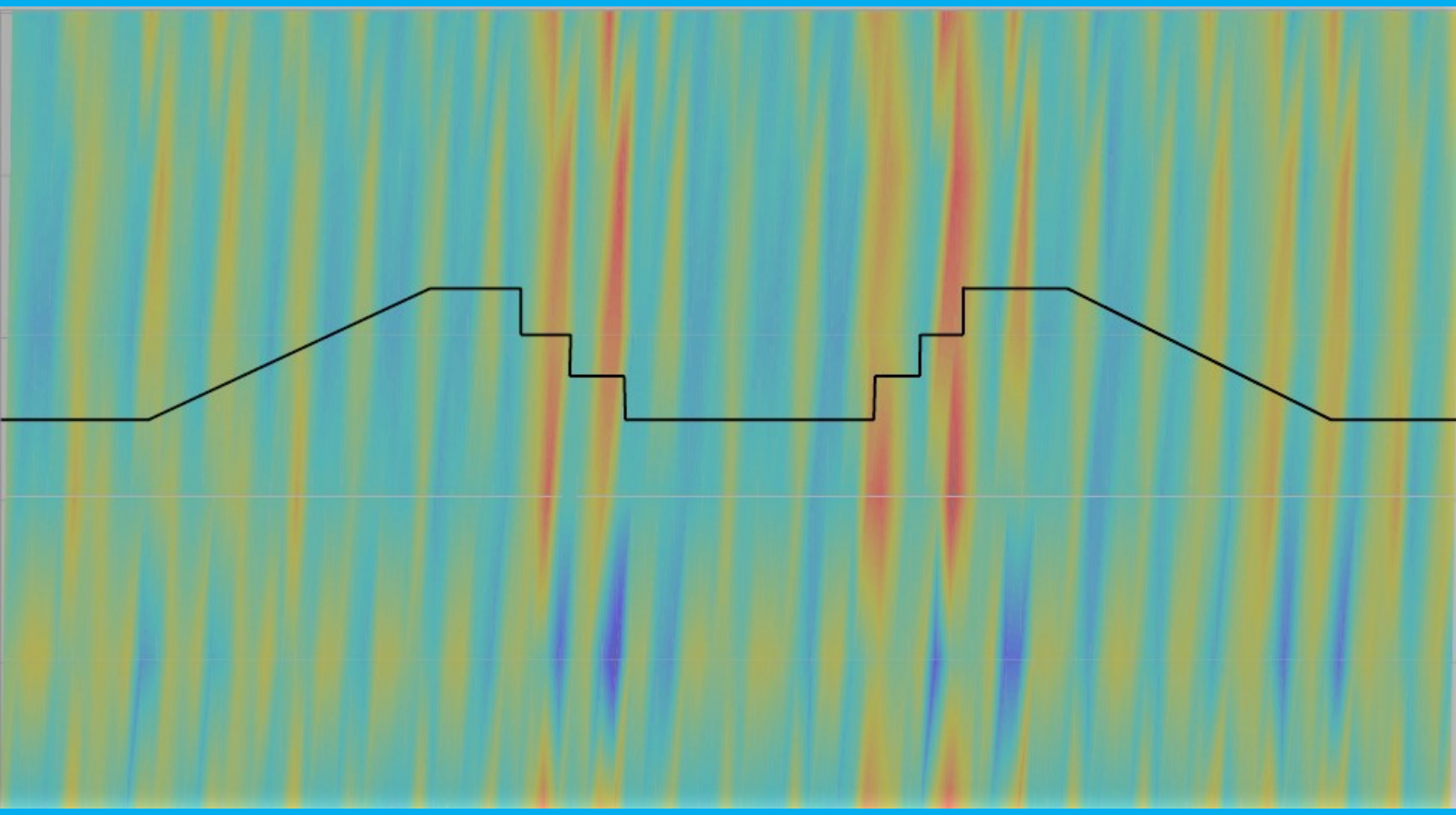
# Locomotion Intent Recognition Using A Multimodal Sensor Fusion Approach

Manav Penubaku



TU Delft

# Locomotion Intent Recognition Using A Multimodal Sensor Fusion Approach

by

**Manav Penubaku**

in partial fulfillment of the requirements for the degree of

**Master of Science**
in Mechanical Engineering

at the Delft University of Technology,
to be defended publicly on Friday September 7, 2018 at 9:30 AM.

| | | |
|---|---|---|
| Supervisor: | Dr. ir. R. Fluit | |
| | Ir. S. Wang , | Reboocon Bionics |
| Thesis committee: | Prof. dr. ir. H. van der Kooij, | TU Delft & University of Twente |
| | Dr. ir. R. Fluit, | Univeristy of Twente |
| | Dr. ing. J. Kober, | TU Delft |
| | Dr. ir. G. Smit, | TU Delft |
| | Ir. S. Wang | Reboocon Bionics |

*This thesis is confidential and cannot be made public until September 7,2023*

TU Delft Delft University of Technology

reboocon bionics

UNIVERSITY OF TWENTE.

# Preface

This thesis concludes my 2 year journey in the pursuit of a Master's degree in Mechanical Engineering. It has been a very pleasant experience during which there has been a lot of personal and professional growth. I had chosen the specific track of Biomechanical Design with the intention of working on prosthetic devices and this thesis is a small step in the direction of the field.

The thesis helped me apply the concepts learned during the course of my Master's and also helped in exploring new fields such as machine learning and pattern recognition. I also got practical exposure to data collection and processing. All of these factors led to a very enriching experience while working on the thesis. The analysis done in this project would aid in the development of active knee prosthesis and I look forward to seeing the results when it is implemented on the device.

This project was made possible by a few people. I would like to thank my mentor Dr. Rene Fluit for all the valuable feedback and regular discussions on the project. I would also like to thank Shiqian Wang, founder of Reboocon Bionics for giving me the opportunity to work on the project and for regular feedback on the project.

I would also like to thank my family and friends for always being supportive of my decisions and their encouragement to pursue my interests.

*Manav Penubaku*
*Delft, August 2018*

# Abbreviations

| | |
|---|---|
| **MPKs** | Micro-processor controlled knee prostheses |
| **P-MPKs** | Passive Knee Prostheses |
| **A-MPKs** | Active Knee Prostheses |
| **NMPKs** | non-microprocessor knees |
| **LW** | Level Ground Walking |
| **Si** | Sitting |
| **St** | Standing |
| **RA** | Ramp Ascent |
| **RD** | Ramp Descent |
| **SA** | Stair Ascent |
| **SD** | Stair Descent |
| **SVM** | Support Vector Machine |
| **RVM** | Relevance Vector Machine |
| **LDA** | Linear Discriminant Analysis |
| **NB** | Naive Bayes |
| **KNN** | K-Nearest Neighbor |
| **ANN** | Artificial Neural Networks |
| **NM** | Nearest Mean |
| **BDT** | Binary Decision Trees |
| **OvO** | One Versus One |
| **OvR** | One Versus Rest |
| **IMU** | Inertial Measurement Units |
| **EMG** | Electromyography |
| **DBN** | Dynamic Bayesian Network |
| **HMM** | Hidden Markov Model |
| **ANN** | Artificial Neural Networks |
| **GMM** | Gaussian Mixture Model |
| **PCA** | Principal Component Analysis |
| **FDA** | Fisher Discriminant Analysis |
| **GRF** | Ground Reaction Force |

| | |
|---|---|
| **pdf** | probability distribution function |
| **EM** | Expectation Maximization |
| **TO** | Toe Off |
| **HC** | Heel Contact |
| **HRB** | heuristic rule based |
| **FSM** | Finite State Machine |
| **SL** | Stride Length |
| **SH** | Stride Height |
| **SEM** | Standard Error of the Mean |
| **AUC** | Area under Receiver Operating Characteristic Curve |
| **VAF** | Variance Accounted For |

# Abstract

An estimated 40,000 people have trauma related above knee amputations in the United States alone. A transfemoral prosthesis is an artificial limb that replaces the amputated limb. Although trauma related amputations are going down annually, there is still the need for prostheses that are capable of restoring normal biological knee function. Active microprocessor knees are a type of transfemoral prosthesis that can supply energy for activities therefore making many activities of daily living like chair and stair negotiation possible, however these devices are not yet commercially available as it does not yet meet the requirement of robust and unambiguous mode switching. One aspect of active microprocessor knees that needs improvement is the intent recognition system that perceives the intent of the user. A novel hybrid intent recognition algorithm based on machine learning using solely mechanical signals was developed and tested in this thesis. The algorithm is capable of distinguishing between the modes Standing, Sitting, Walking, Ramp Ascent, Ramp Descent, Stair Ascent and Stair Descent. The analysis of the algorithm was done on an open source healthy subject gait data set containing a total of 476 trials. The analysis involved determining recognition error rates and decision times for a novel subject's data. The proposed algorithm fuses data from Inertial Measurement Units(IMUs) worn on the shank and knee joint encoders to make the decisions. The algorithm can achieve an overall error rate of 14.28%, the error rate reduces to 2.62% when grouping the Ramp Ascent and Ramp Descent together with the Walking mode. Decision times are, on average 9.59$ms$ after a transition for critical transitions between stair modes and walking. For transitions between less critical modes like sitting and standing, decisions are taken with a maximum delay of 610ms. All transitions were successfully detected in 229 out of the 476 trials. The remaining trials had misclassifications due to improper labelling and variations in gait speed among the users. A preliminary analysis into adding ground reaction forces and moments indicates that the error rates can be decreased with it's use. The research concludes that a hybrid classifier in which ramp walking is treated as level ground walking is a good starting point for implementing on the transfemoral prosthesis.

# Contents

# 1

# Introduction

Lower limb loss due to amputation is still quite prevalent despite medical advancements over the past century. In the United States alone, 1.5 million people are living with limb loss. 90% of the cases are due to cardiovascular disease, 6% due to trauma, 2% due to cancer and 2% due to congenital related diseases [1],[2],[3]. These statistics are for both above-knee (transfemoral) and below-knee (transtibial) amputations - with 20% being for the former case.
Transfemoral prostheses are used to restore limb functionality for above knee amputees. There are two types of transfemoral prosthesis: Micro-processor controlled knee prostheses (MPKs) and mechanical non-microprocessor knees (NMPKs). NMPKs are not capable of restoring normal knee function and have the following drawbacks:the metabolic cost is much higher when walking with the device, users tend walk at a lower self selected speed than they prefer, users tend to develop an asymmetric gait and finally energy demanding tasks like stair ascent and descent cannot be performed. On the other hand, MPKs shown in Fig.1.1, attempt to simulate normal biological knee function through the use of sensory information that offers real time auto adaptive control of the joint movement.

Figure 1.1: Active Transfemoral Prosthesis from the MIT Biomechatronics Lab [4]

There are two types of MPKs, variable damping/Passive Knee Prostheses (P-MPKs) and Active Knee Prostheses (A-MPKs). Most commercial prostheses are P-MPKs which are passive devices and cannot do positive work. P-MPKs allow the user to choose the walking speed[5] and negotiate stairs and ramps[6][7] by adjusting the impedance.
A-MPKs have been gaining interest in research institutes over the recent past because of their ability to inject energy and do positive work. This would make possible daily activities like step-over-step stair negotiation, chair negotiation and ramp ascent[8]. These devices come

equipped with sensors like Inertial Measurement Units (IMU) that have 3-axis accelerometers to measure accelerations, 3-axis gyrometers that measure angular velocities and 3-axis magnetometers that measure magnetic flux; position and velocity encoders for the knee and ankle; load cells to measure contact forces; surface Electromyography (EMG) sensors that measure residual muscle activation and knee torque encoders.
Many studies have investigated the advantages of P-MPKs over NMPKs with regard to kinematics and kinetics of multiple activities. However, no consensus is found in studies investigating the advantages of A-MPKs with respect to P-MPKs[6][9]. This is because the performance of A-MPKs is dependent on the control architecture. This thesis is part of a larger study that aims to demonstrate the benefits of A-MPKs. The presence of multiple sensors makes it possible to implement intelligent control strategies. The following sections contains a description of the control strategy used for A-MPKs.

## 1.1. Control of active lower limb prostheses

Tucker *et al.* [10] proposed a generalized control framework for active lower limb prostheses and orthoses and is as shown in Fig. 1.2. It has a hierarchical structure with three layers: high level, mid level and low level.

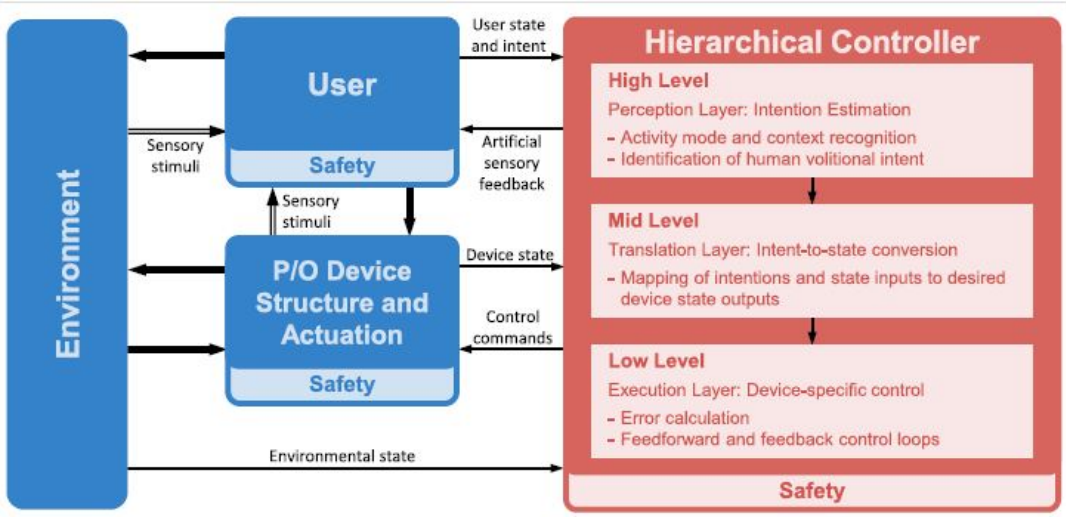



Figure 1.2: Generalized control framework from article by Tucker *et al.* [10]

The high level layer is the layer in which the controller has to determine the user's locomotion intent. These locomotion intents encompass various activities of daily living(ADLs). Activity modes that are of interest in research MPKs include Standing (St), Sitting (Si), Level Ground Walking (LW), Stair Ascent (SA), Stair Descent (SD), Ramp Ascent (RA) and Ramp Descent (RD). Commercial MPKs like the Ottobock Genium[11] also incorporate additional activities of stumble recovery, backwards walking, kneeling and running.

Various methods exist for the high level implementation. One such method is a user defined input from a button interface or an interface in a smart phone. Alternatively some devices require the user to perform a specific manoeuvre that would put the device in the right mode. Another method is to use an algorithm that can automatically infer the user intent. Such an algorithm can be formulated using a heuristic rule based (HRB) approach and is widely used in commercial MPKs, alternatively a machine learning approach can also be used. These methods are further described in section 1.2.

The mid level layer translates the user's intent to desired device states for the low level controller to track. This is implemented as a Finite State Machine (FSM) where the device can only be in one of a finite number of states that are sequentially connected. Each activity of interest has an FSM associated with it . For instance the gait cycle FSM has two phases, the stance phase and the swing phase each comprising of a few states as shown in Fig. 1.3. The

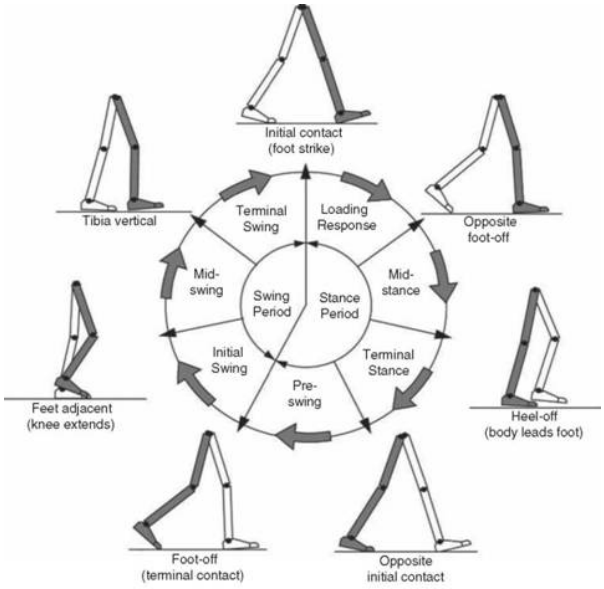



Figure 1.3: Gait phases and states labelled with respect to the leg in grey[12]

state within a gait cycle is determined in this layer and a control law is applied accordingly. This control law can be implemented as a position,velocity, torque, impedance, or admittance controller.
The desired device state that is determined at the mid-level layer is passed on to the low-level controller. This layer computes the error with respect to the current state and sends commands to the actuator. This is done either using feed-forward or feedback control and has to take into account the kinematics and the kinetics of the device.

## 1.2. Locomotion Intent Recognition

This thesis will focus on locomotion intent recognition, which corresponds to the high level layer highlighted in the previous section, an algorithmic approach based on sensor fusion is adopted. Locomotion intent recognition can be done in two ways

1. User Dependent: Data is used from each individual user to come up with a decision model. The performance is then tested on the same user.

2. User Independent: Data obtained from one or more users is used to obtain a decision model. The performance is then tested for a novel user.

As mentioned before there are two possible algorithmic approaches to come up with the decision models.

### 1.2.1. Heuristic Rule Based Approach(HRB)

The HRB approach uses knowledge of gait kinematics to come up with certain rules that define the decision models. These models operate using conditional logic and thresholding schemes. In addition, HRB approaches are straight forward and easy to understand for the user. The amputee can also be trained by a clinician to perform specific movements that trigger the MPK correctly. However, with increasing number of modes, different movements are needed to distinguish between the modes. These new movements can feel unnatural and increases the user cognitive burden. This approach is best suited for the user dependent case, but not for the user independent case as it doesn't generalize well. The reason for this is that inter subject gait variability makes it difficult to draft universal rules while maintaining the simplicity of the rules. Another issue with this approach is that the decision times are dependent on the rules/thresholds used.

### 1.2.2. Machine Learning Approach

Machine learning is a field of computer science which uses statistical techniques to give computers the ability to learn a model with the help of data without being explicitly programmed to do the task[13]. Figure 1.4 shows a block diagram representation of machine learning.

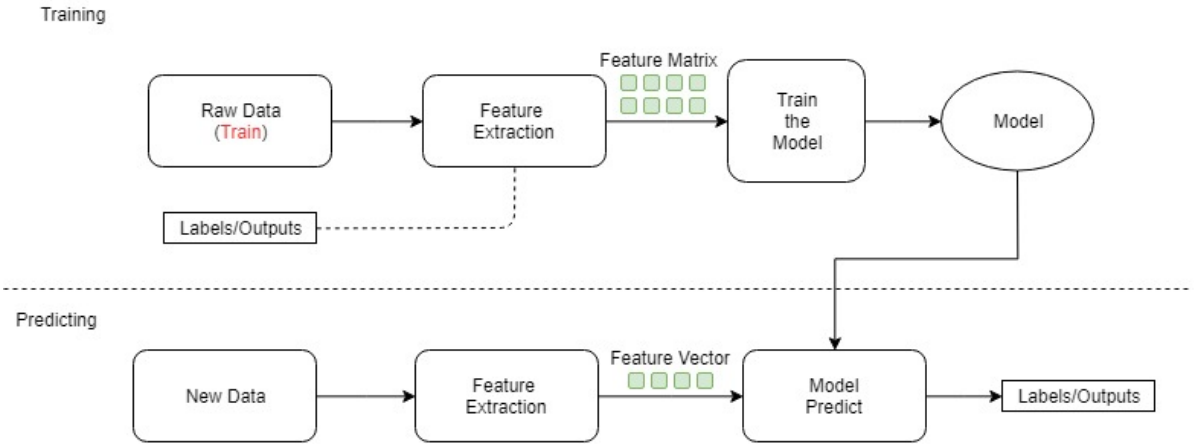



Figure 1.4: Machine Learning Block Diagram

It has a training phase and a predicting phase. During the training phase, raw data is supplied for the specific task that the model will be used for. Features which are important characteristics of the raw data are extracted e.g. initial orientation of shank segment. For sensor data, features are extracted over windows of a specific time interval. Multiple features are usually extracted and assembled as a feature vector. The feature vectors for the training data can then be stacked row-wise to obtain a feature matrix. In order to obtain a model, labels or outputs are needed which governs the task that the model has to perform. The feature matrix along with the labels/outputs are used to train a model. Different algorithms exist for training the model like Support Vector Machine (SVM),Linear Discriminant Analysis (LDA),Artificial Neural Networks (ANN),Naive Bayes (NB) to name a few. During the predicting phase, the trained model takes in feature vectors extracted from new data and predicts the label/output.
Machine learning tasks fall in 2 categories

1. **Supervised Learning**

   In these tasks the algorithm is provided with example input feature matrices and their corresponding labels/outputs, the goal is then to learn a mapping between the input space and the output space resulting in a model that is used to make predictions.

2. **Unsupervised Learning**

   In these tasks the algorithm is only provided with example inputs, labels/outputs are either withheld or aren't available. Therefore the algorithm has to find patterns in the input on it's own. These techniques try and uncover hidden patterns in the data and in some cases are used for feature learning wherein features are automatically discovered from raw data.

Machine learning can be used for broadly 3 different applications

1. **Classification**

   Classification is a supervised learning task. Inputs are divided into two or more classes which are defined as labels which are discrete entities eg:Sitting and Standing modes. The classifier algorithm then learns a model using the class labels given to it during training, this model is then used to assign labels for unseen inputs.

2. **Regression**

   Regression is also a supervised learning task, where the outputs are continuous values rather than discrete labels. The goal is to predict the output based on the input data E.g. determining the torque required to perform a manoeuvre.

3. **Clustering**

   Clustering is an unsupervised learning task. A set of inputs is provided which has to be divided into groups,however unlike the classification task these groups are not known beforehand.

Locomotion intent recognition requires the use of supervised learning for classification. Such an approach is robust and can generalize well. This is because they use features from all available sensor signals to come up with complex decision models. The models then automatically infer the intent for new data. Cognitive burden is lower when using this approach as the user does not have to be aware of any rules. This approach has some drawbacks though. Firstly, the training times can be high based on the algorithm chosen for training the model. Secondly, decision times can also be high based on the framework used to make decisions. One such framework is the majority voting scheme, where a fixed number of outputs are stored in a vector. A decision is made only if a certain proportion of outputs of the vector are in agreement, therefore longer decision times would result.
Despite the drawbacks, a machine learning approach was adopted in this thesis because of it's generalization capabilities. This would make the development of a user independent algorithm feasible. The following section gives an overview of the state of the art in locomotion intent recognition and sets up the objectives for the thesis.

## 1.3. State of the art in user locomotion intent recognition

Supervised learning techniques are the preferred choice of machine learning techniques among researchers.The detailed literature review is present in Appendix- B and some key points have been summarized in the rest of this section.
An offline analysis is a part of almost all of the research in the field. In offline analysis, pre recorded data is used to train and test the algorithm. Performance criteria like the overall classification accuracies/errors have been reported. The overall classification accuracy is usually separated as a steady state accuracy for decisions where the activity mode doesn't change, and a transitional accuracy for decisions where the activity mode does change. Some articles also test the learned model in real time implementations, decision times have been reported for these cases.

### 1.3.1. Sensors

Different sensors have been used for the intent recognition problem. Some articles focus only on surface EMG signals[14]–[18]. EMG signals have many issues. They can be distorted by nearby magnetic fields from the actuator coil and are affected by conditions like humidity and temperature. The EMG sensors are located in the socket, therefore during donning and doffing the electrode position can change which introduces additional variability in the signal. Therefore EMG signals are not repeatable. This leads to a decreasing accuracies with time when used for intent recognition.

Mechanical sensor signals from IMU [17], [19]–[28], goniometers[19], [27]–[31], pressure sensored insoles [32], [33], load cell sensors[34], [35] and laser sensors[36] have also been used by researchers for intent recognition. Mechanical signals are normally used along with EMG signals in a neuromechanical fusion approach, this enhances the feature vector and the highest reported classification accuracies are for this case. However, even this approach has a deterioration in performance because of the EMG signal quality. One fix for this issue as reported in [37], [38] is to use adaptive classifiers.However, it would require data to be stored during the use of the prosthesis and then retraining the model with that data. Such data storage after initial fitting of the prosthesis may not be feasible in all cases.
Mechanical sensors unlike the EMG sensors do not suffer from deterioration of signal quality.Therefore, classifiers based purely on them would give robust results.

### 1.3.2. Machine Learning Algorithms

Most research focuses on distinguishing the modes LW,RA,RD,SA and SD. The reason for this is the presence of discrete gait events like Heel Contact (HC) & Toe Off (TO) shown in Figure 1.3 as initial contact and foot off respectively. These gait events can be used to define specific windows from which features can be extracted. This ensures that the features are repeatable and makes it possible for the algorithm to pick up patterns. Algorithms like LDA,SVM,NB,K-Nearest Neighbor (KNN),ANN,SVM,Dynamic Bayesian Network (DBN),Hidden Markov Model (HMM), Nearest Mean (NM),Binary Decision Trees (BDT) have been used for this. For distinguishing between modes like St,Si where it's hard to define a discrete event, the Gaussian Mixture Model (GMM) has been used. Table 1.1 contains an overview of the range of overall user dependent accuracies obtained for healthy subject gait and amputee gait using the previously mentioned algorithms.

| Classifier | Reference | User Dependent Accuracy | |
|---|---|---|---|
| | | Healthy Subjects | Amputees |
| Linear Discriminant Analysis(LDA) | [14], [19], [25], [39]–[41] | 87-98% | 91-99.5% |
| Naive Bayesian(NB) | [20], [39], [42] | 84-90% | |
| K-Nearest Neighbor(KNN) | [20], [39], [42], [43] | 87% | |
| Artificial Neural Networks(ANN) | [20], [25], [44]–[47] | 83-96% | 90-97.8% |
| Support Vector Machine(SVM) | [14], [15], [20], [24], [38], [42], [48] | 90-98% | 92.5-98.36% |
| Dynamic Bayesian Network(DBN) | [19], [20], [37], [41], [49] | 92-99.4% | 92-98% |
| Hidden Markov Model(HMM) | [20], [32] | 96-98.4% | 95.8% |
| Nearest Mean(NM) | [20] | 98.5% | |
| Binary Decision Trees(BDT) | [20], [42], [50], [51] | 81-93% | |
| Gaussian Mixture Model(GMM) | [28], [30], [31], [52] | 91.3-92.2% | - |

Table 1.1: Overview of machine learning algorithms

Neuromechanical sensor fusion has been the focus in most research, however this method wouldn't perform well outside the lab because of the issues with signal quality mentioned before. Mechanical sensor based algorithms would give repeatable results and are thus more suited for outside the lab implementations. The algorithms highlighted in green in Table 1.1 have been implemented with solely mechanical sensor data. Young *et al.*[41] reported steady state user dependent accuracies of $\sim$ 98% when using LDA and $\sim$ 99% when using DBN. Woodward *et al.*[25] report steady state user dependent accuracies of $\sim$ 98% when using ANN. Liu *et al.*[38] report steady state user dependent accuracies in the range $93-99\%$ when using a HMM. The transitional accuracies are usually $5-12\%$ lower than the steady state accuracies.

All the algorithms based on mechanical sensors, use simple features of the signals such as the mean, standard deviation, minimum , maximum, initial and final values within the window. The use of more complex features like the absolute orientation of the thigh, stride height and stride length can give improved transitional accuracy.

A consistent issue with all algorithms is that they have low accuracies when tested on a novel user. Not many research articles report the user independent accuracies. Highest accuracies are reported for user dependent classifiers, therefore training of the classifier is needed for every new user which isn't desirable. The paper by Young *et al.*[49] suggests to retrain the classifier with level walking data for a novel user, but this only gives an overall classification accuracy of 86.4 %. In yet another article by Young *et al.*[41] they report a user independent accuracy of 92% when using a mode specific DBN classifier that also accounts for time history of the decisions. A mode specific classifier is a classifier that is trained based on the transitions possible from each mode. User independent accuracies have also been reported by Woodward *et al.*[25] for LDA and ANN as 78% and 89% respectively.

In general it isn't possible to reach a 100% accuracy using standard machine learning algorithms. However, in the article by Varol *et al.*[30] they report 100% accuracy in a real time implementation that uses a GMM with a majority voting scheme. This was at the expense of a longer decision time of $\sim 500ms$, but the modes of interest were Si,St and LW for which longer decision times are acceptable.

Despite some promising results in user dependent classifiers, much work still remains to be done in developing an error proof intent recognition system that requires little to no user dependent training. Such a system should also work outside the laboratory in an uncontrolled environment. A single algorithm most likely won't be sufficient for this, a hybrid classifier that utilizes two or more algorithms could give better results and is yet to be investigated.

## 1.4. Objectives

The goal of this thesis is:
"To develop a novel classifier architecture capable of real time intent recognition based on multimodal sensor fusion using mechanical sensors for use in a transfemoral prosthesis".

The locomotion modes of interest for this project include LW, SA, SD, RA, RD, St and Si. The review of the state of the art in locomotion intent recognition revealed that different algorithms are good at detecting different sets of modes. The hybrid classifier optimally combines the outputs of different classifiers obtained from different algorithms. It takes advantage of grouping modes for a classifier, specifically it is possible to achieve 100% accuracy when distinguishing the modes of Si,St and LW. This, when coupled with a 5 mode classifier would have an overlapping mode of LW, within this mode decisions can be weighed to improve the overall user independent accuracy of the system. A comprehensive analysis is needed into each part of the hybrid classifier to find the best algorithms. This also entails an analysis of the effect of window size and complex features on the classification error.

Elaborating on the research questions that will be addressed in this thesis:

1. What is the optimal window size, feature combination for the various classifiers and which algorithm gives the best classifier based on the offline performance.

2. What is the effect of using a hybrid classifier on the user independent error rates.

The project was carried out in collaboration with Reboocon Bionics. The prosthesis being developed by the company has a range of sensors available, these include a 9-axis IMU on the shank of the prosthesis, knee joint position and velocity encoders, motor currents and a strain gauge that can measure sagittal plane forces and moments.

Due to unavailability of the prosthesis, all analysis was done on healthy subject gait data using similar sensors as would be available on the prosthesis. From Table 1.1, the accuracy ranges for the algorithms are on a similar scale for healthy subject and amputee gait data. Therefore the proposed classifier architecture would give a similar performance with amputee gait.

## 1.5. Outline

The layout of the thesis is as follows:

**Chapter 2** contains a description of the gait data used. This includes experimental data collected for healthy subject gait; the data collection protocol, experimental setup and data processing will be described. In addition to this, the open source gait data set contributed by Hu *et al.* [53] which was also used for the analysis will be discussed. Feature extraction is also emphasized in this chapter.

