X_Train_4 = X(training(Z_4,1),:);
Y_Train_4 = Y_4(training(Z_4,1));

% test data

X_Test_4 = X(test(Z_4,1),:);
Y_Test_4 = Y_4(test(Z_4,1));
```

```
%% using Naive Bayes model for classifiction

X_Naive =
NaiveBayes.fit(X_Train_4,Y_Train_4,'Distribution','kernel','Prior','uniform')
;

% predict the test data based on the model

[Y_Naive_4] = predict(X_Naive,X_Test_4);

% Evaluating accuracy
% confusion matrix

[conf_Naive_4,order] = confusionmat(Y_Test_4,Y_Naive_4);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_Naive_4 = 1-trace(conf_Naive_4)/sum(conf_Naive_4(:));

%% using TreeBagger

X_TreeBagger = TreeBagger(22,X_Train_4,Y_Train_4,'oobpred','on');
plot(oobError(X_TreeBagger))
xlabel('number of grown trees')
ylabel('out-of-bag classifcation error')

% predict the test data based on the model

Y_TreeBagger_4 = predict(X_TreeBagger,X_Test_4);

% Ensemble predictions for out-of-bag observations.

Ensemble_TreeBagger_4 = oobPredict(X_TreeBagger);

% Evaluating accuracy
% misclassfication probability

ERR_TreeBagger_4 = error(X_TreeBagger,X_Train_4,Y_Train_4);

% confusion matrix

[conf_TreeBagger_4,order] = confusionmat(Y_Test_4,Y_TreeBagger_4);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_TreeBagger_4 = 1-trace(conf_TreeBagger_4)/sum(conf_TreeBagger_4(:));

%% using Nearest Neighbor
```

```
X_KNN = ClassificationKNN.fit(X_Train_4,Y_Train_4,'NumNeighbors',16);

% predict the test data based on the model

[Y_KNN_4] = predict(X_KNN,X_Test_4);

% Evaluating accuracy
% resubstitution loss; fraction of misclassification from prediction

RLoos_KNN_4 = resubLoss(X_KNN);

% cross validation error

CVmdl_KNN = crossval(X_KNN);
KLoss_KNN_4 = kfoldLoss(CVmdl_KNN);

% loss Classification error

Loss_KNN_4 = loss(X_KNN,X_Test_4,Y_Test_4);

% confusion matrix

[conf_KNN_4,order_KNN] = confusionmat(Y_Test_4,Y_KNN_4);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_KNN_4 = 1-trace(conf_KNN_4)/sum(conf_KNN_4(:));

%% using SVM (suppor vector machine)

X_SVM = svmtrain(X_Train_4,Y_Train_4,'Kernel_Function','rbf');

% predict the test data based on the model

Y_SVM_4 = svmclassify(X_SVM,X_Test_4);

% Evaluating accuracy
% confusion matrix

[conf_SVM_4,order_SVM] = confusionmat(Y_Test_4,Y_SVM_4);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_SVM_4 = 1-trace(conf_SVM_4)/sum(conf_SVM_4(:));

%% Brininging all the indiviual predicted results and the test data
% (True result) in one array
% Naive Bayes model
```

```
Y_Naive_all = [Y_Naive_1,Y_Naive_2,Y_Naive_3,Y_Naive_4];

% TreeBagger

Y_TreeBagger_all =
[Y_TreeBagger_1,Y_TreeBagger_2,Y_TreeBagger_3,Y_TreeBagger_4];

% Nearest Neighbor

Y_KNN_all = [Y_KNN_1,Y_KNN_2,Y_KNN_3,Y_KNN_4];

% SVM (suppor vector machine)

Y_SVM_all = [Y_SVM_1,Y_SVM_2,Y_SVM_3,Y_SVM_4];

% Test data

Y_Test_all = [Y_Test_1,Y_Test_2,Y_Test_3,Y_Test_4];

%% combining the predicted result and the test data (True result)

% Naive Bayes model

u = size(Y_Test_1);
Y_Naive_complete = cell(u);
for k = 1:u;
ifstrcmp(Y_Naive_all(k,1),'Normal');
Y_Naive_complete(k,1) = {'Normal'};
elseifstrcmp(Y_Naive_all(k,2),'Ankle_Brace');
Y_Naive_complete(k,1) = {'Ankle_Brace'};
elseifstrcmp(Y_Naive_all(k,3),'Knee_Brace');
Y_Naive_complete(k,1) = {'Knee_Brace'};
elseifstrcmp(Y_Naive_all(k,4), 'Both');
Y_Naive_complete(k,1) = {'Both'};
else
Y_Naive_complete(k,1) = {'NaN'};
end
end

% TreeBagger

u = size(Y_Test_2);
Y_TreeBagger_complete = cell(u);
for k = 1:u;
ifstrcmp(Y_TreeBagger_all(k,1),'Normal');
Y_TreeBagger_complete(k,1) = {'Normal'};
elseifstrcmp(Y_TreeBagger_all(k,2),'Ankle_Brace');
Y_TreeBagger_complete(k,1) = {'Ankle_Brace'};
```

```
elseifstrcmp(Y_TreeBagger_all(k,3),'Knee_Brace');
Y_TreeBagger_complete(k,1) = {'Knee_Brace'};
elseifstrcmp(Y_TreeBagger_all(k,4), 'Both');
Y_TreeBagger_complete(k,1) = {'Both'};
else
Y_TreeBagger_complete(k,1) = {'NaN'};
end
end

% Nearest Neighbor

u = size(Y_Test_3);
Y_KNN_complete = cell(u);
for k = 1:u;
ifstrcmp(Y_KNN_all(k,1),'Normal');
Y_KNN_complete(k,1) = {'Normal'};
elseifstrcmp(Y_KNN_all(k,2),'Ankle_Brace');
Y_KNN_complete(k,1) = {'Ankle_Brace'};
elseifstrcmp(Y_KNN_all(k,3),'Knee_Brace');
Y_KNN_complete(k,1) = {'Knee_Brace'};
elseifstrcmp(Y_KNN_all(k,4), 'Both');
Y_KNN_complete(k,1) = {'Both'};
else
Y_KNN_complete(k,1) = {'NaN'};
end
end

% SVM (suppor vector machine)

u = size(Y_Test_4);
Y_SVM_complete = cell(u);
for k = 1:u;
ifstrcmp(Y_SVM_all(k,1),'Normal');
Y_SVM_complete(k,1) = {'Normal'};
elseifstrcmp(Y_SVM_all(k,2),'Ankle_Brace');
Y_SVM_complete(k,1) = {'Ankle_Brace'};
elseifstrcmp(Y_SVM_all(k,3),'Knee_Brace');
Y_SVM_complete(k,1) = {'Knee_Brace'};
elseifstrcmp(Y_SVM_all(k,4), 'Both');
Y_SVM_complete(k,1) = {'Both'};
else
Y_SVM_complete(k,1) = {'NaN'};
end
end

% Test data

u = size(Y_Test_4);
Y_Test_complete = cell(u);
for k = 1:u;
ifstrcmp(Y_Test_all(k,1),'Normal');
```

```
Y_Test_complete(k,1) = {'Normal'};
elseifstrcmp(Y_Test_all(k,2),'Ankle_Brace');
Y_Test_complete(k,1) = {'Ankle_Brace'};
elseifstrcmp(Y_Test_all(k,3),'Knee_Brace');
Y_Test_complete(k,1) = {'Knee_Brace'};
elseifstrcmp(Y_Test_all(k,4), 'Both');
Y_Test_complete(k,1) = {'Both'};
else
Y_Test_complete(k,1) = {'NaN'};
end
end

%% Comparing the predicted result and the test data (True result)
% accuracy of Naive Bayes model

% confusion matrix

[conf_Naive_complete,order] = confusionmat(Y_Test_complete,Y_Naive_complete);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_Naive_complete = 1-
trace(conf_Naive_complete)/sum(conf_Naive_complete(:));

% accuracy of TreeBagger

% confusion matrix

[conf_TreeBagger_complete,order] =
confusionmat(Y_Test_complete,Y_TreeBagger_complete);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_TreeBagger_complete = 1-
trace(conf_TreeBagger_1complete)/sum(conf_TreeBagger_complete(:));

% accuracy of KNN

% confusion matrix

[conf_KNN_complete,order_KNN] = confusionmat(Y_Test_complete,Y_KNN_complete);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_KNN_complete = 1-trace(conf_KNN_complete)/sum(conf_KNN_complete(:));

