A Machine Learning Approach for Next Step Prediction using On-Body Inertial Measurement Sensors

Bryan Alan Barrows

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Alan T. Asbeck

Alfred L. Wicks

Tomonari Furukawa

December 12, 2017

Blacksburg, Virginia

Keywords: Gait Analysis, KNN, Machine Learning, Prediction Model, Next Step Prediction, IMU

A Machine Learning Approach for Next Step Prediction using On-Body Inertial Measurement Sensors

Bryan Alan Barrows

ABSTRACT

This thesis presents the development and implementation of a machine learning prediction model for concurrently aggregating interval linear step distance predictions before future foot placement. Specifically, on-body inertial measurement units consisting of accelerometers, gyroscopes, and magnetometers, through integrated development by Xsens, are used for measuring human walking behavior in real-time. The data collection process involves measuring activity from two subject participants who travel an intended course consisting of flat, stair, and sloped walking elements. This work discusses the formulation of the ensemble machine learning prediction algorithm, real-time application design considerations, feature extraction and selection, and experimental testing under which this system performed several different test case conditions. It was found that the system was able to predict the linear step distances for 47.2% of 1060 steps within 7.6cm accuracy, 67.5% of 1060 steps within 15.2cm accuracy, and 75.8% of 1060 steps within 23cm. For separated flat walking, it was found that 93% of the 1060 steps have less than 25% error, and 75% of the 1060 steps have less than 10% error which is an improvement over the commingled data set. Future applications and work to expand upon from this system are discussed for improving the results discovered from this work.

A Machine Learning Approach for Next Step Prediction using On-Body Inertial Measurement Sensors

Bryan Alan Barrows

ABSTRACT (GENERAL AUDIENCE)

This thesis presents the development and implementation of a machine learning prediction model for determining the stepping distance of future steps in real-time walking before their placement occurs. Specialized sensor units for measuring human motion activity are worn on the body for collecting and characterizing human walking behavior in real-time. Two subject participants are asked to walk a planned course consisting of flat, stair, and sloped walking elements. This work discusses the prediction algorithm voting scheme, real-time application design considerations, descriptive data elements for the algorithm, and experimental testing under which this system performed several different test case conditions. Detailed experimental tests are concluded in order to fully understand the extent of the system's performance and the behaviors it exhibits throughout. The approach explored in this work enables researchers and roboticists to develop improvements and construct variations which may become superior to this method.

Acknowledgments

The opportunity to pursue this work was made possible thanks to my graduate adviser and committee chair, Dr. Alan Asbeck. He has been a tremendous source of guidance and support since August 2015 when he offered me the opportunity to work with his team in his Assistive Robotics Lab. Since then, Dr. Alan Asbeck has been a critical part of my development as a researcher and has supported my growth as a professional. He has spent much energy and countless hours giving his best support to his students and lab group.

It is my pleasure to thank my graduate committee faculty members Dr. Alfred Wicks and Dr. Tomo Furukawa of the Mechanical Engineering department for their dedication and commitment to their students. They have offered much of their time and resources in their course lectures and have provided me with many one on one opportunities for guidance and help.

I would also like to thank my father, Danny Barrows, and mother, Donna Barrows, for their endless love and support throughout my entire academic journey. This experience would not have been possible were it not for your devotion.

Personal Motivation

This subsection addresses background the motivation behind the research pursued in my thesis. It includes my personal experiences and information that supports my efforts and passion for my work. There is no technical material included in this subsection, as it is intended to provide depth to the overall work.

My interests in machine learning and wearable robotics first came about when I started my second internship opportunity with the National Aeronautics and Space Administration (NASA) at Langley Research Center. During that summer opportunity, I had begun my second programming course and started learning about mechatronics through my internship work as a design engineer student. I had the opportunity intern a few times again with NASA and work on various projects within several different engineering fields. These internship experiences led me to realize the value of learning and integrating multiple technical disciplines. My third and fourth NASA internships were valuable experiences which involved the development of image processing and machine learning applications for improving flight deck technologies. In this time, I took my first online course in machine learning created by Stanford University and hosted through the Coursera online learning platform. I applied the knowledge I gained in this course to my personal hobby of building small home robotic applications. Additionally, the machine learning fundamentals I learned were applied to my undergraduate control systems and robotics courses. These opportunities provided me with much needed experience in areas aside from my mechanical engineering undergraduate studies that relate heavily to my interests in robotics studies.

My passion for robotics was further supported when I served as the senior design project leader for Old Dominion University's (ODU) Autonomous Surface Vehicle (ASV) team. My role as project lead was to plan, organize, motivate and guide the team to delivery of a fully autonomous aquatic vessel for the 2015 RoboBoat Competition. I enjoyed this ASV leadership position because it allowed me to interact with each of my fellow team members and the discipline-specific tasks that they were responsible for. This experience allowed me to the opportunity to apply my knowledge of computer vision and machine learning I gained through NASA to develop a semi-autonomous navigation system for avoiding obstacles and steering the vessel.

I am very grateful for my undergraduate experiences with mechatronics and robotics

as it eventually led to my opportunity to pursue graduate school at Virginia Polytechnic Institute and State University. During my time at Virginia Tech, I have been able to work with my advisor, Dr. Alan Asbeck, and his Assistive Robotics Lab group to further my knowledge on wearable robotics and begin my research on the topic of lower extremity exoskeleton technology. My coursework in mechatronics, Bayesian robotics, and digital signal processing at Virginia Tech has greatly supported my interests in wearable robotics and autonomy. The combination of these course teachings and my previous experiences has allowed me to exponentially learn and flourish in my ability to do research. My experiences and opportunities within my graduate school endeavors has allowed me the opportunity to continue working with NASA at Langley Research Center. Furthermore, it presented the offer to work as a Pathways Co-op intern at NASA, where I was able to cross over my research topic and work at Langley to support both efforts simultaneously. These experiences have cemented my interests in wearable robotics and have provided me with great opportunity for my career as an engineer and researcher.

Contents

List of Figures

List of Tables

Abbreviations

AP Aggregate Prediction

BCT Best Case Test

CV Cross Validation

IP Interval Prediction

KNN K-Nearest Neighbor

ML Machine Learning

PM Prediction Model

RCT Realistic Case Test

Chapter 1 Introduction

Until recently, the analysis and prediction of intra-step gait behavior has been computationally limited due to sensor technology shortcomings and processing capability of time consuming iterative algorithms. Recent development of sophisticated sensors and high performance computers have enabled a new tier of intensive analytical algorithms to be implemented and executed on large gait data measurements in real-time. Machine learning is a feasible option for analyzing real time data and generating predictive conclusions within a fraction of a second. Additionally, the increasing popularity of machine learning and robotics has facilitated immense growth from companies and researchers to develop technologies that are able to achieve high sensing capability and computational performance. With the abundance of high-quality sensors, powerful machines for processing large volumes of data, and advance algorithms, it is easier than ever today to analyze and predict human gait behavior in real-time. The work presented in this thesis is a product of these technological advances.

This work revolves around the analysis of gait behavior and the prediction of the next step in the gait cycle. The primary center for this research is the use of a machine learning algorithm to predict, in real-time, the next step location of a human when walking.

A wearable inertial measurement motion capture, by Xsens Technologies B. V., is the sensor package used for real-time acquisition of accelerometer state data. The machine learning element uses the k-Nearest Neighbors algorithm to process the accelerometer data and predict future steps.

1.1 Motivation

The vision of electromechanical exoskeletons is for the technology to provide assistance to human operators and be capable of enhancing perceived physical power at little to no physical expense of the user. In the late 1990s, the idea of wearable powered systems was a large work in process that had a limited amounts of research to support it. Now, just two decades later, research has flourished and ideas have materialized with a quickly formed progression of these applications. What merely seemed possible in the past has recently become a reality and advancements are occurring at an abundant rate.

The motivation behind the research in this thesis document is to improve the intelligence of modern day wearable systems. This advancement of intelligence involves investigating the limitations of current systems, specifically the lack of knowledge of the wearer's intent. It is largely desirable for a user's intent or actionable event to be made aware of before its actual occurrence. In many technologies today, the knowledge of the user's intent is very minimal and is often not measured ahead of time. This hinders the exoskeleton system's ability to supply power or response for the desired action by the wearer. The future of this work will provide great opportunity for providing prediction through the use of powerful machine learning capabilities, and will supply knowledge of intent by the human operator.

1.2 System Applications

Applications for this work cover a broad range, and may be incorporated into nearly any conceivable use for a robotic system. Such applications are listed and are not limited to the following:

1. Clinical gait analysis
2. Human Activity and pattern recognition
3. Motion planning and prediction

What this set of applications and many others have in common is the need for predictive modeling or recognition of human behavior. Each of these applications utilize knowledge of previous gait behavior information in order to make a future prediction. Such applications can

benefit greatly from intra-step prediction for determining the next step position.

Clinical Gait Analysis

There has been consistent demand for applications to monitor and analyze abnormal human gait behavior. Previous research studies have analyzed gait behaviors of elderly individuals who are prone to fall and trip hazards in effort to recognize potential causes while walking [1]. Another study was performed to assess how many steps ahead individuals consider in order to walk with comfortable and normal behavior [2]. Both applications rely on a model which analyzes gait behavior after data collection and may provide a classification of behavior after the behavior in question occurs. Moreover, these systems may not adapt well to new situations and outlier cases. Real-time prediction offers a unique solution in that it can provide an expectation before the event actually occurs. The fall or trip of an individual may be anticipated by the system just before it actually occurs, and the planned sequence of future steps may be determined seconds before the human operator pursues their route of steps.

The recognition and prediction of events with walking before their occurrence is critical for clinical and health applications. It helps to prevent undesired behaviors or inform the user of appropriate action.

Human Activity and Pattern Recognition

Recognition of behaviors and physical activity has recently been of high interest. Many studies have been done to detect human activity by analyzing lower body movements. These works involve utilizing machine learning algorithms for automated approaches to recognize movement patterns either using camera sensor systems on-body accelerometers [1], [3], [4]. The same issue occurs as in the application example for clinical gait analysis, where predicted results are not made ahead of time, and are only determined after the occurrence of an action or event. Human activity recognition is a concept that may be applied to many various applications. The research presented in this thesis aims to predict human activity before it occurs and strongly supports the advancement of current recognition systems.

Motion Planning and Prediction

A number of applications, most popularly prosthetic systems, are concerned with motion planning and projecting movement. The user is typically very limited to the amount of control they have over a prosthetic limb or wearable applications and therefore relies on the intelligence of the system. The primary portion of the work in this thesis focuses on the prediction of a future linear step distance. The work does not focus heavily on motion planning, however; the prediction of movement destination distance plays a critical role in the field of motion planning.

1.3 Organization of this Document

Prior Work

The Prior Work section explores previous contributions in the fields of sensing for wearable robotics, machine learning methods for exoskeleton technology, and human gait analysis. This section includes a collection of largely impactful literature and publications in the fields listed above from various research institutes and organizations. These reviewed documents lay the groundwork for the purpose of the presented detail in this thesis and contribute greatly to the completion of this work.

Algorithm Design and Implementation

In this section, an in-depth overview and detailed exploration of the process is presented. The full methodology for the system is described, including the model for this work, the code organization, data flow, and process setup. A conclusion is presented for how this design and implementation may be further developed to provide additional functionality.

Experimental Design

The experimental design section discusses the objectives of the data collection process and scenario testing in detail. A review is provided for the statistical methods used in

characterizing the system's effectiveness. Additionally, this section describes the real-world examples and execution of these goals.

Experimental Results

In the results section, the system's performance is fully presented and evaluated. The results of the model's training, prediction accuracy, and limitations are discussed. Furthermore, a written portion is provided which discloses alternative solutions and future advancements that may be explored to achieve relatable results.

Conclusions and Future Work

A summary of the contributions made in this thesis is provided, highlighting the progression and results of this work. This section covers the insights gained from the research and the algorithm development, as well as the experimentation of the application. Avenues for expanding this research and potential future work are explored in addition to a description of how this work may possibly be improve

Chapter 2 Prior Work

2.1 Human Gait Analysis and Assisted Walking Behavior

Since the early 1990s, assistive wearable robotics has been a popular topic of interest. In order to better research lower extremity assistive technologies, it is necessary to study the biological gait and stepping behavior of humans. Such wearable systems that largely interface with humans and rely on direct human cooperation demand careful consideration of design requirements and specifications. Understanding the basic principles of human walking is critical for conducting this lower extremity exoskeleton research and development of approach. For these reasons, the work in this thesis is driven by prior studies that have further investigated human gait influences from exoskeleton technologies and is greatly driven by the findings of previous work.

The work presented by Dollar and Herr in [5] reviews historical advances made with lower extremity exoskeleton and active orthoses technologies. An informative overview of hardware design, actuation, sensing, and control techniques is provided and discussed. The conceived idea of exoskeleton technology for most parties across the world was primarily for military benefit but was later realized to be assistive for any human being and application. It was found in the 1960s by a researcher for the U.S. Army that there were many fundamentally difficult aspects in researching and developing an exoskeleton device. These include, impact on biological gait, power sources that are portable, sensing of the system and environment, and control of the system. From the work in previous decades, the shift of focus to having an onboard power source or energetic autonomy has increased drastically. Also, the aim to lower metabolic cost of walking (MCW) and metabolic cost of transport (COT) has become increasingly popular. It is of high interest to improve the performance of such attributes while also providing comfortable fittings of an exoskeleton system and allowing for natural gait behavior. In current ongoing research, common issues include power supplies, lightweight actuators, and efficient developments for research tend to not satisfy low weight and/or high efficiency design considerations. It is observed that electromechanical and energy requirements have become the challenge areas for implementing exoskeleton concepts. Computing and sensing methods and technologies have advanced so dramatically in recent years and are not typically limiting factors for exoskeleton design and function. It is advantageous to pursue exoskeleton development for impaired and incapable individuals

because there is significant value in improving such quality of life. However, for performance enhancement of able-bodied individuals, the challenge to develop suitable exoskeleton applications is much greater and requires higher power features to advance the physical capacity of capable human beings compared to those that are incapable. If exoskeletons cannot sufficiently improve the performance of individuals or empower them to complete tasks in which they previously could not otherwise perform, then such a system has very little to offer. This review believes that much of the future work to come will involve improvements of algorithms, power supplies, actuators, sensing strategies, and transmissions for improved mobility and efficiency of exoskeletons.