**Chapter 3** starts off with the description of the hybrid classifier followed by a detailed description of the machine learning algorithms used for modelling each component of the classifier. This is followed by the metrics used to evaluate the performance. The chapter ends with the methods used to test the different algorithms and the structured approach in the development of the hybrid classifier.

**Chapter 4** contains a detailed account of the results obtained and an in depth discussion of the results. Results include the offline analysis of each element of the hybrid classifier

using the various performance metrics. Real time performance of the hybrid classifier in terms of the classification latency of the algorithms is also reported and discussed.

Finally, **Chapter 5** has a short summary on the work done during the thesis, a comparison to the results of existing literature, limitations of the research and also highlights areas for future work.

# 2

# Data Collection and Processing

Machine learning algorithms require a large amount of data for training and testing to properly evaluate their performance. This chapter contains an account of the data sets used for the analysis, all data sets used were for healthy subject gait. In order to do a proper analysis, it was necessary to have data from sensors that would be present on the prosthesis. The primary data set used was the open source data set made available by Hu *et al.* [53], however this data set didn't have any ground reaction force features. To augment the primary data set, gait data was collected at the wearable robotics lab in Univeristy of Twente.

## 2.1. WR-Lab Dataset

Gait data was collected at the Wearable Robotics(WR) Lab in University of Twente. Totally, 3 healthy able bodied subjects(2 male, 1 female;age; 72±11kg; height 176±8cm) were recruited for the experiments which were completed in March 2018.
Before conducting the trials, the subjects had their right leg instrumented. 9-DOF IMUs were placed on the subject's thigh and shank using holsters provided with the sensors. The IMU signals were sampled at 100Hz. The subject also wore the Xsens Force shoe which has a load cell at the heel and toe and IMUs for position and orientation tracking, these signals were also sampled at 100Hz. Instrumenting the leg took 15 minutes. Figure 2.1 shows the sensors used and the instrumented leg for one of the subjects.

### 2.1.1. Data Collection Protocol

In one experimental session, each subject had to perform 3 repetitions of 8 circuits consisting which encompassed all the modes of interest. A 10° slope was used for the ramp activities and the staircase had 6 stairs with each stair having a 20cm rise(height) and 25cm run(width). The 8 circuits were St → Si → St, St → LW → St, LW → SA → LW, LW → SD → LW, LW → RA → LW, LW → RD → LW, St → SA → St, St → SD → St. The 7 circuits involving locomotion were conducted with gait initiation by both the instrumented leg and the sound leg which gave a total of 6 repetitions for those circuits. The subjects were allowed to freely transition between the modes and they could choose their own walking speed. Trials with trips and sensor slippage while walking were excluded from the final data set.

The IMUs and the Force shoe used a different software package, therefore to aid post processing of the raw data a synchronization routine which involved a stomping action was performed. This gave peaks in the IMU signals which was used as the synchronization point. Trials in which the synchronization was missing were excluded.

### 2.1.2. Post-processing

The time series data of the IMU attached to the shank and the IMUs in the Force shoe was analyzed to find the peaks in the signal, the first peak was used as the synchronization point


(a)

(b)

(c)

Figure 2.1: Instrumentation Setup (a)Xsens IMU (b)Xsens Force Shoe (c)Instrumented Leg

and the data before this peak was excluded.
The data had to first be labelled with the gait modes, in order to do this, the gait events of TO and HC were detected using a threshold of $0.05 \times BW$, where BW is the body weight. The TO event was used to label all transitions to modes involving locomotion(LW,SA,SD,RA,RD), for the transition from LW$\rightarrow$ St the HC event was used as transition trigger.
IMU signals were low pass filtered using a 6th order butterworth filter with a cutoff frequency of 25Hz. The ground reaction forces was smoothed using a 4th order Savitzky-Golay filter and the signals were normalized to body mass to reduce the variability of the signals between subjects. In addition to these signals, knee angle and knee angular velocity are useful signals but these couldn't be recorded as sensors like the goniometer were not available. To work around this, an IMU was also attached to the thigh segment during measurement. From the data of the two IMUs the joint angle can be estimated using the joint optimization method proposed by Seel *et al.* [54] which is robust to IMU mounting orientation.
The method assumes the knee to be a perfect hinge joint. Two optimization's have to be done to obtain a mathematical description of the knee joint. In the first optimization, the joint axis coordinates have to be identified. If $g_1(t)$ and $g_2(t)$ are the 3-axis gyroscope signals from the

two IMUs, these signals measured on a hinge joint differ only by a rotation matrix and a joint angular velocity vector. This implies that the projection of these signals into the joint plane have the same lengths for every instant in time. Concretely,

$$\|g_1(t)) \times j_1\|_2 - \|g_2(t)) \times j_2\|_2 = 0, \forall t \tag{2.1}$$

where $\|\|_2$ is the Euclidean distance norm. This equation holds irrespective of the mounting orientation of the sensors. $j_1$ and $j_2$ are identified by minimizing the cost function which is the left hand side of Equation (2.1),$j_1$ and $j_2$ can be written in spherical coordinates as

$$j_1 = (cos(\phi_1)cos(\theta_1), cos(\phi_1)sin(\theta_1), \sin\phi_1) \tag{2.2}$$

$$j_2 = (cos(\phi_2)cos(\theta_2), cos(\phi_2)sin(\theta_2), \sin\phi_2) \tag{2.3}$$

The optimization can be run over the sum of squares cost function to obtain the parameters that describe the joint axis coordinates.
In the second optimization, the joint position coordinates have to be identified. This can be done by exploiting the fact that the acceleration of each sensor is the sum of the acceleration of the joint center and the acceleration induced by the rotation of the sensor around the joint. The acceleration of the joint center has to be the same in the frames of both IMUs up to some rotation matrix. Mathematically, this is written as

$$\|a_1(t) - \Gamma_{g_1(t)}(o_1)\|_2 - \|a_2(t) - \Gamma_{g_2(t)}(o_2)\|_2 = 0, \forall t \tag{2.4}$$

$$\Gamma_{g_i(t)}(o_i) := g_i(t) \times (g_i(t) \times o_i) + \dot{g}_i(t) \times o_i, i = 1, 2 \tag{2.5}$$

where $\Gamma_{g_i(t)}(o_i)$ is the radial and tangential acceleration. Subtracting this from the measured acceleration shifts it by $-o_i$ giving the acceleration at the joint center. $a_1(t)$ and $a_2(t)$ are the 3-axis accelerometer signals. An optimization can be run over the sum of squares cost function which is the left side of Equation (2.4). The constraint described in the same equation is satisfied by every pair of points $o_1$ and $o_2$ on the joint axis, therefore the results of the optimization have to be processed further to shift it as close as possible to the sensors by using the equations

$$o_1 = \hat{o}_1 - j_1\frac{\hat{o}_1 \cdot j_1 + \hat{o}_2 \cdot j_2}{2}, o_2 = \hat{o}_2 - j_2\frac{\hat{o}_1 \cdot j_1 + \hat{o}_2 \cdot j_2}{2} \tag{2.6}$$

Using these optimized values the joint angle can be determined from the accelerometer and gyroscope readings. The gyroscope based flexion angle is calculated by integrating the difference between the angular rates described in the joint axis.

$$\alpha_{gyr}(t) = \int_0^t (g_1(\tau) \cdot j_1 - g_2(\tau).j_2)d\tau \tag{2.7}$$

This estimate is smooth but generally plagued by drift. A drift free estimate of the flexion angle is therefore needed. This can be obtained from the accelerometer signals which are noisy but free of drift. These two estimates are fused with a filter of the form

$$\alpha_{acc+gyr}(t) = \lambda\alpha_{acc}(t) + (1-\lambda)(\alpha_{acc+gyr}(t-\Delta t) + \alpha_{gyr}(t) - \alpha_{gyr}(t-\Delta t)), \lambda \in [0,1] \tag{2.8}$$

Figure 2.2 shows the resulting joint angle estimates for one of the typical trials recorded along with the normalized ground reaction forces. The joint optimization's were done for every trial and the joint angles were estimated. The knee angular velocity was determined by taking the central difference numerical derivative of the knee angle.
All relevant signals including the shank IMU channels, sagittal plane ground reaction forces and moments and the knee angle and velocity were then stored as the final post processed signals. The next step was to extract relevant features from the data sets.

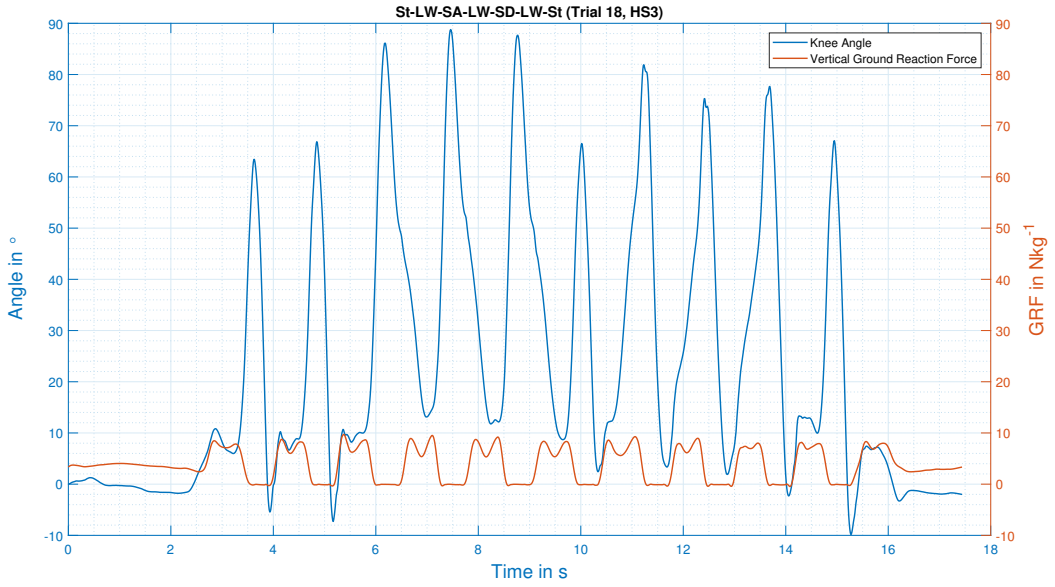



Figure 2.2: Extracted Knee Joint Angles, Normalized Ground Reaction Force for a typical trial

### 2.1.3. Feature Extraction

Features were extracted both in a continuous and discrete way. The signals were segmented into windows from which the features were extracted. For the initial analysis, continuous windows of size 50ms, 100ms,200ms and 300ms were extracted without a sliding frame. An excerpt of the continuous windows is shown in Figure 2.4. For the final analysis sliding windows of size 50ms, 100ms,200ms and 300ms were extracted every 10ms.
For the discrete windows, the signals were segmented using previously identified HC and TO gait events, each analysis window began 100ms,200ms and 300ms before each identified event. An excerpt of the discrete windows extracted for one trial is shown in Figure 2.3.
Simple features that were extracted include the mean, standard deviation, maximum, minimum, initial and final values (6 features/channel). There were a total of 3 sensors(1 IMU at the shank segment, 1 virtual-goniometer, 1 Load Cell), 12 channels(9 IMU, 2 virtual-goniometer, 3 Load Cell) resulting in 84 such features. The magnetometer readings of the IMU were not reliable and hence were dropped from the feature vector, reducing the number of features to 66.

More complex features like the signal energy and signal entropy for the accelerometer channels, as was done in [20] were also extracted. The energy of a discrete signal is calculated using Equation 2.9

$$E_s =< x(n), x(n) >= \sum_{n=0}^{n=N} \left|x(n)\right|^2 \tag{2.9}$$

where,N is the number of samples in the window frame.

Signal Entropy is a frequency domain feature, it is determined using the following steps

1. Signal spectrum $X(\omega_i)$ is calculated using the short time Fourier transform(STFT) of x(n).

2. Power spectral density(PSD) is then determined using Equation 2.10

$$P(\omega_i) = \frac{1}{N}\left|X(\omega_i)\right|^2 \tag{2.10}$$

where, N is the number of frequency bins of the fourier transform.

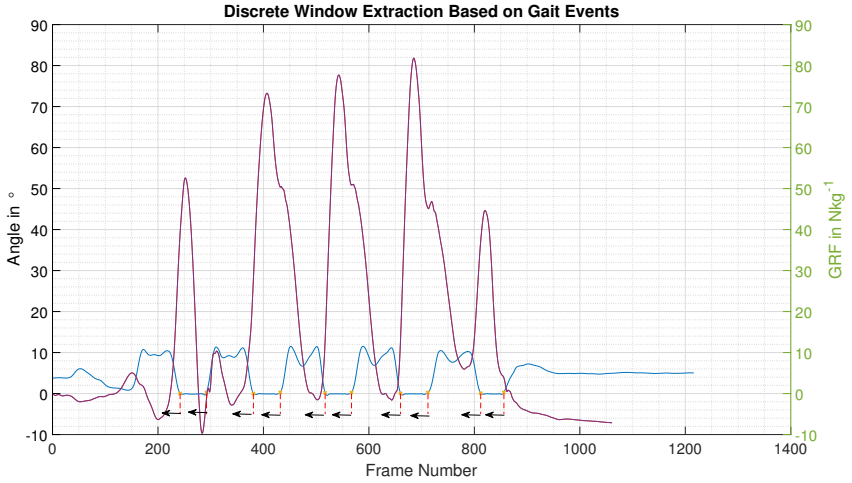



Figure 2.3: Discrete Window Extraction with a window size of 300ms

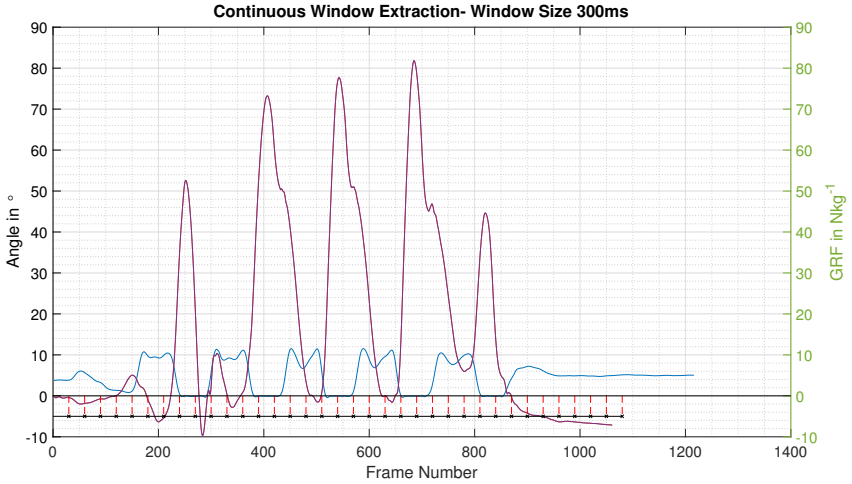



Figure 2.4: Continuous Window Extraction with a window size of 300ms

3. Normalize the PSD to get a probability distribution function.

$$p_i = \frac{P(\omega_i)}{\sum_i P(\omega_i)} \tag{2.11}$$

4. The entropy is then calculated using the expression 2.12

$$PSE = -\sum_{i=1}^{n} p_i ln(p_i) \tag{2.12}$$

The absolute orientation of the thigh segment in the sagittal plane is also a useful feature. The orientation of the thigh is a linear combination of the absolute orientation of the shank and the knee joint angle. As the knee joint angles are already recorded, adding features of the absolute orientation of the shank is sufficient. This can be determined using the accelerometer and gyroscopes of the shank IMU. Figure 2.5 shows the coordinate system of the IMU along with an approximate mounting orientation.

The orientation that is calculated by integrating the gyroscope reading about the y-axis($G_y$) suffers from drift because of large time scale integration. In order to get rid of the drift another

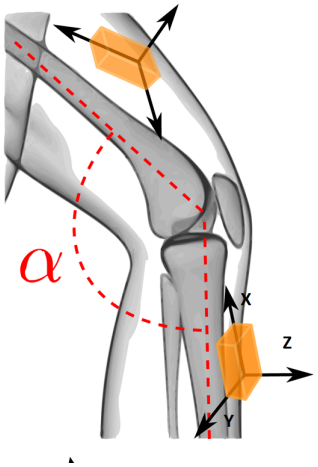



Figure 2.5: Orientation of shank IMU

signal would be needed that is free from drift, it turns out that a tilt angle estimate though noisy is free of drift.It can be determined from the accelerometer signals using Equation 2.13

$$\alpha = sin^{-1}(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}) \tag{2.13}$$

Where, $A_x, A_y$ and $A_z$ are the accelerometer readings along it's 3 axis. An underlying assumption is that the y-axis of the sensor is parallel to the knee axis. Ideally, a still standing trial is used to obtain a rotation matrix that determines the correct orientation of the sensor. The two estimates are fused using a complementary filter, the mathematical formulation is as given in Equation 2.14

$$\hat{\theta}_k = \lambda(\hat{\theta}_{k-1} + G_{y_k}\Delta t) + (1 - \lambda)\alpha_k \tag{2.14}$$

$\lambda$ is generally chosen to be a value close to 1, this ensures that the fused signal follows the smooth orientation estimate obtained from the gyroscope and the mean of the tilt angle. On investigating the equation in the laplace domain, it turns out that the filter is equivalent to low pass filtering the tilt estimate and high pass filtering the orientation estimate from the gyroscope signal.

Interesting features can also be obtained by integrating the accelerometer signals between TO and HC events to obtain an approximation of the stride length and stride height which is vital information that helps in distinguishing SA/SD from LW. The mathematical expressions to calculate the Stride Length (SL) and Stride Height (SH) are as given in Equation 2.15 and 2.16 respectively.

$$SL = \int\int (a_z \sin\theta_k)dt = \sum_{n=1}^{N}(\sum_{n=1}^{N} a_z(n).dt).dt \tag{2.15}$$

$$SH = \int\int (a_x \cos\theta_k - g)dt = \sum_{n=1}^{N}(\sum_{n=1}^{N} a_x(n).dt).dt \tag{2.16}$$

It is important to remove the effect of gravity on the accelerometer readings, the absolute orientation of the shank segment is used for this purpose. By projecting the gravity vector onto the sagittal plane axes of the IMU, a simple vector subtraction will get rid of the gravity components.

### 2.1.4. Data Labelling

In order to use the supervised learning algorithms, the data has to be labelled with the locomotive mode. The gait events were used for the labelling, the standard used by Hu *et*

*al.* [53] was adopted, this ensured that the testing process of the algorithms is streamlined. Every gait event detected was assigned a 4-digit trigger of the form:

Previous Mode_Previous Phase_Current Mode_Current Phase

The standard used for the modes was

0=Sitting; 1=Level Ground Walking; 2=Ramp Ascent; 3=Ramp Descent; 4=Stair Ascent; 5=Stair Descent; 6=Standing

A separate standard was used for the gait phases which was

1=Stance; 2=Swing; 3=Mid Swing

Stance: Gait phase when the foot remains in contact with the ground.
Swing: Gait phase when the foot is not in contact with the ground.
Mid Swing: Point at which the swinging leg passes the stance phase leg.

## 2.2. ENABL3S Dataset

ENABL3S(Encyclopedia of Able-bodied Bilateral Lower Limb Locomotor Signals ) is a publicly available benchmark data set of bilateral neuromechanical collected by Hu *et al.* [53]. The data set has bilateral EMG and joint, limb kinematics recorded via wearable sensors for 10-able bodied subjects transitioning between the same locomotive modes that are of interest for this study. Also only the data of the kinematic signals was used for this thesis.
The signals of interest that were recorded include, sagittal plane joint kinematic signals obtained using electrogoniometers placed on the knee and sampled at 500Hz, 6 DOF IMU placed on the subject's shank and also sampled at 500 Hz. The setup is shown in Figure 2.6

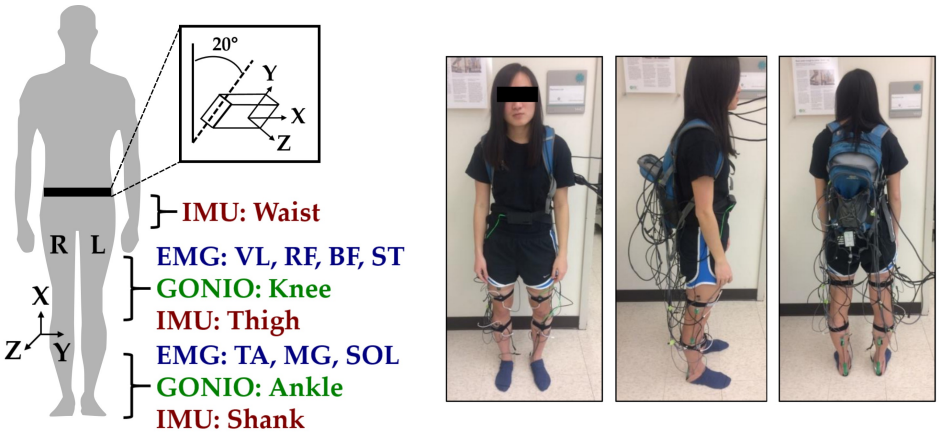



Figure 2.6: Sensor placement

### 2.2.1. Data Collection Protocol

10 healthy able bodied subjects(7 male, 3 female; 25.5 ± 2 years; 174 ± 12 cm; 70 ± 14 kg) without any gait impairments were recruited for the study. The data was collected between January and February 2017.
25 trials each of two circuits: Si → St → LW → SA → LW → RD → LW → St → Si and Si → St → LW → RA → LW → SD → LW → St → Si were performed by each subject. A 4-step staircase with dimensions: 19.685cm rise, 25.4cm run was used and the ramp had a 10° slope. The circuit for the trials is shown in Figure 2.7. Trials in which sensors had to be re-positioned were excluded from the analysis. Subjects were asked to walk at a self selected speed and perform the transitions freely. An experimenter labelled the locomotor intent using a key fob. The standard used for labelling has already been discussed in section 2.1.4.

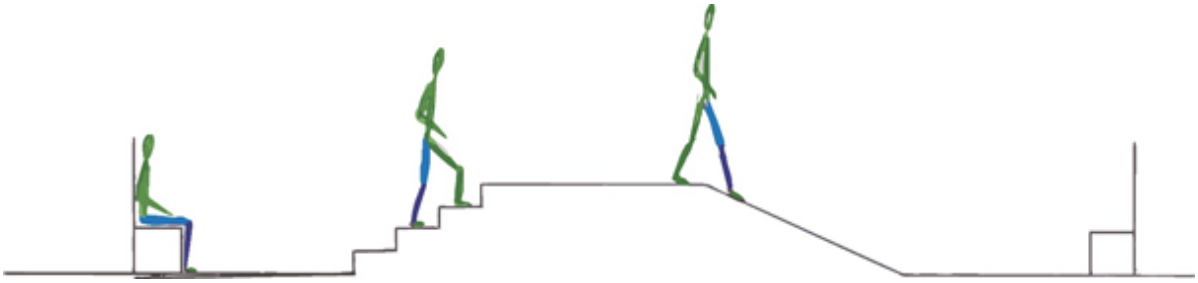


Figure 2.7: Circuit used for trials

### 2.2.2. Post Processing

Event switches could not be used to identify HC and TO events in this data set, so these events were detected using the sagittal plane angular velocity($G_Y$). The signal was first mean-subtracted and low pass filtered using a first order butterworth filter with a cut off frequency of 6Hz, peaks were then identified using a threshold based technique. The largest peak gives the mid swing event, TO events were identified by searching for peaks before each mid swing event and HC events were identified by searching for peaks after mid-swing event. Gait events corrupted by motion artifacts like pauses and trips were excluded.
Goniometer and IMU signals were low-pass filtered using a sixth order butterworth filter with cutoff frequencies of 10Hz and 25Hz respectively. Joint velocities were determined indirectly by computing the central difference based numerical derivative of the joint position obtained from the goniometer channels.

### 2.2.3. Feature Extraction

The data came with features already extracted for a window size of 300ms before each identified HC and TO gait event. The data set included four additional 300ms analysis windows were used (delayed by 30,60,90 and 120ms relative to each event). For each window, features extracted included the mean, standard deviation, maximum, minimum, initial and final values. As the effect of the window size was of interest for this study, additional 50ms, 100ms and 200ms analysis windows were extracted. The feature vectors had to be augmented as well to include more complicated features like the signal energy, signal entropy , stride height and stride length, the mathematical formulations for which have already been discussed.

### 2.2.4. Data Labelling

The event triggers and associated labels are only present for the dynamic modes and were used as the ground truth data for the dynamic classifier analysis. As this data set was collected for both legs, the labelling was done when either leg entered a certain mode. This resulted in slightly different labels when compared to the WR-Lab dataset which was labelled with respect to the instrumented leg. Particularly, for the ENABL3S dataset, the instrumented leg of interest has earlier mode transitions when the other leg is the leading leg for the transition. These labels were retained to see if the classifier can recognize such an early transition.

For the static modes only key fob labels were present in the data set. Since this was done by the experimenter there was quite a lot of error and many data points were wrongly labelled. To partly fix this issue a thresholding scheme was adopted to relabel the data for Si and St modes, a threshold of 20° for the knee joint angle was used based on an evaluation of all subject's data. For relabelling the St and the LW mode, the trials that had the gait initiation TO triggers and gait termination HC triggers were used. However, for nearly 80% of the trials, these triggers weren't present and the key fob label had to be used. This led to the data for the two classes having a considerable overlap.

# 3

# Methods

This chapter deals with all the methods used in the thesis, it starts off with a description of the hybrid classifier, it then runs through some critical design choices for the components of the classifier. This is followed by a description of the algorithmic choices and ways to evaluate the performance. The chapter ends with the methods used to robustly evaluate the algorithms using the performance metrics.

## 3.1. Design of Hybrid Classifier

The aim of the intent recognition problem in this thesis is to be able to distinguish 7 locomotion modes: Si, St, LW, SA, SD , RA and RD. Using a single classifier for all modes is prone to giving higher error rates due to overlapping class distributions which causes a deterioration in the performance. A better approach would be to group certain modes together and use a separate classifier for each group. A natural choice is to separate the static modes of Si, St from the remaining dynamic locomotion modes. Adding LW to the static modes would create a common mode for the two classifiers where misclassifications can be avoided by combining the decisions. Since in daily activities, most time is spent in LW for the dynamic modes. It would be better to have two decisions for this mode to prevent incorrect mode switching.

The hybrid classifier architecture is shown in Figure 3.1. It has two classifiers running in parallel, a static mode classifier that makes decisions between the Si, St and LW, a classifier that outputs probabilities is used for this. The second classifier is used to classify the dynamic locomotion modes LW,SA,SD,RA,RD. When either classifier enters LW mode, the other one is switched on. The decisions are then weighed against each other, therefore a probabilistic decision is preferred from the dynamic mode classifier as well. If either classifier exits the LW mode, the other classifier is switched off.

The static mode classifier operates with features extracted from continuous windows. This classifier makes a lot of decisions because of the use of continuous features, to speed up computation the feature vectors have to be reduced to a lower dimension. Feature reduction techniques are described in section 3.4. An evaluation of the feature reduction techniques is needed to find the method that best retains the variance in the data. The algorithm choices for this classifier include GMM and NB as both methods can output probabilities associated with the decisions. Both algorithms are based on Bayes decision theory and are described in section 3.3.

One might argue that a simple threshold based scheme for the knee angle can be adopted for the static mode classification, however such a threshold would be exceeded several times during standing and walking and will therefore result in misclassifications. It is best to go with a probabilistic classifier as it gives an intricate model based on combining measurements from different sensors.

The dynamic mode classifier has two components, a heel contact classifier and a toe off classifier that make decisions at the respective gait events. Separate models that operate with features extracted from discrete windows preceding each event are trained for this purpose.

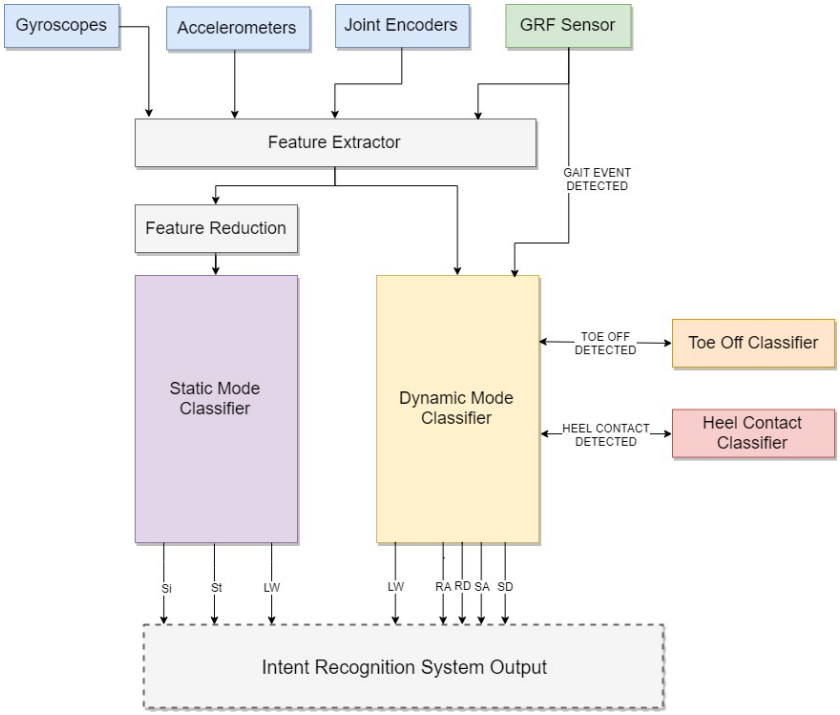



Figure 3.1: Hybrid Classifier Architecture

Feature reduction is not needed for this classifier as decisions are made less frequently. The choice for the second classifier is based on the offline evaluation of the user dependent and user independent performance of the classifiers. Different classifiers were trained using the algorithms SVM,LDA,Relevance Vector Machine (RVM),KNN and NB. Each algorithm belongs to a different category which is described in section 3.3. In addition to the classifier choice, factors like the window size and features have an impact on the performance. Therefore there is the need for a comprehensive analysis for the dynamic classifier, the process for this is explained in section 3.6. The performance criteria for the analysis are highlighted in section 3.5. In a real time implementation, sliding continuous windows are extracted at discrete intervals. When either classifier enters LW mode, the other classifier is switched on. For the dynamic mode classifier, a detection of the gait event is also needed from the Ground Reaction Force (GRF) sensor, the feature vectors are then sent to the respective classifier when an event is detected to make a decision.