% accuracy of SVM

% confusion matrix
```

```
[conf_SVM_complete,order_SVM] = confusionmat(Y_Test_complete,Y_SVM_complete);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_SVM_complete = 1-trace(conf_SVM_complete)/sum(conf_SVM_complete(:));
```

A3: Categorical classification for the left foot; Ankle_Brace Vs Normal gait cycle

```
%%   Packages data from the collected from IMU
%   Clear up workspace

closeall;
clearall;
clc;

%   Load up data

loadprocessed_imu_data

%   Extracts each gait cycle from normal data and
%   stuffs them into a structure.
%   Define structure for housing data

gait_data = struct('subject',[], 'trial', [],.....
'right_foot',struct('t', [], 'accel',[]));

a1 = struct('trial',[],'complete',[]); a2 = struct('trial',[],'complete',[]);
a3 = struct('trial',[],'complete',[]); a4 = struct('trial',[],'complete',[]);
a5 = struct('trial',[],'complete',[]); a6 = struct('trial',[],'complete',[]);
a7 = struct('trial',[],'complete',[]); a8 = struct('trial',[],'complete',[]);
b1 = struct('trial',[],'complete',[]); b2 = struct('trial',[],'complete',[]);
b3 = struct('trial',[],'complete',[]); b4 = struct('trial',[],'complete',[]);
b5 = struct('trial',[],'complete',[]); b6 = struct('trial',[],'complete',[]);
b7 = struct('trial',[],'complete',[]); b8 = struct('trial',[],'complete',[]);
c1 = struct('trial',[],'complete',[]); c2 = struct('trial',[],'complete',[]);
c3 = struct('trial',[],'complete',[]); c4 = struct('trial',[],'complete',[]);
c5 = struct('trial',[],'complete',[]); c6 = struct('trial',[],'complete',[]);
c7 = struct('trial',[],'complete',[]); c8 = struct('trial',[],'complete',[]);

for k=[1,2,3,13,14,15,7,8,9];

%   Stuff known fields into gait structure

gait_data(k).subject = accel_data(1,k).subject;
```

```
gait_data(k).trial = accel_data(1,k).trial;
idx_left=accel_data(1,k).LANK_MAG;

%   Find index of heel strikes for left foot.  Leverage the fact that
%   the gait cycle starts and ends with heel strike leading to large
%   acceleration spikes.  Between the two large spikes there are two
%   smaller ones corresponding to the start of heel-off and swing.

    [~,idx_lp] = findpeaks(idx_left);
    left_cut_1 = mean(idx_left(idx_lp));
    idx_cut_1 = idx_left> left_cut_1;        %  Find all four peaks will be
above average
    left_cut_2 = mean(idx_left(idx_cut_1));
    idx_cut_2 = find(idx_left> left_cut_2);        %  Find points surrounding
heel strikes
    [~,idx] = findpeaks(idx_left(idx_cut_2));    %  Find heel strikes
idx_lhs = idx_cut_2(idx);

%   Stuff data for each cycle into a gait_data structure
%   right foot

%   Stuff data for each cycle into a gait_data structure
%   left foot

num_gait_cycles = length(idx_lhs);
for m = 1:(num_gait_cycles-1)
m_lower = idx_lhs(m);
m_upper = idx_lhs(m + 1);

t_plot = accel_data(1,k).t(m_lower:m_upper) - accel_data(1,k).t(m_lower);
mag_plot = idx_left(m_lower:m_upper);

gait_data(k).left_foot(m).t = t_plot;
gait_data(k).left_foot(m).accel = mag_plot;

% Processing finished cycles only

        [T,S] = size(gait_data(k).left_foot(m).accel);
        Peaks = findpeaks(gait_data(k).left_foot(m).accel);
        [Q,P] = size(Peaks);
if S > 24;
if P > 4;
% gaussian fit to each gait cycle
            [xData, yData] =
prepareCurveData(gait_data(k).left_foot(m).t,....
gait_data(k).left_foot(m).accel );

% Set up fittype and options.
ft = fittype( 'gauss8' );
opts = fitoptions( ft );
```

```
opts.Display = 'Off';
opts.Lower = [-Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0 -
Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0];
opts.StartPoint = [25304.0846037163 0 0.0138089761463768 20574.9756149364
0.609999999999957 0.0240799643087144 19697.9331861504 0.0199999999999818
0.0123971874863801 18576.0098094693 0.109999999999957 0.00978562351870957
14870.6628491009 0.0399999999999636 0.0142068835652844 11052.9580286452
0.139999999999986 0.0122707734005006 10532.7090398849 0.0699999999999932
0.0167500967218961 5502.143072838 0.159999999999968 0.017599409350113];
opts.Upper =
[InfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInfInf];

% Fit model to data.
            [fitresult, gof] = fit(xData, yData, ft, opts );

% Gaussian coffecients for each steps

a1(k).trial(m) = fitresult.a1;
a2(k).trial(m) = fitresult.a2;
a3(k).trial(m) = fitresult.a3;
a4(k).trial(m) = fitresult.a4;
a5(k).trial(m) = fitresult.a5;
a6(k).trial(m) = fitresult.a6;
a7(k).trial(m) = fitresult.a7;
a8(k).trial(m) = fitresult.a8;
b1(k).trial(m) = fitresult.b1;
b2(k).trial(m) = fitresult.b2;
b3(k).trial(m) = fitresult.b3;
b4(k).trial(m) = fitresult.b4;
b5(k).trial(m) = fitresult.b5;
b6(k).trial(m) = fitresult.b6;
b7(k).trial(m) = fitresult.b7;
b8(k).trial(m) = fitresult.b8;
c1(k).trial(m) = fitresult.c1;
c2(k).trial(m) = fitresult.c2;
c3(k).trial(m) = fitresult.c3;
c4(k).trial(m) = fitresult.c4;
c5(k).trial(m) = fitresult.c5;
c6(k).trial(m) = fitresult.c6;
c7(k).trial(m) = fitresult.c7;
c8(k).trial(m) = fitresult.c8;
else
end
else
end
end

end

savegait_data.matgait_data
```

```
%% preparing the cofficient for features in the classification algorithm
% cutting parts of the data that is not fitted, since the cycle is not
% completed
% finding non zero points of cofficient a1 since a1 is always a non-zero
% number

for k=[1,2,3,13,14,15,7,8,9];

cut = find(a1(k).trial);
a1(k).complete = rot90(a1(k).trial[19]);
a2(k).complete = rot90(a2(k).trial[19]);
a3(k).complete = rot90(a3(k).trial[19]);
a4(k).complete = rot90(a4(k).trial[19]);
a5(k).complete = rot90(a5(k).trial[19]);
a6(k).complete = rot90(a6(k).trial[19]);
a7(k).complete = rot90(a7(k).trial[19]);
a8(k).complete = rot90(a8(k).trial[19]);
b1(k).complete = rot90(b1(k).trial[19]);
b2(k).complete = rot90(b2(k).trial[19]);
b3(k).complete = rot90(b3(k).trial[19]);
b4(k).complete = rot90(b4(k).trial[19]);
b5(k).complete = rot90(b5(k).trial[19]);
b6(k).complete = rot90(b6(k).trial[19]);
b7(k).complete = rot90(b7(k).trial[19]);
b8(k).complete = rot90(b8(k).trial[19]);
c1(k).complete = rot90(c1(k).trial[19]);
c2(k).complete = rot90(c2(k).trial[19]);
c3(k).complete = rot90(c3(k).trial[19]);
c4(k).complete = rot90(c4(k).trial[19]);
c5(k).complete = rot90(c5(k).trial[19]);
c6(k).complete = rot90(c6(k).trial[19]);
c7(k).complete = rot90(c7(k).trial[19]);
c8(k).complete = rot90(c8(k).trial[19]);

end
%% putting all the features from the different trials into one features
% with there respective feature type
% creating a matrix for each features and inserting the feature data
% initial conditions

n=1; t=0;
for n =[1,2,3,13,14,15,7,8,9];
if t==0;
        t = t + size(a1(n).complete);
        t = t(1,1);
        feature_a1(1:t,1) = [a1(n).complete];
        feature_a2(1:t,1) = [a2(n).complete];
        feature_a3(1:t,1) = [a3(n).complete];
        feature_a4(1:t,1) = [a4(n).complete];
        feature_a5(1:t,1) = [a5(n).complete];
        feature_a6(1:t,1) = [a6(n).complete];
```