Replicating human gait behavior for assistance and enhancement has been a difficult challenge due to specific physical limitations and requirements by human bipedal locomotion. In [6], Giulia et. al. performed a study for comparing the dynamically balanced human gait to that of the REX exoskeleton by REX Bionics. This study explored the differences between kinematic data collected from gait analysis when using the REX exoskeleton compared to natural human gait data. It was found in this work that the collected gait data from the REX exoskeleton was similar to the natural gait but could not provide natural gait for various joints and movements. Specifically, the static gait maximum flexion gait of the REX occurs earlier and is far lower than the natural gait data. Additionally, the REX exoskeleton reveals larger power generation at the knee and hip as compared to the natural gait data which has the most power at ankle. These highlighted differences are only a few limitations that inhibit natural gait performance when using assistive lower extremity exoskeleton systems.

Similar to the work of [6], the study in [7] summarizes the design of a lower extremity exoskeleton and analyzes its performance with respect to Clinical Gait Analysis (CGA) data used for the study. A primary objective of the proposed work was to develop an optimized system that satisfies a low weight requirements, requires minimal power consumption, and yields high performance. The completed BLEEX system depends on collected CGA data that was adjusted to influence necessary joint angles and torque for optimal performance of the exoskeleton. One important design aspect of BLEEX is that it transfers the exoskeleton load and additional payload to the ground through the mechanical linkages. For this to be possible and for the BLEEX system to be kinematically similar to a human leg, a few Degrees of Freedom (DOF) in the foot and legs needed to be removed and limited. It was concluded in the paper that the ranges of motion of the DOFs in the BLEEX system needed to be slightly less to ensure safety and reliable performance. Lastly, the work in this paper also showed

that the optimized joint angle and torque power curves generated for each joint did not require each joint and linkage to be actuated movement by the user that did not require significant power and meet the minimum power threshold for motion. This work emphasizes the need for compliance between the assistive system and the physical limitations of the human operator.

The works presented in [2] and [8] aim to investigate how visual information influences the control of bipedal walking. From the study in [8], Matthis and Fajen defined a complex terrain walking scenario, which refers to a flat ground path with irregularly spaced obstacles. In their study, two experiments were conducted in order to evaluate walking performance for a complex terrain path under visually limiting conditions. Both experimental cases consisted of having participants walk through a path region populated with obstacle areas that was displayed on a flat floor using a projector. Participants were asked to maintain a consistent walking speed while avoiding projected obstacles, but were not explicitly told that both conditions needed to be satisfied. The trials had a limited visible walking radius around the participant of a varying number of footsteps. In both experiment sets, it was found that it became difficult for participants to avoid stepping on obstacles with little or no decrease in speed when walking with a visual field of view constrained to less than 2 steps. The performance of the participants with a field of view of at least 2 steps or greater were comparable to the trials with no visual limitations. This suggests that a field of view radius as great as 2 to 3 steps is necessary for people to traverse complex terrain and avoid obstacles.

In [2], Patla and Vickers designed an experiment where participants were instructed to travel two defined paths to measure how many steps ahead each individual looks while walking the paths. This study was designed to understand the link between human gaze behavior and bipedal walking, specifically stepping patterns. Two primary gaze behaviors are considered for assessing the visual link to locomotion. The participants were required to step on each footprint in the paths, with one path having regularly spaced footprints and the other irregularly spaced. The travel path type was not an influential factor in the average number of steps that the participants looked ahead. Surprisingly, it was found that the participants' gaze did not fixate on the footprints, but rather the ground in general and sometimes the spaces in between. Additionally, the travel gaze fixation behavior was significantly more predominant than the object gaze fixation behavior. This suggests that the object gaze fixation behavior may not necessarily be needed unless the object being stepped onto or over jeopardizes walking stability or has other physical properties that extend beyond

the flat and level travel paths used in this study. Similar to [8], the phenomena of interest in this study is how locomotion is influenced by vision. Patla and Vickers found that visual context plays a crucial role in determining foot placement and general stepping behavior.

The objective of this research in [9] is to achieve a portable wearable exoskeleton robot that has the capability of influencing human locomotion and reducing metabolic cost of walking (MCW). A tethered pneumatically powered soft clothing-like exoskeleton suit is the focus of this research because of its lightweight textile materials, low inertias which result in low metabolic cost, and the intrinsic transmission of torques through the human wearer's joints. Prior to the design, each of these benefits were heavily researched and understood before implementing for experimentation and further study. Sensor systems including kinematic sensing, gait cycle monitoring, and force control feedback were outfitted on the soft exosuit system in order to achieve smooth gait cycle transitions and walking for the user. Common rigid exoskeleton systems tend to be bulky and have significant inertias, which negatively influences normal human gait and causes unnatural walking performances that are often quite noticeable. The soft exosuit presented in this paper minimizes these issues and allows for natural user gait, and significantly reduces metabolic expenditure. The study showed that the pneumatically-powered soft exosuit is capable of enhancing human walking (no load), and able to reduce the MCW with little change to the gait kinematics during implementation.

In [10], Farris et al evaluates and discusses the findings from testing a powered lower limb exoskeleton capable of achieving stair ascent and decent of for paraplegic individuals. For assisted stair ascent and decent to be possible, the joint torques and power requirements for the hip and knee joints were designed to afford desired motions with large loads. A system was developed that was able to provide the necessary motion and be able to carry such loads by taking into consideration the user's center of pressure (CoP) and mass. As the CoP shifts forward the right leg begins to lift and be placed forward. The left leg is raised and placed on the same subsequent stair tread once a second shift occurs in the sequence, and the process is repeated until completion of ascent or decent behavior or when the CoP is not shifted forward. It was found that the largest torque and power requirements occurred in the right hip during stair ascent. Similarly, for stair decent the right hip also required more power and torque, however; the left knee, which represents the completion of the sequence, had the highest demand in the sequence. This work supports the findings that the largest power and torque requirements on a system of this type will occur during stair ascent and decent, and that suitable requirements be defined and set in order to achieve such motion safely.

Understanding stair ascent and decent walking elements are crucial for improving technological advances in assisting human walking performance as it is a mode of human locomotion experienced almost daily by most of the worldwide population.

2.2 Gait Data Analysis and Machine Learning

Machine learning research and application integration has increased tremendously to accompany the growing field of robotics. Research for wearable technology continuously integrates more advanced subsystems to provide improved usage and state-of-the-art performance. Through machine learning, latent behaviors and phenomenon may be discovered that may not be revealed easily by human analytics. Similarly, guided assistance may be provided to aid users of wearable technology in order to optimize performance and maximize benefits. For these reasons, the research in this thesis is influenced and motivated by previous work that critically analyzes human gait data through powerful machine learning techniques to determine hidden factors and provide further insight into human walking behavior.

A review of machine learning challenges and trends for wearable physiological sensors is presented in [11] which explores recent findings and research related to on-body health monitoring systems. Several algorithms were found suitable for handling real-time data analysis are considered, including support vector machine (SVM), Neural Networks (NN), decision trees, Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), rule based decision-making, and wavelet analysis. These algorithms are desirable because they are equipped by design to handle streams of time series data and are developed for recognizing pattern sequences. Frequently reoccurring challenges presented include high dependency on the data features and type of data collected, computational complexity, and experimental conditions which lack real world situational consideration. Furthermore, it is commonly found that the systems need to be supervised, such that they are provided with distinct data labels which characterize the information being assessed by the methods. The literature review presented provides useful insight to the technical challenges discovered over recent years of research which reveals the large trend of utilizing supervised machine learning algorithms for classifying and prediction real-time physiological behavior.

An approach for achieving locomotion of a user equipped with a lower extremity

exoskeleton system by recognizing and categorizing common ground terrain profiles was studied to better understand common modes of assisted human walking. This work reported in [12] aimed to improve violation control of powered lower limb prosthetics using electromechanical devices rather than strictly mechanical limitations. The process is broken down into two primary parts: Terrain Recognition (TR) model and Locomotion Mode (LM) model. The TR system has three main stages: terrain reconstruction, terrain classification, stand/turn detection. During operation, the IMU (acceleration, angular velocity, and orientation) and laser sensor (terrain measurement) are acquired and processed simultaneously. The terrain classification stage is the final process and is dependent on other two stages. First, the system aims to measure and evaluate the terrain in front of the user by processing laser sensor data from the ground profile. At the same time, stand/turn detection occurs and determines if the whether or not terrain categorization is enabled or disabled. If it is enabled following the proceeding two stages, then the system categorizes what type of ground terrain is in the frontal view of the users; this includes, upper terrain, level ground, lower terrain. The final stage must be disabled when turning because the system requires valid data to reconstruct and classify ground terrain in front, the direction of intended forward motion, of the user. Following the TR system is the LM model which engages the correct mode of locomotion to traverse the terrain in front of the user. Locomotion modes include: walking, ramp ascent, ramp descent, stair ascent, and stair descent. The accuracy of the TR system was roughly 99% for the seven test subjects used in this report. The TR model required only a half second processing delay before the user could execute the resulting locomotion mode.

The work in [3] summarizes an effort for recognizing human-motion activity from on-body accelerometer data by means of machine learning classification. Furthermore, simple base classifier performance is compared to that of combination meta-classifiers in effort to uncover any notable causes for an increase in activity recognition accuracy. Features extracted from a single accelerometer, located near the pelvis, were used as the input for training a set of classifiers, which include: decision trees (C4.5), decision tables, Naïve Bayes, and the nearest-neighbor algorithm, with the decision trees performing the best, at an accuracy of 84%. Meta-classification includes, boosting/bagging, plurality voting, stacking using ODTs and stacking using MDTs. The purpose for formulating this effort as a classification problem is to discover which classifiers are the best for recognizing activities, determine if combining classifiers yields greater performance, and learning which activities are more difficult to recognize. Activities that were recognized include: standing, walking,

running, climbing up/down, vacuuming, brushing teeth and sit-ups. Prominent features extracted and used in this study include, mean, standard deviation, energy (sum off squared discrete FFT component magnitudes), and correlation. Certain features, such as correlation, were particularly useful for distinguishing between activities like walking and running with stair climbing and vacuuming. However, since correlation only differentiates translation in just one dimension, it often cannot distinguish walking from running since they are essentially the same translation. In summary, boosted/bagged Naïve Bayes, SVM and kNN classifiers performed consistently well for all data, regardless of subject independence. Interestingly, SVM performed well overall, however; the boosted SVM classifier outperformed all other models for when data from one subject on a particular day was used for training the model and data from another subject on a different day was used for testing. Overall, it was found that meta-classification techniques yielded better prediction accuracies for activity recognition. It was also found that some activities, like brushing, were in general hard to distinguish and often confused with other activities, such as standing and vacuuming.

Another work by Begg et al, performed in [13], further supports the use of supervised machine learning for understanding human physical behavior. This work proposes a solution for automatic gait recognition between young and elderly human gait types using the Support Vector Machine (SVM) algorithm. Minimum Foot Clearance (MFC), a derived walking metric from a 2D motion analysis system, was found to be particularly useful in distinguishing different gait behavior between young and old subject participants. Additional features were extracted from MFC histograms and Poincare plots. The study demonstrates that the MFC metric is highly informative for indicating age, with the hypothesis that a lower foot clearance is common among elderly people and the clearance height correlates strongly to tripping and falls. Many of the MFC-derived features for SVM classification were valuable statistical metrics, while some others were more directly extracted since they were highly observable via the MFC plots. Hill-climbing feature selection was pursued for identifying the most significant features in order to reduce the dimensionality of the feature space and create a less complex SVM classifier model. The results in this experiment of young-elderly gait recognition showed that the SVM classifier is able to perform better when a fewer number of features, which are more qualitative overall, are chosen. Out of the set of most significant features, it was found by the feature selection algorithm that the MFC Coefficient of Variation feature had the most distinguishable difference between the two age groups. The average accuracies of the SVM classification validation test using all features was 81% and greater.

Some combinations of three features were able to produce classification models with accuracies greater than 90%. Overall, the MFC metric and many of its derived features were significant enough to use for training an SVM classifier and distinguishing between young and elderly steady-state gait behavior with high accuracy.

Further research on feature extraction and selection is performed in [14], where a human activity recognition approach is derived from feature extraction and selection techniques. The work done in this study aimed to identify the most important features that had high informational content and unique distinguishability for recognizing human physical activities. Physical, frequency-domain, and statistical features were defined and extracted for use in the feature selection approach. On-body accelerometers by MotionNode were used to measure acceleration data and extract features from six participants. Physical activities included: walking forward, walking left, walking right, stair ascent/descent, jumping, running, standing, and sitting. Fixed window of 2 second lengths and with 50% overlap were used for feature extraction. Common frequency-domain and statistical features, such as, mean, variance, correlation coefficients, root mean square, DC component, and entropy were computed. Additionally, physical features computed from multiple fused sensor modalities were extracted, and are important for quantifying coordinated motion between multiple limbs and joints. Important physical features include: movement intensity mean (AI) and variance (VI) which are useful in differentiating static and dynamic activity, Averaged Velocity along Heading Direction (AVH) and Correlation between Acceleration along Gravity and Heading Directions (CAGH) which are valuable for measuring motion intensities, and finally Averaged Rotation Angles related to Gravity Direction (ARATG) which is necessary for detecting human rotation of joints about the gravity direction and distinguishing between forward and sideways walking.