## 3.2. Multiclass Classification Strategies

The intent recognition problem is a multiple mode problem as was highlighted in the previous section. A majority of the machine learning algorithms are formulated for binary classification. The method used to transform the problem to a binary classification is therefore a design consideration for the hybrid classifier. Two strategies exist in machine learning literature[55] for the training and testing of such classifiers which are described in the following sections.

### 3.2.1. One Versus Rest(OvR)

The OvR strategy involves training a single classifier per class called a base classifier, all the training samples belonging to that class are assigned as positive samples while the remaining samples are assigned as negatives. As discrete labels are predicted by the base classifiers for a test point, it would result in ambiguous decisions with multiple classes being assigned. This strategy would be more effective if the base classifiers output a confidence score rather than just a class label, a non ambiguous decision can then be made based on the base classifier with maximum confidence. The confidence score is a real valued function e.g. probabilities.

The training algorithm for a K class OvR constructed from a binary learner L is as follows

**Inputs:**

- L, a training algorithm for binary classifiers,
- Training feature vectors X
- Labels y where $y_i \in \{1, ....., K\}$

**Output:**

- A list of classifier models $f_k$ for $k \in \{1, ....., K\}$

**Procedure:**

- For each k in 1,.....,K
  - Relabel as vector z where $z_i = 1$ if $y_i = K$ and $z_i = 0$ otherwise
  - Apply L to X, to obtain $f_k$

For a test sample x, decisions are made based on the outputs of the K classifiers, concretely

$$\hat{y} = \arg \max_{k \in \{1,.....,K\}} f_k(x) \tag{3.1}$$

### 3.2.2. One Versus One(OvO)

The OvO strategy involves training a set of base classifiers that separate two classes at a time. Therefore, totally $\frac{K(K-1)}{2}$ base classifiers are required. Each base classifier receives the subset of samples of the pair of classes from the original training set and learns to distinguish between those classes. For a test point, the discrete labels are determined for all the classifiers and a majority voting scheme is applied that chooses the class with the highest number of positive predictions. Even this method can give ambiguous decisions but the region of ambiguity in the feature space is much smaller.
Figure 3.2 which shows the decision boundaries and the ambiguous region for a 3-class problem. Note that the OvR example is slightly different and uses only 2 base classifiers , it makes an assignment to the third class if a negative decision is received from both base classifiers. Normally, one would use K base classifiers in the approach and this would create more ambiguous regions. It is for this reason that the OvO approach is preferred.

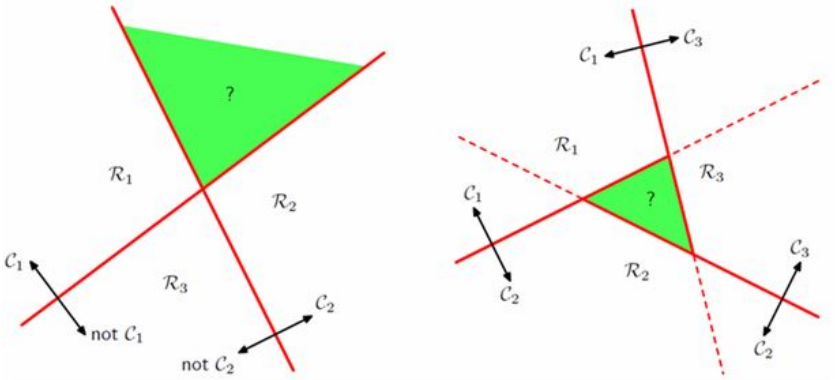



Figure 3.2: Decision Ambiguity for OvR(left) and OvO(right)

## 3.3. Machine Learning Algorithms

Algorithms are needed to find models for the classifiers that make up the hybrid classifier. The following sections list the different categories of algorithms and the specific ones used for the analysis from each category.

### 3.3.1. Linear Classifiers

Linear Classifiers use a set of discriminant functions that find linear decision boundaries to separate the classes in the data. Multiclass LDA is one such typical linear classification algorithm that was used for the analysis.[55] It involves finding a linear transformation $\phi$ that maximizes a cost function of the form

$$J(\Phi) = \frac{\left|\Phi^T \hat{\Sigma}_b \Phi\right|}{\left|\Phi^T \hat{\Sigma}_w \Phi\right|} \tag{3.2}$$

$\hat{\Sigma}_w$ is the intra-class covariance matrix and is calculated as

$$\hat{\Sigma}_w = \sum_{i=1}^{k} \sum_{x \in c_i} (x - \bar{x}_i)(x - \bar{x}_i)' \tag{3.3}$$

where, $\bar{x}_i$ is the mean feature vector for each class, $x$ is the feature vector for the various training examples. $\hat{\Sigma}_b$ is the inter-class covariance matrix and is determined using the equation

$$\hat{\Sigma}_b = \sum_{i=1}^{k} m_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})' \tag{3.4}$$

where, $m_i$ is the number of training examples for each class, $\bar{x}$ is the total mean vector given by $\bar{x} = \frac{1}{m} \sum_{i=1}^{k} m_i \bar{x}_i$. It turns out that the transformation $\Phi$ can be obtained by solving a generalized eigenvalue problem of the form

$$\hat{\Sigma}_b \Phi = \lambda \hat{\Sigma}_w \Phi \tag{3.5}$$

Once the transformation function $\Phi$ is determined, decisions are made based on a distance metric like the Euclidean distance $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$. A new instance $z$ is classified based on the criterion

$$\arg_k \min d(z\Phi, \bar{x}_k \Phi) \tag{3.6}$$

where $\bar{x}_k$ is the centroid of the k-th class.
Figure 3.3 shows the multiclass LDA for a problem with 3 classes and the obtained decision boundaries. The simplicity of these models makes it a good option for the dynamic mode classifier. LDA performs well when the classes have non overlapping distributions. The use of mechanical signals and discrete windows before specific gait events ensures repeatability of feature vectors for the classes, it also gives class distributions with less overlap. For these reasons LDA was used in the analysis.

### 3.3.2. Classifiers based on Bayes Decision Theory

These classifiers use probabilistic arguments obtained using the statistical nature of the extracted features. They require modelling a probability distribution function (pdf) based on the training feature vectors and labels. Using the models, posterior probabilities that represent the probability that an unclassified feature vector belongs to a class $\omega_i$ are determined. The assignment is then made to the class with the highest posterior probability. Two methods based on this principle have been used for the analysis, these are Gaussian Mixture Models(GMM) and NB classifier.

1. **Gaussian Mixture Models**
   GMMs model the feature vectors with a mixture of gaussians of the form

$$p(x \mid \omega_i) = \sum_{m=1}^{M} \lambda_m^i p_m^i(x), i = 1, 2...., K \tag{3.7}$$

   where $x$ is the feature vector, K is the number of classes, M is the number of component gaussians, $\lambda_m^i$ is the mixture parameter of the $i^{th}$ component subject to the constraints

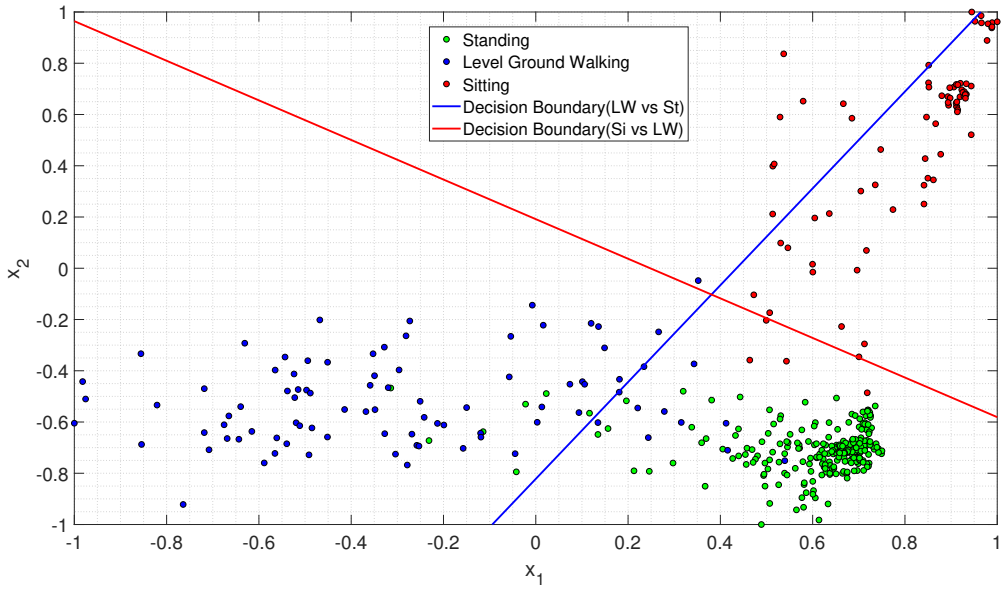



Figure 3.3: Multiclass LDA for static modes showing decision boundaries

$\sum_{m=1}^{M} \lambda_m^i = 1$ and $\lambda_m^i \geq 0$, $p_m^i(x)$ is a gaussian distribution function of the form

$$p_m^i(x) = \frac{1}{\sqrt{(2\pi)^D \left|\Sigma_m^i\right|}} \exp\{-\frac{1}{2}(x - \mu_m^i)'(\Sigma_m^i)^{-1}(x - \mu_m^i)\} \tag{3.8}$$

Given the dimension D of the feature vector, the gaussian is characterized by a Dx1 mean vector $\mu_k^i$ and a DxD full covariance matrix, $\Sigma_k^i$ . The parameters are determined in an iterative fashion using the Expectation Maximization (EM) algorithm[56].Each GMM requires the estimation of $K(1 + D + D(D + 1)/2) - 1$ parameters [55].
A new test point with a feature vector $x_s$ is assigned to the class with the maximum probability

$$\omega_s = \arg\max(p(x_s \mid \omega_i)) \tag{3.9}$$

An issue with the GMM is that the number of required training data points scales exponentially with the dimension of the feature space. This is referred to as the curse of dimensionality. Crudely speaking, if N points are sufficient to obtain an accurate estimate of the probability distribution function in one- dimensional space, $N^l$ points are needed for an l-dimensional space. This really limits the GMM as a classifier and some amount of feature reduction/selection is needed to use GMMs effectively, this is further described in section 3.4. The probabilistic nature of the GMM makes it an option for the static mode classifier. It was not considered for the dynamic mode classifier as the dimension of the feature vector space is high, this leads to convergence issues for the EM algorithm.

2. **Naive Bayes Classifier**
NB classifier is a simplified version of GMM that attempts to avoid the curse of dimensionality by assuming that the individual features $x_j, j = 1, 2, ...., l$ are statistically independent. Under the assumption the class conditioned probability can be written as the product of probability distribution functions for each individual feature[55].

$$p(x \mid \omega_i) = \prod_{j=1}^{l} p(x_j \mid \omega_i), i = 1, 2, ..., K \tag{3.10}$$

Therefore, for each class we need to determine "l" one-dimesional pdfs. $l \times N$ training data points would be sufficient for this which is significantly lower. The classifier assigns an

unknown sample $x_s = [x_1, x_2, ...., x_l]^T$ to the class

$$\omega_s = \arg\max_{\omega_i} \prod_{j=1}^{l} p(x_j \mid \omega_i), i = 1, 2, ...., K \quad (3.11)$$

Since NB gives lighter models with less number of parameters, it is a feasible option for both the static and dynamic mode classifiers and was used in both analyses.

### 3.3.3. Instance-based learning

These methods compare test feature vectors with instances seen in the training set. This requires that the training data be stored in the memory which may not be feasible for large scale problems. The KNN algorithm is one such algorithm that uses this principle.
The training phase involves storing the data as feature vectors and class labels of the training samples. In the classification phase, the user has to define a parameter $'k'$, which is the number of nearest neighbors. Given an unlabelled feature vector, the algorithm assigns a label based on the most frequent label of the k training samples nearest to the test point. Various distance criterion can be used to determine the nearest neighbors, for continuous variables the euclidean distance is widely used. The performance of the algorithm is heavily dependent on the user chosen parameter k. As a thumb rule, k is chosen to be 10% of the number of points in the training set. It is also possible to optimize the parameter based on cross validation if a large enough training set is available [57].
Figure 3.4 shows an example of a 2 dimensional KNN classification for a 2 class problem. It illustrates the variability introduced by the choice of the parameter $'k'$. The test sample is the green circle which should be classified to either the first class marked by blue squares or to the second class marked with red triangles. If k=3, the point is assigned to the second class as there are 2 triangles and 1 square inside the inner circle. If k=5, the point is assigned to the first class as there are 3 squares and 2 triangles in the region covered by the outer circle. Therefore this algorithm is extremely sensitive to the choice of the parameters[55].
The advantage of using this algorithm is that is robust to noisy training data where feature vectors are not properly clustered for the classes. However, since the method involves computing distances to the training feature vectors, the computation time scales with the number of training points. This is the reason it is only a feasible option for the dynamic mode classifier analysis.

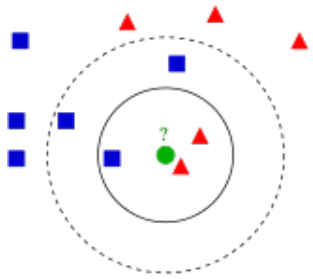



Figure 3.4: k-NN classification example

### 3.3.4. Non Linear Classifiers

Classifiers that find a non linear decision boundary to separate the classes constitute non linear classifiers. The classifiers that are capable of finding such non linear boundaries are

1. **Support Vector Machine**
   SVM is a sparse machine learning method developed by Vapnik[58] that outputs compact models. In a multi-class setup, an SVM constructs a set of boundary hyper-planes such that the margin is maximized. The margin is the distance from the hyper-plane to the nearest training data point of any of the classes. Two types of margins are possible,

a hard margin that considers points in the vicinity of the boundary and a soft margin that considers training points far away from the decision boundary as well. The SVM as such finds a linear decision boundary to separate the classes. However, by applying the kernel trick on the data the feature space can be transformed such that a hyper-plane in this transformed space is equivalent to a nonlinear decision boundary in the original input space. Figure 3.5 shows the decision boundary in the original feature space and the transformed feature space. The kernel trick involves computing a non linear func-

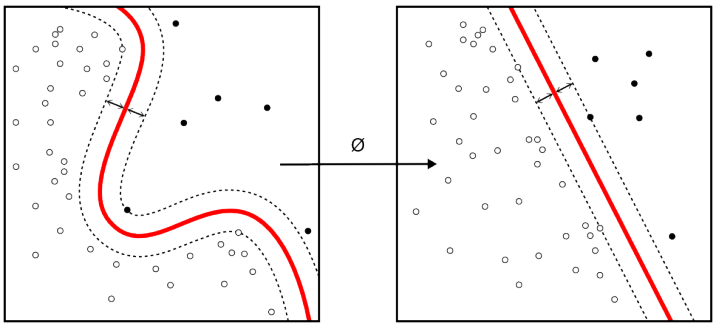



Figure 3.5: SVM Decision Boundaries: original input space(left), transformed feature space(right)
$\phi$ is the kernel function

tion of the distance of a given feature vector to all the remaining feature vectors in the training data set. Different options are available for the kernel functions, the most powerful one is the radial basis function and was chosen for the analysis. It is formulated like a gaussian distribution as

$$K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \text{ for } \gamma > 0 \tag{3.12}$$

where $x_i, x_j$ are two training feature vectors and $\gamma$ is a parameter that defines how sharply peaked the gaussian is at the training data point.
The values of the kernel functions for the N training points are usually assembled in a gram matrix $G(x, x^T)$ the columns of which represent the basis functions $\phi_1, ...., \phi_N$ of each training point.

$$G_{i,j} = K(x_i, x_j) \tag{3.13}$$
$$G = [\phi_1, \phi_2, ....., \phi_N] \tag{3.14}$$

Classification by a non linear SVM is done according to equation 3.15 where, $\omega_1, \omega_2$ are the classes.

$$\text{assign x in } \omega_1(\omega_2) \text{ if } y(x \mid w) = \sum_{i=1}^{N} w_i K(x_i, x) + w_0 > (< 0) \tag{3.15}$$

The training phase involves selecting the weights $w_i$ as well as the training vectors $x_i$ that maximizes the margin. The removal(pruning) of irrelevant training vectors is done by removing the corresponding columns of the gram matrix. The final selected vectors are called the support vectors and these define the decision boundary. The SVM is essentially a binary classifier and one has to adopt either of the approaches mentioned in section 3.2 for the multiclass scenario. Generally the OvO is preferred over the OvR approach for modelling the multiclass SVM. In the OvR approach data-sets can be heavily skewed towards one class because of assigning all negative classes as a single negative class. Additionally, each of the resulting base classifier's was trained in the task of finding if a test point is within a certain class or not, such tasks are independent and do not represent the actual task of finding which class the training point belongs to. The OvO approach uses a voting scheme to classify test points which is more logical, the disadvantages are that more base classifiers need to be trained which leads to more

training time and increased computation time for test points as more support vectors are used.
Some drawbacks of SVM are:

- It only outputs a class label, the probabilities of the underlying classes cannot be extracted. This is helpful when using a hybrid classifier architecture as they help capture the uncertainty in the prediction.
- Although the method is sparse, SVMs make a liberal use of the kernel functions and the number of support vectors generally grows linearly with the training set size. Some post-processing is required to reduce the computational complexity[59] .

Since SVM also relies on data that is repeatable to find proper decision boundaries, it is best suited for the dynamic mode classifier. The algorithm also handles high dimensional feature spaces well through the use of kernel functions, which is advantageous.

2. **Relevance Vector Machine**
RVM is a method developed by Tipping[60]. It tackles the issues of the SVM by adopting a Bayesian model of equation 3.15. Such an approach gives sparser models thus resulting in quicker computations for test points. RVM can also be formulated for $M > 2$ classes and there is no need to adopt the multiclass strategies used for SVM which have many disadvantages. The model used for a M class RVM is again a linear combination of weights and the kernel transformed feature vectors

$$a_m = w_m^T K(x_i, x) \tag{3.16}$$

These outputs are then combined using a softmax function to find the class posterior probabilities

$$y_k(x) = \frac{\exp a_m}{\sum_j \exp(a_j)} \tag{3.17}$$

The Bayesian modelling is introduced through the weight parameters, which are now modelled with prior probability distributions with each weight having a separate precision hyperparameter $\alpha$. The parameters are determined using an iterative algorithm on the training data set $T$ by minimizing the negative log likelihood function

$$-\ln p(T \mid \mathbf{w_1}, ...., \mathbf{w_M}) = -\prod_{n=1}^{N} \prod_{m=1}^{M} y_{nm}^{t_{nm}} \tag{3.18}$$

where the target values $t_n$ for each training point n is labelled using a 1-of-M coding scheme. For instance ,in a 5 class problem this would be $[0, 1, 0, 0, 0]$ when the 2nd class is positive. $t_{nm}$ just represents the $k^{th}$ element of the label. In the iterative training phase, one of the steps involves inversion of the $N \times N$ gram matrix which is computationally expensive. The RVM therefore takes a longer time to train than SVM.

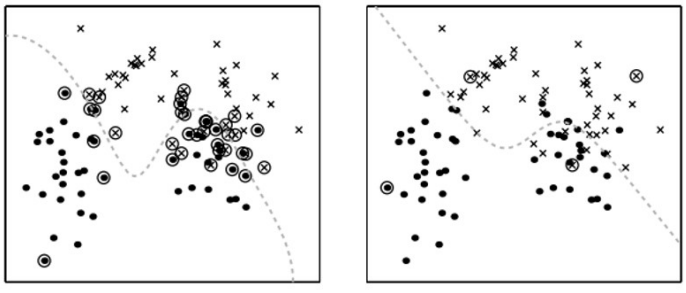


Figure 3.6: SVM(left) and RVM(right) classifiers performance on Ripley's Gaussian mixture data set[60].

Once the model is trained, equation 3.17 can be used to classify a new point as the class having the highest probability. The RVM gives much sparser models than the SVM as it has a very effective pruning procedure and results in computational savings for test points which is a more critical performance criterion for a machine learning algorithm. Figure 3.6 shows the selected training vectors using SVM and RVM for a standard gaussian data set, the classification errors are on a similar scale of 10%. It also showcases the reduction in the complexity of the model determined by RVM.

Just like the SVM this algorithm is a good consideration for the dynamic mode classifier. It could potentially perform better in the hybrid classifier because of it's probabilistic nature.

## 3.4. Feature Reduction

The static mode classifier requires the reduction of the number of features to avoid an excessive number of parameters in the model. This helps in training the algorithm more efficiently. Two methods have been used in the literature for feature reduction

### 3.4.1. Principal Component Analysis(PCA)

PCA aims to find a projection direction into a lower dimensional space while retaining most of the variability of the data in the original space. Mathematically, it is defined as finding an orthogonal linear transformation that transforms the data into a new coordinate system such that the greatest variance by some projection of the data lies on the first coordinate, the second greatest variance on the second coordinate, and so on.
This technique doesn't use the labels of the training data, it just takes in the feature vectors and outputs a lower dimension feature vector based on the covariance matrix of the data. For PCA to work properly, some preprocessing has to be performed either in the form of feature scaling if the features are on different scales or mean normalization which makes the features have a zero mean [55].
The procedure for PCA, once the data is preprocessed is

- Compute Covariance Matrix

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T \tag{3.19}$$

- Perform singular value decomposition of the covariance matrix, the columns of the left singular vectors matrix are the projection vectors and the singular values matrix code how much variance is explained by the corresponding projection vectors
  $U = [u^{(1)}u^{(2)}.....u^{(n)}] \in \mathbb{R}^{n\times n}$
  $$S = \begin{bmatrix} S_{11} & 0 & \cdots & 0 \\ \vdots & S_{22} & \cdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ 0 & \cdots & \cdots & S_{nn} \end{bmatrix}$$

- Choose number of components k required
  $U_{reduced} = [u^{(1)}u^{(2)}.....u^{(k)}] \in \mathbb{R}^{n\times k}$ for $k < n$

- Compute new feature vectors and find variance explained by the transformed features

$$x^{(i)}_{reduced} = U_{reduced} \times x^{(i)} \tag{3.20}$$

  The variance accounted for(VAF) is computed from the singular values as

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \tag{3.21}$$

A VAF higher than 90% is acceptable for machine learning problems as you don't lose out on important information that helps to separate the classes.

### 3.4.2. Fisher's discriminant analysis(FDA)

This method is based on the LDA method, it takes into account the class labels unlike the PCA and usually yields better results for supervised learning applications because the labels are known for the training set. Additionally, no mean normalization is required when using FDA[55].
The procedure for FDA is as follows

- Compute Between Class($S_B$) and Within Class($S_W$) covariance matrices using equations 3.4 and 3.3 respectively.
- Perform SVD for the matrix $S_W^{-1}S_B$
- Choose k columns of the left singular vectors matrix $U_{reduced}$
- Compute new feature vectors and find variance explained by the transformed features

$$x_{reduced}^{(i)} = U_{reduced} \times x^{(i)} \tag{3.22}$$

The variance accounted for(VAF) is computed from the singular values as

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \tag{3.23}$$

## 3.5. Performance Metrics

Comparing the different algorithms requires the selection of performance metrics, the following sections describe metrics that were used in the analysis.

### 3.5.1. Confusion Matrices

In the context of binary classification, confusion matrices are an overview of all the decisions made by the classifier and is usually represented by a two dimensional table with predicted outcomes along one dimension and actual outcomes along the other. The confusion matrix is shown in Figure 3.7 and is composed of four types of outcomes: True Positives(TP), True Negatives(TN), False Positives(FP) and False Negatives(FN).

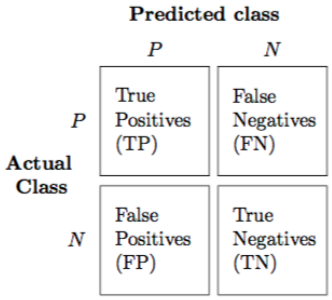



Figure 3.7: Confusion Matrix for binary classification

For a M class problem, the confusion matrix is a $M \times M$ matrix that has the predicted class labels along one dimension and the true class labels along the other. This metric helps evaluate where the most misclassifications are happening and provides insight into the classes that are hard to predict with the given set of features.

### 3.5.2. Area under Receiver Operating Characteristic Curve(AUC)

This metric is again defined for a binary classification problem for classifiers that output probabilities. It is derived from the receiver operating characteristic curve(ROC) which is a graph of the diagnostic ability of the classifier as the probability threshold used to make a decision is varied. Two metrics from the confusion matrix, the true positive rate(TPR) defined

as $\frac{TP}{TP+FN}$ and the false positive rate(FPR) defined as $\frac{FP}{FP+TN}$ are needed to plot the curve, it is determined for probability threshold intervals in the range [0,1]. Figure 3.8 shows an example of the ROC, the area under the curve is the blue area and represents the probability that the classifier will assign a higher score to a randomly chosen positive integer than to a randomly chosen negative integer. The AUC is therefore a good metric to evaluate the discrimination capability of the classifier. The dashed line represents the ROC of a classifier that just randomly predicts 1 or 0 for test points, any classifier whose ROC is below that line is a shouldn't be chosen. Such an ROC could be the an indication of incorrect labelling of data.

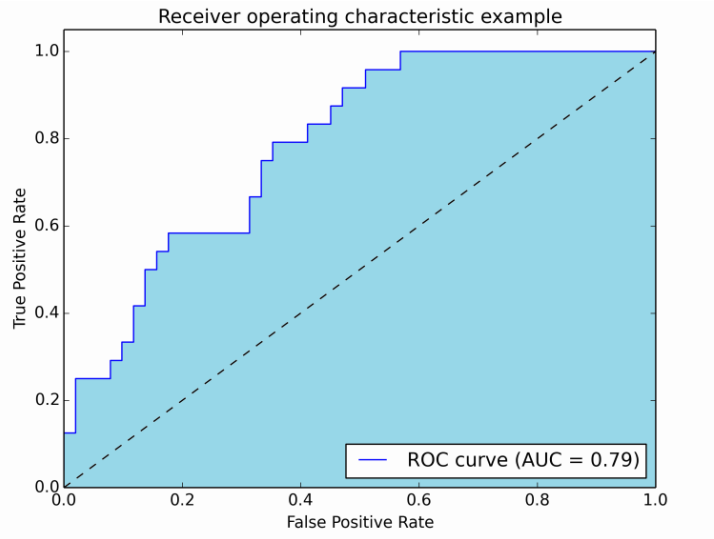



Figure 3.8: Receiver Operating Characteristic Curve

For a multiclass problem with M classes, Hand *et al.*[61] use an approach similar to an OvO classification approach. The AUC is calculated for two classes$(\omega_i, \omega_j)$ at a time and is then averaged over the total number of pairwise classifiers.

$$AUC_{total} = \frac{2}{M(M-1)} \sum_{\{\omega_i,\omega_j\} \in M} AUC_{(\omega_i,\omega_j)} \tag{3.24}$$

This measure is insensitive to skewed class distributions. This metric was used to evaluate the algorithms used for the static mode classifier.

### 3.5.3. Classification Error

The classification error is defined as

$$\delta = \frac{\text{Number of incorrect predictions}}{\text{Total number of predictions}} \tag{3.25}$$

For the locomotion mode recognition problem the error can be broken down as a steady state error($\delta_{ss}$) where the mode doesn't change and a transitional error($\delta_{tr}$) where the mode is different before and after the gait event of interest. Transition decisions are much more critical than steady state decisions, therefore a weighted error is used to evaluate the performance while calibrating the parameters of the intent recognition model learned by the classifier.

$$\delta_w = (1 - P_{ss}) \times \delta_{ss} + P_{ss} \times \delta_{tr} \tag{3.26}$$

$$P_{ss} = \frac{\text{Number of steady state steps}}{\text{Total number of steps}} \tag{3.27}$$

### 3.5.4. Decision Time

Decision time is an important metric used to evaluate the performance for the real time implementation of the algorithm, it gives an estimate of the classification delay which is the extra time needed after the gait event to make the decision. For the static mode classifier, it is the time required after a certain transition for the switching of modes, the computation time is negligible as the models used are simple.
For the dynamic mode classifier, the decision time is dependent on the delay induced by the sliding interval for the windows and the computation time

$$\text{Decision Time} = \text{Interval delay} + \text{Computation Time} \tag{3.28}$$

The computation time was determined in MATLAB using the tic toc function. The interval delay was determined based on the time-stamp after the gait event when the classifier was called.

## 3.6. Methods of Testing

A systematic approach was used to evaluate and choose the best algorithms for the hybrid classifier. A robust evaluation of the algorithms requires that the data sets be split into training, cross validation and test sets. This is done to avoid the problem of over-fitting wherein the model is fit very well to the training data, resulting in low errors when tested on the training set itself.

A common practice is to only perform cross validation where the data set is split into folds, the performance is then evaluated iteratively. In each iteration, one of the folds acts as the test set and is excluded from training. Figure 3.9 shows the cross validation method.

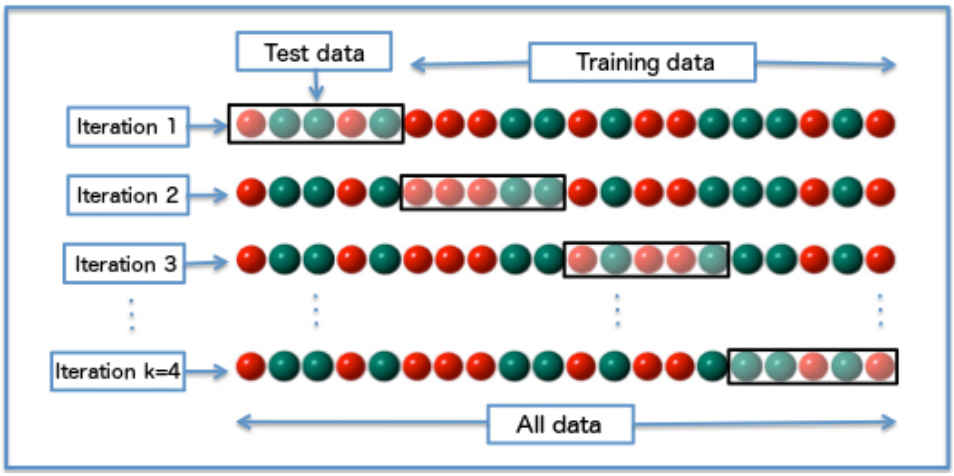



Figure 3.9: Cross Validation Procedure

Just doing the cross validation isn't sufficient to evaluate the performance. This is because each algorithm has a few governing hyper-parameters that need to be set by the user, cross validation is therefore used to calibrate the model by choosing the model with the least weighted error. An additional test data set is then needed to find the classification errors using the calibrated model. This test data set hasn't been seen by the algorithm at all during training and cross validation and is therefore a good measure of generalization.