```
        feature_a7(1:t,1) = [a7(n).complete];
        feature_a8(1:t,1) = [a8(n).complete];
        feature_b1(1:t,1) = [b1(n).complete];
        feature_b2(1:t,1) = [b2(n).complete];
        feature_b3(1:t,1) = [b3(n).complete];
        feature_b4(1:t,1) = [b4(n).complete];
        feature_b5(1:t,1) = [b5(n).complete];
        feature_b6(1:t,1) = [b6(n).complete];
        feature_b7(1:t,1) = [b7(n).complete];
        feature_b8(1:t,1) = [b8(n).complete];
        feature_c1(1:t,1) = [c1(n).complete];
        feature_c2(1:t,1) = [c2(n).complete];
        feature_c3(1:t,1) = [c3(n).complete];
        feature_c4(1:t,1) = [c4(n).complete];
        feature_c5(1:t,1) = [c5(n).complete];
        feature_c6(1:t,1) = [c6(n).complete];
        feature_c7(1:t,1) = [c7(n).complete];
        feature_c8(1:t,1) = [c8(n).complete];
        n = n+1;
        p = t+1;
elseif t==p;
        t = t + size(a1(n).complete);
        t = t(1,1)-1;
        feature_a1(p:t,1) = [a1(n).complete];
        feature_a2(p:t,1) = [a2(n).complete];
        feature_a3(p:t,1) = [a3(n).complete];
        feature_a4(p:t,1) = [a4(n).complete];
        feature_a5(p:t,1) = [a5(n).complete];
        feature_a6(p:t,1) = [a6(n).complete];
        feature_a7(p:t,1) = [a7(n).complete];
        feature_a8(p:t,1) = [a8(n).complete];
        feature_b1(p:t,1) = [b1(n).complete];
        feature_b2(p:t,1) = [b2(n).complete];
        feature_b3(p:t,1) = [b3(n).complete];
        feature_b4(p:t,1) = [b4(n).complete];
        feature_b5(p:t,1) = [b5(n).complete];
        feature_b6(p:t,1) = [b6(n).complete];
        feature_b7(p:t,1) = [b7(n).complete];
        feature_b8(p:t,1) = [b8(n).complete];
        feature_c1(p:t,1) = [c1(n).complete];
        feature_c2(p:t,1) = [c2(n).complete];
        feature_c3(p:t,1) = [c3(n).complete];
        feature_c4(p:t,1) = [c4(n).complete];
        feature_c5(p:t,1) = [c5(n).complete];
        feature_c6(p:t,1) = [c6(n).complete];
        feature_c7(p:t,1) = [c7(n).complete];
        feature_c8(p:t,1) = [c8(n).complete];
        n = n+1;
        p = t+1;
end
    t=p;
end
```

```
%% bringing all features in one data set for ease of modeling
% the data taken as indiviual binary classification

X = [feature_a1,feature_a2,feature_a3,feature_a4,feature_a5,feature_a6,....
    feature_a7,feature_a8,feature_b1,feature_b2,feature_b3,feature_b4,.....
    feature_b5,feature_b6,feature_b7,feature_b8,feature_c1,feature_c2,.....
    feature_c3,feature_c4,feature_c5,feature_c6,feature_c7,feature_c8];

% response data for the each cycle or row data
% that is the class of the data
% frist by separating the normal and the others

% initial conditions

n=1; t=0;
for n=[1,2,3,13,14,15,7,8,9];
if t==0;
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(1:t,1) = {'Normal'};
        p = t+1;
elseif (2<=n)&&(n<=3);
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(p:t,1) = {'Normal'};
        p = t+1;
elseif (13<=n)&&(n<=15);
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(p:t,1) = {'Normal'};
        p = t+1;
elseif (7<=n)&&(n<=9);
        t = t + size(a1(n).complete);
        t = t(1,1);
Y(p:t,1) = {'Ankle _Brace'};
        p = t+1;
else
end
    n = n+1;

end

%% partitioning data for training and test (for validation)

Z = cvpartition(Y,'holdout',0.3);

% training data

X_Train = X(training(Z,1),:);
```

```
Y_Train = Y(training(Z,1));

% test data

X_Test = X(test(Z,1),:);
Y_Test = Y(test(Z,1));

%% using Naive Bayes model for classifiction

X_Naive =
NaiveBayes.fit(X_Train,Y_Train,'Distribution','kernel','Prior','uniform');

% predict the test data based on the model

[Y_Naive] = predict(X_Naive,X_Test);

% Evaluating accuracy
% confusion matrix

[conf_Naive,order] = confusionmat(Y_Test,Y_Naive);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_Naive = 1-trace(conf_Naive)/sum(conf_Naive(:));

%% using TreeBagger

X_TreeBagger = TreeBagger(22,X_Train,Y_Train,'oobpred','on');
plot(oobError(X_TreeBagger))
xlabel('number of grown trees')
ylabel('out-of-bag classifcation error')

% predict the test data based on the model

Y_TreeBagger = predict(X_TreeBagger,X_Test);

% Ensemble predictions for out-of-bag observations.

Ensemble_TreeBagger = oobPredict(X_TreeBagger);

% Evaluating accuracy
% misclassfication probability

ERR_TreeBagger = error(X_TreeBagger,X_Train,Y_Train);

% confusion matrix

[conf_TreeBagger,order] = confusionmat(Y_Test,Y_TreeBagger);
```

```
% calculate what percentage of the Confusion matrix is off the diagonal

Error_TreeBagger = 1-trace(conf_TreeBagger)/sum(conf_TreeBagger(:));

%% using Nearest Neighbor
X_KNN = ClassificationKNN.fit(X_Train,Y_Train,'NumNeighbors',16);

% predict the test data based on the model

[Y_KNN] = predict(X_KNN,X_Test);

% Evaluating accuracy
% resubstitution loss; fraction of misclassification from prediction

RLoos_KNN = resubLoss(X_KNN);

% cross validation error

CVmdl_KNN = crossval(X_KNN);
KLoss_KNN = kfoldLoss(CVmdl_KNN);

% loss Classification error

Loss_KNN = loss(X_KNN,X_Test,Y_Test);

% confusion matrix

[conf_KNN,order_KNN] = confusionmat(Y_Test,Y_KNN);

% calculate what percentage of the Confusion matrix is off the diagonal

Error_KNN = 1-trace(conf_KNN)/sum(conf_KNN(:));

%% using SVM (suppor vector machine)

X_SVM = svmtrain(X_Train,Y_Train,'Kernel_Function','rbf');

% predict the test data based on the model

Y_SVM = svmclassify(X_SVM,X_Test);

% Evaluating accuracy
% confusion matrix

[conf_SVM,order_SVM] = confusionmat(Y_Test,Y_SVM);