Feature analysis is further explored By Zhang et al in [14] where three feature selection models are assessed: Relief-F, Single Feature Classification (SFC), and Sequential Forward Selection (SFS). These feature selection techniques each showed that the designed physical features play an important role in the resulting classification accuracy. A total of 50 features is the optimal limit, as the misclassification error for each feature selection set of 50 features and greater tapers off and changes very little. SFC yields good results overall with only very little computation time, compare to that of SFS, and SFC also has the greatest amount of selected physical features. However, SFS overall generates the best classification performance for all feature set sizes, as it has approximately the same misclassification error at only 5 selected features as it does at 50. A hierarchical feature selection and classification

model, using SVMs, is employed to achieve correct classification of activities across the wide range of possible variations. The multi-layer classification system is able to classify activities which only differ largely in one distinguishable dimension, such as gravity in stair ascent/descent or intensity between walking and running. In conclusion, the SFS method achieved the best feature selection performance and was improved by using a hierarchical feature selection and classification framework that classified the data from multiple perspectives. However, the downfall in the hierarchical model is that it is developed from domain knowledge rather than being determined from the data and nature of the human activities.

A more real-time based approached by is investigated by Mannini and Sabatini in which utilizes statistical pattern recognition and the HMM for classifying static postures, such as lying, standing, sitting, and dynamic motions, like running, walking, and several other distinguishable activities [4]. An annotated dataset of on-body accelerometer signals is used to test and validate the classification algorithms employed in this study. Since the accelerometer output is non-stationary real-time data, several derived feature variables are obtained to properly represent the raw accelerometer output, and are ultimately used for training the classifier algorithms as they possess higher quality information content. Metrics taken from feature evaluation include the averaged accelerometer signal DC acceleration component, the computed variance from each de-trended frame of data samples, the signal energy via Short Time Frequency Transforms (STFT), the frequency-domain entropy derived from the STFT coefficients, and the correlation coefficients between all pairs of accelerometer signals. In addition to the chosen features, several lenient thresholds are sanctioned in order to distinguish between critical properties of motion, like presence or absence of motion and sensor sensitivity or noise output, in order to better classify behavior. As in [14], the frequency-based features play an important role in characterizing repetitive information within a data window, where windowing is heavily relied on.

In [4], the study utilizes a preexisting dataset from another experiment which is composed of five dual-axis accelerometers located at the right foot, left knee, right wrist, left upper arm, and pelvis. Unlike the other experiment, this study trained its single frame classifiers on a subject-specific bases rather than overall. In addition, windowing was applied for 512 samples (6.7 seconds) with 50% overlap in order to generate a representation of a set of different actions. An optimal 17-dimensional feature vector was chosen, composed of 4 DC components and 13 correlation coefficients. The DC components provided enough postural information and the correlation coefficients gave enough information regarding

coordinate body motion in order to accurately classify the physical activity of interest.

HMMs are of particular interest in [4] because of their proficiency in recognizing sequences of events and patterns in data. Several single frame classifiers were trained and tested, including Naïve Bayes, GMM, SVM, KNN, Neural Networks, and several others; each of which yielded a classification accuracy greater than 90%. The employed continuous HMM multi-frame sequential classifier peaks at a classification accuracy of 99.1% which is superior to its simple single-frame GMM contender. In conclusion, this study found that the HMM-based sequential pattern recognition approach shows very promising results with many data and scenario dependent advantages. It is believed that subject-specific training could improve accuracies and results of the HMM-based sequential classifier due to the uniqueness exhibited by individuals when perform various physical activities.

A novel alternative to [4] is presented in [15], where gesture recognition is pursued using the theory of Random Projection (RP), and by utilizing Dynamic Time Warping (DTW) in combination with Affinity Propagation (AP) for training the system. In [15], accelerometer gesture data is stored in an off-line lookup database and is comprised of 18 possible classification gestures. HMMs were considered, however; they were deemed insufficient as they require an explicit number of states, and vary in complexity due to their dependence on dimensionality of the feature space. The learning process of the data includes, first convolving for smoothness, clustering by applying DTW and then AP, and finally recognition by a combination of applying DTW and then formulating the problem as l_1 minimization. The DTW algorithm is critical in this study as it minimizes the cumulative distance measure between the instances of aligned samples. AP is the desired clustering method because it computes clusters using a collection of similarities between data points as inputs rather than raw data. The Random Projection method is used to project the sparse nature of the accelerometer gesture traces, represented by the best match selection of exemplars from AP, onto a unified lower-dimensional subspace. This study shows that subject-dependent recognition obtains near perfect accuracy, while other forms of mixed subject dependency and independence yield results comparable to that of other works addressing accelerometer-based gesture recognition. Compared to continuous HMMs and approaches used in other literature, The proposed system outperforms the continuous and discrete HMM variants, as well as several proposed systems from other literature works, in recognizing test gestures from both mixed-user and user-independence settings.

Similar to [15], the work in [16] investigates a hybrid time series prediction model that incorporates the HMM algorithm and DTW. The DTW algorithm performs good initial

clustering of the data which is then cross-referenced and cleaned by the HMM clustering method. These two partitioned clustering methods behave similarly to the K-Means and KNN algorithm, however; they consider a sequence of data rather than a static collection. The HMM clustering behaves as a check against the DTW clustering in the event that any sequence of data is misrepresented, and re-categorizes the incorrectly assigned sequences. The results from this complementary combination are improved compared to each individually tested case. Even though the twice-over clustering is redundant, the accuracy of the system is improved by assessing the incoming time series data window sequences using different analytical approaches.

Chapter 3 Algorithm Design and Implementation

3.1 Approach

Machine learning is one of several potential approaches for characterizing human walking behavior and determining a next step linear distance. Other approaches, such as regression analysis, probabilistic modeling, and dynamic controls systems were considered for determining future outcomes. Unlike machine learning, these alternatives do not easily provide instantaneous predictions on unseen data instances, where a collection of real-time data may be classified by a closest probable match. These alternative systems also require pre-defined assumptions addressing the kinematics of human walking behavior. This pre-definition of assumptions and checking of kinematic motion models becomes especially convoluted when many different walking modes and unexpected behaviors are highly probable to occur. Because of the many different modes and variation of walking, a high-order regressive model or collection motion model sets would be needed to appropriately characterize all possible behaviors. These were not desirable approaches for the work in this thesis. A machine learning approach enables distinct predictions between the various forms of walking are needed in order to properly classifying the occurring step.

In a physical motion model or regression model, a mathematical definition is developed which describes the walking behavior and builds a relationship between multiple characteristic variables of human locomotion. As the next step occurs, the linear step distance may be estimated using the mathematical relationship which is modeled after a generalized behavioral form of walking. A probabilistic motion model is one possibility for predicting future motion of the body. In human walking, both legs must be considered in order to accurately model the behavior of the human operator. This modeling would be required to consider a large variable set, as the body has many degrees of freedom in each leg. During normal walking, each hip has three degrees of freedom, each knee has one degree of freedom, and each ankle has three degrees of freedom, see Figure 1. For this approach to be used, each leg has a minimum 7 DOF in the general case, totaling at 14 DOF when considering both legs. A motion model would include the lengths of the limb segments, their inertias, and the entire body's velocity measured at the center of mass. With all of these characteristics, a dynamical model of the system could be constructed.

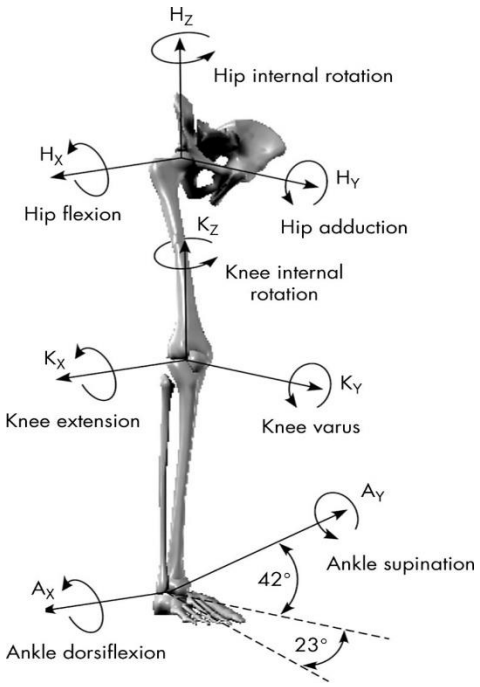

Figure 1: Human leg kinematic model (image from Evaluation of a two dimensional analysis method as a screening and evaluation tool for anterior cruciate ligament injury, McLean et. al)

To use the probabilistic model approach, the state would include the 14 DOF join angles and their derivatives, for a total of 28 variables. To predict the walking linear step distance (1 DOF), a function is necessary which would estimate the foot's position given all of the joint angles in the two legs. With the included derivative joint velocities, an estimate of the foot's landing position could be constructed. However, this problem is highly underdetermined and there are many possible final linear foot step distances for a given initial state. As such, some sort of estimate or probability distribution of the possible final linear foot step distances would be necessary for accurate prediction.

Motion modeling, compared to machine learning, is more desirable in such situations where predicting linear step distance does not require a significant number of derived features or a complicated high DOF system. The large-modeled system has too many input variables and potential variations of outcome to accurately account for each walking mode class. Furthermore, a probabilistic motion model derived for multiple modes of walking is likely to perform poorly with instances with unfamiliar step data or unusual stepping trajectories. There are countless swing phase motions that may occur for different classes of terrain and ground profiles. Due to these reasons, machine learning is desirable as it may build an all-in-one model of multiple ground profile classes and walking behaviors given a sufficient collection of training exemplars. Additionally, a motion model based prediction

made from unseen data which may be similar to multiple swing phase motions that makes it difficult to distinguish which type of walking the human operator may be concurrently experiencing. Thus, in such scenarios, it would be difficult to confidently define and identify outlier instances using a mathematical physical motion model with no inference capability. The need to estimate the probability distribution of the final linear step distance implies that a machine learning approach is necessary. As such, given that a probabilistic motion model may be complex and yet still lead to the same machine learning requirements by design, it was decided to use a straightforward supervised machine learning approach without any motion model. However, motion modeling could still be very useful in determined which features may be most useful for machine learning, and for determined which joint angles are best to use for predicting the linear step distance of the foot during walking.

Complications arise given real-time dependency and volatility of anticipating human intent during walking. For example, if a human intends to walk on flat terrain but steps on or over an elevated obstacle, the characteristic behavior of this sudden mid-step change in walking throw off the estimations by the previously mentioned alternative models whereas a machine learning model may accept new incoming mid-step data and recognize the change in human intent.

Another approach that was considered was the use of a regression model to determine the future linear step distance. The following regressive model below describes the relationship between independent variable X, dependent variable Y, and unknown parameter vector A.

$$E(Y|X) = f(X, A)$$

The above regression hypothesis determines the output variable Y, given the input state X and characteristic parameters of walking, denoted as A. For human walking step prediction, the X input vector for the regression is a set of characteristic features, such as acceleration, velocity, joint angles, joint angle velocities, and instantaneous position. The parameter set, A, is defined as the coefficient set and unknown characteristic variables derived from the empirical data to form a minimized squared error estimate. Lastly, the Y output is the determined linear step distance from the toe-off starting step position to where the foot is placed on the ground, given the X inputs and the minimized error characteristic set, A. This model may be applicable for determining the linear step distances during walking for a single walking mode (i.e., flat terrain walking) but by design would perform poorly for other modes of walking. Furthermore, this approach has the downsides that only specific

trajectories could be predicted, making the system less robust and not perform as well in the event of unexpected inputs or unusual stepping motions.

Additional walking behaviors can be accounted for by using a high-order regression, with a significantly increased unknown parameter vector space, or by forming multiple regressive models for each intended behavior, which is a largely tedious task. The former option must account for a high-dimensional space whereby the system may generalize well and may also provide a probabilistic outcome. The high-dimensional probabilistic approach is affordable by methods such as the logistic regression or SVM, by which both commonly categorize as supervised machine learning algorithms. The exploration of regressive models circles back around and points toward the affordable option of using machine learning.

3.2 Algorithm Design Considerations

In this section, the formulation of a simulated real-time machine learning prediction model is developed for estimating the future foot placement of a human subject under natural walking conditions. This literature begins with three primary phases of the model development, feature selection, model training, and prediction. Feature selection in this work is the process by which physical and statistical measures are generated and chosen to represent the human walking behavior described by the prediction model. The prediction model is trained using the K-Nearest Neighbor (KNN) algorithm to learn specific states of human activity in natural walking. Finally, from this inferred learning, the real-time machine learning results are aggregated to form a single prediction outcome representing the foot placement.

There are two distinct forms of machine learning techniques that have been popularized in the field: supervised and unsupervised approaches. Supervised learning approaches for classification such as artificial neural networks, Naïve Bayes Classifier, SVM and Decision Tree Classifiers require the input data vector to be supplied with a set of categorical labels. In contrast, unsupervised learning techniques, like clustering algorithms, GMMs Neural Networks, and HMM, are able to infer the label categories without any supplemental information [17]. This additional label information required for supervised learning is attainable from the walking data due to the nature of the experimental procedure. Label metrics are processed after data collection and do not need to rely on unsupervised inference. Since a low overhead algorithm is desired and the label sets are readily available, it was decided in this work to focus solely on a supervised learning approach.