### 3.6.1. Static Mode Classifier Testing

The first step in the analysis of the static mode classifier is to evaluate the feature reduction techniques, the Variance Accounted For (VAF) is used to determine the best technique.Following this for the GMM, parameters such as the number of gaussians used to fit the resulting feature space and the window size used to extract the features need to be eval-

uated. The AUC metric was used to evaluate the performance. For the NB classifier, the best window size has to be determined. The procedure adopted was as follows

1. Divide data into 10 folds.
2. Perform feature reduction on training data using PCA or LDA and save the transformation weights.
3. Fit Gaussian mixtures for $n_m = 2, ....., 6$ on the training data.
4. Reduce the features of test data using transformation weights calculated for training data.
5. Determine AUC for the test data and average over the folds.
6. Fit a Naive Bayes model to the data.
7. Repeat for different window sizes.
8. Compare performance over the number of gaussians, window size and classifier choice.

In order to get a stable estimate of the parameters of the GMM, normalization in the range $[-1, 1]$ is done for all the reduced features.

### 3.6.2. Dynamic Mode Classifier Testing

The performance of the dynamic classifiers were evaluated for both user dependent and user independent cases. Many factors influence the performance of the algorithms like the window size, feature sets. A systematic evaluation was therefore needed. The step wise procedure adopted for this is as follows:

First the window sizes were evaluated for all the classifiers
**A: User Dependent : Effect of Window Size**

1. Data for one subject is split with 80% going for the training and cross validation and 20% for testing.
2. Training data divided into 5 folds and cross validation is performed.
3. The model which gives the best weighted accuracy is chosen.
4. Steady State, Transitional and Overall errors determined for test set when classifying with the chosen model.
5. Repeat steps 1-3 for each window size.
6. Repeat steps 1-4 for each subject.
7. Determine mean and standard deviations of classification errors over the subjects.
8. Compare performance over the different window sizes.

After choosing the window size, the effect of having a delayed window that incorporates data after a gait event was evaluated. The procedure adopted is the same as in step A but with interval of delay as the parameter of interest.
After this evaluation, the effect of adding new features is investigated.The additional features are split into 3 groups: stride features which are only available for the heel contact classifier, orientation features and energy and entropy features.

**B: User Dependent : Effect of Additional Features**

1. Data for one subject split with 80% going for the training and cross validation and 20% for testing.
2. Training data divided into 5 folds and cross validation is performed.

3. The model which gives the best weighted error is chosen.

4. Steady State, Transitional and Overall errors determined for test set when classifying with the chosen model.

5. Repeat steps 1-3 for each subject.

6. Repeat steps 1-4 for each set of additional features.

7. Determine mean and standard deviations of classification errors over the subjects.

8. Compare performance with baseline classifier trained with simple features.

All further analysis is done with the chosen window size and with an augmented feature set that has the useful additional features. The best algorithms are also chosen for the next part of the analysis based on the user dependent results.

**C: User Independent**

For N subjects, the following procedure is adopted to find the effect of including more subjects data in the training set.

1. Leave out data set of Nth subject for testing.

2. Randomly select data sets of subject's 1 through $(N-1)$ and pool the data.

3. Train the classifier with the pooled data set.

4. Determine the Steady state, Transitional and Overall errors for the Nth subject.

5. Run 10 iterations of steps 2-4, with subject as a a random variable.

6. Determine mean and standard deviations of the classification errors over the iterations.

7. Repeat steps 1-6 by varying number of subject's in the training set from 1 to $N-1$

8. Compare performance over the number of subjects.

In addition to this, all classifiers are tested based on leave one out cross validation. The procedure for this is

1. Select data set of nth subject for $n = 1, \ldots\ldots, N$ as test set.

2. Pool data of remaining subjects as training set.

3. Classifier trained for the pooled data set.

4. Steady state, Transitional and Overall errors are determined for the test set.

5. Mean and standard deviations of the classification errors determined over the subjects.

6. Repeat for the different classifiers.

7. Compare performance over the classifiers.

The final algorithm is selected based on generalization capability which is judged based on the user independent analysis.

### 3.6.3. Hybrid Classifier Testing

The best algorithms are chosen for the components of the hybrid classifier. The real time performance is of interest for this classifier and it would use continuously extracted features. The static mode classifier utilizes continuous windows extracted at every interval of $\Delta t = 10ms$, the predictions for each window are stored in a voting vector. A majority voting scheme is then used wherein 80% of the decisions should be in agreement for the classifier to switch modes. The use of the majority voting scheme necessitates an additional parameter which is the length of the voting vector. The classification error and decision time are used to decide the optimal voting vector length in a user dependent analysis. This was only done for the optimal static mode classifier obtained through the analysis described in section 3.6.1. The voting vector lengths were varied from 10 to 100 in increments of 5. Ideally, a 0% error should be achieved at a certain voting length.
The dynamic mode classifier also uses windows of a fixed size extracted continuously at every 10ms interval. Since decisions are made when a gait event is detected, using discrete windows is problematic as the algorithm would send feature vectors that have been extracted with a smaller window to the trained classifier. Such feature vectors would be misclassified as the region of feature space they span would be different. The nature of the feature extraction used would result in a maximum delay of 10ms, in addition to this the computation also induces a slight delay.
The hybrid classifier was tested using leave one out cross validation with respect to the subjects. The error rates were determined separately for the static and dynamic classifiers based on all decisions within every trial recorded for that subject. It is difficult to find the decision time for the static classifier because of the user independent nature of the analysis. Instead, computation times were determined for the static classifier and decision times for the dynamic classifier. The performance metrics were then averaged over the trials. A total of 476 trials was evaluated for the final analysis.
The overall performance of the hybrid classifier is evaluated based on the decisions for each trial. Trials are flagged successful if all decisions are correct and unsuccessful if there is incorrect switching to a different mode. The time taken to correct decisions for the static mode classifier was also noted. The unsuccessful trials were further evaluated to find which component causes the error.

# 4

# Results & Discussion

This chapter compiles the results obtained for the analysis of each component of the hybrid classifier.

## 4.1. Static Mode Classifier

The static mode classifier operates on dimension reduced features. Therefore feature reduction was first performed on the training data set and visualized to see the separation in classes. Figure 4.1 shows the 3 dimensional feature space when FDA was used for dimension reduction and has much better class separation than when PCA was used for dimension reduction which is shown in Figure 4.2. The plots are for normalized feature values and serve as the input to the GMM and NB classifier.

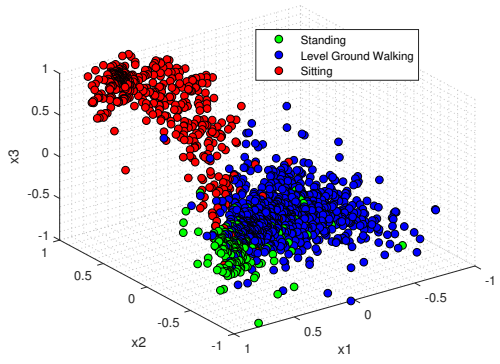



Figure 4.1: FDA dimension reduction to 3 dimensions

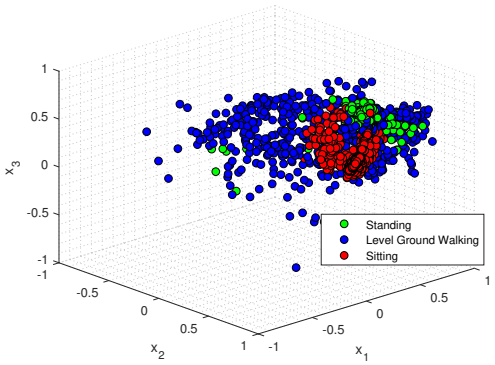



Figure 4.2: PCA dimension reduction to 3 dimensions

For reduction to 3 dimensions, on average FDA retains 99.99% of the variance in the data. PCA on the other hand retains a slightly lower 93.08%. PCA is unable to retain as much variance as FDA because the dimension of the input feature space is high. 26 features were used in total for the analysis, a reduction to 3 dimensions is therefore a steep decrease in dimensionality. It was noticed that at least 5 dimensions are needed to retain 99% of the variance. To avoid convergence problems for the static mode classifier algorithms ,PCA was dropped for further analysis.

A range of Gaussian mixtures were fit to the feature reduced data and the performance was evaluated with the AUC metric on the test data set. Figure 4.3 shows the effect of the window size on the AUC score. It scales linearly with larger windows having a higher AUC score. The performance of the classifier with the 2 dimensional data is only slightly better when compared to the 3 dimensional data. The AUC score reduces by $\approx 0.002$. Normally, an increase in dimension should correspond to a higher AUC score. However, this isn't the case because

the 3rd component of the FDA only accounts for $1e-6$ of the variance. This is much smaller compared to the other components and therefore no gains are noticed in the AUC score. A

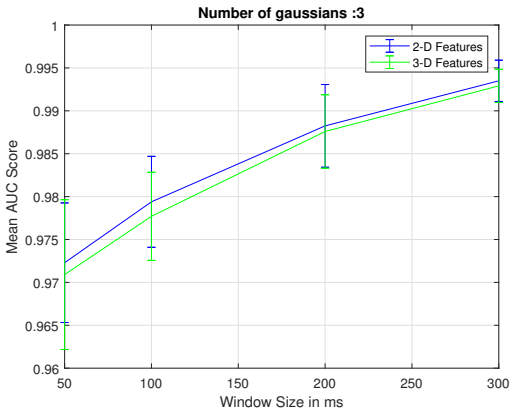



Figure 4.3: AUC score for a GMM fit with 3 gaussians for different window sizes

mixture with 3,4,3 and 3 Gaussian distributions gives the best AUC scores for windows of size 50,100,200 and 300 ms respectively. Figure 4.4 shows the effect of the number of gaussians on the AUC score, in general the AUC doesn't vary much with this parameter. Therefore the least complex model is chosen as that speeds up the computation time for test points. On analyzing the results, the mixture with 3 gaussians for features reduced using FDA to 2

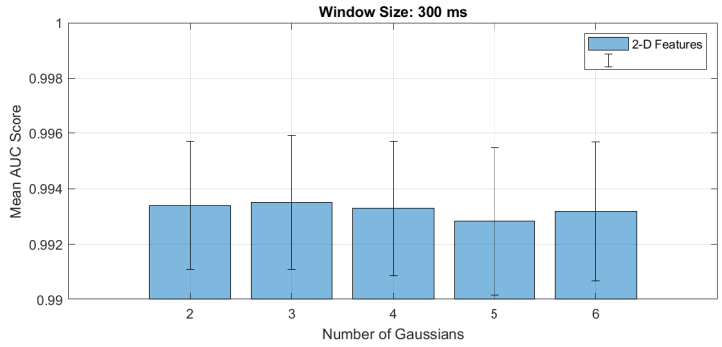



Figure 4.4: AUC scores with varying number of gaussians for a window size of 300ms

dimensions gives the best performance. The window size has a more pronounced effect on the performance, the classifier achieves a high AUC score of 0.994 when a window size of 300ms is used.
A NB model was also fit to the data, Figure 4.5 shows how this model compares to the best GMM for the different window sizes. The GMM consistently has a higher AUC score than the NB model. The NB model assumes statistical independence of features, any correlation between the features would give an inaccurate model and lower AUC scores. GMM on the other hand uses a full covariance matrix that picks up on correlations in the features. This is the reason why it performs better.
These results were obtained for windows that did not have any overlap, however the actual implementation in the hybrid classifier will have overlapping windows with a majority voting scheme to smooth out the decisions, the performance of this modified classifier will be described further in section 4.3. The 2 dimensional GMM with 3 components and 17 parameters in total was used for that analysis. The final implementation also uses a user independent model for the static classifier which is trained using pooled data.

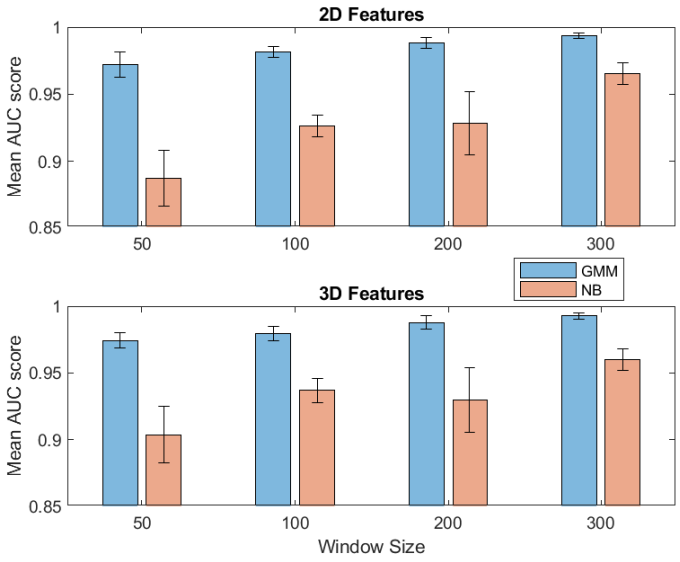



Figure 4.5: GMM versus NB for static mode classification

# 4.2. Dynamic Mode Classifier

## 4.2.1. User Dependent Classification

A systematic approach was used to evaluate the user dependent classification results. First the effect of the window size and classifier choice was investigated.

*Effect of Window Size*

Three window sizes from 100 to 300ms in intervals of 100ms were investigated to observe if any linear relationship exists between error rate and window size. A maximum window size of 300ms was used as it has approximately half of the gait information between two successive gait events which captures most of the variability in the gait mode. Only basic features were used for this analysis. Figure 4.6 shows the mean classification errors and $\pm$1Standard Error of the Mean(SEM) for the heel contact classifier for each of the algorithms across these window sizes. A clear linear trend can be observed for the LDA and SVM algorithms in both the steady state and the transitional error, for the RVM only the steady state error decreases. The error reduces significantly for the LDA algorithm, for the SVM the error reduction isn't as significant and this is because the feature vector is mapped to a new space in which it maximizes the margin between classes, this mapping is more important for a better performance from the SVM algorithm. The KNN and NB algorithms give higher errors than the rest of the algorithms and are not affected by the window size, this is because the parameters used in these algorithms are not very robust and do not pick up on subtle differences in the feature vectors obtained using different window sizes. For instance, changing the number of neighbors, a parameter of the KNN algorithm affects the final classification error more than the modified feature vectors. The results reported here were obtained using a heuristic[57], where the number of neighbors is the square root of the number of training points.

Based on these results, a 300ms window was chosen for further analysis as it gives the best results for a majority of the algorithms. Since the static mode classifier also uses a 300ms window, there is no need to store two different feature vectors for the two classifiers which is another reason why this window size was focused on.

*Effect of delayed windows*

It could be useful to add information after a gait event has occurred, this would mean that the decision from the classifier would be delayed but it can be a reasonable trade-off if lower error rates are achieved and if the delay doesn't have an impact on balance. For the 300

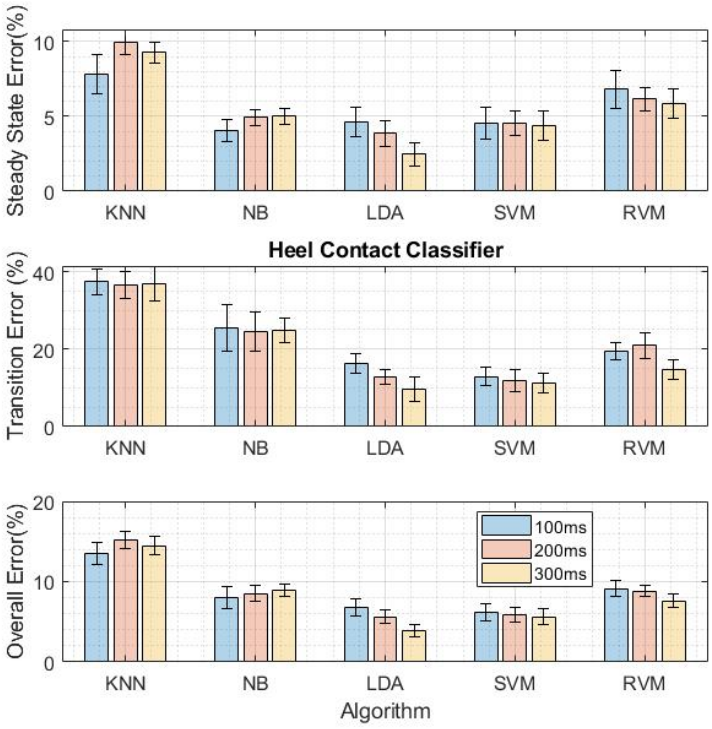



Figure 4.6: Classification Error Versus Window Size

ms window the effect of such delays was analyzed, 4 delays were off interest: 30ms, 60ms ,90ms and 120ms. This was investigated for the three best classifiers, Figures 4.7 and 4.8 show the mean error rates for the different delays for the heel contact and toe off classifiers respectively when compared to the baseline results of having no delay. For RVM, the error rates only show an improvement with a delay of 120ms which is too high for implementing on a prosthesis. The steady state error goes down with increasing delay for the heel contact classifier for LDA and SVM, no such trend is observed in the transitional error for the LDA which has the best transitional error at a delay of 60ms, however the overall error remains at the same level. The transitional error increases with delay for SVM whereas the overall errors remain at the same level so using a delay does not improve the performance of the classifiers enough across all choices to warrant it's inclusion in the hybrid classifier.
For the toe off classifiers, no clear trend exists in the error rates across the algorithms. For LDA the best error is achieved with a delay of 30ms and for the SVM and RVM when a delay of 60ms is used. The critical transitional errors are quite low when using the respective delays so it would make sense to include it in the implementation for the prosthesis provided it doesn't impact gait stability. A separate feasibility study is needed to see the impact of delayed control actions on gait stability and if it is perceived by the user, a choice can then be made on whether delayed windows should be used. For the rest of the analysis in this thesis though it doesn't make sense to continue with the delayed windows as the overall error rates remain at the same level.

*Effect of Additional Features*

For the algorithms where the window size didn't significantly reduce the error, adding additional features can be helpful. For the SVM and RVM algorithms it could improve the mapping between the input feature space and the kernelized feature space thus giving better performance, for the KNN and NB algorithms it could improve the class separability and thus reduce the number of misclassifications. The additional features were grouped into 3 sets, and the effect of each set on the classification errors when compared to the baseline classifier

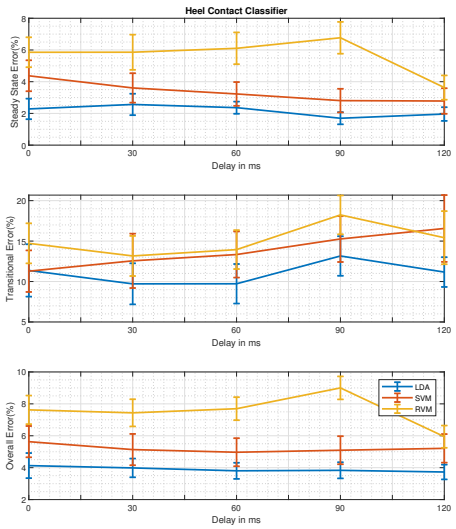



Figure 4.7: Classification errors for different delays in heel contact classifier

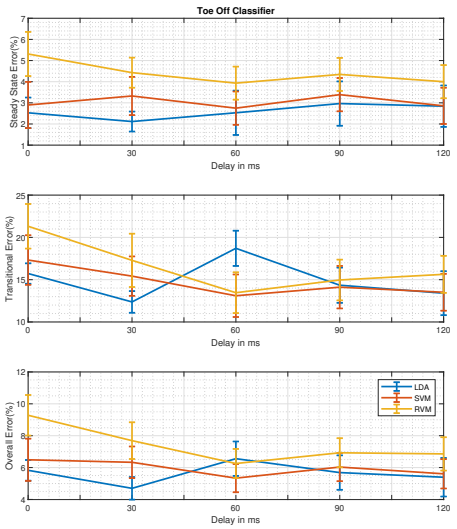



Figure 4.8: Classification errors for different delays in toe off classifier

was then investigated. Since the RVM takes a longer time to train and it shares the same underlying machinery as SVM have of using a kernel to transform the feature vector, the analysis was done for the SVM only and the final set of features was used to evaluate the performance of the RVM.
Figure 4.9 shows the mean classification error $\pm$1SEM of the heel contact classifier when using stride features compared to the baseline performance. Adding the stride features reduces

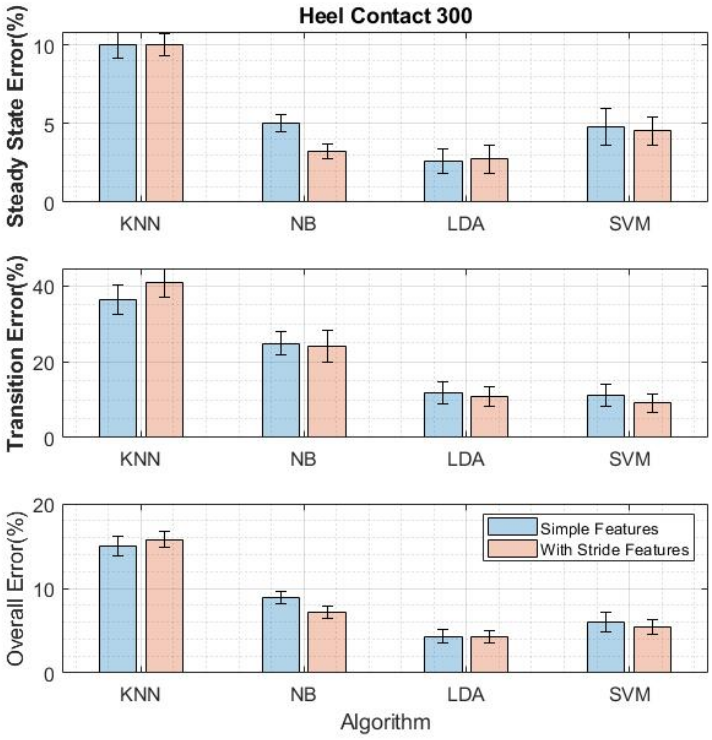



Figure 4.9: Baseline Classification Error compared to Classification Error when using stride features

both the steady state and transitional errors for the SVM and NB algorithms only marginally. For the LDA algorithm, the transitional errors go down marginally and the steady state performance remains the same whereas for the KNN algorithm adding these additional features worsens the classification error. The stride features were obtained by integrating accelerations in the vertical direction and in the walking direction, however these directions were obtained by only using the sagittal plane orientation of the shank estimated using a complementary filter. Therefore the stride length and height are not precisely estimated which is the reason why there is only marginal reduction in error. A more accurate 3D orientation estimated using a non linear kalman filter can vastly improve the value of the stride features and can help in distinguishing specifically the stair modes from the other modes.
Figure 4.10 shows the mean classification error ±1SEM of the heel contact classifier when using orientation features compared to the baseline performance. Adding the orientation fea-

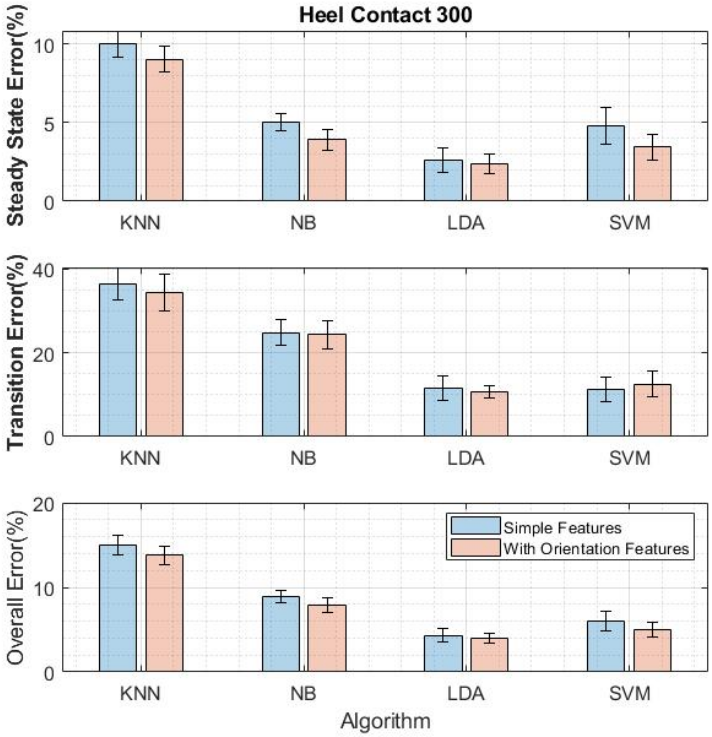



Figure 4.10: Baseline Classification Error compared to Classification Error when using orientation features

tures decreases the steady state error for all the algorithms in general, it gives a significant reduction for the SVM and NB algorithms. The transitional errors remain at the same level when compared to the baseline. The orientation of the thigh is indirectly encoded as it is the linear combination of the orientation of the shank and the relative orientation between the thigh and shank which is the knee angle recorded by the goniometer. This orientation doesn't vary much between the ramp modes, LW and SD however since the hip flexes more during SA the orientation of the thigh is different for this mode. The orientation features thus only add information for one mode. To gain further insight into the misclassifications, the confusion matrix for the SVM classifier for one subject shown in Figure 4.11 was plotted to see which modes are being mislabelled the most. A majority of the misclassifications were between the ramp modes and LW, a 100% recognition rate is achieved for SA and only one decision was wrong for SD mode. The confusion matrix confirms as predicted that the orientation features don't add much value for the ramp modes.
Figure 4.12 shows the mean classification error ±1SEM of the heel contact classifier when using energy and entropy of the accelerometer channels compared to the baseline performance. These features only reduce the steady state error whereas the transitional error remains at

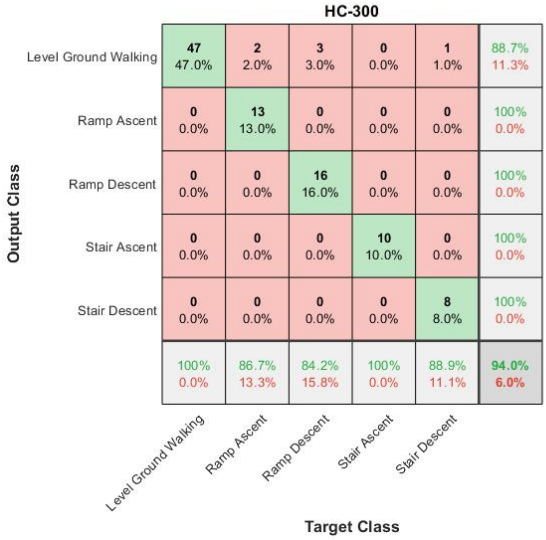



Figure 4.11: Confusion Matrix for one subject

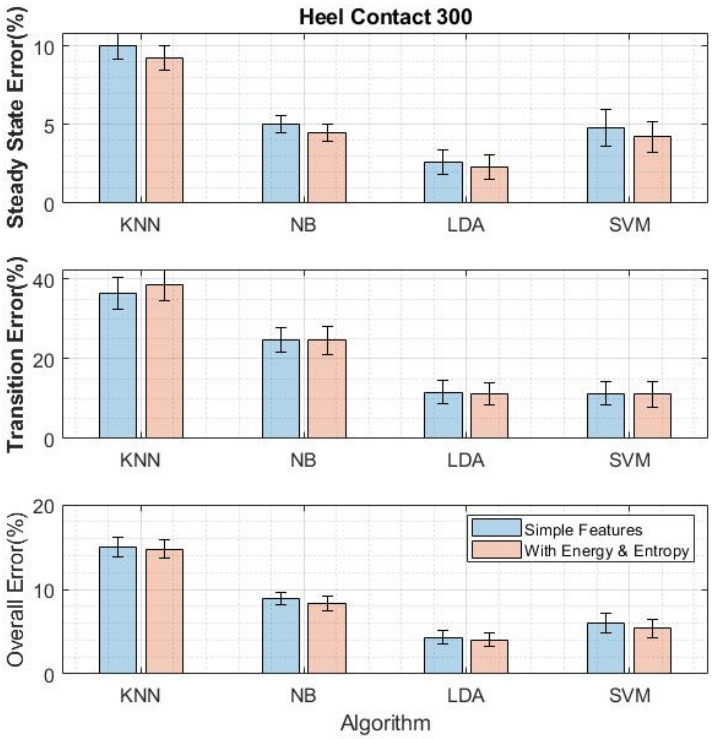



Figure 4.12: Baseline Classification Error compared to Classification Error when using signal energy and entropy

the same level, since these features are computationally expensive to calculate they were not used for the final hybrid classifier implementation. The energy and entropy are higher for the stair modes but most misclassifications happen between ramp modes as shown earlier which is why not much improvement is noticed in the transitional error.
Figures 4.13 and 4.14 show the error rates of the classifiers when ground reaction force features are added to the baseline features. The WR-Lab data set was used for this analysis. A reduction was obtained across all errors for the SVM algorithm but not for LDA for which

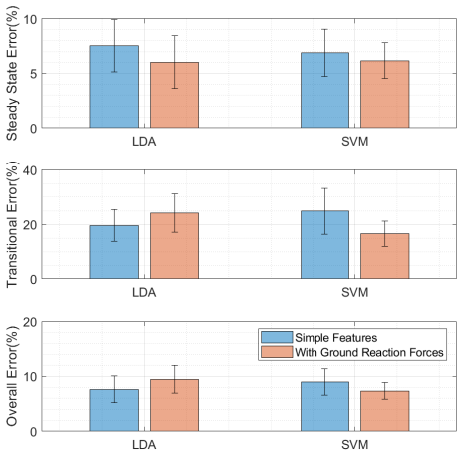



Figure 4.13: Effect of Ground Reaction Forces on Heel Contact classifier performance

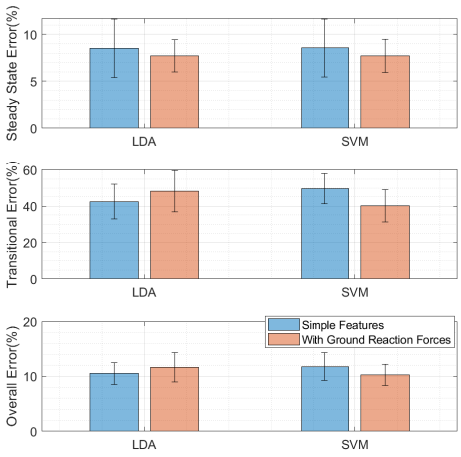



Figure 4.14: Effect of Ground Reaction Forces on Toe Off classifier performance

only the steady state error goes down, this is because the number of transition steps in this investigation was quite small when compared to the dimension of the feature space and the algorithm is unable to fit a proper model that adapts to these steps. The SVM is capable of adapting to higher dimension data sets with few training examples and therefore the results can be relied on, it would therefore make sense to add sagittal plane ground reaction forces to the feature vectors for the prosthesis classifier.