% calculate what percentage of the Confusion matrix is off the diagonal
```

```
Error_SVM = 1-trace(conf_SVM)/sum(conf_SVM(:));
```

Appendix B : Curve fitting observations

B1: Gait cycle curve fit with observations using Vicon data set

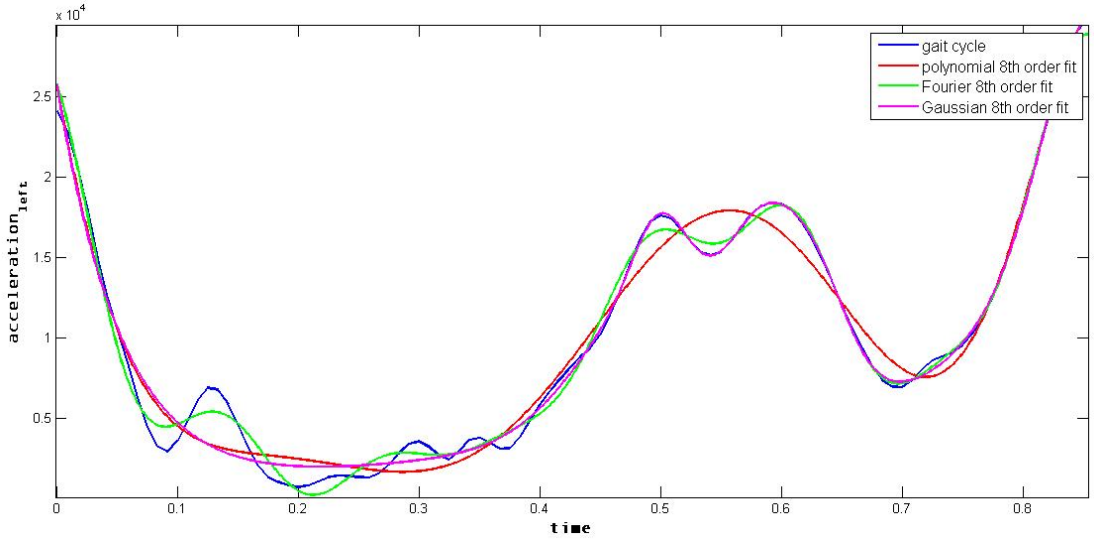


Figure B-1 Vicon gait cycle curve fit observation for the left foot

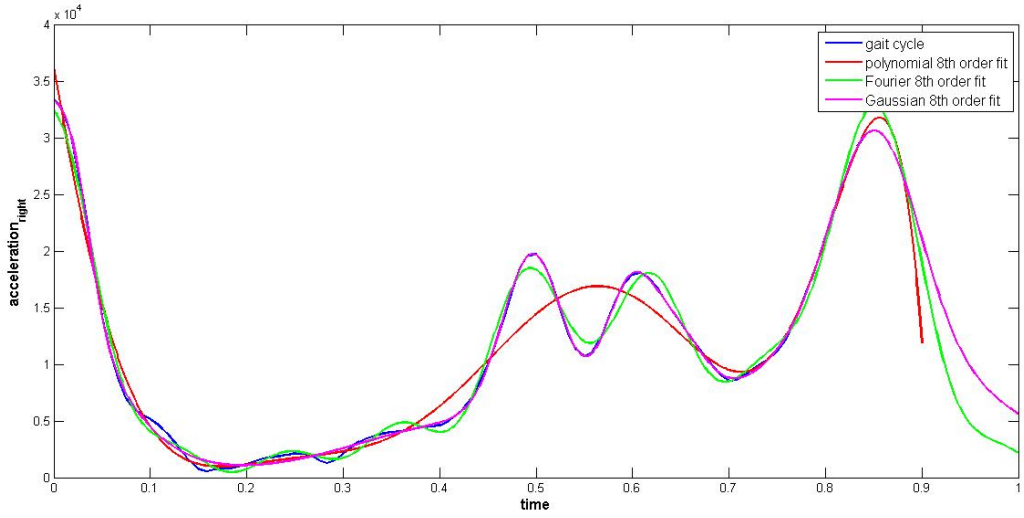


Figure B-2 Vicon gait cycle curve fit observation for the right foot

B2: Gait cycle curve fit with observations using IMU data set

B2-1: Normal gait cycle

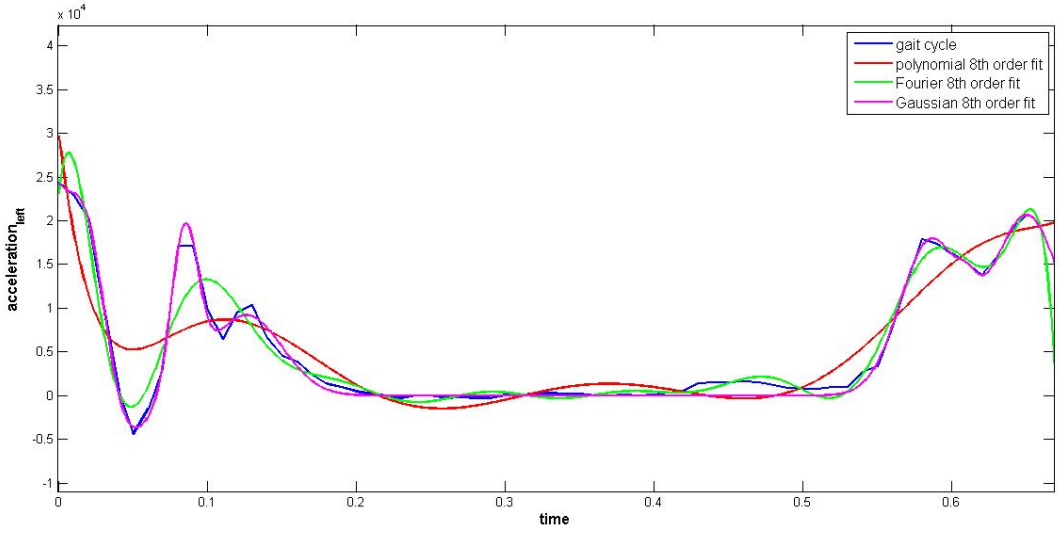