According to [13], advanced machine learning techniques, such as Neural Networks, offer unique findings that support this research. The work goes on to state that such learning unsupervised algorithms do not require gait feature set labels for automatic classification of gait behavior. It is important to note that unsupervised learning and complex convergence methods were of interest in this research, however; they were not suitable due to their complexity, processing demands, and potential to derive an incorrect set of instance beliefs. Specific intra-step walking data instances must be identified by means of accurate prediction, and computations are required within a short period of time for simulated real-time performance. The unsupervised approach may tend to make classification distinctions of gait behavior without any awareness and additional information of the categories which classify the data. Due to these constraints and differences, such computationally biased algorithms were not considered in this work. The supervised learning approach is able to make distinct predictions to a given set of instance beliefs, are generally fast, and introduce a relational variable, known as a label, to help strengthen classification ability of the states using a well-defined set of data exemplars.

3.3 Machine Learning Algorithm

To provide near instantaneous classification, it is necessary to devise an algorithm that will process a stream of measured data efficiently enough to make a viable prediction of the next future step position based on the current walking state. Predictions of the occurring step using no previous prior sequential data is desirable as it allows for a decision to be made that is not influenced by discrepancies and trends from recently collected past information. This idea is made possible through the development of a software algorithm which relies on machine learning and statistical calculations to discover discrete relationships between measured data. Interval predictions of incoming windowed data are aggregated as the next step occurs and provides a final prediction that is concurrent with the measured data.

Data pre-processing is one of the most vital steps for machine learning in this work. Preprocessing of the data is necessary for detecting the beginning of a footstep, normalizing the data set, and calculating the distances between footsteps. Other preprocessing included in this work consists of filtering data, interpolation, and selecting data features. Detection of each footstep starting point is accomplished by assessing the zero acceleration and zero velocity points measured from the Xsens IMU foot unit data. When the foot unit measures zero acceleration and velocity over a short period of time, it is indicative of motionless activity

and having been planted on a stationary surface. The true starting point of the next footstep is measured as the last stationary data point of contact before detecting motion. Upon detecting motion, it signals the start of the next occurring footstep, where the simulation real-time prediction begins.

A threshold is applied to the detection of footstep starting positions. It is defined such that when a zero acceleration and velocity point has not occurred for longer than 0.25 seconds, then the activity since the last measured zero point is considered the occurring real-time step. This process is necessary as many zero points may be measured between the initial contact of the foot to the ground and throughout the stance phase of the gait.

3.3.1 Feature Extraction and Selection

The extracted and selected features generated from the raw sensor data are supplied to the machine learning classification algorithms in the form of an input vector. For human physical activity prediction, the exemplars of input data are associated with the observed behaviors (class instances) under consideration. The classification models considered in this work use the input feature vector to learn a decision set, to which it associates the input data to the class instances [17].

Windowing is employed following the initial data preparation and step detection. The windowing process consist of dividing the data stream into small time segments as the data is being measured. Each windowed segment of step data is processed using machine learning to produce an interval prediction of the occurring step's final step distance. In order to accurately identify step behavior from a small portion of measured data, kinematic and statistical features are derived to relate the measured information to human walking behavior. In studies [3][14][18], physical, frequency-domain, and statistical features were shown to each have high added value for accurately predicting human physical activity and

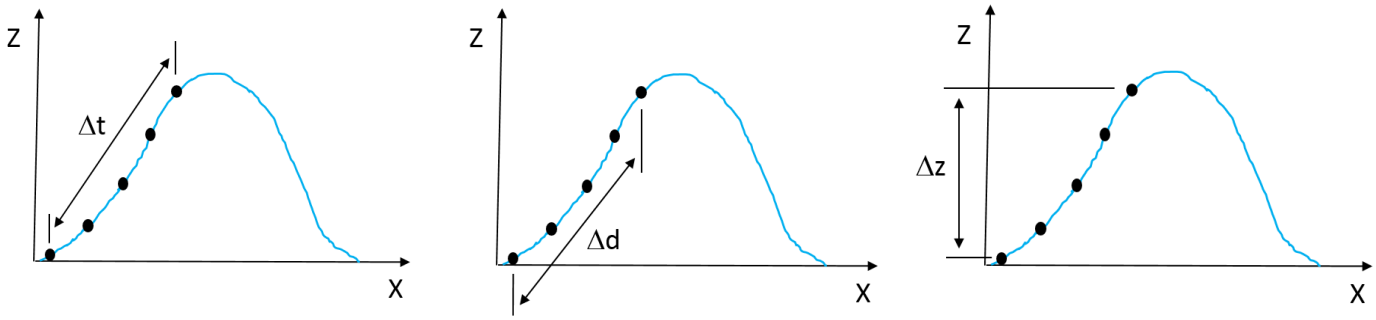

Figure 2: Global window features - Time Computation (left), Total Change in Distance (center), Total Change in Height (right)

walking behavior. However, the nature of the experiments in those studies is vastly different in the repetitive motion is not crucial, nor common in the windowed segments in this research. As such, frequency-domain features were deemed unnecessary, and physical kinematic features were more heavily used. Features describing the entire windowed data segment, referred to as global features, are representative of the physical behavior in larger incremental changes, see Figure 2.

The features in this work include window time computations, linear pairwise distances between inline data points, acceleration, velocity, and average changes in kinematic behavior over the span of the windowed data. The most notable features are the pairwise distance between sequential window points and the derived features including acceleration and velocity components. The simple average and average change of the kinematic acceleration and velocity components over the window are derived to provide physical features which characterize the data represented in the windowed data segment. The derived features are relatively simple and require minimal computational power and time. The physical features are shown by Figures 3 and 4.

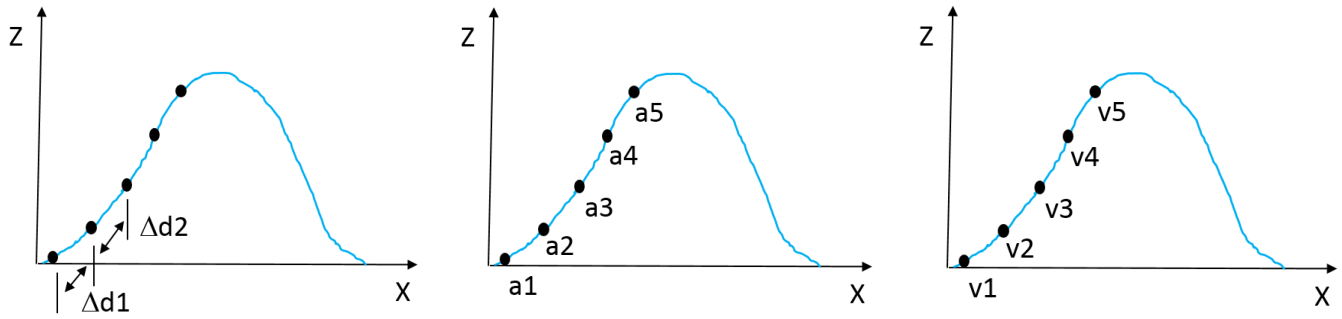

Figure 3: Physical features – Pairwise Distance (left), Raw Acceleration (center), Raw Velocity (right)

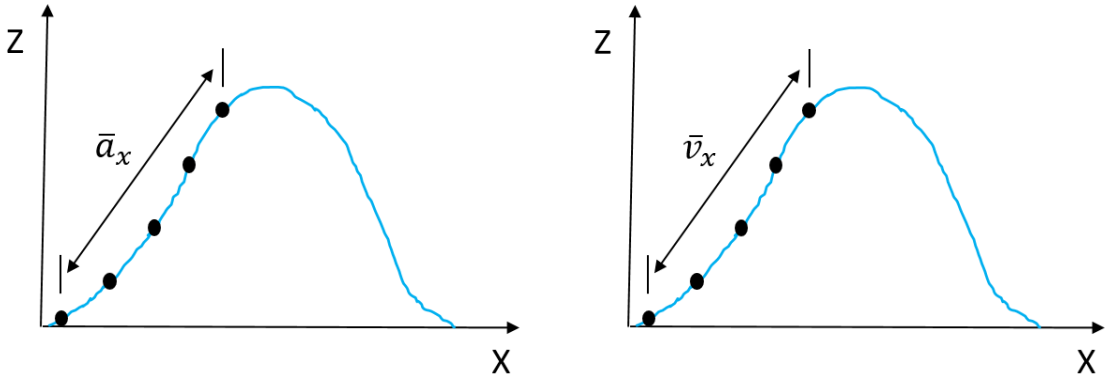

Figure 4: Derived Physical features - Average Change in Acceleration (left), Average Change in Velocity (right)

The Pairwise Distances between sequential points is necessary for understanding the change in velocity and position of each data point. It follows such that for a window size of N, there are $N-1$ total pairwise distance features derived and included in the feature vector.

$$PD = \sqrt{\left(p_{i_x} - p_{i-1_x}\right)^2 + \left(p_{i_y} - p_{i-1_y}\right)^2 + \left(p_{i_z} - p_{i-1_z}\right)^2}$$

The average of kinematic components over the window is applied for acceleration and velocity components only. This is shown by the following Average Change in Acceleration quantities:

$$\mathrm{ACA_x} = \frac{\sum_{n=2}^{N} a_{x_n} - a_{x_{n-1}}}{N-1}$$
$$\mathrm{ACA_y} = \frac{\sum_{n=2}^{N} a_{y_n} - a_{y_{n-1}}}{N-1}$$
$$\mathrm{ACA_z} = \frac{\sum_{n=2}^{N} a_{z_n} - a_{z_{n-1}}}{N-1}$$

And for the Average Change in Velocity quantities:

$$\mathrm{ACV_x} = \frac{\sum_{n=2}^{N} v_{x_n} - v_{x_{n-1}}}{N-1}$$
$$\mathrm{ACV_y} = \frac{\sum_{n=2}^{N} v_{y_n} - v_{y_{n-1}}}{N-1}$$
$$\mathrm{ACV_z} = \frac{\sum_{n=2}^{N} v_{z_n} - v_{z_{n-1}}}{N-1}$$

Similarly, as with the pairwise distance magnitudes for position, the sequential pairwise magnitudes are calculated between points for acceleration and velocity. The same quantity of $N-1$ features are also found for each, given a window size of N.

$$\mathrm{PD_a} = \sqrt{\left(a_{n_x} - a_{n-1_x}\right)^2 + \left(a_{n_y} - a_{n-1_y}\right)^2 + \left(a_{n_z} - a_{n-1_z}\right)^2}$$

$$\mathrm{PD_v} = \sqrt{\left(v_{n_x} - v_{n-1_x}\right)^2 + \left(v_{n_y} - v_{n-1_y}\right)^2 + \left(v_{n_z} - v_{n-1_z}\right)^2}$$

Using these derived features, a series of experimental tests are performed for many combinations of the features. Raw data and extracted features are used in combination within the supplied feature vector input. The testing of subset feature spaces is crucial for determining the overall best performing classifier model. This experimental work involving feature selection results is further explored and presented in Chapter 5.

Feature selection is a necessary step which consists of selecting a subset of relevant data features with higher quality information within the original feature set to represent the nature of the data. The objective of feature selection is to determine the most informative and influential features used to recognize and represent human walking behavior to provide accurate step prediction. Using the kinematic physical features, derived from the measurement parameters of human motion, the statistical features and overall feature space is enhanced with a broad reach of representative metrics [14]. Redundancy of data tends to incorrectly model the systems behavior and leads to loss of accuracy by incorrect predictions. For this reason, it was desirable to include many varying derived physical inertial components of the measured foot IMU data, derived statistical metrics and representative global window features such as overall time and distance in attempt to recognize the model's behavior as accurately as possible.

Traditionally, in human gait analysis, features such as mean, variance, correlation, and Fast Fourier Transform (FFT) coefficients computed from motion measurement sensors are depended on for recognition [3]. However, in this work repetitive cyclic motion is not of interest as the recognition of a sequential series of data to predict short term behavior does not rely on any intra-window repetition. What makes the work in this literature different from other works is that it does not heavily utilize frequency-domain and statistical features for prediction. Furthermore it is commonly thought to compose all available data features into one feature set to use as the input to a machine learning classifier. The fundamental disadvantage of this approach is that much of the raw data features and additional derived features may not aptly characterize the data and may be redundant. The inclusion of such irrelevant features hinders the classification accuracy by not providing good quality features or by supplying too much data as the input. Additionally, too many features may even confuse the classifier model and cause failure in its ability to discriminate various states in the data. Even worse is the potential of the "curse of dimensionality", where the performance drastically decreases as more features are added. This is further worsened when there is not enough training data to reliably learn all of the distinguishable states and parameters of the generated model. It is for these reasons that feature selection must be carefully executed and

tested before proceeding further with analysis. To differentiate between sample data, supervised learning techniques need salient and complimentary features. Using inappropriate or redundant features may deteriorate the performance of a classification algorithm. Features that have optimal discriminative power between classes, is significant in data mining. The feature selection process is defined in this work as a process of testing a subset of appropriate features from the original set. Feature selection is an important step in maximizing the performance of the KNN algorithm as it greatly influences the step behavior prediction and reduces the complexity of the feature space, all while improving the overall data set classification rate [17].