*Effect of Grouping Modes*

The investigation of the confusion matrices yielded that the ramp modes are hard to distinguish from LW. For the prosthesis, it is possible to either use similar low level control settings for RA and LW modes or just have a LW mode with a slope estimator. This enables the grouping of locomotion modes making it either a 4 mode(LW,SA,SD,RA) or 3 mode(LW,SA,SD) problem for the intent recognition system. Figure 4.15 shows the effect this has on the errors of the various algorithms. These results were obtained with the additional stride and orientation features included.

Grouping the modes significantly reduces the misclassification rate as was expected, LDA and SVM give the lowest overall error of $\approx$ 2% with the SVM giving the lowest transitional error of 4.87%. The transitional decisions are the most critical decisions the classifier has to make. RVM also gives a low overall error of 2.51% and selects on average only 11 feature vectors from the training set compared to 60 which the SVM uses in it's model. However the training time is rather high which is a real downside and it grows exponentially with the number of feature vectors in the training set. KNN is the worst performing algorithm, this could be due to the dimension of the feature space and overlapping class distributions that the instance based learning cannot adapt to. NB does give a low overall error but the critical transitional error is 17.69%,which is too high to implement on the prosthesis.

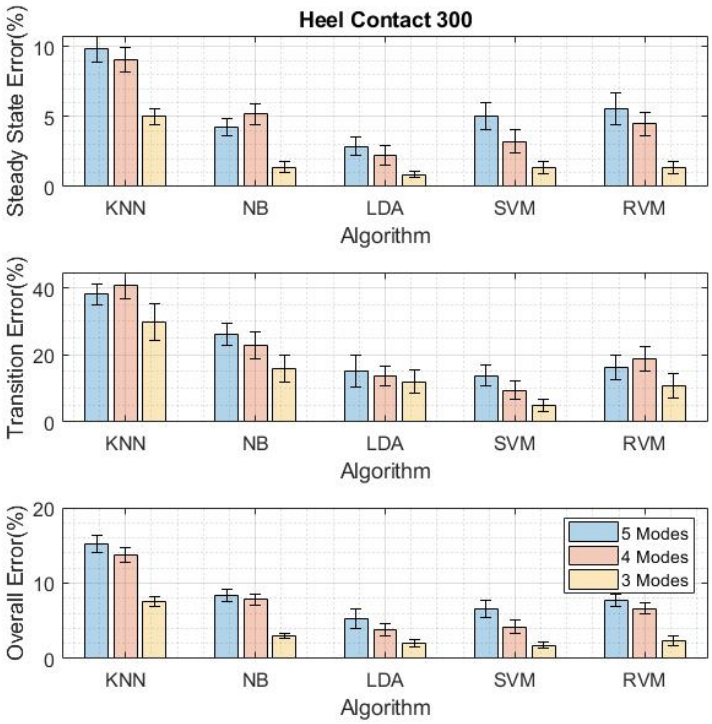



Figure 4.15: Effect of grouping modes on heel contact classifier

### 4.2.2. User Independent Classification

The user dependent classification yielded the LDA and SVM to be the best algorithm choices for intent recognition. The performance of the algorithms generally deteriorates when tested on a novel subject due to inter subject gait variability. For a more robust evaluation, the generalization capability of these algorithms was investigated. One major factor that effects the performance is the training set size which is described in the following section.

*Effect of training set size*

Figure 4.16and 4.17 shows the mean classification errors of the heel contact and toe off classifiers respectively trained with different number of subject data sets in the training set for 3 mode classification, all errors go down with an increase in the training set size. Plateauing is present for the steady state error and not much improvement is noticed after 5 subjects for the toe off classifier with the error settling around the 3% mark, the transitional error however goes down at a faster rate and doesn't plateau for the heel contact classifier so a better performance can be obtained by adding more data.

SVM has very good generalization capability. For models trained with just one subject's data gives a much lower steady state error than LDA. This trend is consistent with increasing number of subjects as well. In addition to this, the standard error for the SVM is lower over the iterations indicating that is independent of a random choice in the data used in the training set which is advantageous. The error from the LDA algorithm seems to converge to that of the SVM for the toe off classifier, however this could be because of the choice of the novel subject. The model could be picking up on correlations in the data between subjects used for the analysis. To evaluate if this is the case leave one out cross validation was done where each subjects data was used in turn as the test set

*Leave One Out Cross Validation*

Figure 4.18 and 4.19 show the results of the leave one out cross validation for the heel contact and toe off classifiers respectively when using different number of modes. The SVM gives the

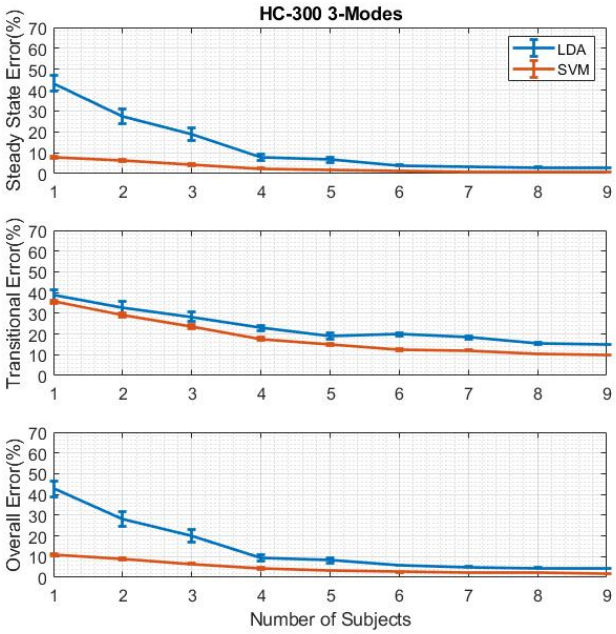



Figure 4.16: Effect of training set size on classification error for Heel Contact classifier, Window Size=300ms

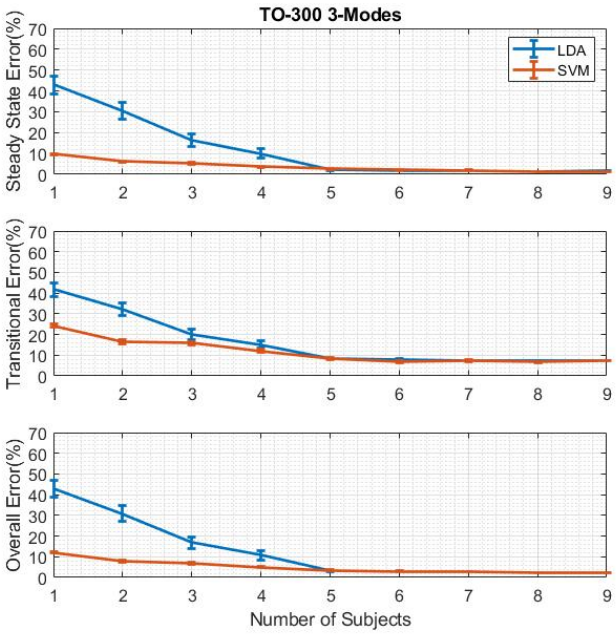



Figure 4.17: Effect of training set size on classification error of Toe Off Classifier, Window Size=300ms

lower error rate across all the models and is therefore a good choice for the hybrid classifier implementation.
Not much improvement is noticed when the number of modes is reduced from 5 to 4 especially for the transitional errors which means that there is still a lot of variability in RA mode among the subjects and the models cannot adapt to it. A significant improvement is noticed when all ramp modes are relabelled as LW which is why this configuration was used for the analysis of the hybrid classifier.

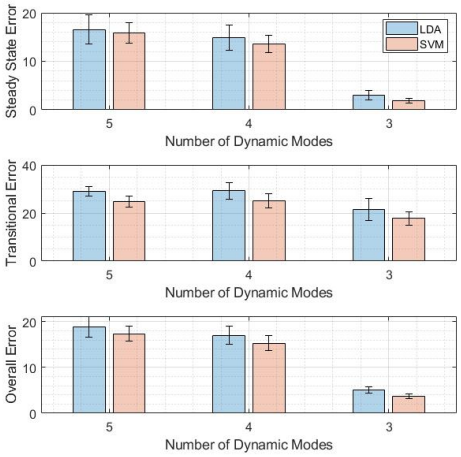



Figure 4.18: Leave One Out Cross Validation for Heel Contact Classifier

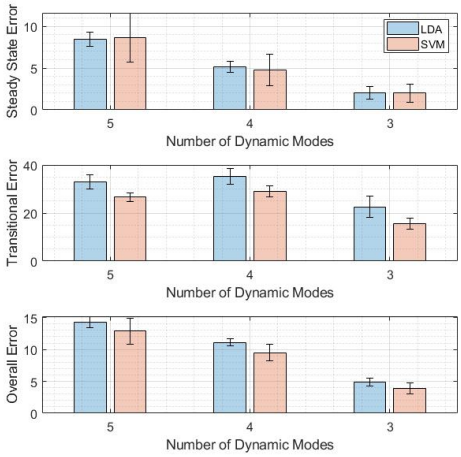



Figure 4.19: Leave One Out Cross Validation for Toe Off Classifier

## 4.3. Hybrid Classifier

The hybrid classifier combines the two classifiers discussed in the previous two sections and optimally combines the decisions to give an output. This classifier operates with 300ms sliding continuous windows extracted at an interval of 10ms as was previously used in a real time implementation by Varol *et al.*[30] for a multimodal intent recognition system that aimed to classify Si, St and LW. The interval size can impact the results when looking at time scales of 10ms but that would induce additional delays in the decisions, also using a smaller time scale is computationally more expensive as feature vectors have to calculated more frequently. A 10ms interval is an optimal trade off between delay and computational burden.

### 4.3.1. Optimal Voting Length

A sliding window implementation requires the use of a voting scheme to smooth out the decision from the static classifier, the length of the voting vector has an impact on the classification error, Figure 4.20 shows the mean classification error against the voting vector length for user dependent classification. It wasn't possible to reach a 0% error for the static classifier, this is because the data set used for the study had a lot of mislabelled data points especially between St and LW, in addition to this a threshold of 20° was used while relabelling the Si and St modes which is high, most studies use a threshold of 5°. The error rate saturates at the 6% mark for a voting vector length of 50 and higher, this would give a minimum overall delay of $d = \frac{f}{2} + l_{max} \times \delta t = 650ms$, where $\delta t$ is the sliding interval and the window size $f = 300ms$. Ideally a delay below 500ms is preferred which would be possible on adding ground reaction forces and moments to the feature set. A preliminary investigation was done for the optimal voting length when using GRF for the WR-Lab dataset. Figure 4.21 shows the mean error rate against the voting length. It is possible to achieve a 0% error for voting length 25 and higher, this would result in a minimum delay of 400ms, which is within the acceptable limit.

The hybrid classifier uses an user independent model of the static classifier. The performance would deteriorate even further because of inter subject variability. In order to avoid a high misclassification rate a transition probability matrix was included for the static classifier that mitigates transitions from LW to Si and vice versa. The transition probability matrix was a diagonal matrix with the class probabilities as the elements of the diagonal, for the

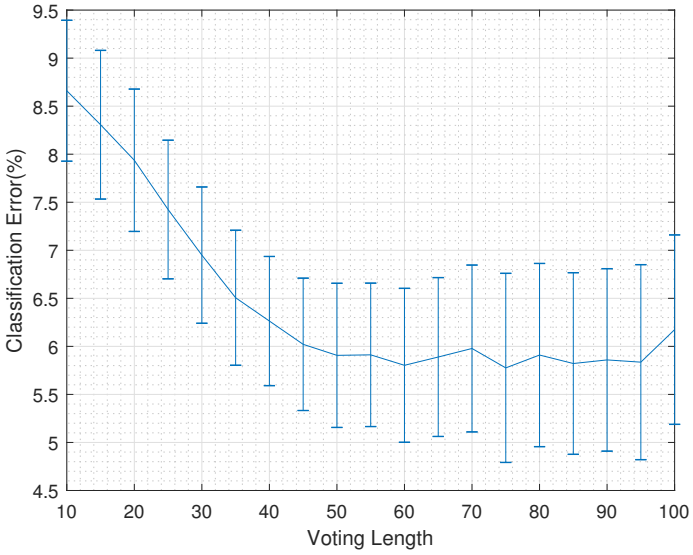



Figure 4.20: Analysis of Voting Length

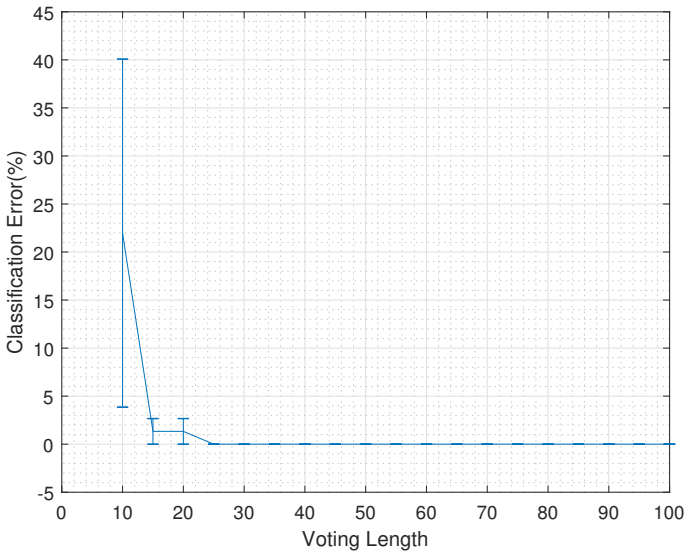



Figure 4.21: Analysis of Voting Length when using GRF features

final analysis the probabilities were set to the following values

$$T_{Si} = \begin{matrix} Si & LW & St \\ \begin{pmatrix} 0.4 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.4 \end{pmatrix} & \begin{matrix} Si \\ LW \\ St \end{matrix} \end{matrix}, T_{LW} = \begin{matrix} Si & LW & St \\ \begin{pmatrix} 0.02 & 0 & 0 \\ 0 & 0.6 & 0 \\ 0 & 0 & 0.38 \end{pmatrix} & \begin{matrix} Si \\ LW \\ St \end{matrix} \end{matrix}, T_{St} = \begin{matrix} Si & LW & St \\ \begin{pmatrix} 0.33 & 0 & 0 \\ 0 & 0.33 & 0 \\ 0 & 0 & 0.33 \end{pmatrix} & \begin{matrix} Si \\ LW \\ St \end{matrix} \end{matrix} \quad (4.1)$$

These probabilities were selected based on the likelihood of the respective transitions happening, for instance directly going into Si mode from LW is a very unusual transition so a low probability is associated to that element of the matrix, on the other hand, it is possible that one goes from Si to LW when there is an insignificant standing period and therefore this transition has a higher probability than the reverse case. This assumption allowed the use of a shorter voting length of 35 which would give an overall delay of 500ms, the error rate

obtained was 4.92% for the user dependent analysis.

### 4.3.2. Error Rates and Computation Times

Table 4.1 shows the leave one out cross validation results, with separate error rates for the static and dynamic mode classifiers. Computation times have been reported for the static classifier and decision times for the dynamic classifier. The choice of SVM a non probabilis-

| STATIC CLASSIFIER(GMM) | | DYNAMIC CLASSIFIER(SVM) | | | |
|---|---|---|---|---|---|
| Error Rate(%) | Computation Time(ms) | Steady State Error(%) | Transitional Error(%) | Overall Error(%) | Decision Time(ms) |
| $11.3 \pm 0.5$ | $0.08 \pm 0.004$ | $1.53 \pm 0.5$ | $18.3 \pm 1.7$ | $3.4 \pm 0.5$ | $9.76 \pm 0.19$ |

Table 4.1: Performance Metrics for non probabilistic hybrid classifier

tic classifier for the hybrid classifier had some drawbacks as the outputs of the classifiers couldn't be combined for the overlapping LW mode, this led to high error rates of $\approx 20\%$ for the static classifier. The solution adopted for this was to utilize the decision of the dynamic classifier to correct that of the static classifier, this helps to transition quicker and also mitigates the confusion between St and LW. The results in Table 4.1 are for the system that includes the above said rule, this gives an error reduction of $\sim 9\%$. The error rates were obtained using all the decisions from the static classifier, since this occurs every 10ms it is based on many decisions and doesn't account for the delay in the decisions. Instead, it is a measure of the degree of overlap between the ground truth and the prediction. In the overall performance of the hybrid classifier, only the transition decisions for the static modes are considered which is a better measure.
A high transitional error of $\sim 18.3\%$ was obtained for the dynamic classifier. The transition errors predominantly occur when the right leg is the trailing leg, the reason for this can be understood from Figure 4.22 which shows the knee angle for one such trial and the true and predicted labels at the different gait events, the misclassification occurs in the heel contact classifier. The data is labelled such that if either leg enters a certain mode then all the gait events of both legs are labelled with the new mode till the next transition occurs. This is an issue as the trailing leg is still in the previous mode and therefore the gait kinematics and kinetics resemble that mode more. This leads to the classifier making a wrong prediction. The toe off classifier also has a similar issue. In all cases, the prediction at the next gait event is correct so the system is capable of switching to the right mode if labelled correctly.
The data was relabelled with respect to the right leg. Specifically, Table 4.2 shows the obtained error rates when the data was relabelled with respect to the right leg. The transitional error decreases to $\sim 7\%$ and this decreases the overall error , the steady state error on the other hand goes up, any errors in predicting LW mode can be rectified with a static classifier that can detect all transitions correctly.

| STATIC CLASSIFIER(GMM) | | DYNAMIC CLASSIFIER(SVM) | | | |
|---|---|---|---|---|---|
| Error Rate(%) | Computation Time(ms) | Steady State Error(%) | Transitional Error(%) | Overall Error(%) | Decision Time(ms) |
| $11.3 \pm 0.5$ | $0.08 \pm 0.004$ | $2.19 \pm 0.5$ | $7.17 \pm 2.4$ | $2.62 \pm 0.5$ | $9.59 \pm 0.26$ |

Table 4.2: Performance Metrics for non probabilistic hybrid classifier with relabelled data

The use of a probabilistic framework that combines the decisions of the static and dynamic classifiers can also improve the performance. To investigate this, an RVM model which was trained on the pooled data was tested with one novel subject's data and the error rates were compared to the classifier that uses the non probabilistic SVM model. The Gaussian mixture models give probabilities on different scales for the 3 modes whereas RVM outputs probabilities in the range $[0, 1]$, to compare the two the output of the GMM was first standardized and then the probabilities were scaled with respect to each other. A threshold of 0.9 was set for the overlapping LW mode and if any of the classifiers output a higher probability the decision of both classifiers was changed to LW. This rule yielded a reduction in the error for the static classifier. Table 4.3 shows the obtained error rates, decision and computation times for the algorithm.

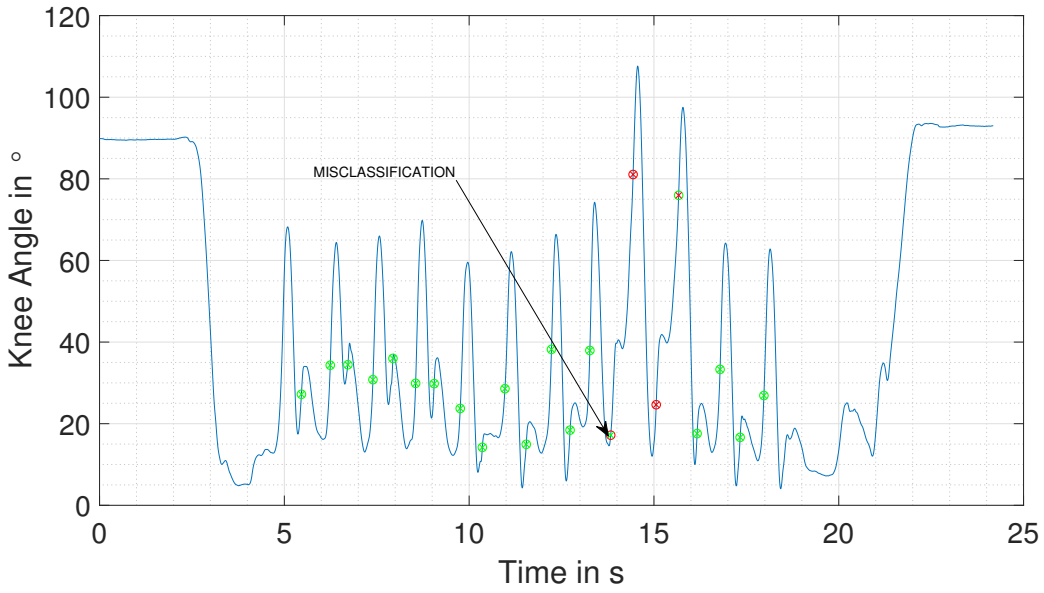



Figure 4.22: Example of a typical misclassification

| STATIC CLASSIFIER(GMM) | | DYNAMIC CLASSIFIER(RVM) | | | |
|---|---|---|---|---|---|
| Error Rate(%) | Computation Time(ms) | Steady State Error(%) | Transitional Error(%) | Overall Error(%) | Decision Time(ms) |
| 9.94 | 0.06 | 1.13 | 7.29 | 2.75 | 6.52 |

Table 4.3: Performance Metrics for probabilistic hybrid classifier

### 4.3.3. Overall Performance

Figure 4.23 shows the ground truth and the predictions of the hybrid classifier for one of the successful trials. The classifier is able to successfully detect all transitions for this trial, the incorrect labelling of the data has an effect on the predictions between St and LW and can be observed as an early onset of St for this particular trial. The user wouldn't be able to notice the change if he is close to gait termination, the use of a controller that has an overlap between the static modes for the prosthesis could also make the delay in transition imperceptible. The user independent nature of the classifier causes a longer decision time for some transitions. The maximum delay for a decision was 610ms and it occurred for the transition between St and LW. On the other hand, the hybrid classifier allows early detection of the transitions between Si and St for the static classifier. This is due to differences in the knee velocity among the subjects.

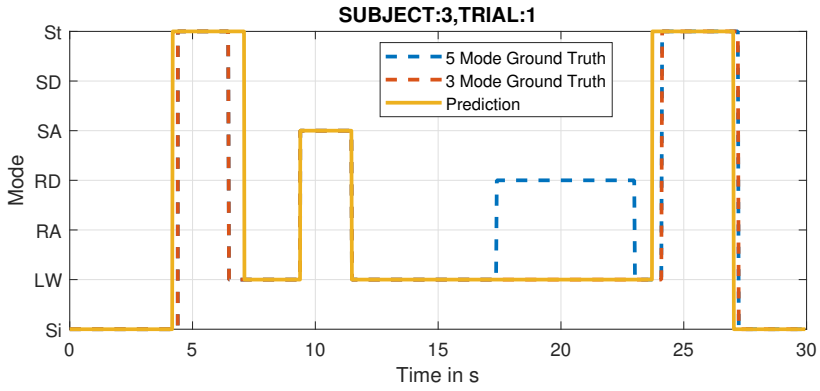



Figure 4.23: Mode Profiles: Ground Truth and Predictions for a successful trial

Figure 4.24 shows the ground truth and the predictions of the hybrid classifier for one of

the unsuccessful trials. Errors arise in both the static and dynamic classifiers for this trial. There is one incorrect decision each for the toe off and heel contact classifier for the SD mode. The static mode classifier switches to St mode during LW mode, this can be attributed to the mode distribution overlap. The decision is only corrected after $\sim 300ms$ in this trial.

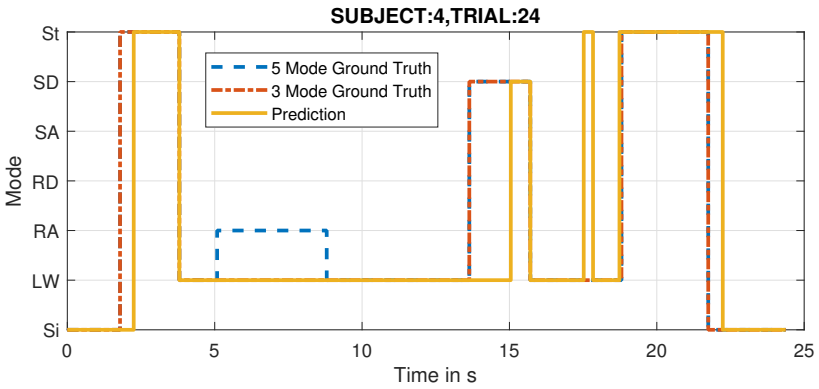



Figure 4.24: Mode Profiles: Ground Truth and Predictions for an unsuccessful trial

Out of the 476 trials that were investigated, 229 trials were successful and had a 0% error. The remaining 244 trials were unsuccessful and had several issues. In 22 of these trials, direct switching occurs between the static modes Si and LW, mainly because the St mode separating the two lasted for a small duration. One such trial is shown in 4.25 where the classifier fails to recognize the St mode at the end of the trial. The use of the majority voting scheme requires that a certain mode persists for at least 80% of the voting length, which is 280ms provided the modes don't have overlapping distributions. This is not the case with the data set and is the reason why the transitions were missed.

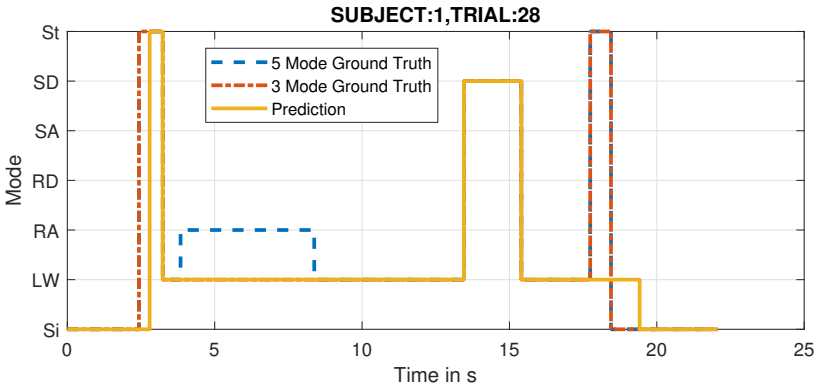



Figure 4.25: Mode Profiles: Ground Truth and Predictions for an unsuccessful trial with missed static mode transitions

In all the remaining incorrect trials the transitions are detected correctly for the static modes. However, the classifier wrongly switches to St mode for periods of average $410ms$ while in LW mode. The minimum time it stays in the incorrect mode depends on the voting length. These set of trials also had errors from the dynamic mode classifier, Figure 4.26 shows the confusion matrix for all the decisions made by this classifier. Most misclassifications occur for the SD mode. Two subjects in particular had a lot of incorrect decisions, the reason for this is their lower gait speed. The average trial times for the subjects are plotted in Figure 4.27, the data of subjects 4 and 10 were the ones with the most misclassifications. As the window size is fixed, all the simple statistical features will be different as the percentage of gait cycle covered by the window is smaller. The gait speed is therefore an important aspect that should be considered in the design of user independent classifiers.

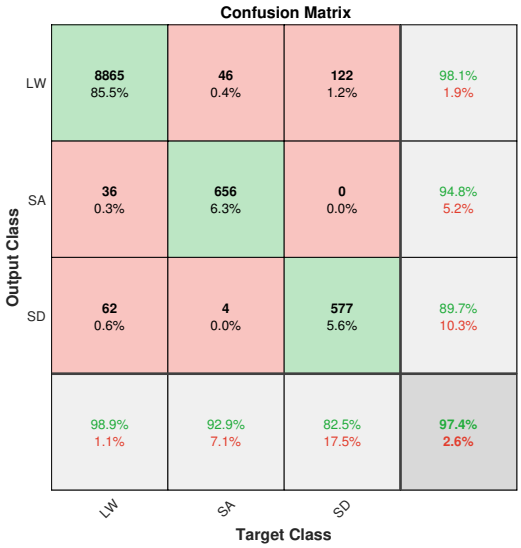



Figure 4.26: Confusion Matrix for user independent classification

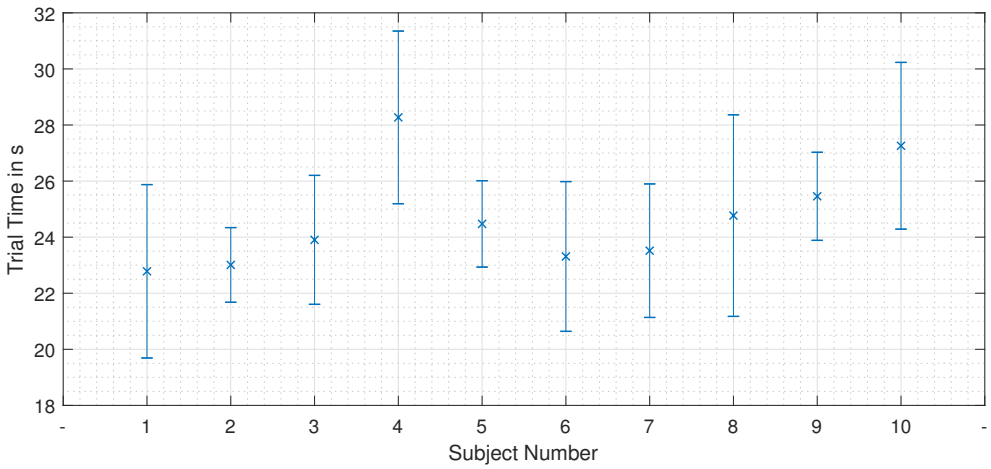



Figure 4.27: Trial times(Mean $\pm$ 1SD) for each subject

# 5

# Conclusions

## 5.1. Summary

Locomotion Intent Recognition is an important element in the hierarchical control of an active transfemoral prosthesis. Intent recognition can be treated as as supervised machine learning problem and a variety of a algorithms can be used to train models that are used to predict the intent. The performance of the different algorithms is dependent on the statistical properties of the data, therefore a robust analysis is needed to choose the best algorithm that generalizes well to the data set. This thesis demonstrated that a hybrid classifier that combines the outputs from models obtained using different algorithms aids in achieving a lower error rate.

The performance of a hybrid classifier framework was evaluated using healthy subject gait data. The classifier only uses mechanical sensors and includes a 6-axis inertial measurement unit and a knee position sensor. The hybrid classifier has a static classifier that separates the classes of Si, St & LW, and a dynamic classifier that separates the classes of LW, SA, SD, RA & RD. Each of these classifiers were evaluated separately offline based on different parameters such as the classifier choice, window size & features.

The static classifier required the use of feature reduction to reduce the dimension of the feature space. FDA retains more than 99% of the variance in the data whereas PCA retains 92% when the feature vectors were reduced to 3 dimensions. Two probabilistic algorithms: Naive Bayes & GMM were then compared using the AUC metric. The feature vectors for 4 different window sizes 50,100,200& 300ms reduced using FDA were compared to find the optimal window size. GMM give a higher AUC score consistently across all window sizes and the highest AUC score was obtained for a 300ms window. The 3 component mixture model gave a good trade-off between the number of parameters and performance. The static classifier also required the use of a majority voting scheme to smooth out the decision when using sliding windows, the lowest error that could be obtained was $\approx$ 5.5% when using a voting vector length of 50 giving a total delay of 650ms. The non zero error rate is partly due to incorrect labelling of the data for the St and LW mode. To reduce the overall delay and the error a series of transition probability matrices were used that prevented strange transitions from Si to LW and vice versa, with this framework a voting vector length of 35 could be used giving an overall delay of 500ms which is acceptable for the static modes.