Figure B-3 IMU normal gait cycle curve fit observation for the left foot

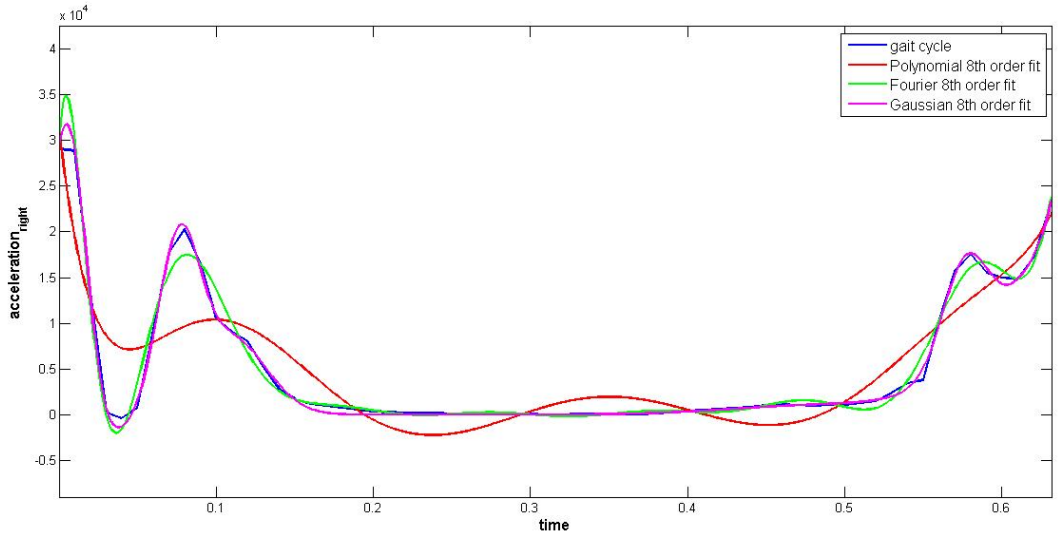


Figure B-4 IMU normal gait cycle curve fit observation for the right foot

B2-2: Ankle braced gait cycle

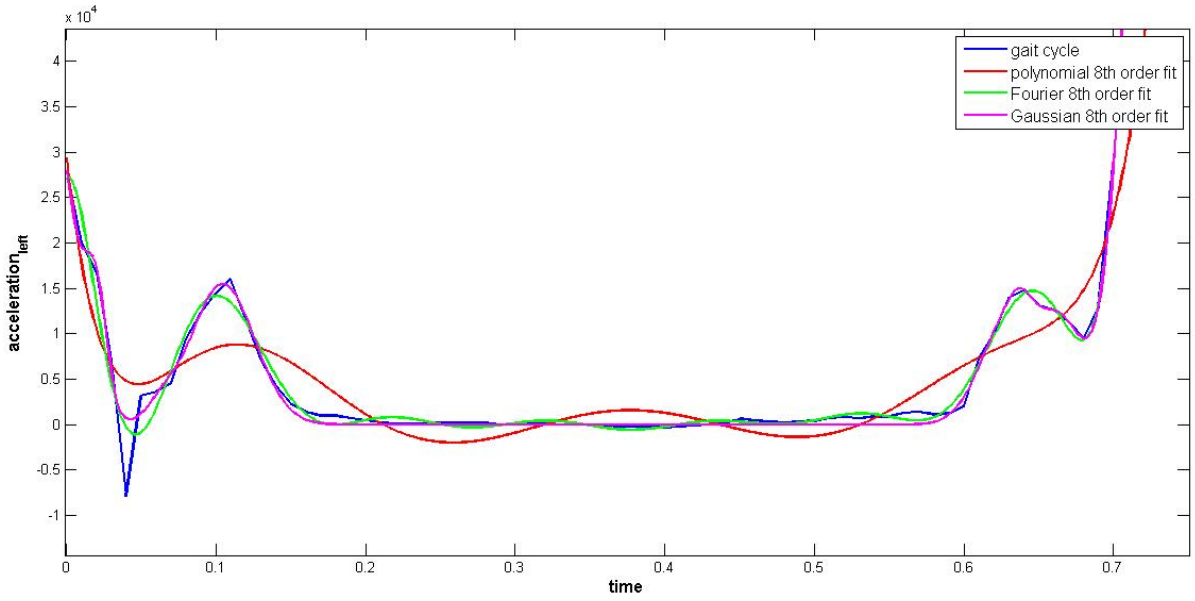


Figure B-5 IMU Ankle braced gait cycle curve fit observation for the left foot

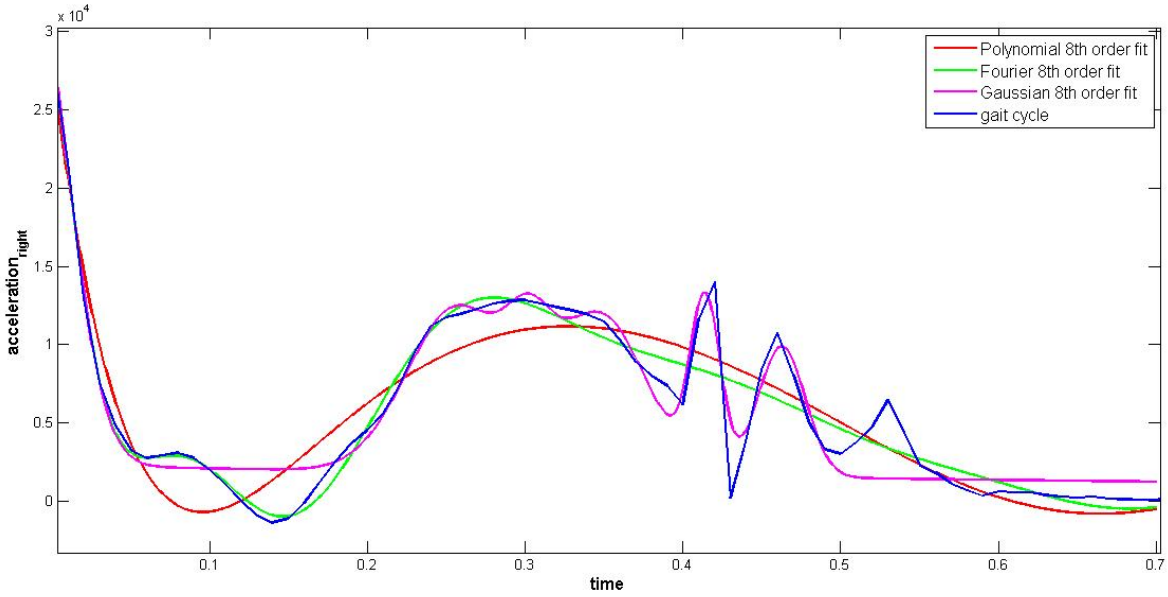


Figure B-6 IMU Ankle braced gait cycle curve fit observation for the right foot

B2-3: Knee braced gait cycle

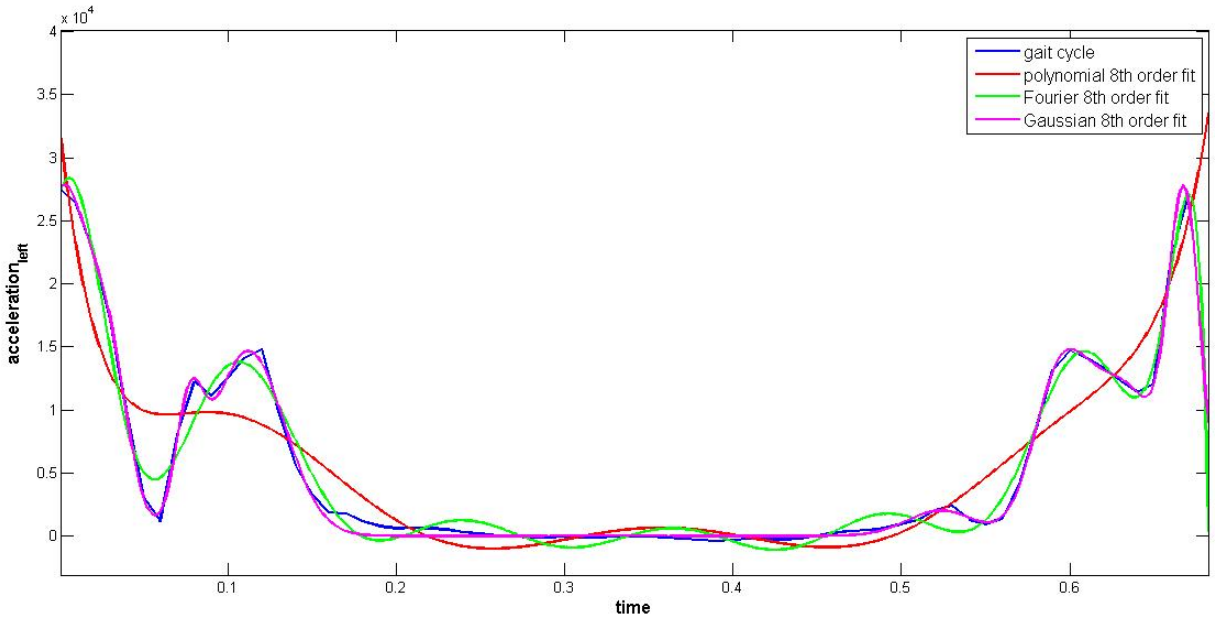


Figure B-7 IMU Knee braced gait cycle curve fit observation for the left foot

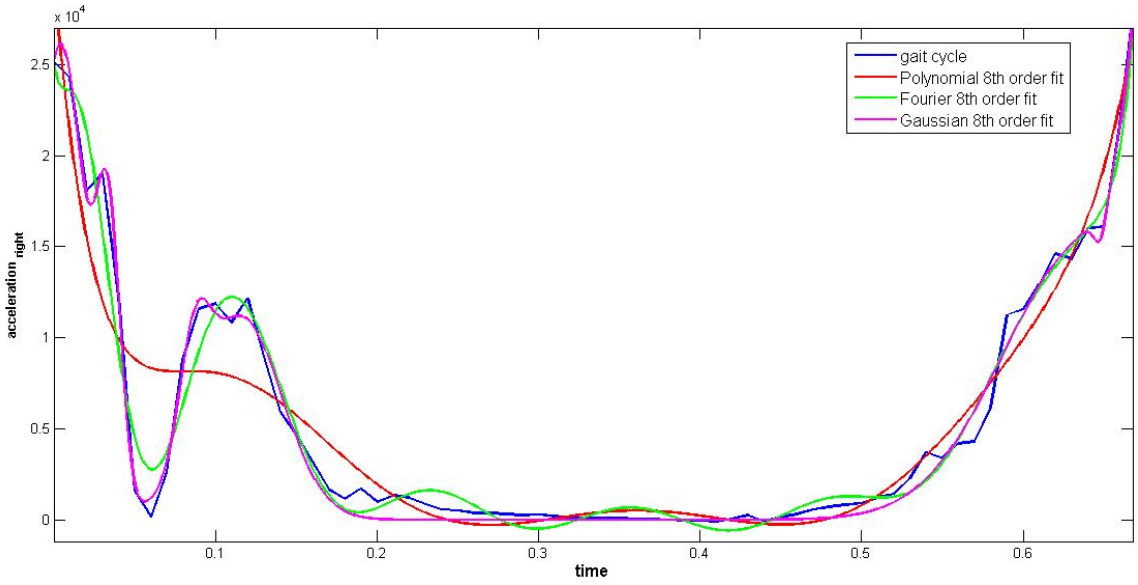