To achieve the best prediction performance from the model, the dimensionality of the feature vector should be kept small [13]. The smaller the size of the feature vector, the better the classifier performance becomes. Additionally, smaller dimensionality of the feature vector may reduce computational cost. It is for this primary reason that the number of features is limited to a very small amount in this work.

3.3.2 Prediction Model

The prediction model system was developed with the primary focus of having the ability to make instantaneous predictions based on real-time observations of unique sequential exemplar from a stream of new data. This consideration was taken into account when developing the simulated real-time software functions ["methods"] and is largely responsible for the selected machine learning algorithm implementation. Supervised machine learning algorithms, including KNN), decision tree classifiers, Naïve Bayes, and Support Vector Machines, were considered in this work because of their design to accept labeled data for training and prediction. These methods in many cases offer faster processing speeds and allow for training to be performed on the basis of supplied labels, which provide additional information regarding measured data.

Preliminary research indicated the best choice algorithm for use in this work to be the KNN model. The KNN model is considered a lazy learner, meaning that it directly learns from training observations on the basis of lookup comparisons, rather than building fully parameterized model [4]. The other considered supervised learning models operate more eagerly and with limited knowledge, in that they require more than the KNN algorithm [19], [20]. They build parameterized models that are strictly limited to the scope of knowledge deduced by a supplied training data set [4]. What sets the KNN algorithm apart is its

profound ability to learn incrementally. The KNN method adapts well to learning and predicting data streams, where it can determine the closest matched instance of new data. It performs exceptionally well at observing and classifying new and complex instances of data, compared to other supervised learning methods. This makes the KNN algorithm a highly desired method for this work.

Fundamentally, the KNN algorithm performs on the foundation of a formulated majority vote between the K most similar data exemplars to provide a latent observation. Essentially, an observation is made based on the closest similarities in a given set of data instances using a provided set of data characteristics, known as features, which are representative of an unknown response [21].

Similarity within the KNN algorithm is defined according to the Euclidean distance metric between two data points. Such that:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

Euclidean distance was the best choice for use in this work because of the measured data's strong spatial context. Acceleration, velocity, and position is highly relatable through Euclidean space since they are spatial metrics [19].

In theory, given that a positive integer K, an observation X and a similarity metric D, the KNN classifier perform prediction through the following steps:

- First, the algorithm runs through the entire training set and computes similarity D and observation X between each data instance. The K number of neighboring points in the training data that are closest to X are represented by A. Commonly, K is chosen to be an odd number in order to prevent ties in prediction.
- The conditional probability is then estimated for each class, that is, the fraction of points in AA with that given class label.

$$P(y = j \mid X = x) = \frac{1}{K} \sum_{i \in A} I(y^{(i)} = j)$$

As a result, the input X is classified and assigned to the category with the highest total probability.

The KNN algorithm is a relatively fast algorithm when relying on smaller sized data sets. It is desirable because of its simple non-parametric nature, where it relates data effectively to provide unknown observations [19], [20]. The algorithm may provide accurate observations for results that are unusual and not highly characterized. KNN requires relatively little training time, and excels at providing quick analysis of provided data. Furthermore, the algorithm easily supports multiclass scenarios, used in this research, which allows for an unknown instance of data to be classified into one of many other categories. On the contrary, the KNN may be slow on large data sets and perform poorly if it is trained on skewed data. Training sets with uneven data instances tend to cause poor performance, where predictions are biased toward classes with high volumes of exemplars.

3.4 Next Step Prediction

It would be undesirable to use the KNN algorithm for predicting the occurring next step all at once. This would require nearly all of the data pertaining to the step in question to be collected and would potentially require far too many features to informatively represent the step. Therefore, the KNN algorithm is applied to each segment of the sliding window to determine predictions incrementally as the data is processed in series, as in [4], [14]. However, as each new data point is collected, a new step distance prediction is made with respect to the more recent data window. The collection of predictions becomes integral to determining the step distance as each part of the swing phase gait is essentially accounted for. In theory, each preceding window prediction will provide more information in order to allow for a more stable assessment of the next step distance to be made. This overall prediction assessment of the distance becomes an aggregation of each individual window prediction based on a voting method.

Simply, the voting system is the most frequently occurring distance prediction with respect to the training data. In this work, where there is only one decisive algorithm, the collection of multiple KNN predictions acts similar in behavior to the popular Bootstrap Aggregating (Bagging) method, by which multiple random sub-sample sets from a set of exemplars are used to determine the model. The difference in this setting is that the KNN utilizes windowed sub-sample sets of a single instance of collected data or step to determine a single outcome. The aggregation of each windowed prediction is weighted equally and accounts for the many kinematic characteristics and descriptors throughout the full stepping motion.

As the stream of step data is processed in this work, the aggregate prediction is updated, shown in Figure 1. In doing so, the algorithm offers an overall step distance prediction outcome throughout the entire next step duration. This presents the opportunity for fluctuation in the aggregate distance prediction as new votes are continuously included. However, a stable prediction outcome to be made a significant time in advance before completion of the occurring step. Regardless of whether or not an aggregated prediction outcome becomes stable early on or remains unsettled throughout, it attempts to predict the step's distance from the step's starting contact point until it is determined to be complete. The algorithm concludes its prediction attempts once it recognizes a zero acceleration and zero velocity point has occurred.

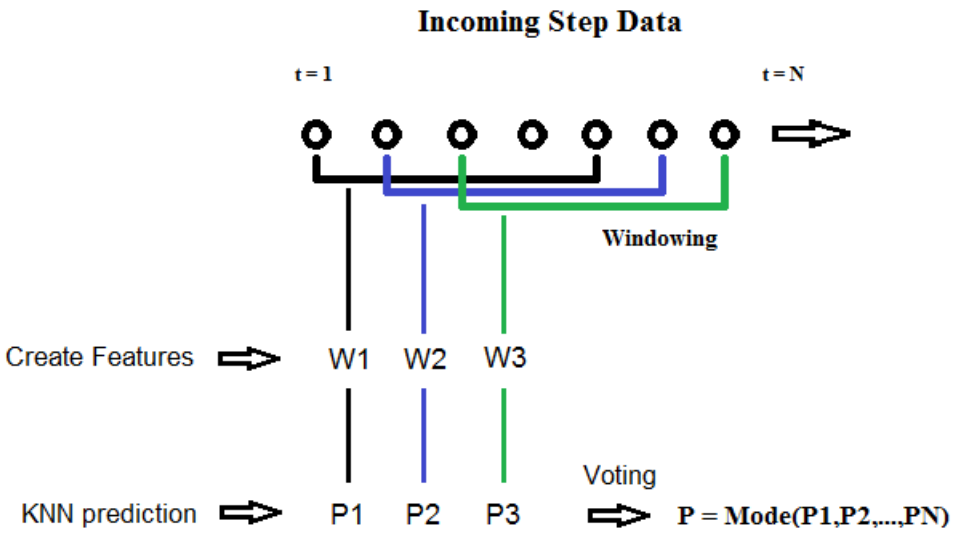

Figure 5: Aggregation Process of Interval Step Predictions

In the formulation of this approach, it was believed that the future step distance could be closely determined by assessing primarily the data collected of the pre-swing and early swing phase during the gait cycle. Significant behavioral information is presented in the early stages of the gait cycle which was believed to be enough knowledge of intent for determining the relative stepping distance. Because of this, very little measured data is necessary to begin the making predictions from the windowed series data. The results of this belief are presented further in Chapter 5.

3.5 Model Validation

A Cross Validation (CV) test was performed as a metric for validating the prediction model. This approach and its performance with respect to other data tests are provide rigid analysis of the overall algorithms prediction performance. The CV test is a model evaluation method that offers a better assessment of the accuracy of the system. It provides further insight than residual split testing of training, validation, and testing data subsets. What makes the CV test particularly useful is that is provides a statistical proof of how well the model evaluates and predicts new data that it has not already seen. This process is conducted further as it is performance several times for multiple partitions of the entire data set. A portion of the data is removed and set aside before the KNN training begins. After the training is complete, the data that was removed is then used to test and validate the performance of the KNN model on the unseen data. However, rather than simply relying on the small partition of removed data to test the model, this process is repeated several times. Though it may seem redundant, the averaged result of the multiple CV partitions provides for a better assessment of the algorithms performance.

Instead of relying solely on the common training and testing data, this study employs the more advance *K*-fold CV approach. This CV test improves over the traditional training and testing holdout method by dividing the overall data set into *K* subsets, and the underlying holdout method is performed *K* times. The data is randomized and each *K* subsets is individually interchanged, also at random, for testing.

Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set *k-1* times. The variance of the resulting estimate is reduced as *k* is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch *k* times, which means it takes *k* times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set *k* different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

Chapter 4 Experimental Design

4.1 Testing Considerations

It was desired for the data collected in the experiment to represent many behaviors typically found in natural human walking on a daily basis. Walking on natural level and irregular terrain was performed in order to capture the natural walking behavior and general approach to foot placement [2]. Irregular terrain in this context is defined as ground profiles with slightly slanted profile, slight unevenness, and small differences in elevation. It is typical outdoor terrain that is not complex terrain or requires additional consideration or planning when walking.

Several distinct terrain profiles were included in order to differential walking data and observe walking behavior. These specially assessed terrains include, flat ground, entry/exit doors, ascending/descending slopes and ascending/descending stairs. Each of these unique terrain items offer unique walking characteristics which can be assessed for classification and prediction of future foot placements.

4.1.1 Data Collection

The untethered Xsens Awinda system, shown in Figure 2, measures data from each IMU at a sampling frequency of 240 Hz. All IMU devices for the Xsens system are synchronized using a patented Xsens protocol that ensures a delay of 10 microseconds or less for simultaneous collection from all units. A 30 millisecond delay of latency occurs between the accelerometer data acquisition and recording of the data to the file. Upon initial configuration of the Xsens system on the wearer, a data recording unit is enabled. After the first phase of the recording unit is enabled, the recording unit undergoes a detection process for recognizing each of the IMU sensor units. The automatic detection of the lower body system and other included sensor units allows the Xsens system to prepare and utilize proper calibration. Once ready for data collection, the recording unit is engaged and a 3-5 second stand-still N-pose is required before walking. The stand-still N-pose position consists of the user standing with their feet at shoulder length apart and both arms down along the sides of the body. This particular pose allows the Xsens to know the relative position of the

user and each IMU unit for accurate calibration. The N-pose calibration provides a small set of data measurements for the Xsens system to know the position of each sensor unit relative to one another on the wearer. This is necessary due to the drastic change in placement, distance, and orientation of each unit when an active session occurs. Without this essential step, the Xsens system cannot possibly know how to estimate calculations for the joint angles and motion of the sensors relative to the wearer's body anatomy. The Xsens data recorder begins to blink once acquisition has started and the experimental trial proceeds as directed.

The raw data collected by the Xsens system and used in this work includes accelerometer readings, gyroscope readings, and magnetometer readings from each IMU. These readings are then fused by a proprietary algorithm by Xsens, in conjunction with a human body model, to generate estimates of the velocity and position for each IMU, in addition to the wearer's estimated joint angles and joint angular velocities [22]. The Xsens sensor fusion process estimates these metrics for each body segment with respect to a globally-fixed coordinate system. In this reference coordinate system X is positive toward magnetic north, Y is normal to X by a rightward Cartesian coordinate system, and Z is positive upward against gravity, see Figure 9. Each IMU is composed of its own local coordinate system, however; when the Xsens Awinda system is calibrated, the orientation and reference of each unit become fixed to the default global coordinate system described above.

During setup of the Xsen system, the user's total height and offset of their shoe insole from the ground is manually entered into the BioMech system for data collection before the calibration occurs. These dimensions infer the limb segment lengths, which are recognized during calibration of the multiple IMU orientations and respective positions. A body model, which estimates the velocity and position of each foot through integration, is used for determining these simultaneously collected raw data measurements. The other IMU segments along the shin, thigh and pelvis are necessary in order for these integration estimations to be possible and mitigate drift as the Xsens system collects measurements. For this reason, the more IMU devices that are connected with the Xsens system, the greater the calibration accuracy, data component calculations, and reduction of measurement drift.

Figure 6: Fully Setup Untethered Xsens MVN System

The resolution of the Xsens data is high enough for the research work performed in this thesis. On average, a step may take as long as 1.5 seconds or short as 0.5 seconds to occur. At the Xsens sampling rate, more than enough data points are provided to perform step prediction of the proceeding movement. For future real-time applications, it may be suggested to test a subsampled rate for analyzing measured data points due to the high resolution.

Figure 7: IMU Sensor Placement Top Profile

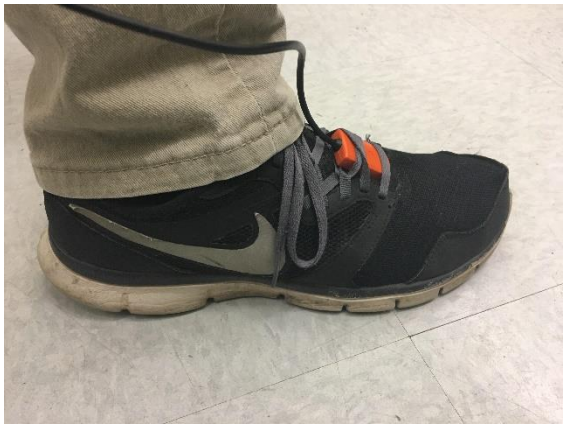

Figure 8: IMU Sensor Placement Side Profile

The Xsens IMU foot sensors, shown in Figures 7 and 8, are perhaps the most sensitive in terms of sensor placement. Each foot sensor must be fastened securely on top of the foot and not shift during data collection. Small shifts can cause large amounts of error which can ultimately skew the entire data set and results. Fastening strips and tightened shoe laces were enough to keep the sensors from experiencing undesired movement.