The dynamic classifier uses windows extracted preceding the gait events of heel contact and toe off. The algorithmic choices for this include LDA, SVM, RVM, KNN, & NB. 3 window sizes of 100,200 & 300ms were investigated and the steady state, transitional & overall errors were used to compare the performance. The input features were split into 4 categories: simple features that had statistical descriptors like the mean, standard deviation, maximum, minimum, initial & final values; stride features that had the stride length and stride height; orientation features that had the initial and final value of the absolute orientation of the shank; energy and entropy features.

An optimal window size of 300ms was obtained when using the simple feature set. LDA, SVM give the lowest error rates and the next best algorithm is RVM which gives a very light model but at the expense of a much higher training time. Adding information after a gait event only yielded about $0.5 - 1\%$ improvement in the overall error rate, this was based on the analysis of 4 delays of 30,60,90 & 120ms. No correlation was observed between the delay and the transitional error rate, the steady state error goes down when increasing the delay for the heel contact classifier. Since a delayed decision is not preferred and the performance didn't improve much, delayed windows were omitted from the hybrid classifier analysis.
The additional sets of features were then added to the feature vectors of the optimal window size. The stride features gave a lower transitional error whereas the orientation features and the energy and entropy features yielded a reduction in the steady state error across all classifiers. A good orientation estimate will improve the accuracy of the calculated stride features and that will aid in better differentiating the stair modes from the level walking mode. The confusion matrices gave further insight into the effect of the additional features, they don't have an impact on the ramp mode classifications indicating that there is an overlap between these modes and level walking and the additional features don't have enough information that would help separate the modes.
From a control viewpoint for the prosthesis, it is possible to use similar low level control settings for the ramp modes so the intent recognition system may not explicitly need to make decisions on these modes. A significant improvement in error rates was obtained when grouping the ramp modes with level walking, both LDA & SVM attain about $\sim 2\%$ overall error rate. The transitional error for SVM is 4.87% which is the lowest for all classifiers

Ideally, for the prosthesis, pretrained models should be able to generalize to a novel subject so that retraining of algorithms is not required to ensure good performance. SVM gives a lower error rate than LDA in the user independent analysis and is capable of reaching an average steady state error rate of 3% when leave one out cross validation is done. The average transitional error is around the 20% mark, the high error rate is because of the way the data is labelled, relabelling the feature vectors with respect to the gait events of the instrumented leg would yield a better performance.
An analysis on the size of the training data size on the mean classification error was also done, for the toe off classifier the learning curve saturates for higher than 5 subjects data in the training set. For the heel contact classifier only the steady state error rate saturates for higher than 6 subjects data in the training set, the transitional error on the other hand could be improved with increasing the training data set size even more. A preliminary analysis was done for RVM up to 6 subjects in the training set, it gives good generalization just like the SVM, however the training times limited a comprehensive evaluation and hence the results were omitted.

The two classifiers were then combined in a hybrid classifier and the decision times and error rates were determined. The static mode classifier gave an error of 11.3% when using the transitional probability matrices along with a preference for the decision from the dynamic classifier. A voting length of 35 was used for the static classifier, the decision times varied a lot because of the user independent nature of the models. Some decisions were made earlier than anticipated and in other cases delays greater than 500ms were observed. The computation times were 0.08ms on average. After relabelling the data with respect to the right leg, the dynamic mode classifier gave an average steady state error of 2.19% and an average transitional error of 7.17% for 3 mode classification. The decision times were on average $9.59ms$. The use of a probabilistic framework with RVM as the dynamic classifier was investigated for one subject. Combining the decisions gave a further reduction of 2% for the static classifier. For the dynamic modes the hybrid classifier reaches a steady state error of 1.13% and a transitional error of 7.29% . This framework also made quicker predictions, requiring on average only 6.52ms. The downside of using RVM is the training time. It is almost 10 times higher than SVM and it further scales with the training set size, however it should be feasible to use once an analysis is done on the minimum number of subjects data required to achieve a good generalization.

The overall performance was evaluated using the decisions for each of the 476 trials tested.

All transitions were detected correctly in 229 trials. The remaining trials mainly had confusion between St and LW for the static mode classifier and between SD and LW for the dynamic mode classifier. The misclassifications from the dynamic classifier were mainly due to differences in gait speeds among the subjects.

Since the 3 mode dynamic classifier gives a low overall error rate of 2.62%, it is a promising avenue for implementing on the transfemoral prosthesis. The hybrid classifier should be trained with properly labelled data to get accurate models so that the hybrid classifier can make better decisions.

## 5.2. Comparison with existing literature

The lowest error rates reported in the literature used additional sensors like ankle angle and velocity encoders, motor currents and ground reaction forces, moments. A number of research articles report the steady state and transitional errors for user dependent classification of the dynamic modes. For LDA, the lowest error was reported by Young *et al.*[41] , a 2% steady state and 14% transitional error was obtained. The respective errors of the LDA based classifier in this thesis are 2.85% and 12.18%. The steady state accuracy can improve with the inclusion of more sensors.

The static mode classifier used in the research by Varol *et al.*[30] reached a 0% error when using a majority voting scheme. The results obtained in this study had a 5% error rate, this was due to the use of higher thresholds which led to more violations and due to the way the data was labelled.

The proposed hybrid classifier gives better performance than the user independent classifier errors reported in Young *et al.*[49]. They use LW data to retrain the algorithm and report a steady state error of ~ 13% and a transitional error of ~ 36% for 5 mode classification. For the hybrid classifier using 5 dynamic mode the errors obtained were 13.2% and 22.8% respectively. In particular,a lower transitional error is achieved.

A 3 mode user independent classifier was implemented in the article by Young *et al.*[41]. They make use of a DBN classifier that takes in feature vectors extracted from windows at 8 different points in the gait cycle. Additionally they make use of a mode specific classifier that is trained specifically for transitions that are possible from a given mode, therefore for every mode other than LW it is a binary classifier. They report a steady state error of ~ 2% and transitional errors of ~ 4%. The steady state errors of the hybrid classifier are in the same range but the transitional errors are higher. The reason for the difference is that their classifier had more information from the use of additional sensors and the time history of the signal encoded in the 8 windows extracted.

## 5.3. Limitations

The limitations associated with the work in this thesis are:

- The analysis was done on healthy subject gait. The gait patterns when wearing a prosthesis are different when compared to normal gait, this could have an effect on the error rates. Additionally, the results were for data collected in a controlled environment. Error rates need to be investigated in an uncontrolled environment.

- Incorrect labelling was used for the static modes which gave imperfect models for the classifiers. This had an impact on the performance of the hybrid classifier.

- The analysis was done with filtered signals, in a real time implementation it would induce delays which is undesirable as it delays the decision from the classifier.

- The decision times were determined on a computer running at 2.3GHz. The decision times would be longer on the actual micro controller because of a lower processing speed.

## 5.4. Future Work

The scope of future work involves testing out the hybrid classifier architecture on the powered knee prosthesis with models trained with amputee gait, the availability of a strain gauge on the prosthesis that measures the sagittal plane ground reaction forces and moments, motor currents and knee torque would give the classifier more useful information and therefore lower error rates can be achieved especially for the static classifier.
The fully probabilistic framework with RVM based dynamic classifier needs further investigation. There is also the possibility of using Platt Scaling[62] to obtain posterior probabilities from the SVM models allowing the use of the hybrid classifier with the probabilistic framework, the performance can then be compared to RVM and the best option can be chosen.

A smarter intent recognition system that is able to detect less probable modes like obstacle avoidance and backwards walking can also be developed by adding layers in parallel to the existing framework. From a safety point of view, stumble detection is an important event that the system should be able to recognize and using a classifier based on frequency domain features would be able to find any such event. A commonly occurring mode in daily activities that wasn't analyzed in this thesis is turning, a separate classifier can be used for this and only the IMU signals should suffice to distinguish level walking versus turning. The proposed classifier architecture in this study would serve as the base for the development of a smart intent recognition system. This would help in reducing the cognitive burden for the user by smoothly and effortlessly transitioning between activity modes.

# A

# MATLAB Code

## A.1. Feature Extraction

```
% Written by Manav Penubaku
% Code to extract features from processed signals
clear all
clc
%% Extract features of a 300ms ,200ms & 100ms window from processed data
window_size=[50,100,200,300];
fs=1000;
dt=1/fs;
g=-9.81;% Acceleration Due to Gravity

frame_len=window_size*fs/1000;

cd Z:\NutFolder\rbionics\HCTRL\SensorFusion\DATA\Rouse_Data_Set
f_sub=dir('AB*');% Find data folders of each subject

for n_win=1:length(frame_len)
    ns=frame_len(n_win);
    for n_1=1:length(f_sub)
        cd(strcat(f_sub(n_1).folder,'\',f_sub(n_1).name,'\Processed'));
        f_trial=dir('AB*');
        combined_dataset_hc=[];
        combined_dataset_to=[];
        for n_2=1:length(f_trial)
            data=csvread(f_trial(n_2).name,1,0);
            relevant_data=data(:,[1:6,46,50]);
            % Determine Absolute Orientation of Shank
            acc_data=relevant_data(:,1:3);
            gyr_data=relevant_data(:,4);
            alpha=atan2d(acc_data(:,2),sqrt(acc_data(:,1).^2+acc_data(:,3).^2));
            lambda=0.99;
            abs_or=zeros(length(alpha),1);
            for n=2:length(alpha)
                abs_or(n)=lambda*(abs_or(n-1)+gyr_data(n-1,1)*dt)+(1-lambda)*alpha(n);
            end

            % Correct Accelerometer readings using rotation matrices
            acc_data_corrected=zeros(size(acc_data));
            for n=1:length(abs_or)
                R_n=rotation_matrix(abs_or(n));
                acc_data_corrected(n,:)=R_n*(acc_data(n,:).'*g);
                acc_data_corrected(n,1)=acc_data_corrected(n,1)+g;
            end

            % Retreive indices of gait events
            hc_index=data(data(:,55)>1000,54);
            hc_triggers=num2str(data(data(:,55)>1000,55));
            to_index=data(data(:,57)>1000,56);
```

```
to_triggers=num2str(data(data(:,57)>1000,57));

% Initialize feature vectors
features_hc=NaN(length(hc_index),52);
features_to=NaN(length(to_index),50);

%% Calculate stride height and stride length for available indices
stride_height=[];
stride_length=[];

clear n_clip;
if length(hc_index)<length(to_index)
    n_clip=length(to_index)-length(hc_index);
    rel_ind=hc_index>to_index(1:end-n_clip);
    t_in=to_index(1:end-n_clip);
    h_in=hc_index;
    clear n_clip;
elseif length(hc_index)>length(to_index)
    n_clip=length(hc_index)-length(to_index);
    rel_ind=hc_index((n_clip+1):end)>to_index;
    t_in=to_index;
    h_in=hc_index((n_clip+1):end);
    stride_height(1:(n_clip+1))=NaN;
    stride_length(1:(n_clip+1))=NaN;
    if n_2==1 && n_1==7
        h_in=hc_index([1:2,5:end]);
        t_in=to_index(1:end-1);
        rel_ind=logical(ones(length(t_in),1));
    end
else
    rel_ind=hc_index>to_index;
    t_in=to_index;
    h_in=hc_index;
end
if (rel_ind==0)
    t_in=to_index(1:end-1);
    h_in=hc_index(2:end);
    rel_ind(:)=1;
    rel_ind(end)=[];
    stride_height(1)=NaN;
    stride_length(1)=NaN;
end
ind_list=[t_in(rel_ind) h_in(rel_ind)];

for n_3=1:length(ind_list)
    a_x_data=acc_data_corrected(ind_list(n_3,1):ind_list(n_3,2),1);
    a_z_data=acc_data_corrected(ind_list(n_3,1):ind_list(n_3,2),3);

    if exist('n_clip','var')
        stride_height(n_3+n_clip)=integrator(a_x_data,dt,1,2);
        stride_length(n_3+n_clip)=abs(integrator(a_z_data,dt,1,2));
    else
        stride_height(n_3+1)=integrator(a_x_data,dt,1,2);
        stride_length(n_3+1)=abs(integrator(a_z_data,dt,1,2));
    end
end

stride_height(stride_height==0)=[];
stride_length(stride_length==0)=[];
stride_height( rel_ind)=NaN;
stride_length( rel_ind)=NaN;


if n_2==1 && n_1==7
    stride_height=[stride_height(1:2) NaN NaN stride_height(3:end)];
    stride_length=[stride_length(1:2) NaN NaN stride_length(3:end)];
end

%% Write features
for n=1:length(hc_index)
    % Pull out data for window
```

```
                windowed_data=relevant_data((hc_index(n)-ns):(hc_index(n)-1),:);
                % Mean
                features_hc(n,1:8)=mean(windowed_data);
                % Standard Deviation
                features_hc(n,9:16)=std(windowed_data);
                % Minimum
                features_hc(n,17:24)=min(windowed_data);
                % Maximum
                features_hc(n,25:32)=max(windowed_data);
                %Initial Value
                features_hc(n,33:40)=windowed_data(1,:);
                % Final Value
                features_hc(n,41:48)=windowed_data(end,:);
                % Initial Value Absolute Orientation
                features_hc(n,49)=abs_or((hc_index(n)-ns),:);
                % Final Value Absolute Orientation
                features_hc(n,50)=abs_or((hc_index(n)-1),:);
                % Stride Height
                features_hc(n,51)=stride_height(n);
                % Stride Length
                features_hc(n,52)=stride_length(n);
                features_hc(n,53:55)=sum(acc_data_corrected((hc_index(n)-ns) .....
                :(hc_index(n)-1),:).^2);
                X_f=fft(acc_data_corrected((hc_index(n)-ns):(hc_index(n)-1),:));
                X_f=abs(X_f);
                S_XX=X_f.*X_f/ns;
                S_XX_i          =    S_XX./sum(S_XX);
                features_hc(n,56:58)=-sum(S_XX_i.*log(S_XX_i));
            end
            for n=1:length(to_index)
                windowed_data=relevant_data((to_index(n)-ns):(to_index(n)-1),:);
                features_to(n,1:8)=mean(windowed_data);
                features_to(n,9:16)=std(windowed_data);
                features_to(n,17:24)=min(windowed_data);
                features_to(n,25:32)=max(windowed_data);
                features_to(n,33:40)=windowed_data(1,:);
                features_to(n,41:48)=windowed_data(end,:);
                features_to(n,49)=abs_or((to_index(n)-ns),:);
                features_to(n,50)=abs_or((to_index(n)-1),:);
                features_to(n,51:53)=sum(acc_data_corrected((to_index(n)-ns): .....
                (to_index(n)-1),:).^2);
                X_f=fft(acc_data_corrected((to_index(n)-ns):(to_index(n)-1),:));
                X_f=abs(X_f);
                S_XX=X_f.*X_f/ns;
                S_XX_i          =    S_XX./sum(S_XX);
                features_to(n,54:56)=-sum(S_XX_i.*log(S_XX_i));
            end
            features_hc(:,59)=str2num(hc_triggers(:,3));
            features_hc(:,60)=str2num(hc_triggers(:,1));   %#ok<*ST2NM>
            features_to(:,57)=str2num(to_triggers(:,3));
            features_to(:,58)=str2num(to_triggers(:,1));
            combined_dataset_hc=[combined_dataset_hc;features_hc];
            combined_dataset_to=[combined_dataset_to;features_to];
        end
        rootpath='Z:\NutFolder\rbionics\HCTRL\SensorFusion\DATA\Classification_Data\';
        cd(strcat(rootpath,f_sub(n_1).name,'\Discrete'));
        filename=strcat('hc_',f_sub(n_1).name,'_',num2str(window_size(n_win)),'.csv');
        csvwrite(filename,combined_dataset_hc);
        filename=strcat('to_',f_sub(n_1).name,'_',num2str(window_size(n_win)),'.csv');
        csvwrite(filename,combined_dataset_to);
    end
end

function [sig_int]=integrator(sig,dt,flag_sum,order)
% flag_sum=1, if sum is wanted for time interval
% flag_sum=0, if discrete integrated signal required
% order,        derivative order of the signal
if flag_sum
    if order==2
        v_h=0;sig_int=0;
        for t_1=1:length(sig)
```

```
            sig_int=sig_int+v_h*dt;
            v_h=v_h+sig(t_1)*dt;
        end
    else
        sig_int=0;
        for t_1=1:length(sig)
            sig_int=sig_int+sig(t_1)*dt;
        end
    end
else
    if order==2
        v_h=zeros(length(sig),1);sig_int=zeros(length(sig),1);
        for t_1=2:length(sig)
            sig_int(t_1)=sig_int(t_1-1)+v_h(t_1-1)*dt;
            v_h(t_1)=v_h(t_1-1)+sig(t_1-1)*dt;
        end
    else
        sig_int=zeros(length(sig),1);
        for t_1=2:length(sig)
            sig_int(t_1)=sig_int(t_1-1)+sig(t_1-1)*dt;
        end
    end
end

end

function R=rotation_matrix(alpha)
    R=[cosd(alpha) 0 sind(alpha);
        0 1 0;
        -sind(alpha) 0 cosd(alpha)];
end
```

## A.2. Classifier Training & Testing

```
%% Subject Independent Performance Testing
% Classification Accuracy based on Leave One Out Cross Validation.
% Written by Manav Penubaku
% 20-06-2018
clear all

N_C=5; % Number of Classes

%% Setup classifier
% SVM Classifier
classifier_svm = templateSVM('KernelScale',10,'KernelFunction','gaussian','Standardize',1 );
% LDA Classifier
classifier_lda = templateDiscriminant('DiscrimType','linear');

% Multiclass classification method
code='onevsone';

classifier=classifier_svm;
%% Load Data
cd Z:\NutFolder\rbionics\HCTRL\SensorFusion\DATA\Classification_Data
f_sub=dir('AB*');% Find data folders of each subject
pool_data=cell(length(f_sub),1);
cd(strcat(f_sub(1).folder,'\',f_sub(1).name,'\Discrete'))
f_model=dir('hc*300.csv');
for n_model=1:size(f_model,1)
    % Write data for each model of the subjects into cell matrix
    for n_sub=1:size(f_sub,1)
        cd(strcat(f_sub(n_sub).folder,'\',f_sub(n_sub).name,'\Discrete'))
        f_model=dir('hc*300.csv');
        f_model(n_model).name
        data=csvread(f_model(n_model).name);
        n_features=size(data,2);
        data(data(:,n_features-1)==6,:)=[];
        if N_C==4
```

```
            data(data(:,n_features-1)==2,n_features-1)=1;
            data(data(:,n_features)==2,n_features)=1;
        elseif N_C==3
            data(data(:,n_features-1)==2,n_features-1)=1;
            data(data(:,n_features-1)==3,n_features-1)=1;
            data(data(:,n_features)==2,n_features)=1;
            data(data(:,n_features)==3,n_features)=1;
        end
        pool_data{n_sub}=data;
    end

    for n=1:size(pool_data,1)
        test_data=pool_data{n};
        % Remove test data from the pooled data set
        swap_data=pool_data;
        swap_data{n}=[];
        train_data=cell2mat(swap_data);

    %% Setup KNN classifier

        N_N=floor(sqrt(size(train_data,1)));% Number of neighbors

        if rem(N_N,N_C)==0
            N_N=N_N-1;
        end

        classifier_knn = templateKNN('NumNeighbors',N_N,'Standardize',1);
    %% Train Classifier
        intent_model=fitcecoc(train_data(:,1:48),train_data(:,n_features-1),.....
        'Learners',classifier,'Coding',code);
        save(strcat(f_sub(n).name,'_HC'),'intent_model');

        true_label=test_data(:,n_features-1);
        % Test Classifier
        pred_label=predict(intent_model,test_data(:,1:48));
        ss_tr=(test_data(:,n_features-1)==test_data(:,n_features));
        true_label_ss=true_label(ss_tr,1);
        true_label_tr=true_label( ss_tr,1);
        pred_label_ss=pred_label(ss_tr,1);
        pred_label_tr=pred_label( ss_tr,1);
        % Determine Steady State, Transition & Overall Accuracies
        ss_a(n)=length(find(true_label_ss==pred_label_ss))/length(true_label_ss)*100;
        tr_a(n)=length(find(true_label_tr==pred_label_tr))/length(true_label_tr)*100;
        ov_a(n)=length(find(true_label==pred_label))/length(true_label)*100;
    end
    acc_mu(n_model,:)=[mean(ss_a) mean(tr_a) mean(ov_a)];
    acc_sigma(n_model,:)=[std(ss_a) std(tr_a) std(ov_a)]./sqrt(length(ss_a));
end

%% Train and test subject independent gausssian mixture model
% Written by Manav Penubaku

%% Initialize parameters
% Reduced Dimension
d=2;

% Number of classes
N_c=3;

k=3;% Number of gaussians in mixture

% Gaussian Mixture Model settings
options=statset('MaxIter',10000);
cov='full';

%% Fit GMM on pooled datasets
cd Z:\NutFolder\rbionics\HCTRL\SensorFusion\DATA\Classification_Data
addpath C:\Users\Manav\Documents\Thesis\MATLAB\Classifiers
f_sub=dir('AB*');% Find data folders of each subject
pool_data=cell(length(f_sub)-1,1);
for n_sub=1:size(f_sub,1)
```

```
    cd(strcat(f_sub(n_sub).folder,'\',f_sub(n_sub).name,'\Continuous_300'))
    pool_data{n_sub}=xlsread('St_Si_LG.xlsx');
end

for i=1:size(pool_data,1)
    swap_data=pool_data;
    swap_data{i}=[];
    train_data=cell2mat(swap_data);

    test_data=cell2mat(pool_data(i));

    % Separate Feature Vectors from Labels
    X=train_data(:,1:22);
    Y=train_data(:,23);

    [X_reduced,weights]=lda_reduction(X,Y,d,0);

    % Determine minimum and range of feature vectors
    X_MIN=min(X_reduced);
    X_RANGE=range(X_reduced);

    % Normalize feature vector
    Xnorm_01=bsxfun(@rdivide,bsxfun(@minus,X_reduced,X_MIN),X_RANGE);
    Xnorm=2*Xnorm_01-1;

    Si_train=Xnorm(Y(:)==0,:);
    LW_train=Xnorm(Y(:)==1,:);
    St_train=Xnorm(Y(:)==6,:);

    myflag=true;
    while myflag
        try
            GMModels.St=fitgmdist(St_train,k,'Options',options,....
            'CovarianceType',cov,'ProbabilityTolerance',1e-6);
            myflag=false;
        end
    end

    myflag=true;
    while myflag
        try
            GMModels.Si=fitgmdist(Si_train,k,'Options',options,.....
            'CovarianceType',cov,'ProbabilityTolerance',1e-6);
            myflag=false;
        end
    end

    myflag=true;
    while myflag
        try
            GMModels.LW=fitgmdist(LW_train,k,'Options',options,.....
            'CovarianceType',cov,'ProbabilityTolerance',1e-6);
            myflag=false;
        end
    end
    GMModels.WEIGHTS=weights;
    GMModels.MIN=X_MIN;
    GMModels.RANGE=X_RANGE;


    Y_test=train_data(:,23);

    X_test=train_data(:,1:22);
    X_test_reduced=X_test*GMModels.WEIGHTS;
    Xnorm_test=2*bsxfun(@rdivide,bsxfun(@minus,X_test_reduced,X_MIN),X_RANGE)-1;

    probs_Si=pdf(GMModels.Si,Xnorm_test);
    probs_St=pdf(GMModels.St,Xnorm_test);
    probs_LW=pdf(GMModels.LW,Xnorm_test);

    probs_all=[probs_Si probs_LW probs_St];
```

```
    ranges=range(probs_all);
    GMModels.GMRange=ranges;

    ST_MODES=[0,1,6];

    pred_label=sum((probs_all==max(probs_all.').').*ST_MODES,2);

    acc=sum(pred_label==Y_test)./sum(Y_test==Y_test)

    [n,p] = size(X_test);
    isLabels = unique(Y_test);
    nLabels = numel(isLabels);

    % Convert the integer label vector to a class-identifier matrix.
    [ ,grpOOF] = ismember(pred_label,isLabels);
    predLabelMat = zeros(nLabels,n);
    idxLinear = sub2ind([nLabels n],grpOOF,(1:n)');
    predLabelMat(idxLinear) = 1; % Flags the row corresponding to the class
    [ ,grpY] = ismember(Y_test,isLabels);
    YMat = zeros(nLabels,n);
    idxLinearY = sub2ind([nLabels n],grpY,(1:n)');
    YMat(idxLinearY) = 1;

    figure;
    plotconfusion(YMat,predLabelMat);
    h = gca;

    label_names=cell(3,1);
    label_names{1}='Sitting';
    label_names{2}='Level Ground Walking';
    label_names{3}='Standing';

    h.XTickLabel = [label_names; {''}];
    h.YTickLabel = [label_names; {''}];


    save(strcat(f_sub(i).name,'_GMM'),'GMModels')
end
```