Figure B-8 Knee braced gait cycle curve fit observation for the right foot

B2-4: Ankle&Knee braced gait cycle

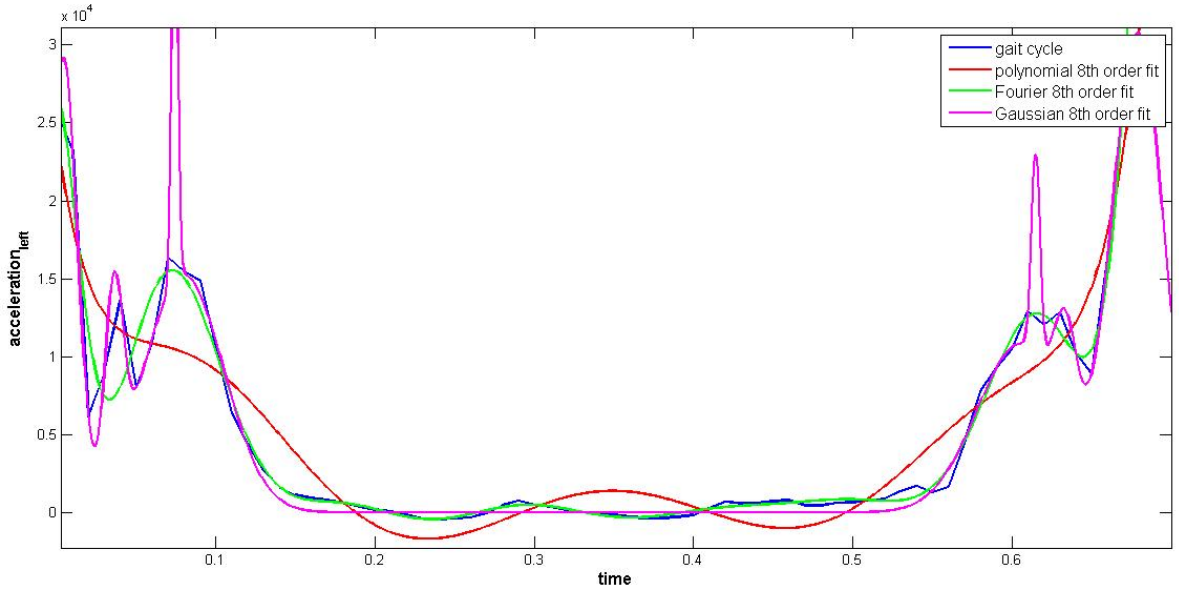


Figure B-9 IMU Ankle&Knee braced gait cycle curve fit observation for the left foot

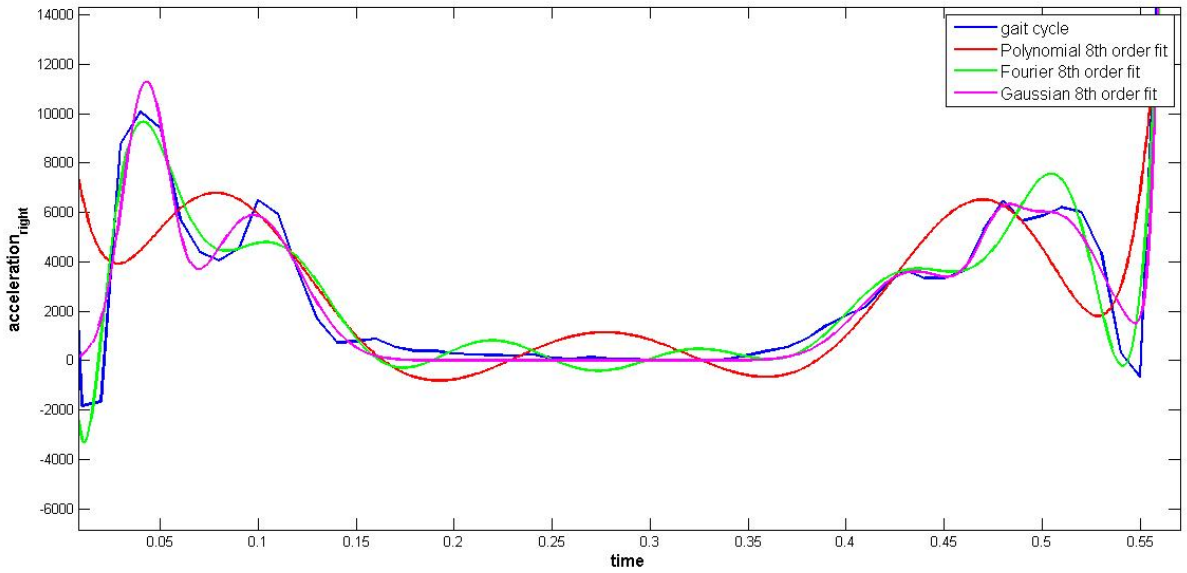


Figure B-10 IMU Ankle&Knee braced gait cycle curve fit observation for the right foot

Appendix C : Sample Curve fittings

C1: Normal Gait cycle curve fit using IMU data set

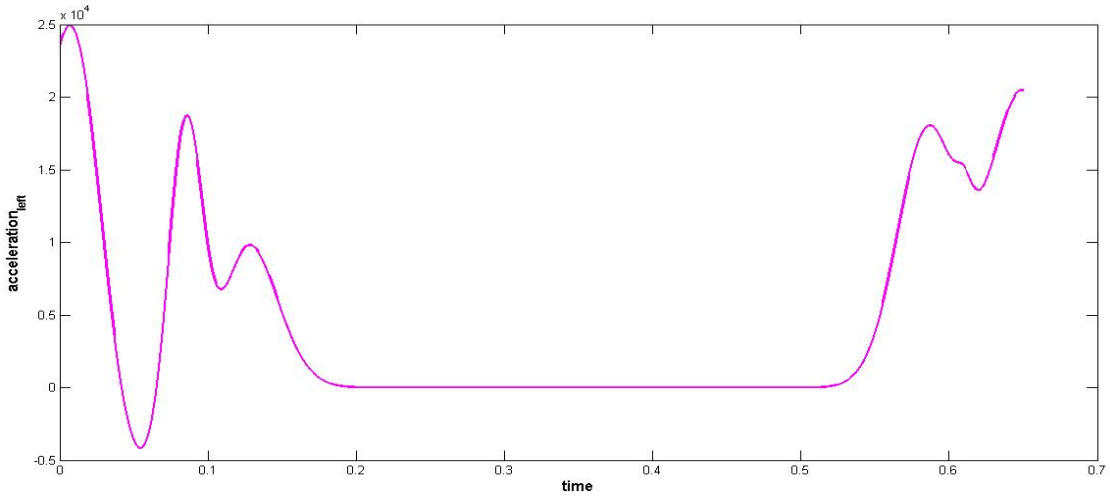


Figure C-1 IMU normal gait cycle curve fit for the left foot

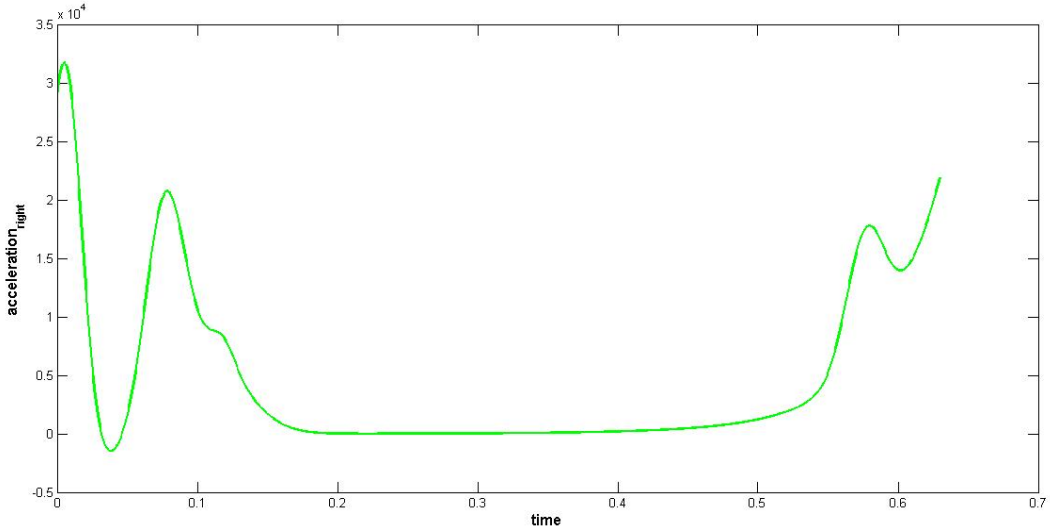