4.2 Experimental Procedures

The experiment begins with the system user setup after consenting to participate in the data collection testing. Before the Xsens system setup process, a few physical measurements are taken of the user in order to help with calibration and accuracy of the data for analysis. These measurements include, total body height, waist height from the ground, and the shoe insole height. Other measurements including IMU sensor distance offsets, joint distances, and orientations are not manually measure, but are instead calculated by the Xsens system upon calibration of the units. In preparation for the walking data collection, the IMU sensors are attached to the body via Velcro body strips. The Velcro strips must be securely fastened to the body in order to prevent any shifting and change in position as the wearer is walking. Only a lower body configuration is necessary in this work. However, the upper body pelvis unit is included in the experimental configuration to provide better sensor calibration and system awareness for each IMU unit.

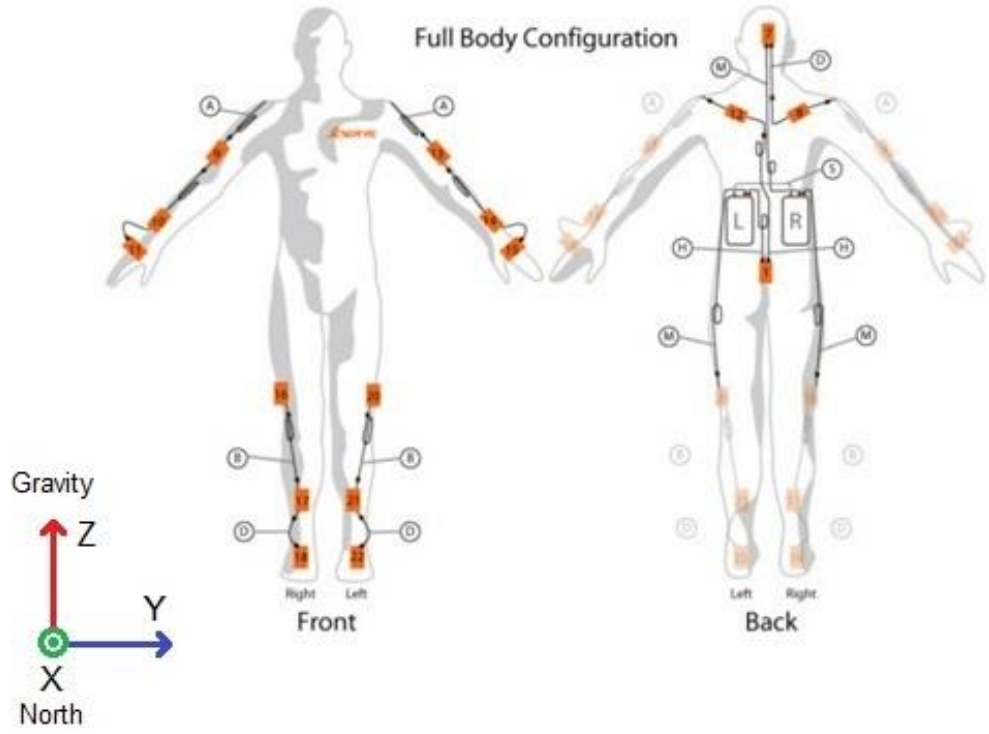

Figure 9: Full Body Xsens Configuration Schematic and Global Earth-Fixed Coordinate System

Sensors are placed midway on each upper leg section, each lower leg section, and at each mid foot location, as shown in Figure 9. In order to help improve calibration accuracy, the IMU sensors are oriented similarly when positioned, such that all of the sensor units are facing the same general direction in 3D space. Calibration of all sensor units are performed simultaneously by the Xsens BioMech Software Suite after the physical system setup is complete. As shown in Figure 10, the BioMech Software provides a 3D simulation which represents the worn Xsens configuration in real-time.

Throughout the data collection trial, the subject is accompanied and guided along a fixed course of travel over flat terrain, ascending terrain and descending terrain elevations. Prominent terrain elements of interest in this work, including stairs and slopes, are desirable in the experimental procedures due to their unique kinematic characteristic behavior. The motion when walking on stair step terrain is substantially different than that of normal flat walking. Even for sloped walking, there are several overlapping similarities and differences in walking over flat terrain and stair terrain. The particular interest of this work is to have the user walk the various types of terrain in order to properly learn from the walking behavior and predict the correct future step regarding the terrain. The user is asked multiple times

throughout the experiment to repeat portions of the course in order to collect sufficient behavioral data. Repetition of travelling specific terrain elements is necessary in order to collect enough walking data for confidently training the classifier model. Once the course is completed, the data is collected from the Xsens Data logger unit, and checked for accuracy on using the BioMech Simulator desktop application. Once the data has been validated, all wearable system materials are carefully removed from the user. The user is asked for any comments or questions regarding the full procedure before concluding the experimental trial.

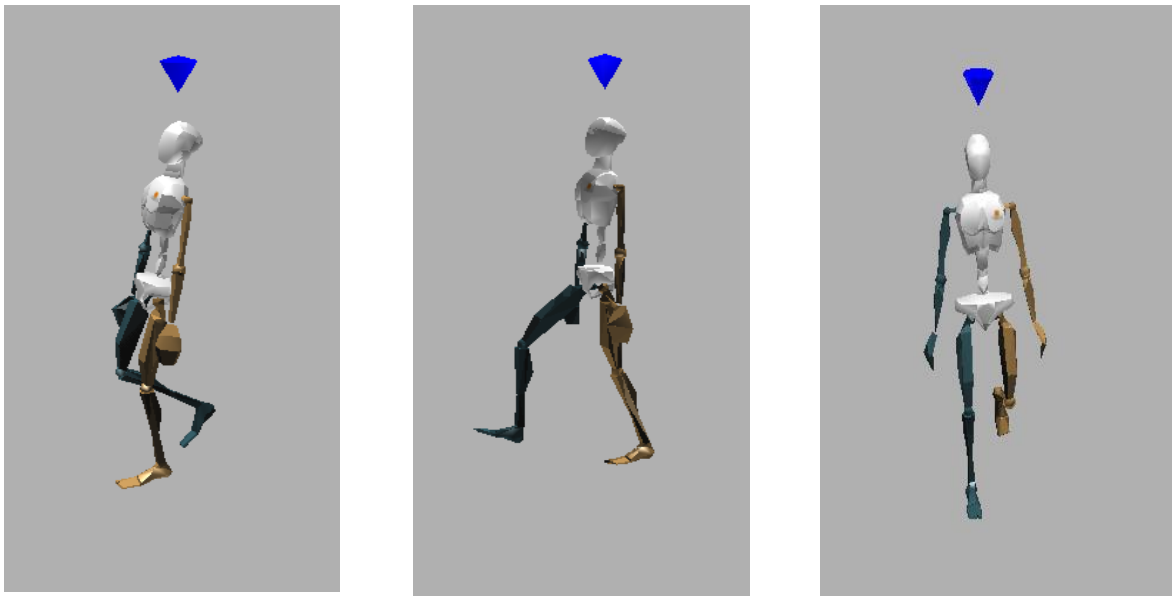

Figure 10: Xsens BioMech Motion Simulator Model

4.2.1 Walking Environment

The testing course displayed in Figure 11 was chosen specifically to include the desired elements encountered in natural everyday walking scenarios. Each of these natural terrains require a large number of training step data in order for the machine learning prediction algorithm to properly classify states and predict future foot placement for similar traveled ground profiles. The course first starts in the Virginia Tech ARL where the user walks at a medium pace to the first stairwell which exits the building. Near this exit of the building, there is an approximately 20ft long hill slope at a roughly 30 degree incline which the subject walks up and down a total of 5 times. Following this terrain element is a short flat walking path that leads to a set of stairwells with intermittent flat tops. The subject is asked to walk this part of the course back and forth 4 times. Next to the stairwell set is a sloped grass profile which matches the stair set profile.

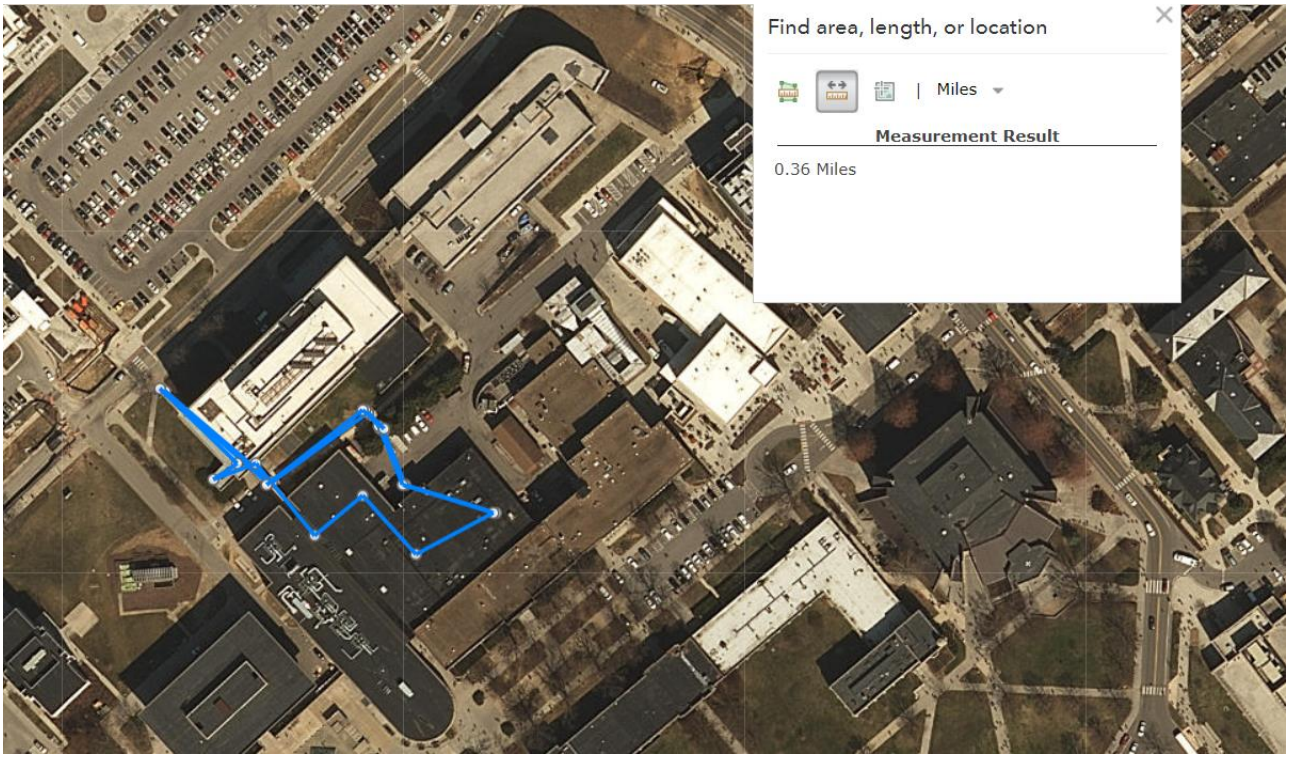

Figure 11: Satellite Image of Indoor-Outdoor Walking Course for Experimental Data Collection

Ordered course elements with step counts:

- Flat walking medium pace (80 steps)
- U shaped stair set with platform (25 steps)
- Flat walking medium pace (45 steps)
- Sloped gravel terrain up/down x5 (110 steps)
- Flat walking faster pace (60 steps)
- Straight stair set with central landing flat tops up/down x4 (260 steps)
- Sloped grass terrain up/down x3 (180 steps)
- U shaped stair set with platform (25 steps)
- Flat walking slower pace (80 steps)
- Straight stair set up/down x2 (45 steps)
- Flat walking medium pace (80 steps)

The approximate total number of steps per participant is roughly 1060 steps.

4.3 Materials

The bulk of the work in this text is completed using a combination of commercially available hardware, software and outdoor testing space for data collection. The Xsens MVN BioMech sensor package was chosen for collecting kinematic data to represent real-time human motion. Matlab is used for data analysis, evaluation and validation of the experiment. Combined, these available technologies provide a swift and effective process for collecting and analyzing data.

4.3.1 Computation and Sensing

It should be reiterated that no additional computation or analytics outside of the Xsens MVN sensor package were performed during data collection. For the research performed in this work, the computation was entirely performed after data collection. There are several reasons for performing the analysis after data collection. Primarily, the preprocessing step for cleaning the data and constructing the feature vector requires too much processing power to process in real-time. System simplification, software optimization and proper hardware were limitations in this work which did not permit inline real-time processing and prediction. Therefore, the processing of the data and classification prediction in this work was performed in a simulated real-time fashion.

The Xsens sensor placements are directly responsible for the type of measured data and quality of that data. The location of the units along both legs allow for the necessary joint angles and limb positions to be accurately estimated. It is important that the Xsens sensors are oriented properly and do not move after calibration during testing. By using knowledge of the surrounding IMU sensors, each Xsens unit can better determine the acceleration, velocity, and position accuracies of the body section it represents.

The data files obtained from the Xsens Data Logger are mvnx which is an XML protocol that is configured to work with Microsoft's Excel software. Instead, Matlab is used to directly load and parse the Xsens data files and configure the data set in temporary memory storage for analysis. Xsens publicly provides a Matlab script for efficiently and safely loading the full data set into memory and organizing the recorded data metrics for use in further processing, see Appendix A. Each metric is manually extracted then used for computation for the preprocessing and learning algorithm sequences. After building the

classifier and performing prediction, the data set is no longer needed in the further computation performed in this work.