## A.3. Hybrid Classifier Implementation

```
%% Code to test Hybrid Classifier Architecture
% Written by Manav Penubaku 09-07-2018
clear all

%% Initialize Parameters
Fs=1000;              % Sampling Frequency
dt=1/Fs;              % Time Step
WS=300;               % Window Size=300ms
del_T=1;             % Interval=10ms;
FL=WS/1000*Fs;        % Frame Length
FS=del_T/1000*Fs;     % Frame Interval

g=-9.81;

VL=500;                % Voting Length for Static Classifier

N_C=3;                % Number of Locomotion Modes..

% Labels for Modes
DYN_MODES=[1 2 3 4 5];
ST_MODES=[0;1;6];

% Transition Probabilities for Static Classifier
TP_Si=[0.4 0 0;
       0 0.2 0;
       0 0 0.4];
```

```
TP_St=[1/3 0 0;
       0 1/3 0;
       0 0 1/3];

TP_LW=[0.02 0 0;
       0 0.6 0;
       0 0 0.38];

%% Load Folders with Data and Models

cd Z:\NutFolder\rbionics\HCTRL\SensorFusion\DATA\Rouse_Data_Set
addpath C:\Users\Manav\Documents\Thesis\MATLAB\Classifier_Models\SI\3-Modes_AF
% Subject List
sub_list=dir('AB*');

DYN_PC=zeros(length(sub_list),4);
ST_PC=zeros(length(sub_list),2);

%% Leave One Out Cross Validation Performance
for n_sub=1:length(sub_list)
    % Load in Classifier Models
    n_sub
    load(strcat(sub_list(n_sub).name,'_GMM'));
    hc_model=load(strcat(sub_list(n_sub).name,'_HC'));
    to_model=load(strcat(sub_list(n_sub).name,'_TO'));

    % Access data of test subject
    cd(sub_list(n_sub).folder)
    cd(strcat(sub_list(n_sub).name,'\Processed'));
    trials=dir('*.csv');

    % Initialize Performance Metrics
    DYN_ERR=zeros(length(trials),3);
    DYN_DT=zeros(length(trials),1);
    ST_ERR=zeros(length(trials),1);
    ST_DT=zeros(length(trials),1);

    flag_test=0;
    acc_weights=zeros(length(trials),3);

    for n_trial=24:length(trials)
        pred_vec_HC=[];
        pred_vec_TO=[];
        pred_vec_GMM=[];
        dec_time_HC=[];
        dec_time_TO=[];
        dec_time_GMM=[];
        true_vec_GMM=[];
        pred_gmm_com=1;

        data=csvread(trials(n_trial).name,1,0);

        % Relevant fields in dataset
        r_data=data(:,[1,2,3,4,5,6,46,50]);
        data_vec=zeros(FL,size(r_data,2));

        % Gait Events and Triggers
        hc_indices=data(:,54)-1;
        hc_indices(hc_indices==-1)=[];
        hc_triggers=data(:,55);
        hc_triggers(hc_triggers==0)=[];
        [r1, ]=find(num2str(hc_triggers)==' ');% Find incomplete triggers
        hc_triggers(r1,:)=[];
        hc_indices(r1,:)=[];

        to_indices=data(:,56)-1;
        to_indices(to_indices==-1)=[];
        to_triggers=data(:,57);
        to_triggers(to_triggers==0)=[];
        [r2, ]=find(num2str(to_triggers)==' ');% Find incomplete triggers
```

```
to_triggers(r2,:)=[];
to_indices(r2,:)=[];

% Relabel data correctly
thres=20;
hc_triggers=num2str(hc_triggers);
hc_PM=str2num(hc_triggers(:,1));
hc_CM=str2num(hc_triggers(:,3));
to_triggers=num2str(to_triggers);
to_PM=str2num(to_triggers(:,1));
to_CM=str2num(to_triggers(:,3));

real_modes=data(:,53);
real_modes(real_modes==1)=NaN;
real_modes(real_modes==2)=NaN;
real_modes(real_modes==3)=NaN;
real_modes(real_modes==4)=NaN;
real_modes(real_modes==5)=NaN;

real_modes([hc_indices;to_indices])=[hc_CM;to_CM];
real_modes=fillmissing(real_modes,'previous');

ind=find(-data(:,46)<thres);

real_modes(1:ind(1))=0;

if range(ind)>4000
    real_modes(ind(end):end)=0;
end

St_LG=min([data(data(:,57)==6112,56) data(data(:,61)==6112,60)]);
LG_St=min([data(data(:,55)==1361,54) data(data(:,59)==1361,58)]);
if  isempty(LG_St)
    real_modes(LG_St:ind(end))=6;
end
if  isempty(St_LG)
    real_modes(ind(1):St_LG)=6;
end

true_labels=real_modes;

% Initialize iteration variables
n=1;
counter=1;
safe_count=50;
safe_count_lim=50;

% Define voting vector for static classifier
VV=NaN(1,VL);
v_counter=0;
flag_LW=0;
probs_all=[];
probs_all_dec=[];
flag_LW=0;

% Setup variables for determining absolute orientation
abs_or=zeros(length(data),1);
alpha=zeros(length(data),1);
lambda=0.99;

% Setup variables for determining stride features
acc_data_corrected=zeros(length(data),3);
flag_HC=0;
flag_TO=0;
event_count=1;
v_h=0;
v_l=0;
s_h=zeros(1,2);
s_l=zeros(1,2);

while n =length(data)
```

```
data_vec(counter,:)=r_data(n,:);

% Estimate orientation for current time step
alpha(n)=atan2d(r_data(n,2),sqrt(r_data(n,1).^2+r_data(n,3).^2));
if n==1
    abs_or(n)=0;
else
    abs_or(n)=lambda*(abs_or(n-1)+r_data(n-1,4)*dt)+(1-lambda)*alpha(n);
end

% Correct accelerations using orientation
R_n=rotation_matrix(abs_or(n));
acc_data_corrected(n,1:3)=R_n*(r_data(n,1:3).'*g);
acc_data_corrected(n,1)=acc_data_corrected(n,1)+g;

% Determine stride height and length
if sum(n==to_indices)==1
    flag_TO=1;
elseif sum(n==hc_indices)==1
    flag_HC=1;
end

if flag_TO==1
   s_h(event_count)=s_h(event_count)+v_h*dt;
   v_h=v_h+acc_data_corrected(n,1)*dt;
   s_l(event_count)=s_l(event_count)+v_l*dt;
   v_l=v_l+acc_data_corrected(n,3)*dt;

   if flag_HC==1
       flag_TO=0;
       flag_HC=0;
       v_h=0;
       v_l=0;
       event_count=event_count+1;
       s_h(event_count)=0;
       s_l(event_count)=0;
   end
end

% Dynamic Mode Classifier
if sum(n==hc_indices)==1
    tic
    FV_HC(1:8)=mean(data_vec);
    FV_HC(9:16)=std(data_vec);
    FV_HC(17:24)=min(data_vec);
    FV_HC(25:32)=max(data_vec);
    FV_HC(33:40)=data_vec(1,:);
    FV_HC(41:48)=data_vec(end,:);
    FV_HC(49)=abs_or(n-FL+1);
    FV_HC(50)=abs_or(n);
    FV_HC(51)=s_h(end-1);
    FV_HC(52)=abs(s_l(end-1));
    FV_HC(53:55)=sum(acc_data_corrected((n-FL+1):n,:).^2);
    X_f=fft(acc_data_corrected((n-FL+1):n,:));
    X_f=abs(X_f);
    S_XX=X_f.*X_f/FL;
    S_XX_i= S_XX./sum(S_XX);
    FV_HC(56:58)=-sum(S_XX_i.*log(S_XX_i));
    pred_HC=predict(hc_model.intent_model,FV_HC);
    t_HC=toc;

    if pred_HC==1
        flag_LW=1;
    else
        flag_LW=0;
    end

    pred_vec_HC=[pred_vec_HC;pred_HC];
    dec_time_HC=[dec_time_HC;t_HC];
```

```
elseif sum(n==to_indices)==1
    tic
    FV_TO(1:8)=mean(data_vec);
    FV_TO(9:16)=std(data_vec);
    FV_TO(17:24)=min(data_vec);
    FV_TO(25:32)=max(data_vec);
    FV_TO(33:40)=data_vec(1,:);
    FV_TO(41:48)=data_vec(end,:);
    FV_TO(49)=abs_or(n-FL+1);
    FV_TO(50)=abs_or(n);
    FV_TO(51:53)=sum(acc_data_corrected((n-FL+1):n,:).^2);
    X_f=fft(acc_data_corrected((n-FL+1):n,:));
    X_f=abs(X_f);
    S_XX=X_f.*X_f/FL;
    S_XX_i= S_XX./sum(S_XX);
    FV_TO(54:56)=-sum(S_XX_i.*log(S_XX_i));
    pred_TO=predict(to_model.intent_model,FV_TO);
    t_TO=toc;

    if pred_TO==1
        flag_LW=1;
    else
        flag_LW=0;
    end

    pred_vec_TO=[pred_vec_TO;pred_TO];
    dec_time_TO=[dec_time_TO;t_TO];
end

% Static Mode Classifier
if counter==FL
    FV_GMM(1:8)=mean(data_vec);
    FV_GMM(9:16)=std(data_vec);
    FV_GMM(17:19)=sum(data_vec(:,1:3).^2);%Energy In Accelerometer Signal
    X_f=fft(data_vec(:,1:3));
    X_f=abs(X_f);
    S_XX=X_f.*X_f/FL;
    S_XX_i= S_XX./sum(S_XX);
    FV_GMM(20:22)=-sum(S_XX_i.*log(S_XX_i));
    X_GMM=FV_GMM*GMModels.WEIGHTS;
    X_GMM=2*bsxfun(@rdivide,bsxfun(@minus,X_GMM,GMModels.MIN),....
    GMModels.RANGE)-1;

    prob_Si=pdf(GMModels.Si,X_GMM);
    prob_LW=pdf(GMModels.LW,X_GMM);
    prob_St=pdf(GMModels.St,X_GMM);
    probs=[prob_Si;prob_LW;prob_St];
    if isempty(probs_all)
        probs=probs./std(probs_all.').';
    end
    probs_all(:,end+1)=probs;

    if isempty(pred_vec_GMM)
        if pred_vec_GMM(end)==0
            probs=TP_Si*probs;
        elseif pred_vec_GMM(end)==1
            probs=TP_LW*probs;
        else
            probs=TP_St*probs;
        end
    end

    probs_all_dec(:,end+1)=probs;

    if max(probs)==0
        VV(v_counter+1)=NaN;
        disp('No probabilities')
    else
        VV(v_counter+1)=ST_MODES(probs==max(probs));
    end
    v_counter=v_counter+1;
```

```
                % Start timer
                tic
                if v_counter==VL
                    preds=unique(VV);
                    freqs=zeros(size(preds));
                    for j=1:length(preds)
                        freqs(j)=sum(VV==preds(j));
                    end

                    % Make decisions based on frequency of predictions
                    if max(freqs)>=0.8*VL && safe_count>=safe_count_lim && flag_LW==0
                        pred_vec_GMM(end+1)=preds(freqs==max(freqs));
                        true_vec_GMM(end+1)=true_labels(n-VL-(FL/2)+1);
                        safe_count=0;
                    elseif isempty(pred_vec_GMM)
                        pred_vec_GMM(end+1)=preds(freqs==max(freqs));
                        true_vec_GMM(end+1)=true_labels(n-VL-(FL/2)+1);
                    elseif n>=max([hc_indices;to_indices])
                        if max(freqs)>=0.8*VL
                            pred_vec_GMM(end+1)=preds(freqs==max(freqs));
                            true_vec_GMM(end+1)=true_labels(n-VL-(FL/2)+1);
                        else
                            pred_vec_GMM(end+1)=pred_vec_GMM(end);
                            true_vec_GMM(end+1)=true_labels(n-VL-(FL/2)+1);
                            safe_count=safe_count+1;
                        end
                    elseif flag_LW==1
                        pred_vec_GMM(end+1)=1;
                        true_vec_GMM(end+1)=true_labels(n-VL-(FL/2)+1);
                        safe_count=safe_count+1;
                    else
                        pred_vec_GMM(end+1)=pred_vec_GMM(end);
                        true_vec_GMM(end+1)=true_labels(n-VL-(FL/2)+1);
                        safe_count=safe_count+1;
                    end
                    t_GMM=toc;

                    dec_time_GMM=[dec_time_GMM;t_GMM];
                    v_counter=v_counter-1;
                    VV(1:end-1)=VV(2:end);
                end

                counter=counter-FS;
                data_vec(1:(FL-FS),:)=data_vec((FS+1):end,:);
            end

            counter=counter+1;
            n=n+1;
    end

    % Classification Errors for Dynamic Classifier

    if N_C==3
        hc_PM(hc_PM==2)=1;
        hc_PM(hc_PM==3)=1;
        hc_CM(hc_CM==2)=1;
        hc_CM(hc_CM==3)=1;
        to_PM(to_PM==2)=1;
        to_PM(to_PM==3)=1;
        to_CM(to_CM==2)=1;
        to_CM(to_CM==3)=1;
    end

    CM=[hc_CM;to_CM];
    PM=[hc_PM;to_PM];

    dyn_preds=[pred_vec_HC;pred_vec_TO];

    % Determine relevant predictions(exclude standing) for dynamic classifier
    rel_preds_hc=(sum(hc_PM==DYN_MODES,2)) & (sum(hc_CM==DYN_MODES,2));
```

```
rel_preds_to=(sum(to_PM==DYN_MODES,2)) & (sum(to_CM==DYN_MODES,2));

rel_preds=(sum(PM==DYN_MODES,2)) & (sum(CM==DYN_MODES,2));

events=[hc_indices;to_indices];
events=events(rel_preds);
events=[events dyn_preds(rel_preds) CM(rel_preds) PM(rel_preds)];
[ ,I]=sort(events(:,1));

pred_vec_HC( rel_preds_hc,:)=[];
hc_PM( rel_preds_hc,:)=[];
hc_CM( rel_preds_hc,:)=[];

pred_vec_TO( rel_preds_to,:)=[];
to_PM( rel_preds_to,:)=[];
to_CM( rel_preds_to,:)=[];

dyn_preds( rel_preds,:)=[];
PM( rel_preds,:)=[];
CM( rel_preds,:)=[];

ss=sum(dyn_preds(CM==PM) ==CM(CM==PM))./sum(CM==PM);
tr=sum(dyn_preds(CM ~=PM) ==CM(CM ~=PM))./sum(CM ~=PM);
ov=sum(dyn_preds ==CM)./sum(CM==CM);


if tr ==0
    flag_test=1;
end

DYN_ERR(n_trial,:)=[ss tr ov];

acc_weights(n_trial,:)=[sum(CM==PM) sum(CM ~=PM) sum(CM==CM)];

% Decision Times Dynamic Classifier
DYN_DT(n_trial)=mean([dec_time_HC;dec_time_TO])*1000;

true_profile=true_labels(1:end-FL-VL+1);
true_profile_3=true_profile;
true_profile_3(logical(sum((true_profile_3==[2 3]).'')))=1;
pred_profile=NaN(size(pred_vec_GMM));
pred_profile(logical(sum(pred_vec_GMM==[0;6]))) .....
=pred_vec_GMM(logical(sum(pred_vec_GMM==[0;6])));
pred_profile(events(I,1))=events(I,2);
pred_profile=fillmissing(pred_profile,'previous');

% Debugging probabilities
if flag_test==1
    figure(1)
    subplot(3,1,1)
    plot(probs_all_dec(1,:))
    subplot(3,1,2)
    plot(probs_all_dec(2,:))
    subplot(3,1,3)
    plot(probs_all_dec(3,:))

    figure(2)
    ax1=subplot(2,1,1);
    plot(true_profile,'--','LineWidth',2)
    grid on
    hold on
    plot(true_profile_3,'--','LineWidth',2)
    plot(pred_profile,'LineWidth',2);
    title(strcat('SUBJECT:',num2str(n_sub),',TRIAL:',num2str(n_trial)))
    hold off
    ax2=subplot(2,1,2);
    plot(-r_data(:,7))
    linkaxes([ax1,ax2],'x')
    flag_test=0;
end
```

```
        % Classification Error Static Classifier

        pred_vec_GMM( sum(true_vec_GMM==ST_MODES))=[];
        true_vec_GMM( sum(true_vec_GMM==ST_MODES))=[];
        ST_ERR(n_trial)=sum(pred_vec_GMM ~=true_vec_GMM)./sum(true_vec_GMM==true_vec_GMM);
        ST_DT(n_trial)=mean(dec_time_GMM)*1000;

    end
    DYN_ERR(isnan(DYN_ERR(:,2)),2)=0;
    DYN_ERR_WM=sum(DYN_ERR.*acc_weights)./sum(acc_weights);
    DYN_PC(n_sub,:)=[DYN_ERR_WM mean(DYN_DT)];
    ST_PC(n_sub,:)=mean([ST_ERR ST_DT]);
end

function R=rotation_matrix(alpha)
    R=[cosd(alpha) 0 sind(alpha);
        0 1 0;
        -sind(alpha) 0 cosd(alpha)];
end
```

# B

# Literature Review

## 1. Introduction

Wearable robotics for lower extremity are gaining increased attention as they are capable of restoring and augmenting human gait. These devices are controlled using sophisticated control architectures. Tucker *et al.* [10] in their review article propose a generalized framework for one such control architecture, the hierarchical controller. They suggest a three layer structure: a high level perception layer that classifies the intent of the user, a middle level translation layer that maps intentions and state inputs to state outputs and a low level execution layer that carries out the device specific control through feed-forward and feedback mechanisms. Sensors are vital for every layer of the controller, wearable robots are usually equipped with multiple sensors that include Electromyography(EMG) sensors, Inertial Measurement Units(IMU), joint position and velocity encoders, load cells, motor current and joint torque encoders to mention a few.

This report contains a review of the state of the art research in high level intent recognition for lower extremity wearable robotics. The main focus area was intent recognition in prostheses: trans-femoral and trans-tibial as this review is intended to aid in a project that involves developing an intent recognition system for an active trans-femoral prosthesis. Intent recognition algorithms can either be based on heuristic rules or machine learning. The intents of interest may include locomotion modes like Level Ground Walking, Stair Ascent, Stair Descent, Ramp Ascent,Ramp Descent, Cycling, Sitting, Standing, Bending, Backwards Walking amd Obstacle Avoidance.

Heuristic rule based algorithms makes use of thresholds/rules for the sensor signals which are set based on knowledge of gait kinematics and kinetics to determine the intent. The rules are simple and are intuitive for the user. The main drawback of these algorithms is the generalizability as it is difficult to come with a universal set of rules for intent recognition. Also, the number of rules increases with the number of modes of interest which would be a cognitive burden for the user.

Machine learning algorithms on the other hand are capable of generalizing well and the user cognitive burden is low as they don't have to be aware of rules, the algorithm uses a model based on data from all the sensors to make decisions. Supervised learning is a machine learning method that is widely used for intent recognition, the process flow for supervised learning is shown in Figure B.1. The method requires input data that has associated discrete labels. These labels are usually the class names encoded as numbers. The data is used to train a model of the task,the model is then used to make predictions for new data. The model can only output the discrete labels it has seen during training. The main drawbacks of this method are the need for training which can be time consuming for some algorithms and computation times based on how the trained model is used to make predictions. Given the drawbacks, there is a lot of scope in improving algorithms that are based on machine learning. The main focus area for this review was thus chosen as supervised learning methods for intent recognition.

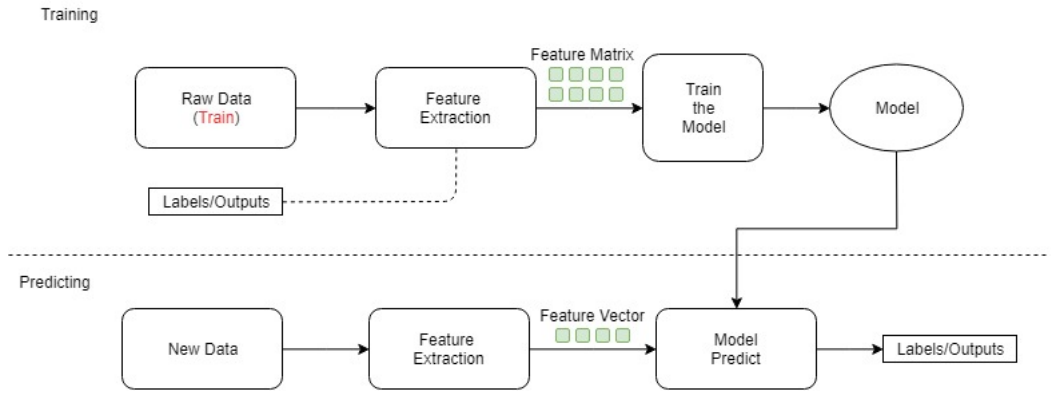



Figure B.1: Block Diagram for Supervised Machine Learning

In addition to the main focus area, a short review was also conducted into sensor fusion for IMUs, which encompasses filtering algorithms. IMUs have 3 sets of 3 axis sensors:accelerometers, gyrometers and magnetometers. IMU signals have a range of issues due to which sensor fusion is a necessary, the gyrometer is subject to drift when integrating to find the angular position, computing position from the accelerometer by double integration can amplify the noise in the signal giving unreliable estimates, the magnetometer can have sudden disturbances due to the presence of ferromagnetic materials like the actuator coil in the vicinity.The signals from the 3 sets of sensors need to be combined through the use of filtering algorithms to get an accurate estimate of the pose and position of the prosthesis. The position and orientation estimates are useful for the intent recognition algorithm as well as for the low level control of the device. Many methods exist for filtering and therefore a review is needed to find the optimal method. The optimal method has a good trade-off between computation time and accuracy.

## 2. Methods

Scopus was used to obtain a repository of relevant literature, the custom keyword search for the main focus area was (”Transfemoral” OR ”Above Knee” OR ”Lower Limb”) AND (”Prosthesis” OR ”Prostheses”) AND (”Intent Recognition” OR ”Pattern Recognition” OR ”Intention Detection”). The search gave 102 results and the abstracts were screened after which 40 papers were chosen for an in depth analysis. Figure B.2 shows that the field is gaining increased interest in the past few years.

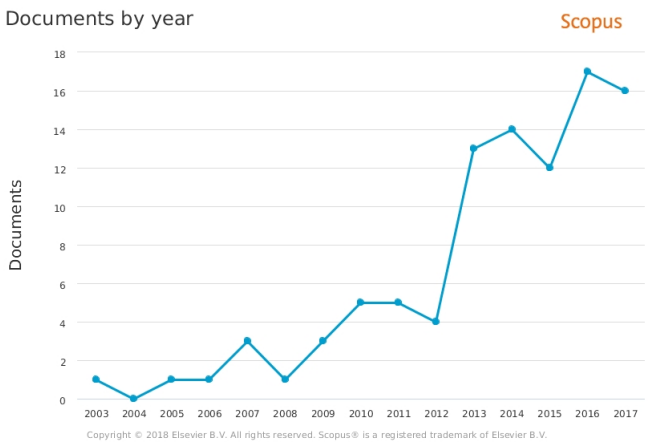



Figure B.2: Relevant publications 2000-2017

To augment the obtained research papers for the main focus area, a broad survey was done into classifying user intent using sensors mounted on the body, this is not directly linked to

wearable robotics but will aid in the project as the sensors used in these research articles are present in the prosthesis for which the algorithms will be developed.
For the are of IMU filtering, the search strings ”Human motion estimation using inertial measurement units”, ”Position and orientation estimation using inertial measurement units” and ”IMU sensor fusion” was used on Google Scholar. These search strings gave an overview of the algorithms used in general IMU filtering as well as algorithms specific to human motion estimation.
The results of the literature review are in the next section, it has two main sub sections each with a focus area, one being intent recognition and sensor fusion methods for IMUs. This is followed by a section containing some concluding remarks on open research areas.

# 3. Results and Discussion

## 3.1. Intent Recognition

The results for this area are split based on the flow of the intent recognition problem. Figure B.3 shows the different steps involved in intent recognition.[63]
It has five phases, a signal acquisition phase wherein the sensor signals are sampled, a fil-

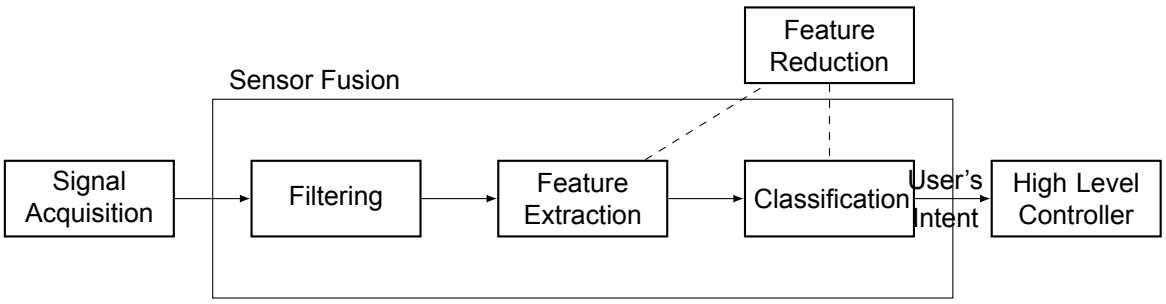



Figure B.3: Block Diagram for Intent Recognition

tering phase wherein noise in the signal is attenuated, a feature extraction phase wherein important characteristics of the filtered signal are determined, feature reduction is an optional phase wherein the number of input features is reduced, the next phase is the classification phase wherein a discrete label is assigned which is the output of the high level controller.

### Signal Acquisition

When looking at the literature, for the signal acquisition phase, it was observed that different sensors have been used by different research groups for the classification of human locomotion modes, each having their own advantages and disadvantages. For lower extremity wearable robotics, the common sensor frameworks and corresponding research articles are the following:

1. Electromyography(EMG) Sensors [14]–[18]
   The signals from EMG sensors are good for classification problems as muscle activation precedes the actual physical movement of the limb and therefore the decision can be made in the early phase of a certain mode which is usually taken earlier than decisions made by using solely mechanical sensor data. Some of the issues to consider with these signals are the noise levels and also the signal quality when surface EMG is used, which is affected by factors such temperature, humidity and skin conductance. EMG signals can also be distorted by magnetic fields when using a motorized prosthesis. These issues prevents the use of EMG sensors outside the lab. The noise levels can be dealt with by using appropriate filtering techniques, however methods to deal with improving signal quality of surface EMG needs further investigation. Intramuscular EMG signals are known to have better signal quality than surface EMG as the electrodes are directly inserted into the muscle with an invasive procedure. However, in an article by Hargrove *et al.* [64] that compares the accuracies of surface EMG based classification

algorithms to intramuscular EMG based classification algorithms for 10 motions of an arm prosthesis, both signal sources provide similar accuracies. No such comparison was found for lower limb prosthesis.

2. Mechanical Sensors
   A wide variety of mechanical sensors like IMUs[17], [19]–[28] goniometers[19], [27]–[31] ,capacitive sensors[34], [35] and pressure sensored insoles [32], [33] have been used for the classification problem. Mechanical sensors generally have a good signal to noise ratio (SNR), however delays are introduced as decisions can be made only after a certain movement has occurred and a certain mode is entered. The delays are not an issue, as long as the control action can be taken based on the classification decision without destabilising the movement.

3. Combination of EMG and Mechanical Sensors(Neuromechanical Sensor Fusion)[18], [36], [38], [40], [48], [49], [65], [66]
   Many researchers see this as the best sensor framework for classification problems, it is one way of dealing with the EMG signal quality issue. By fusing appropriate signals a fast and accurate classifier can be obtained. However there isn't any significant proof for it's superior efficiency. According to Tkach *et al.*[18], neuromechanical sensor fusion gives the best classification accuracy when predicting ambulation mode transitions in trans-tibial amputees. They looked at mode transitions between level walking, ascent and descent of both stairs and ramps, obtained error rates were 2.3% for the fused signal values ,7.8% and 20.2% for mechanical sensors and EMG sensors alone. This was only investigated for the Linear Discriminant Analysis(LDA) algorithm using time domain features. There are more sophisticated algorithms for classification which could yield higher accuracies and therefore neuromechanical sensor fusion can be avoided.

   The issues with using a combination of sensors is that more pre-processing is needed for the signals in terms of re-sampling, amplification and filtering. A larger amount of data is produced as a lot of features are extracted from the different sensors leading to longer training times for the supervised learning machinery and it is also computationally expensive to manipulate the feature vectors.

### Filtering

Acquired signals obtained have to be pre-processed by filtering. This is an important step when EMG signals are used as they have low frequency mechanical artefacts and high frequency aliasing effects, due to this it is common practice to band-pass filter the raw signal which removes all components of the signal except those in a defined band of frequencies. Reliable information for EMG signals is present between 20 to 500 Hz [67]. For the mechanical sensors, low pass filtering might be needed depending on the noise levels of the signal. Noise generally constitutes higher frequency content in the measured signals and can be dealt with either by using moving average filters or by low pass filtering the signal. In a real time implementation, filtering will induce a delay and this will have an effect on the decision times.

Although this a very minor stage of sensor fusion, it can have a large impact on classification accuracies as one can extract critical information within certain frequency bands which would help in separating the movement classes.

### Feature Extraction

All machine learning algorithms require as input features which are certain characteristics of the signal. Feature extraction involves extracting useful information from filtered signals, it reduces the amount of data that needs to be stored as this data is used to train the supervised learning algorithms. Features can range from simple ones like mean, standard deviation , maximum and minimum to more complex ones like spectral power, entropy and work done. It is important to select the right features as some features give better separation between classes which would improve the classification accuracy.

Feature extraction with time domain signals requires segmentation, in which the filtered signal is divided into windows with a defined length in time domain and features are extracted from these windows. The window size is a parameter that affects classification accuracy as longer windows contain more information about the current mode and as a result improves the accuracy. However using longer windows introduces controller delays which could destabilize the system, therefore a trade off exists between controller delay and classification accuracy. According to Smith *et al.* [68] for upper arm prostheses an optimal window length of 150 to 250ms is suggested and the work by Simon *et al.*[29] suggests that delaying ambulation mode transition decisions by 90ms does not have an effect on gait stability for an amputee walking with a trans-femoral prosthesis. Some amount of overlap can exist between windows as well which allows more frequent classifications which is sent to the middle and low level controllers and also there are fewer sudden changes in the output of the machine learning algorithm.

Zhang *et al.* [69] conducted an investigation into the timing to switch control for task transitions. They defined five states in a gait cycle: initial double support, single support, terminal double support,swing flexion and swing extension. They investigated switching control in these five stages for one full gait cycle before and after the prosthetic foot stepped on the upcoming terrain(ramp, stairs, level ground). A subjective feedback was used along with a quantified balance index which was the full body angular momentum to check for instabilities. The results show that there exists 3-4 gait phases for which switching the control mode permitted smooth transitions. They showed that for transitions from level ground walking the critical timing was the beginning of the swing phase so that appropriate flexion torque could be generated, for transitions to level walking the critical timing is the initial contact of the prosthetic foot. Therefore, the decisions need not always be taken at heel strike and toe off, this could have impact on the classification accuracy.

The method of the sliding window with a majority voting scheme where a group of consecutive decisions are weighed, is very popular among researchers as it gives the best results and it is robust to outliers in the sensor readings. Varol *et al.*[30] report different voting vector lengths of 55,45,65 and 60 for frame lengths of 50,100,200 & 400 ms respectively with frames being generated every 10ms. Interestingly ,the number of voting vectors doesn't scale with frame length.

#### Feature Reduction

An optional step after feature extraction is feature reduction. It is common practice to extract a large number of features from the data, however using such large feature vectors is computationally expensive for the classification algorithm as the number of parameters increases, in addition to that some classification algorithms suffer from the curse of dimensionality where the number of training points increases exponentially when the dimension of the feature space is high. It is also difficult to visualize feature spaces and analyze separation between classes when the dimension is greater than 3. A fix for this is to use feature reduction techniques like Principal Component Analysis(PCA)[22], [30], [37], [70] and Linear Discriminant Analysis(LDA)[37].

#### Classification

Classification is the process of assigning discrete labels (Level Ground Walking,Stair Ascent,Sitting etc.) to the extracted features. The label has to capture the intent of the user, control action is taken based on the output of the classifier which is sent to a low level controller that makes sure that the control signals are applied. Many different classification algorithms have been used for human movement classification, Table B.1 contains an overview of the classification algorithms and obtained overall accuracies for locomotion mode classification and those that have been implemented for trans-femoral prostheses are highlighted in green along with overall accuracies obtained for healthy subjects and amputees. The classification algorithms performance can be evaluated in two ways, subject dependent: in which the training and test data set is from the same subject, subject independent: in which the training and test data sets are from different subjects. There is not a lot of subject

independent analysis in the literature, it is only present in the two articles by Young *et al.* [49] and [71].

| Classifier | Reference | Accuracy | |
|---|---|---|---|
| | | Healthy Subjects | Amputees |
| Linear Discriminant Analysis(LDA) | [14], [19], [25], [39]–[41] | 87-98% | 91-99.5% |
| Naive Bayesian(NB) | [20], [39], [42] | 84-90% | |
| K-Nearest Neighbor(KNN) | [20], [39], [42], [43] | 87% | |
| Artificial Neural Networks(ANN) | [20], [25], [44]–[47] | 83-96% | 90-97.8% |
| Support Vector Machine(SVM) | [14], [15], [20], [24], [38], [42], [48] | 90-98% | 92.5-98.36% |
| Dynamic Bayesian Network(DBN) | [19], [20], [37], [41], [49] | 92-99.4% | 92-98% |
| Hidden Markov Model(HMM) | [20], [32] | 96-98.4% | 95.8% |
| Nearest Mean(NM) | [20] | 98.5% | |
| Binary Decision Trees(BDT) | [20], [42], [50], [51] | 81-93% | |
| Gaussian Mixture Model(GMM) | [28], [30], [31], [52] | 91.3-92.2% | - |

Table B.1: Overview of classification algorithms

Table B.2 contains a detailed overview of the classifier implementations for transfemoral prostheses using mechanical sensor signals only.

Analyzing Table B.1 leads to the conclusion that SVM, LDA and DBN are the most promising methods for classification. LDA is predominantly used for EMG signal features and for neuromechanical sensor fusion, it's performance with mechanical sensor data is not as good. SVM is a sparse kernel based technique and is a really good algorithm when the feature space has a high dimension, it outputs vectors from the training set that are vital for classification and the distance to these vectors is used to determine the classes for the training set. There is another sparse kernel based technique called Relevance Vector Machine(RVM) which uses the same framework as SVM but the parameters have a probability associated with them which can be used to formulate a predictive distribution that could improve the generalization of the algorithm.

HMM is a fast classifier that is suitable for sequential temporal data, a conditional distribution has to be determined for transitions between modes and these are assembled in a matrix which is used as a state matrix to update the state based on the observation, the accuracy of this method depends on how well the transitional probability matrix captures the reality. HMM is a special case of the DBN which assumes stationary signals over time and this can effect the accuracy and generalization of the algorithm. A more general DBN which incorporates time history and relaxes the assumption of stationarity does yield higher accuracies, this was done by Young *et al.*[19] where they had different models at eight points in the gait cycle and propagated the locomotion mode probabilities from one time point to the other, they obtained a steady state accuracy of 98% which is higher than the HMM equivalent which yields 95.2%.

Among the classification algorithms that haven't been used for activity mode recognition in prostheses, KNN and NM haven't been used as they suffer from the curse of dimensionality[72], where for larger dimension feature vectors the number of data points required increases exponentially to maintain the same level of performance of the classifier. BDT hasn't been used as there already exists heuristic rule-based decision trees where the rules are defined from existing knowledge about gait patterns. BDT is a method that comes up with rules based on the training data set and these rules are not intuitive, it is a bit of an overkill as there is not much improvement in the classification accuracy when compared to the heuristic rule-based classifier.