Figure C-2 IMU normal gait cycle curve fit for the right foot

C2: Ankle braced Gait cycle curve fit using IMU data set

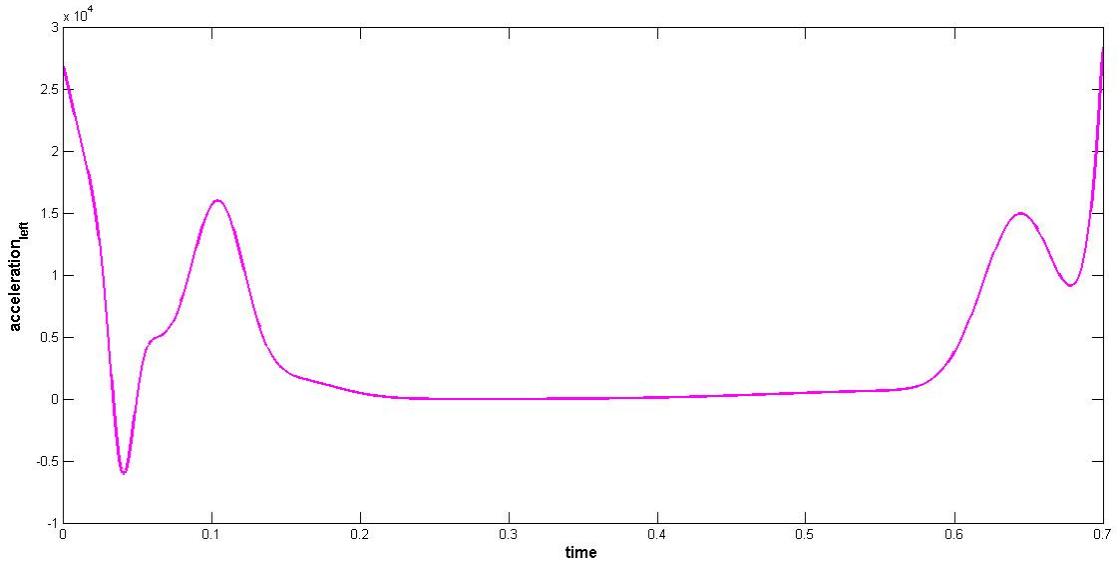


Figure C-3 IMU Ankle braced gait cycle curve fit for the left foot

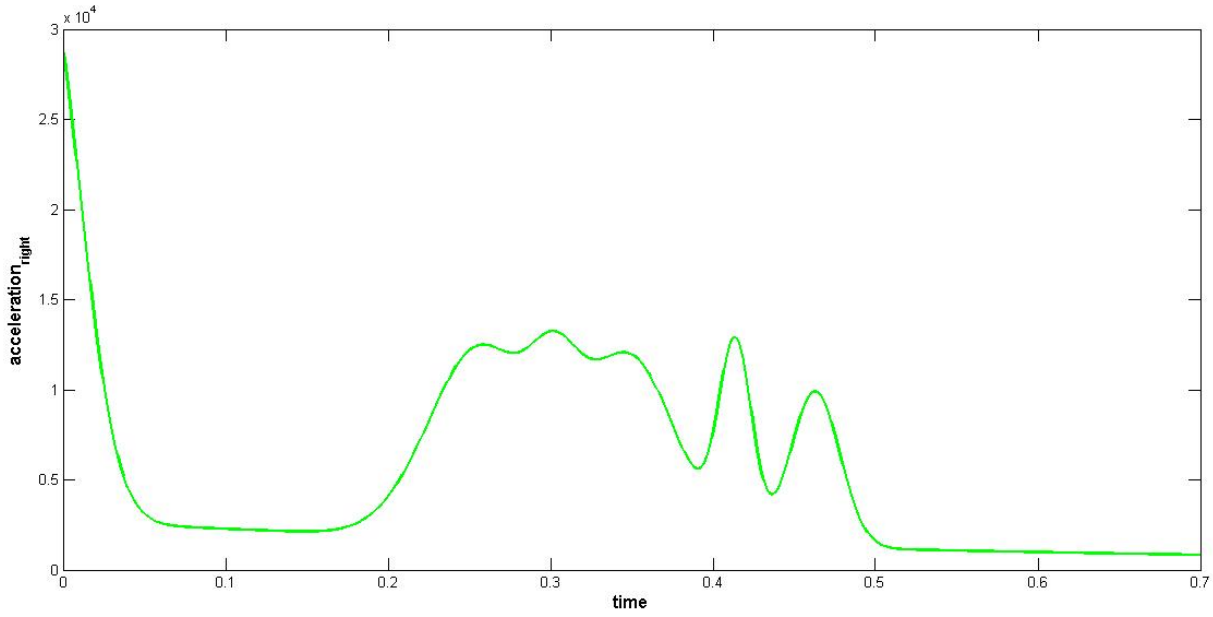


Figure C-4 IMU Ankle braced gait cycle curve fit for the right foot

C3: Knee braced Gait cycle curve fit using IMU data set

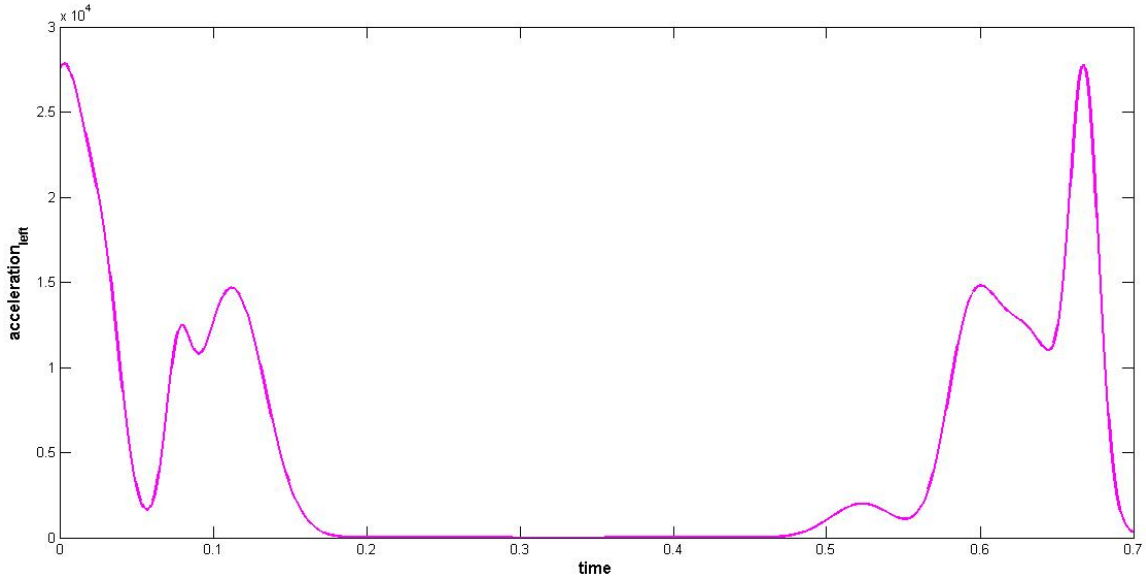


Figure C-5 IMU Knee braced gait cycle curve fit for the left foot

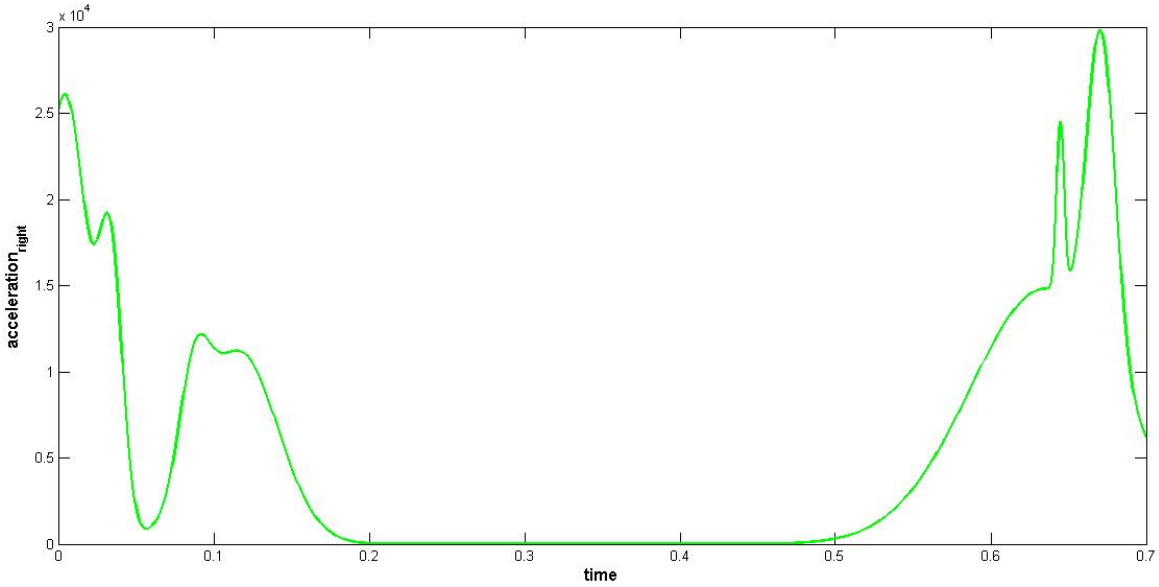


Figure C-6 IMU Knee braced gait cycle curve fit for the right foot