4.3.2 Xsens IMU Motion Capture System

Xsens MVN is a wearable commercial tri-axial inertial sensor system designed for motion capture and motion-based data acquisition. The Xsens MVN package is reliant on inertial and magnetic sensors for measuring physical activity and motion in real-time. Sensor fusion algorithms are employed by the Xsens sensor units to provide an estimated measure of each unit's specific orientation relative to a globally defined reference system. The sensor fusion algorithms utilize integrations and transformations to accurately determine the velocity, position, and joint angles relative to each active unit. To provide better accuracy, several units must be worn and synchronized with one another before collecting measurements. The synchronization and simultaneous measurement of each Xsens MVN unit prevents short term accelerometer drift and provide reference of other surrounding units for improving the calculated component integrations. Each sensor's awareness of the others prevents drift and any offset of values in the short term; however, given enough time all sensors will eventually introduce drift with respect to their shared global frame of reference.

Figure 12: Battery Pack (left) and Data Logger (right)

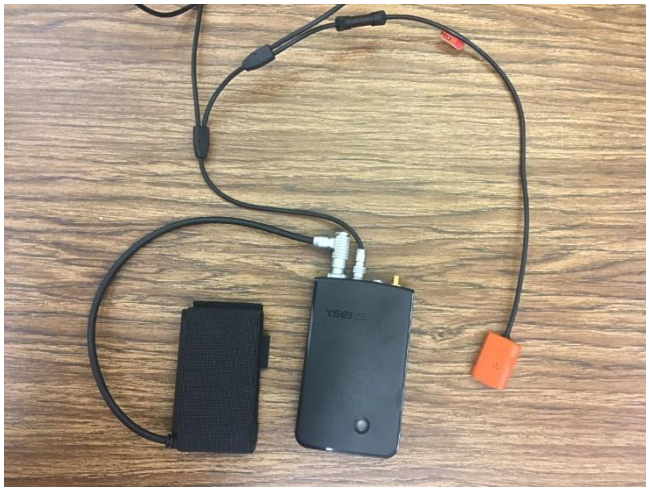

Figure 13: Fully Wired Connection of the Xsens MVN Package

Assembly of the Xsens MVN system involves several crucial components and procedures. Since the system is untethered and enables the user to walk remotely indoors and outdoors, a data logger recording unit and battery pack is required at all times during the data collection trials, see Figures 12 and 13. All components of the wearable Xsens system are fastened using the flexible body straps shown in Figure 14. The lower body IMU sensor set must first be connected to the data logger unit. The data logger unit should not turned on nor should it be connected to the battery pack for protection of the sensor units. Although several sensor input connections are expected by the data logger, it is not required that all of them are used. The input connections can properly be recognized by the data logger unit once it is powered and after all necessary hardware setup is complete. Both the battery pack and data logger should be fastened to the back of the waist band near the pelvis sensor unit, see Figure 15. When fully charged, the Xsens MVN system can be used for several hours, and a significant amount of collected data can be saved to the data logger's onboard memory storage.

Figure 14: Fastening Body Straps for Xsens Hardware

Assessment and validity of the sensor equipment and measured sensor data quality is necessary for providing meaningful results. Current competing industry standard equipment, such as camera-based measurement systems, have great rapport across commercial users, have been well documented and thoroughly tested for many years. According to [23], in a comparison study between the Xsens MVN sensor system and Optotrak camera system, the Xsens MVN sensor package is sufficiently accurate for motion capture. However, when compared to industry standard camera-based motion capture systems, there are several noticeable differences which stand out. Between the two systems,

there is a large anatomical frame discrepancy which results in large differences in Range of Motion (ROM) and joint estimation angles. According to Brennan et al, the joint angle estimation error is caused by the sensor itself because of segment pose estimation inaccuracy. Though these findings challenge the accuracy of the Xsens MVN package, the mean joint angle estimation errors ranging from 1.38° to 6.69° are not large enough to elicit concern in this work [23]. For analytical and experimental purposes in this research, the measure data from the Xsens MVN sensor package is good enough quality and consistent enough to rely on.

All Xsens MVN sensor units for the lower body segment, including the pelvis region, were required to be worn during data acquisition to perform precise calibration. These active sensors were necessary to ensure proper calibration, to minimize accelerometer sensor drift, and to improve the accuracy of the data. The calibration process calculates the orientation offset for each IMU with respect to the global frame of reference on the wearer during a short standing still-motion phase. Time stamped data collection recorded in each trial is captured by the Xsens BioMech Software in real time and exported as playback data table files.

Figure 15: Pelvis IMU and Hardware Placement with Respect to the User

Chapter 5 Experimental Results

5.1 Collected Data and Preliminary Results

The first initial effort of studying the experimental results involved the assessment of the prediction step profiles. A set of predicted step 3D profiles and actual step 3D profiles were compared in order to gain a first look at the correlation between the matched step profiles to the closeness of the predicted linear step distance. A few examples of these step comparisons are shown in Figures 16 and 17, which to a large extent visually validates the prediction model's ability to make accurate step predictions with similar 3D profiles based off of the linear step distance metric alone.

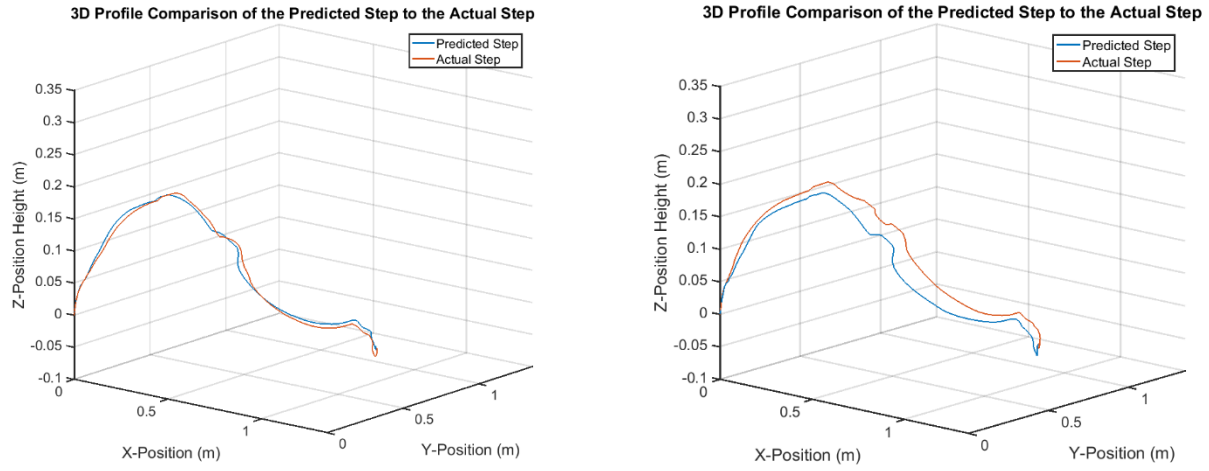

Figure 16: 3D Step Profiles - Close Flat Step Match (left), Offset Flat Step Match (right)

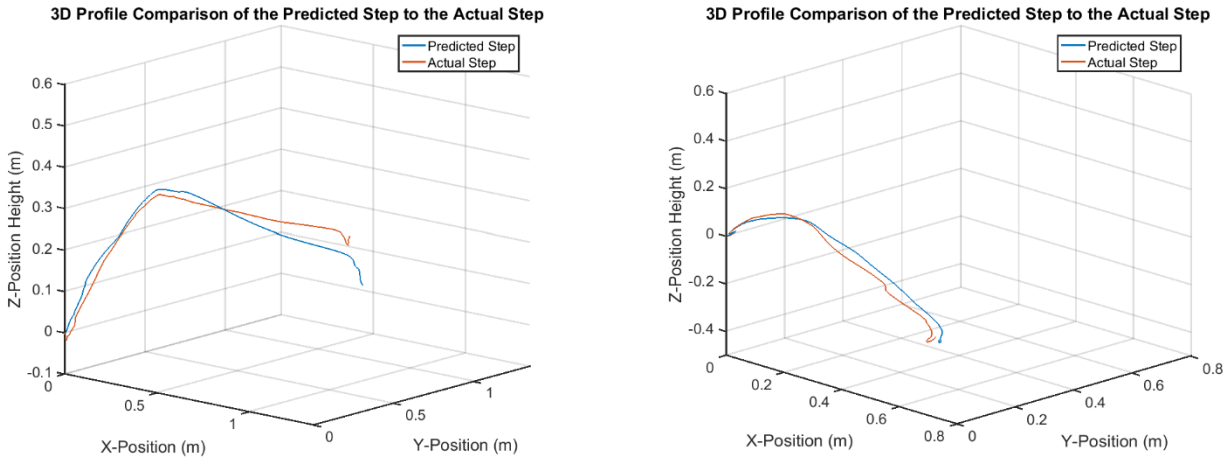

Figure 17: 3D Step Profiles - Stair Ascent Match (left), Slope Descent Match (right)

5.2 Experimental Test Results

Experimental data sets were collected from two participants and consist of approximately 1060 steps each. These data sets were considered separately for the analytical experiments performed in this chapter. Throughout all tests, the KNN algorithm parameters are fixed to assess eight nearest neighbors using the Euclidean distance metric. In these experiments, perhaps the most crucial preliminary testing performed is the selective feature subset testing. The experimental trials performed in this work consider the eight following features for describing the sliding window: Pairwise Distance (PD), Time Computation (TC), Total Change in Height (TCH), Total Change in Distance (TCD), Average Change in Acceleration (ACA), Average Change in Velocity (ACV), Raw Acceleration (RA), and Raw Velocity (RV). The baseline for the testing of the feature space is the assessment and comparison of the various feature subsets to the inclusion of all features, except for RA and RV. Interestingly, it was found that the features which are representative of the window as a whole, such as the TC, TCH and TCD, do not play an integral contribution to the overall classification accuracy. These features are not as informative of the physical walking behavior, like the acceleration and velocity components, and therefore do not influence the classification as greatly.

Although the acceleration and velocity data is highly informative of the walking and step behavior, the appropriate features need to be studied to determine which are desirable for use. In the feature subset experimentation, it was quickly found that the simple averages of the acceleration and velocity components do not perform as well as the ACA and ACV features, and where therefore not included in the preliminary tests. As shown by Table 1, the ACA and ACV features do not perform as well when included with the other global window features. Furthermore, the substitution of ACA and ACV features with RA and RV shows to have the highest prediction accuracy. The raw data itself is highly descriptive of the step behavior. It is understandable that its pure form would provide high accuracies. Step distance predictions within 7.6cm are considered excellent, within 15.2cm are considered good, within 23cm are considered average, and within 30.5cm are considered poor. Step distances that are greater than 30.5cm are considered unacceptable in any practical case.

	Feature Subset Influence on Classification Accuracy within Centimeter Distance			
	Excellent (7.6 cm)	**Good (15.2 cm)**	**Average (23 cm)**	**Poor (30.5 cm)**
PD, TC, TCD, TCH, ACA, ACV	28.8%	46.8%	56.7%	64.5%
PD, TC, TCD, ACA, ACV	29.5%	47.8%	56.8%	64.4%
PD, TC, TCH, ACA, ACV	29.5%	46.0%	55.8%	64.0%
TC, TCD, TCH, ACA, ACV	28.4%	45.7%	54.5%	62.8%
PD, TC, TCD, TCH, ACV	31.1%	52.2%	61.0%	72.3%
PD, TC, TCD, TCH, ACA	27.1%	46.3%	55.7%	63.6%
PD, TC, ACA, ACV	29.1%	47.0%	57.1%	64.8%
TC, TCD, ACA, ACV	28.3%	46.7%	56.9%	64.3%
TC, TCH, ACA, ACV	28.8%	46.7%	56.1%	63.5%
PD, TC, TCD, TCH	33.1%	54.3%	63.6%	74.2%
PD, TC, TCD, TCH, RA	43.9%	67.1%	74.2%	81.5%
PD, TC, TCD, TCH, RV	24.4%	44.3%	57.1%	66.4%
PD, TC, TCD, TCH, RA, RV	47.2%	67.5%	75.8%	83.6%

Table 1: Feature Subset Influence on the Classification Accuracy

The best performing feature subset includes raw acceleration and raw velocity data. In contrast, these features add larger dimensionality to the feature vector. For example, a window size of 20 measured step data points adds the same number as the window size for each x, y, and z component for both acceleration and velocity. These two physical characteristic elements account for a total of 120 features out of the total 142 features for the chosen window size of 20. Though the accuracy is greatly improved, so is the computational demand. Approximately 21 million data points in the feature space must be calculated for the tested data set size of 850 steps. That requires the prediction algorithm to

compute roughly 25,000 feature values for each step in the combined training and testing data sets. This also means that the algorithm would require the same number of computations against the training set for each new added step being predicted. For practical use, the computational expense and time needed to predict every step is much greater then desired for fast prediction. More is discussed near the end of this section that addresses the prediction speed and feasibility of real-time processing.

Another important variable considered in the experimentation throughout this work is the size of the data set used for training and testing the prediction model. Features PD, TC, TCH, and TCD were selected to perform these tests. It was decided beforehand, based on the KNNs algorithmic limitations described in Chapter 3, that at least 1000 steps were needed in order to provide the prediction algorithm with enough historical information to accurately predict future linear step distances. This number was chosen to not be particularly high in order to clear extensive processing demands of the KNN method, as its design requires it to perform exponentially more calculations with larger data sets. To understand more about this important data set metric, subsampling tests were performed in order to explore its influence on step distance prediction accuracy.