As stated before,the accuracies listed in Table B.1 represent the overall accuracy for the methods, however, it is important to separate the steady state accuracy from the transitional accuracy as this gives an insight into which transitions are difficult to detect and also into the robustness of the algorithms. Many of the research papers only have the overall accuracy,the exceptions are the research conducted by Woodward *et al.*[25], Spanias *et al.*[37] and Young *et al.*[27].

| Classifier (Reference) | [28] & [30] | [71] | [25] | [27] | [32] |
|---|---|---|---|---|---|
| Sensors | 7;Knee/ankle angle and angular velocity, interacting force and torque | 13; Knee/ankle angle, axial load cell and IMU on the shank, motor current to knee and ankle actuators | 18; Knee/ankle angle and angular velocity, 6-DOF load cell, IMU on the shank, motor current to knee & ankle actuators | 13; Knee/ankle angle and angular velocity,axial load, IMU on the shank, motor current to knee & ankle actuators. | Thigh gyroscope & Accelerometer |
| Window | 50/100/200/400 ms sliding window with 10ms increment | 300 ms preceding toe off and heel contact(LDA) and 8 300ms windows in the gait cycle(DBN) | 300 ms preceding toe off and heel strike | 50-450ms before toe off and heel strike(250 ms was found optimal) | 800 ms after heel contact |
| Selected Features | 2 per channel; Mean & Standard Deviation. Resulting 14 features reduced to 3 using PCA[18] & LDA[20] | 6; Mean, standard deviation, maximum and minimum, initial and final values. | 6; Mean, standard deviation, maximum and minimum, initial and final values | 4; Mean, standard deviation, maximum and minimum | Intra-class correlation coefficient of thigh gyroscope and accelerometer data, fused using Dempster-Shafer's theory. |
| Classifier | GMM with majority voting scheme | LDA & DBN (user-independent classification reported as well) | ANN(LDA was also used) , User independent classification reported as well. | LDA | HMM gave best accuracy |
| Delay of classification | 500-800ms[20] 430-605ms[18] | - | 15ms | - | - |
| Accuracy | 100%(when voting scheme is used) | 86%,84%(transition,LDA & DBN respectively) 98%,99%(steady state,LDA & DBN respectively) | ~99.5%(steady state) ~93.5%(transition) | ~96%(steady state) ~82%(transition) | 95.8%(93-99% steady state |
| Identified Activities | Walking, Standing & Sitting | Walking, Ramp & Stair Ascent/Descent | Walking, Ramp & Stair Ascent/Descent | Walking, Ramp & Stair Ascent/Descent | Walking, Ramp & Stair Ascent/Descent |
| Test-subjects | 1 Transfemoral Amputee(TFA) & 1 Healthy Subject | 8TFA | 6 TFA | 6 TFA | 3 healthy subjects, 2 TFA |

Table B.2: Detailed overview of classification algorithms using mechanical sensors

Woodward *et al.* report a steady state accuracy of 99.5% and a transitional accuracy of 93.5% for the ANN method when tested within-subject. They obtained a steady state accuracy of 98.9% and a transitional accuracy of 91.4% for the LDA method when tested subject-dependent. The performance of the algorithms deteriorated even further giving a transitional accuracy of 88.9% and 77.8% for ANN and LDA respectively when tested subject-independent where the training and testing was done for the pooled data of all subjects. This showcases that separating the accuracies is essential for a thorough investigation of the intent recognition algorithm.
Spanias *et al.*[37] report a steady state accuracy of 99% and a transitional accuracy of 98% when using their version of the DBN which is the same as used in Young *et al.*[19] but with a 90ms delayed decision. This accuracy is quite high and the method looks promising. They use an adaptive algorithm in this study, which maintains it's accuracy over experiments conducted over multiple days. However they do not mention if the algorithm can generalise between subjects, this is an area where adaptive algorithms can be beneficial.
Recently, there has been a focus on developing adaptive classification algorithms because they become less reliable with variation in input signals caused by physical changes at the sensor interface and human physiological changes. This issue is especially predominant in EMG based classification algorithms, [38] address the issue by using three different adaptive classification algorithms: entropy-based adaptation(EBA), LearnIng from Test data(LIFT) and Transductive Support Vector Machine(TSVM) and comparing their performance in an offline evaluation of data collected from two able bodied subjects and one trans-femoral amputee. They conclude that EBA gives the best performance and is computationally efficient however in the concluding remarks they state that the algorithm may require excessive data storage which is used for retraining and adapting the classifier when looking at timescales of weeks and months.

## 3.2. Sensor Fusion for IMUs

#### Orientation and Position Estimation

The most popular methods for IMU sensor fusion are the complementary filter, non linear extensions of the Kalman Filter(KF) such as the Extended Kalman Filter(EKF) and Unscented Kalman Filter(UKF). Attempts have been made to use particle filters [73][74] as they can generalize the representation of the probability density functions but this seems to be an overkill as it is the most computationally expensive fusion filter and it provides only a slight improvement in the accuracy.
Favre *et al.*[75] came up with an algorithm which estimates the orientation by using the readings from a 3D gyroscope and a magnetic tracker. They used two cascaded complementary kalman filters, the gyroscope bias was estimated using an open loop kalman filter when the magnetic tracker was not disturbed, periods of magnetic disturbance was identified using the gyroscopic and magnetic data. The 2nd filter is an extended kalman filter which is used to estimate the optimal orientation. Their algorithm does behave robustly when compared to the non corrected orientation estimate but the results are not compared to an optical tracking system estimate.
Schepers *et al.*[76] use an extended kalman filter for fusion of inertial(3D gyroscope, 3D accelerometer) and magnetic sensor readings which are taken with respect to a magnetic source containing 3 orthogonally placed circular coils. The position and orientation of the IMU is calculated relative to the magnetic source by integration of angular velocity and acceleration, they then use an EKF to determine the uncertainty associated with the position and orientation estimates and the magnetic source is actuated only if the uncertainty exceeds a predefined threshold, the axis along which most information is sent is chosen for actuation. They report an root mean square(RMS) error of $0.047 \pm 0.010$m for position and $3.6^\circ \pm 1^\circ$ for orientation. This algorithm performs well but the drawback is that the user needs to wear the magnetic source.
Madgwick *et al.*[77] came up with a computationally inexpensive method to determine the orientation. The method is also effective at low sampling rates of 10Hz. It has both magnetic distortion and gyroscope bias drift compensation. The performance was evaluated with re-

spect to an optical measurement system orientation estimate, the algorithm also gives lower RMS error than a Kalman based algorithm for orientation sensor. So far, for human motion estimation, EKF based algorithms have been widely used. UKF is known to have many advantages over EKF and it's use could give more reliable state estimates. Merwe[78] applied the UKF to problems such as state estimation, dual estimation and parameter estimation and they achieve a better level of accuracy than the EKF at a comparable level of complexity, it is also robust to initialization issues. The UKF has only been applied to upper limb motion estimation[79][80][81]. The algorithm is computationally expensive as reported by Rhudy *et al.*[82] and this could be a reason why it hasn't been applied to lower limb motion estimation yet.

Abdulrahim *et al.*[83] applied the zero velocity update(ZUPT) method to MEMs based pedestrian navigation. They detect the stance phase using an IMU placed on the foot. From gait knowledge it is known that during the stance phase the velocity value should be zero, however the velocity estimate obtained by integrating the accelerometer signal is non zero, this constitutes the residual velocity(RV) which captures the errors during this period. The RVs are used as a measurement update for a linear KF to better estimate the position, this framework was compared to the conventional Dead Reckoning which just sets the velocity value to zero during stationary periods. They reported that the KF+ZUPT framework gives the highest position tracking accuracy.

#### Joint Angle Estimation

IMUs can also be used to indirectly determine the joint angles, this is done in studies where healthy subject gait is of interest and goniometers are unavailable. Directly determining the orientation of the IMUs attached to the body segments can give an estimate of the joint angles, it is inaccurate as the axis of rotation may not pass through the actual joint axis.The work by Cooper *et al.*[84] aims to estimate the knee angle in the flexion-extension axis from signals obtained from IMUs attached to the shank and the thigh. They use KF to estimate the Euler angle orientation of the IMUs and then model the knee as a pure hinge joint from which the flexion-extension angle can be determined by assuming that the orientations with respect to the knee joint is known. The drawback of this method is that the user has to specify the rotation axis of the joint relative to the IMUs.

Seel *et al.*[54] proposed a method to use the gyroscope readings to optimise a cost function that would give the joint axis, this method works independent of mounting orientation of sensors and is quite robust. They also utilized the accelerometer readings to determine the joint position.The joint angle can be determined from the accelerometer readings which is noisy and by integrating the gyroscope readings with respect to the joint axis which has drift. The two can then be combined using a regular kalman filter, complementary filter or by a weighted function they mention in the paper. They compare their results with an optical marker based estimate of the joint angles for both human and prosthetic legs. Obtained RMS errors were 3.3° and 0.71° for the knee flexion angle of the contralateral and prosthetic leg respectively.

## 4. Conclusion

A literature review was done into the use supervised learning methods for intent recognition, which has been gaining increased interest over the past few years in a bid to develop smart devices that infer the intent of the user automatically without the user having to perform a manoeuvre or press a button to enter a certain mode. Signals obtained from different sensors like EMG, IMUs, goniometers etc. paired with different classification algorithms have been used for intent recognition but there remains many open research questions.
Firstly, the steady state and transitional performance algorithms like KNN and NB hasn't been reported in the literature. Secondly, a thorough investigation is needed into using different classification algorithms on neuromechanical signals and comparing it's performance to classification algorithms of mechanical signals to see if there is any benefit of incorporating EMG signals. Thirdly, the focus of most literature is subject dependent classification which would give a higher accuracy than subject independent classification algorithms. De-

veloping a subject independent classifier requires an investigation into hybrid classifiers that combine different algorithms, such a classifier could use the RVM that gives a probabilistic decision and is therefore capable of improved generalization. Fourthly, some classification algorithms haven't been tested on amputees, there could be deterioration in performance which would limit it's eventual application to the wearable robot. Fifthly, selected features could have more of an influence on the classification accuracy than the selected classifier, an investigation into accuracies obtained with advanced features like absolute orientation of the segments and stride features(step height and step length)for lower extremity intent recognition is a potential research area.

A short survey was done into estimating orientation and position of segments from IMUs, a accurate position and orientation estimate can be obtained by using non linear filter. EKF is widely used but the RMS error in tracking is higher compared to UKF. Open research areas are the use of the UKF for estimating joint angles for lower limb movements and comparing it's performance to other algorithms, it would be interesting to see if it is computationally worth it to implement it. Madgwick's algorithm gives better estimates for orientation compared to Kalman based algorithms. The algorithm also only has a few tunable parameters which makes it user friendly and a low computation load. These features make it a popular choice for embedded systems.

# Bibliography


[1] K. Ziegler-Graham, E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer, "Estimating the prevalence of limb loss in the united states: 2005 to 2050," *Archives of physical medicine and rehabilitation*, vol. 89, no. 3, pp. 422–429, 2008.

[2] P. F. Adams, G. E. Hendershot, and M. A. Marano, "Current estimates from the national health interview survey, 1996," 1999.

[3] T. R. Dillingham, L. E. Pezzin, and E. J. MacKenzie, "Limb amputation and limb deficiency: Epidemiology and recent trends in the united states," *Southern medical journal*, vol. 95, no. 8, pp. 875–884, 2002.

[4] *Biomechatronics | Transfemoral Powered Prostheses*. [Online]. Available: http://biomech.media.mit.edu/portfolio%7B%5C_%7Dpage/cseaknee/ (visited on 05/14/2018).

[5] H. Herr and A. Wilkenfeld, "User-adaptive control of a magnetorheological prosthetic knee," *Industrial Robot: An International Journal*, vol. 30, no. 1, pp. 42–55, 2003.

[6] E. J. Wolf, V. Q. Everding, A. L. Linberg, B. L. Schnall, J. M. Czerniecki, and J. M. Gambel, "Assessment of transfemoral amputees using c-leg and power knee for ascending and descending inclines and steps.," *Journal of Rehabilitation Research & Development*, vol. 49, no. 6, 2012.

[7] M. J. Highsmith, J. T. Kahle, D. J. Lura, A. L. Lewandowski, W. S. Quillen, and S. H. Kim, "Stair ascent and ramp gait training with the genium knee," *Technology & Innovation*, vol. 15, no. 4, pp. 349–358, 2014.

[8] E. D. Ledoux and M. Goldfarb, "Control and evaluation of a powered transfemoral prosthesis for stair ascent," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 7, pp. 917–924, 2017.

[9] E. J. Wolf, V. Q. Everding, A. A. Linberg, J. M. Czerniecki, and C. J. M. Gambel, "Comparison of the power knee and c-leg during step-up and sit-to-stand tasks," *Gait & posture*, vol. 38, no. 3, pp. 397–402, 2013.

[10] M. R. Tucker, J. Olivier, A. Pagel, H. Bleuler, M. Bouri, O. Lambercy, J. del R Millán, R. Riener, H. Vallery, and R. Gassert, "Control strategies for active lower extremity prosthetics and orthotics: A review," *Journal of neuroengineering and rehabilitation*, vol. 12, no. 1, p. 1, 2015.

[11] P. Kampas and R. Pawlik, *Orthopedic device comprising a joint*, US Patent 9,161,847, Oct. 2015.

[12] *PHASES OF THE GAIT CYCLE - Pediatric Rehabilitation - Academic library - free online college e textbooks*. [Online]. Available: https://ebrary.net/7410/health/phases%7B%5C_%7Dgait%7B%5C_%7Dcycle (visited on 05/14/2018).

[13] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design '96*, J. S. Gero and F. Sudweeks, Eds. Dordrecht: Springer Netherlands, 1996, pp. 151–170, ISBN: 978-94-009-0279-4. DOI: 10.1007/978-94-009-0279-4_9. [Online]. Available: https://doi.org/10.1007/978-94-009-0279-4_9.

[14] J. D. Miller, M. S. Beazer, and M. E. Hahn, "Myoelectric walking mode classification for transtibial amputees," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 10, pp. 2745–2750, 2013, ISSN: 00189294. DOI: 10.1109/TBME.2013.2264466.

[15] L. Chen, P. Yang, X. Xu, L. Zu, and X. Guo, "Above-knee Prosthesis Control Based on Posture Recognition by Support Vector Machine," *2008 IEEE Conference on Robotics, Automation and Mechatronics*, vol. 00, pp. 307–312, 2008. DOI: 10.1109/RAMECH.2008.4681522. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=%7B%5C&%7Darnumber=4681522%7B%5C&%7DcontentType=Conference+Publications%7B%5C&%7DsearchField=Search%7B%5C_%7DAll%7B%5C&%7DqueryText=.QT.movement+intent.QT..

[16] T.-Y. Zhang and Y.-B. Fan, "Motion recognition based on emg signals of residual limb in transfemoral amputee," vol. 31, 478–482 and 494, Dec. 2016.

[17] O. Mazumder, A. S. Kundu, P. K. Lenka, and S. Bhaumik, "Multi-channel Fusion Based Adaptive Gait Trajectory Generation Using Wearable Sensors," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 3-4, pp. 335–351, 2017, ISSN: 0921-0296. DOI: 10.1007/s10846-016-0436-y. [Online]. Available: http://link.springer.com/10.1007/s10846-016-0436-y.

[18] D. C. Tkach and L. J. Hargrove, "Neuromechanical sensor fusion yields highest accuracies in predicting ambulation mode transitions for trans-tibial amputees," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 3074–3077, 2013, ISSN: 1557170X. DOI: 10.1109/EMBC.2013.6610190.

[19] A. J. Young, A. M. Simon, and L. J. Hargrove, "A training method for locomotion mode prediction using powered lower limb prostheses," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 3, pp. 671–677, 2014, ISSN: 15344320. DOI: 10.1109/TNSRE.2013.2285101.

[20] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010, ISSN: 14248220. DOI: 10.3390/s100201154.

[21] D. Novak, P. Reberšek, S. M. M. De Rossi, M. Donati, J. Podobnik, T. Beravs, T. Lenzi, N. Vitiello, M. C. Carrozza, and M. Munih, "Automated detection of gait initiation and termination using wearable sensors," *Medical Engineering and Physics*, vol. 35, no. 12, pp. 1713–1720, 2013, ISSN: 13504533. DOI: 10.1016/j.medengphy.2013.07.003.

[22] L. Lei, Y. Peng, L. Zuojun, G. Yanli, and Z. Jun, "Leg amputees motion pattern recognition based on principal component analysis and BP network," *2013 25th Chinese Control and Decision Conference (CCDC)*, pp. 3802–3804, 2013. DOI: 10.1109/CCDC.2013.6561611. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6561611.

[23] C. Dixon and K. Tuyls, "Towards autonomous robotic systems: 16th annual conference, TAROS 2015 Liverpool, UK, September 8-10, 2015 proceedings," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9287, pp. 180–185, 2015, ISSN: 16113349. DOI: 10.1007/978-3-319-22416-9.

[24] J. Mai, Z. Zhang, and Q. Wang, "Intelligent Robotics and Applications," vol. 8102, pp. 280–289, 2013, ISSN: 03029743. DOI: 10.1007/978-3-642-40852-6. [Online]. Available: http://link.springer.com/10.1007/978-3-642-40852-6.

[25] R. B. Woodward, J. A. Spanias, and L. J. Hargrove, "User intent prediction with a scaled conjugate gradient trained artificial neural network for lower limb amputees using a powered prosthesis," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2016-Octob, pp. 6405–6408, 2016, ISSN: 1557170X. DOI: 10.1109/EMBC.2016.7592194.

[26] B. Chen, E. Zheng, and Q. Wang, "A locomotion intent prediction system based on multi-sensor fusion," *Sensors (Switzerland)*, vol. 14, no. 7, pp. 12 349–12 369, 2014, ISSN: 14248220. DOI: 10.3390/s140712349.

[27] A. J. Young, A. M. Simon, N. P. Fey, and L. J. Hargrove, "Intent recognition in a powered lower limb prosthesis using time history information," *Annals of Biomedical Engineering*, vol. 42, no. 3, pp. 631–641, 2014, ISSN: 15739686. DOI: 10.1007/s10439-013-0909-0.

[28] H. A. Varol, F. Sup, and M. Goldfarb, "Real-time gait mode intent recognition of a powered knee and ankle prosthesis for standing and walking," *Proceedings of the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2008*, pp. 66–72, 2008, ISSN: 2155-1774. DOI: 10.1109/BIOROB.2008.4762860.

[29] A. M. Simon, K. A. Ingraham, J. A. Spanias, A. J. Young, S. B. Finucane, E. G. Halsne, and L. J. Hargrove, "Delaying Ambulation Mode Transition Decisions Improves Accuracy of a Flexible Control System for Powered Knee-Ankle Prosthesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 8, pp. 1164–1171, 2017, ISSN: 15344320. DOI: 10.1109/TNSRE.2016.2613020.

[30] H. A. Varol, F. Sup, and M. Goldfarb, "Multiclass real-time intent recognition of a powered lower limb prosthesis," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 3, pp. 542–551, 2010, ISSN: 00189294. DOI: 10.1109/TBME.2009.2034734. arXiv: NIHMS150003.

[31] ——, "Powered sit-to-stand and assistive stand-to-sit framework for a powered transfemoral prosthesis," *2009 IEEE International Conference on Rehabilitation Robotics, ICORR 2009*, pp. 645–651, 2009, ISSN: 1945-7898. DOI: 10.1109/ICORR.2009.5209582. arXiv: NIHMS150003.

[32] Z. Liu, W. Lin, Y. Geng, and P. Yang, "Intent pattern recognition of lower-limb motion based on mechanical sensors," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 651–660, 2017, ISSN: 23299274. DOI: 10.1109/JAS.2017.7510619.

[33] S. Crea, M. Donati, S. M. M. De Rossi, C. Maria Oddo, and N. Vitiello, "A wireless flexible sensorized insole for gait analysis," *Sensors (Switzerland)*, vol. 14, no. 1, pp. 1073–1093, 2014, ISSN: 14248220. DOI: 10.3390/s140101073.

[34] E. Zheng, B. Chen, Q. Wang, K. Wei, and L. Wang, "A wearable capacitive sensing system with phase-dependent classifier for locomotion mode recognition," *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 1747–1752, 2012, ISSN: 21551774. DOI: 10.1109/BioRob.2012.6290721.

[35] E. Zheng, L. Wang, Y. Luo, K. Wei, and Q. Wang, "Non-contact capacitance sensing for continuous locomotion mode recognition: Design specifications and experiments with an amputee," *IEEE International Conference on Rehabilitation Robotics*, 2013, ISSN: 19457898. DOI: 10.1109/ICORR.2013.6650410.

[36] D. Wang, L. Du, and H. Huang, "Terrain Recognition Improves the Performance of Neural-Machine Interface for Locomotion Mode Recognition," *2013 International Conference on Computing, Networking and Communications (Icnc)*, pp. 87–91, 2013, ISSN: 2325-2626.

[37] J. Spanias, A. M. Simon, S. Finucane, E. Perreault, and L. Hargrove, "Online adaptive neural control of a robotic lower limb prosthesis," *Journal of Neural Engineering*, 2017, ISSN: 1741-2560. DOI: 10.1088/1741-2552/aa92a8. [Online]. Available: http://iopscience.iop.org/article/10.1088/1741-2552/aa92a8.

[38] M. Liu, F. Zhang, and H. H. Huang, "An adaptive classification strategy for reliable locomotion mode recognition," *Sensors (Switzerland)*, vol. 17, no. 9, 2017, ISSN: 14248220. DOI: 10.3390/s17092020.

[39] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *International Conference on Pervasive Computing*, Springer, 2004, pp. 1–17.

[40] L. Du, F. Zhang, M. Liu, and H. Huang, "Toward design of an environment-aware adaptive locomotion-mode-recognition system," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2716–2725, 2012, ISSN: 00189294. DOI: 10.1109/TBME.2012.2208641. arXiv: NIHMS150003.

[41] A. J. Young and L. J. Hargrove, "A classification method for user-independent intent recognition for transfemoral amputees using powered lower limb prostheses," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 2, pp. 217–225, 2016.

[42] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Aaai*, vol. 5, 2005, pp. 1541–1546.

[43] K.-T. Song and Y.-Q. Wang, "Remote activity monitoring of the elderly using a two-axis accelerometer," in *Proceedings of the CACS Automatic Control Conference*, 2005, pp. 18–19.

[44] K. Van Laerhoven and O. Cakmakci, "What shall we teach our pants?" In *Wearable Computers, The Fourth International Symposium on*, IEEE, 2000, pp. 77–83.

[45] J. Mantyjarvi, J. Himberg, and T. Seppanen, "Recognizing human motion with multiple acceleration sensors," in *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, IEEE, vol. 2, 2001, pp. 747–752.

[46] C. Randell and H. Muller, "Context awareness by analysing accelerometer data," in *Wearable Computers, The Fourth International Symposium on*, IEEE, 2000, pp. 175–176.

[47] M. Islam and E. T. Hsiao-Wecksler, "Detection of Gait Modes Using an Artificial Neural Network during Walking with a Powered Ankle-Foot Orthosis," *Journal of Biophysics*, vol. 2016, 2016, ISSN: 16878019. DOI: 10.1155/2016/7984157.

[48] F. Zhang, Z. Dou, M. Nunnery, and H. Huang, "Real-time implementation of an intent recognition system for artificial legs," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 2997–3000, 2011, ISSN: 1557170X. DOI: 10.1109/IEMBS.2011.6090822.

[49] A. J. Young, A. M. Simon, N. P. Fey, and L. J. Hargrove, "Classifying the intent of novel users during human locomotion using powered lower limb prostheses," *International IEEE/EMBS Conference on Neural Engineering, NER*, pp. 311–314, 2013, ISSN: 19483546. DOI: 10.1109/NER.2013.6695934.

[50] J. Bussmann, W. Martens, J. Tulen, F. Schasfoort, H. Van Den Berg-Emons, and H. Stam, "Measuring daily behavior using ambulatory accelerometry: The activity monitor," *Behavior Research Methods, Instruments, & Computers*, vol. 33, no. 3, pp. 349–356, 2001.

[51] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE transactions on information technology in biomedicine*, vol. 10, no. 1, pp. 156–167, 2006.

[52] F. R. Allen, E. Ambikairajah, N. H. Lovell, and B. G. Celler, "Classification of a known sequence of motions and postures from accelerometry data using adapted gaussian mixture models," *Physiological measurement*, vol. 27, no. 10, p. 935, 2006.

[53] B. Hu, E. Rouse, and L. Hargrove, "Benchmark datasets for bilateral lower-limb neuromechanical signals from wearable sensors during unassisted locomotion in able-bodied individuals," *Frontiers in Robotics and AI*, vol. 5, p. 14, 2018, ISSN: 2296-9144. DOI: 10.3389/frobt.2018.00014. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2018.00014.

[54] T. Seel, J. Raisch, and T. Schauer, "IMU-Based Joint Angle Measurement for Gait Analysis," *Sensors*, vol. 14, no. 4, pp. 6891–6909, 2014, ISSN: 1424-8220. DOI: 10.3390/s140406891. [Online]. Available: http://www.mdpi.com/1424-8220/14/4/6891/.

[55] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[56] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[57] M. Jiřina and M. Jiřina jr, "Classifier based on inverted indexes of neighbors," Technical Report No, Tech. Rep., 2008.

[58] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[59] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[60] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[61] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine learning*, vol. 45, no. 2, pp. 171–186, 2001.

[62] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[63] D. Novak and R. Riener, "A survey of sensor fusion methods in wearable robotics," *Robotics and Autonomous Systems*, vol. 73, pp. 155–170, 2015, ISSN: 09218890. DOI: 10.1016/j.robot.2014.08.012. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2014.08.012.

[64] L. J. Hargrove, K. Englehart, and B. Hudgins, "A comparison of surface and intramuscular myoelectric signal classification," *IEEE transactions on biomedical engineering*, vol. 54, no. 5, pp. 847–853, 2007.

[65] F. Zhang and H. Huang, "Source selection for real-time user intent recognition toward volitional control of artificial legs," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 5, pp. 907–914, 2013, ISSN: 21682194. DOI: 10.1109/JBHI.2012.2236563. arXiv: NIHMS150003.

[66] A. J. Young, A. Simon, and L. J. Hargrove, "An intent recognition strategy for transfemoral amputee ambulation across different locomotion modes," *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, vol. 2013, pp. 1587–1590, 2013, ISSN: 1557170X. DOI: 10.1109/EMBC.2013.6609818.

[67] M. S. Redfern, R. E. Hughes, and D. B. Chaffin, "High-pass filtering to remove electrocardiographic interference from torso emg recordings," *Clinical Biomechanics*, vol. 8, no. 1, pp. 44–48, 1993.

[68] L. H. Smith, L. J. Hargrove, B. A. Lock, and T. A. Kuiken, "Determining the optimal window length for pattern recognition-based myoelectric control: Balancing the competing effects of classification error and controller delay," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 2, pp. 186–192, 2011, ISSN: 15344320. DOI: 10.1109/TNSRE.2010.2100828.

[69] F. Zhang, M. Liu, and H. Huang, "Investigation of timing to switch control mode in powered knee prostheses during task transitions," *PLoS ONE*, vol. 10, no. 7, pp. 1–14, 2015, ISSN: 19326203. DOI: 10.1371/journal.pone.0133965.

[70] H. Vallery, E. H. Van Asseldonk, M. Buss, and H. Van Der Kooij, "Reference trajectory generation for rehabilitation robots: Complementary limb motion estimation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 17, no. 1, pp. 23–30, 2009, ISSN: 15344320. DOI: 10.1109/TNSRE.2008.2008278.

[71] A. J. Young and L. J. Hargrove, "A Classification Method for User-Independent Intent Recognition for Transfemoral Amputees Using Powered Lower Limb Prostheses," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 2, pp. 217–225, 2016, ISSN: 15344320. DOI: 10.1109/TNSRE.2015.2412461.

[72] E. Keogh and A. Mueen, "Curse of dimensionality," in *Encyclopedia of Machine Learning and Data Mining*, Springer, 2017, pp. 314–315.

[73] G. To and M. R. Mahfouz, "Quaternionic attitude estimation for robotic and human motion tracking using sequential monte carlo methods with von mises-fisher and nonuniform densities simulations," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 11, pp. 3046–3059, 2013.

[74] Z.-Q. Zhang and J.-K. Wu, "A novel hierarchical information fusion method for three-dimensional upper limb motion estimation," *IEEE transactions on instrumentation and measurement*, vol. 60, no. 11, pp. 3709–3719, 2011.

[75] J. Favre, J. Chardonnens, and K. Aminian, "An orientation measuring system suitable for routine uses made by the fusion of a 3D gyroscope and a magnetic tracker," *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, pp. 3938–3941, 2007, ISSN: 05891019. DOI: 10.1109/IEMBS.2007.4353195.

[76] H. M. Schepers, D. Roetenberg, and P. H. Veltink, "Ambulatory human motion tracking by fusion of inertial and magnetic sensing with adaptive actuation," *Medical and Biological Engineering and Computing*, vol. 48, no. 1, pp. 27–37, 2010, ISSN: 01400118. DOI: 10.1007/s11517-009-0562-9.

[77] S. O. H. Madgwick, A. J. L. Harrison, and A. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm.," *IEEE Int. Conf. Rehabil. Robot.*, vol. 2011, p. 5975346, 2011, ISSN: 1945-7901. DOI: 10.1109/ICORR.2011.5975346.

[78] R. van der Merwe, "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models," no. April, 2004.

[79] L. Peppoloni, A. Filippeschi, E. Ruffaldi, and C. A. Avizzano, "A novel 7 degrees of freedom model for upper limb kinematic reconstruction based on wearable sensors," in *Intelligent Systems and Informatics (SISY), 2013 IEEE 11th International Symposium on*, IEEE, 2013, pp. 105–110.

[80] Z.-Q. Zhang, W.-C. Wong, and J.-K. Wu, "Ubiquitous human upper-limb motion estimation using wearable sensors," *IEEE Transactions on Information technology in biomedicine*, vol. 15, no. 4, pp. 513–521, 2011.

[81] M. El-Gohary, L. Holmstrom, J. Huisinga, E. King, J. McNames, and F. Horak, "Upper limb joint angle tracking with inertial sensors," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, IEEE, 2011, pp. 5629–5632.

[82] M. Rhudy, Y. Gu, J. Gross, S. Gururajan, and M. R. Napolitano, "Sensitivity analysis of extended and unscented kalman filters for attitude estimation," *Journal of Aerospace Information Systems*, vol. 10, no. 3, pp. 131–143, 2013.

[83] K. Abdulrahim, T. Moore, C. Hide, and C. Hill, "Understanding the performance of zero velocity updates in mems-based pedestrian navigation," *International Journal of Advancements in Technology*, vol. 5, no. 2, pp. 53–60, 2014.

[84] G. Cooper, I. Sheret, L. McMillian, K. Siliverdis, N. Sha, D. Hodgins, L. Kenney, and D. Howard, "Inertial sensor-based knee flexion/extension angle estimation," *Journal of biomechanics*, vol. 42, no. 16, pp. 2678–2685, 2009.