C4: Ankle&Knee braced Gait cycle curve fit using IMU data set

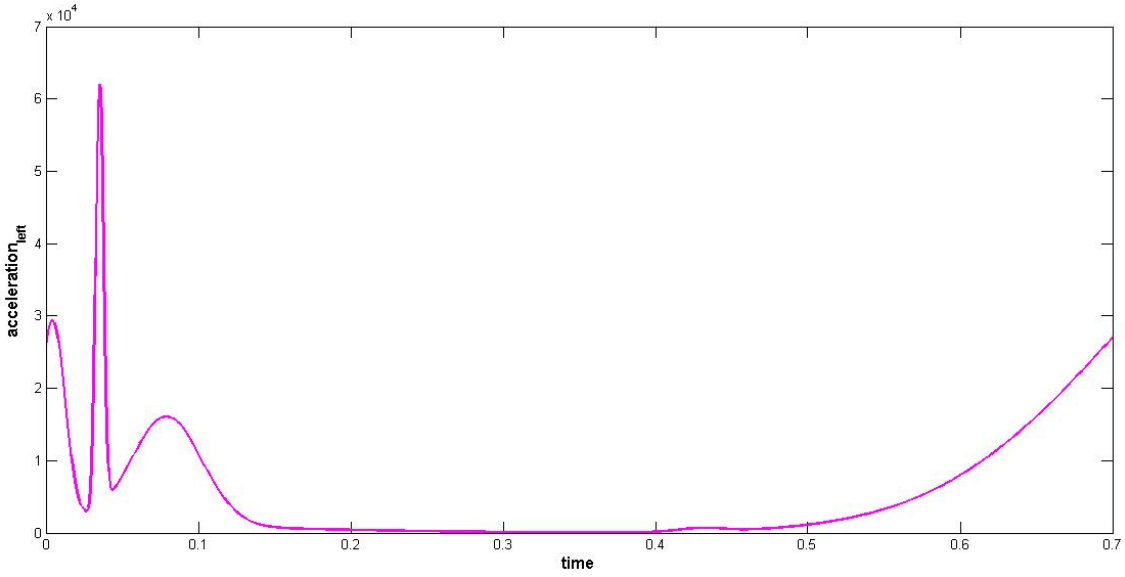


Figure C-7 IMU Ankle&Knee braced gait cycle curve fit for the left foot

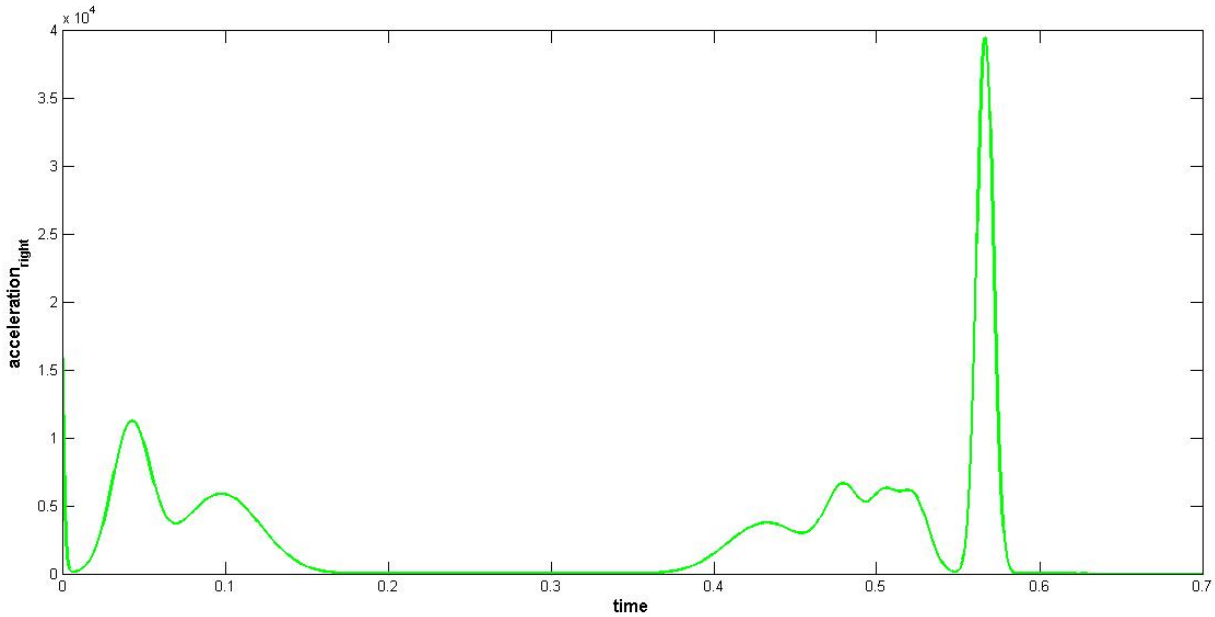


Figure C-8 IMU Ankle&Knee braced gait cycle curve fit for the right foot

Appendix D : Illustration of similarity between normal gait patterns using Vicon motion capture data

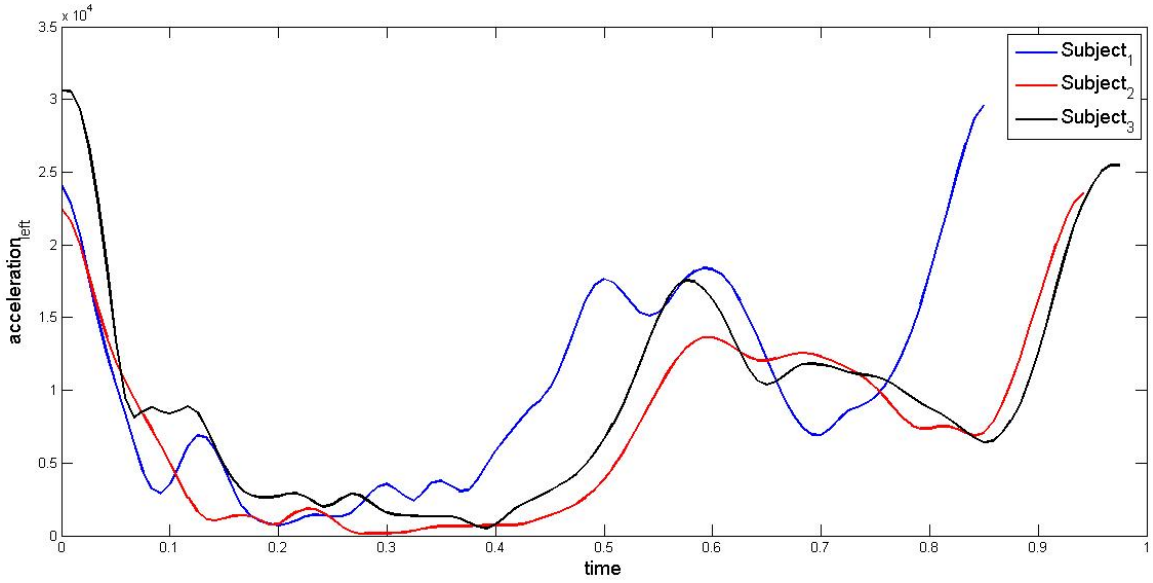


Figure D-1Left foot Vicon collected acceleration plot form three subjects

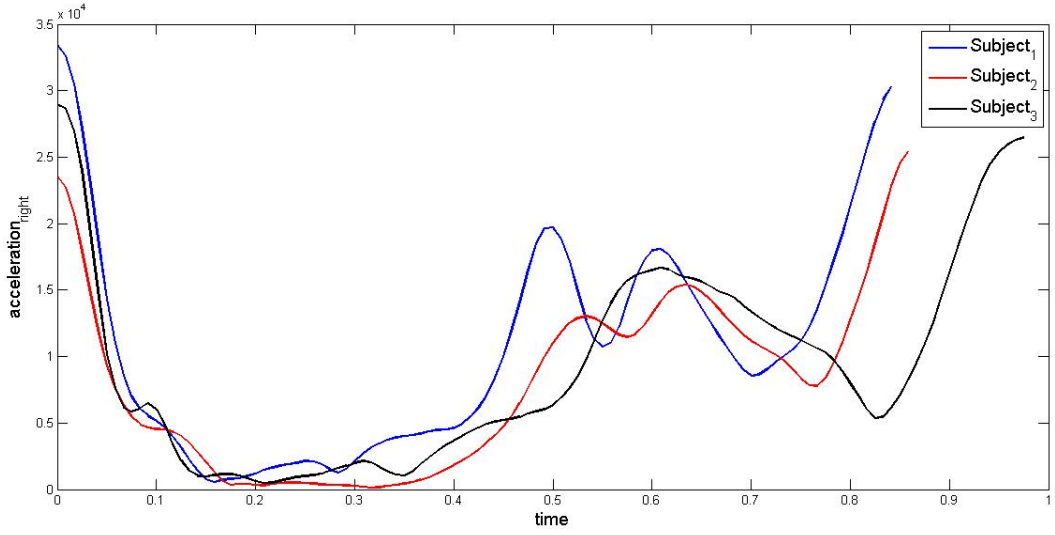


Figure D-2Right foot Vicon collected acceleration plot form three subjects