Data Set Size Influence on Classification Accuracy within Centimeter Distance

	Excellent (7.6 cm)	Good (15.2 cm)	Average (23 cm)	Poor (30.5 cm)
1060 Steps (100% data)	34.6%	56.3%	65.7%	76.1%
530 Steps (50% data)	27.6%	49.6%	61.8%	70.5%
353 Steps (33% data)	22.2%	44.7%	58.4%	70.4%
265 Steps (25% data)	26.7%	43.2%	51.1%	63.9%
159 Steps (15% data)	17.6%	42.8%	53.5%	60.4%
106 Steps (10% data)	15.0%	29.9%	43.9%	55.1%
53 Steps (5% data)	16.9%	32.1%	35.0%	47.4%

Table 2: Data Set Size Influence on the Classification Accuracy

In Table 2, all features were considered when testing the subsampling rate of the total data set size. From these preliminary tests it was discovered that the higher quantile prediction accuracies, which include excellent and good overall step distance prediction accuracies, were heavily influenced by the larger data set sizes subsampled at higher rates. A significant change in all prediction accuracy occurs as the data set is subsampled by only half of its original number of steps. This suggests that the larger data size is desirable in order to prevent significant loss of prediction accuracy. Naturally, the KNN algorithm performs better with more data. However, the processing demand and computational complexity increases drastically directly with the increase in the input data set. For these purposes in this work, the subsampling tests validated that the 1000 number of steps requirement in the data set was a suitable choice. A total acceptable prediction accuracy loss from 76.1% to 47.4% occurs when the data set is subsampled at a much lower rate of 5%. This larger decrease in acceptable predictions compared to the decrease in higher accuracy predictions indicates that many of the steps within the good and average ranges of step distance accuracy become even less reliable.

Another crucial parameter to understand is the window size used for generating each intermediate prediction result. Window sizes consisting of 10, 20, 40, and 80 data points were considered for testing. For these tests comparing window sizes, only features PD, TC, TCH, and TCD were considered for better accuracy, which exclude the acceleration and velocity components. On average, a typical step consists of roughly 190 data points throughout. A maximum window size of 80 measured data points is deemed necessary as the limit size for testing due to its required percentage of step completion before usage. The amount of measured data points in series must be equal or greater than the window size in order for prediction to occur. In this instance of the maximum window size, the step would need to have already been at least one third of the way completed before the first intermediate prediction is made. Unfortunately, this restricts the how soon through the step that a prediction can be used for anticipating the linear step distance. In Table 3, the experimental results show that the prediction accuracy increases directly with the window size. This reveals that the occurring next step distance may be predicted more accurately when a larger quantity of measured data from the step is included in the windowing. The involved dynamic presented from these results indicates that an optimal level of prediction accuracy to step-time prediction can be found on a per-application basis. Such ideal circumstances depend on the necessary prediction accuracy and how soon throughout the step a prediction is needed.

Window Size Influence on Classification Accuracy within Centimeter Distance				
	Excellent (7.6 cm)	**Good (15.2 cm)**	**Average (23 cm)**	**Poor (30.5 cm)**
Window Size 10	32.0%	54.8%	61.9%	73.5%
Window Size 20	34.6%	56.3%	65.7%	76.1%
Window Size 40	34.9%	57.5%	67.3%	77.8%
Window Size 80	36.8%	59.5%	68.6%	80.7%

Table 3: Window Size Influence on the Classification Accuracy

In order to better understand the window size influence, the interval predictions (IPs) of each step were recorded throughout the entire stride of the step until foot placement. Figure 18 gives better insight as to how the windowing influences the step prediction and how soon a stable overall prediction is made. The percentage through the step for varying window sizes is relative. Therefore the 5% step completion for the window size of 40 is further into the step by 20 data measurements and ultimately not assigned a prediction as soon compared to the smaller window size of 20. The step IPs cannot begin as soon for larger window sizes even though they yield higher overall step distance prediction accuracies.

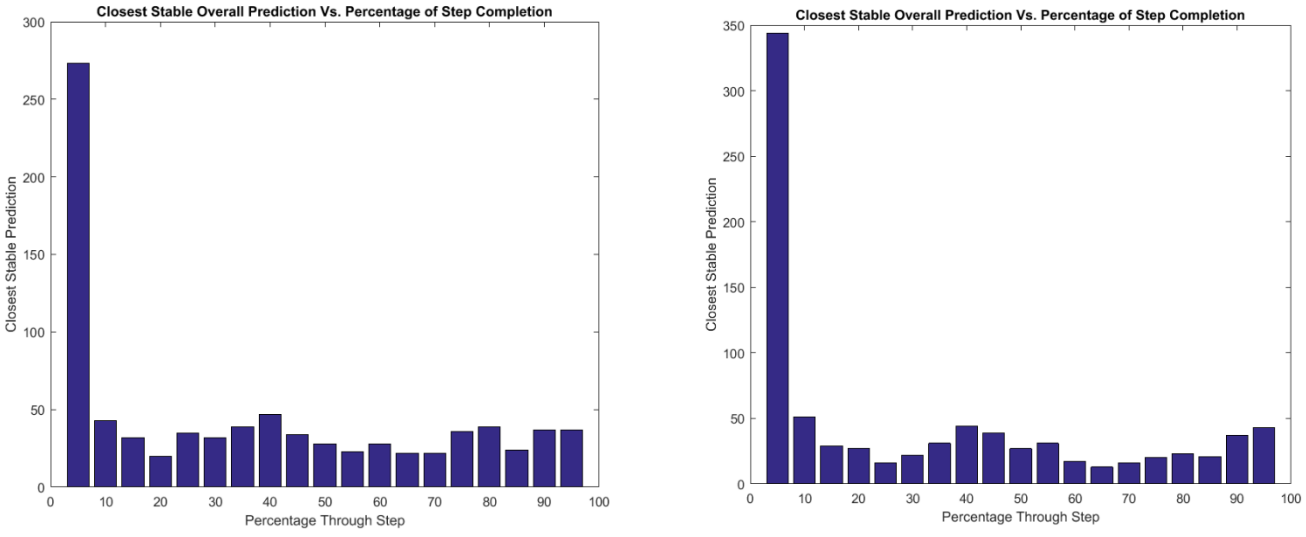

Figure 18: Percentage Through Steps until Stable Aggregate Prediction for a Window Size of 20 (left) and a Window Size of 40 (right)

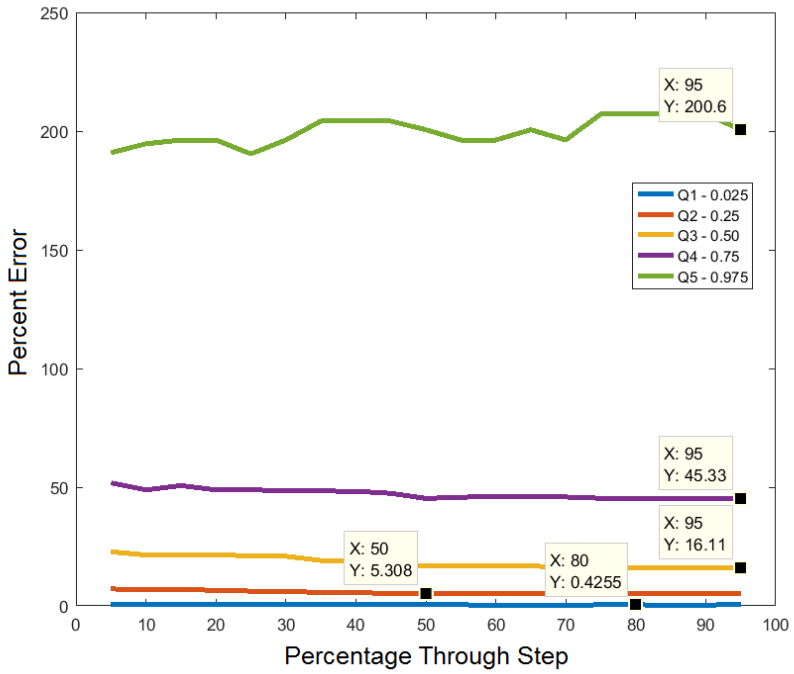

Figure 19: First Trial Quantile Errors throughout the Predicted Step Test Data

Figure 19 shows the quantile prediction performance throughout the occurrence of all steps of the first test case from Table 2, which uses PD, TC, TCD, TCH, ACA, and ACV as features. It was believed before performing the preliminary feature experimentation that this particular feature space combination would perform best. The 2.5% and 25% quantiles are within desirable accuracies, however; the other quantile accuracies are higher than desirable and much less in comparison to other feature subsets.

To further explore these intricate prediction model design variables, two distinct experimental tests were performed using 100% of the data set and an optimal window size of 20. These feature subset tests are defined as a best case, which includes all best performing features (PD, RA, RV, TC, TCD, and TCH) regardless of computation time and dimensionality of the feature space, and a realistic case with a reduced feature space (PD, TC, TCD, and TCH) for optimal executional expense. The best case test feature space includes RA and RV features in addition to the global window features to provide better characterization of the stepping behavior. The realistic case is only different from the best case in that it does not include the RA and RV features, and it ultimately results in a much smaller feature space. Both the acceleration and velocity data for the best case test each account for 60 features from the x, y, and z components. That in combination with the other features constitutes a total feature space dimensionality of 142, whereas the realistic test's dimensionality is a much smaller size of 22. The next largest feature space contribution is

the 19 pairwise linear distances between each point considered in the windowing, which is quite descriptive of the natural stepping behavior.

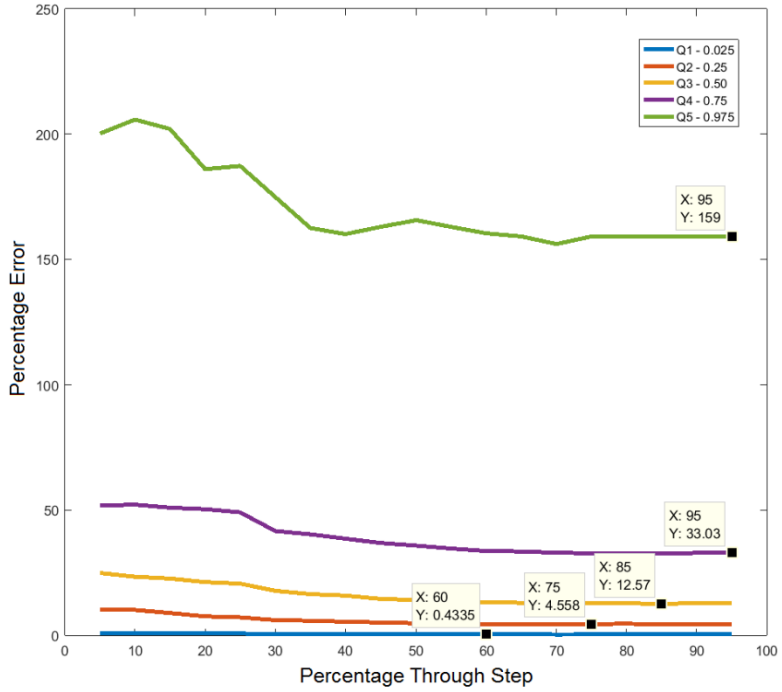

Figure 20: Realistic Case Quantile Errors throughout the Predicted Step Test Data

Between Figures 20 and 21, there is noticeable difference in the prediction accuracies for the 50% and 75% quantiles. This is crucial in terms of performance, as the best case test provides much better step distance prediction accuracy overall for at least 50% of the data and for 75% of the data. The 2.5% and 25% quantiles for both the best and realistic test cases are relatively the same. The 97.5% quantile for both remains well over 100% error and is of less concern for analysis given that it mostly represents outliers and anomalies.

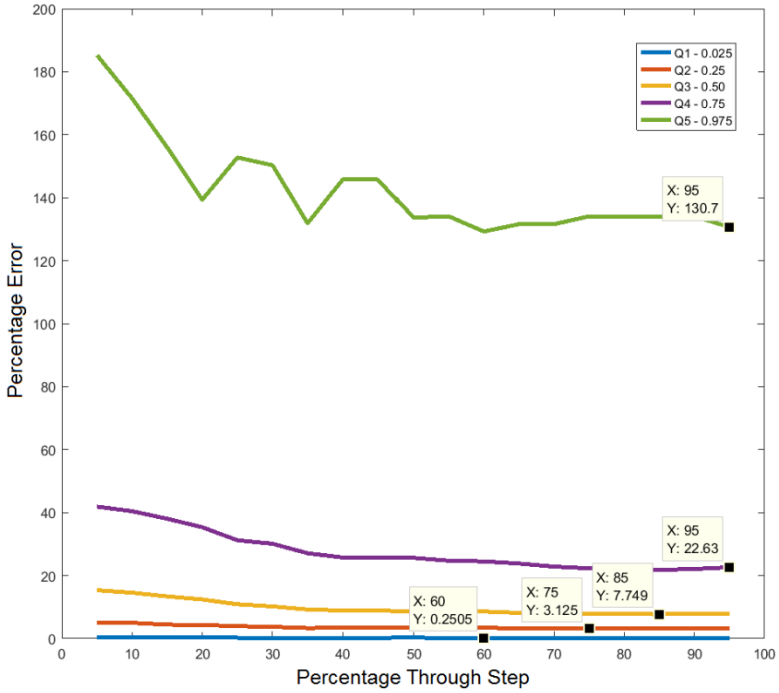

Figure 21: Best Case Quantile Errors throughout the Predicted Step Test Data

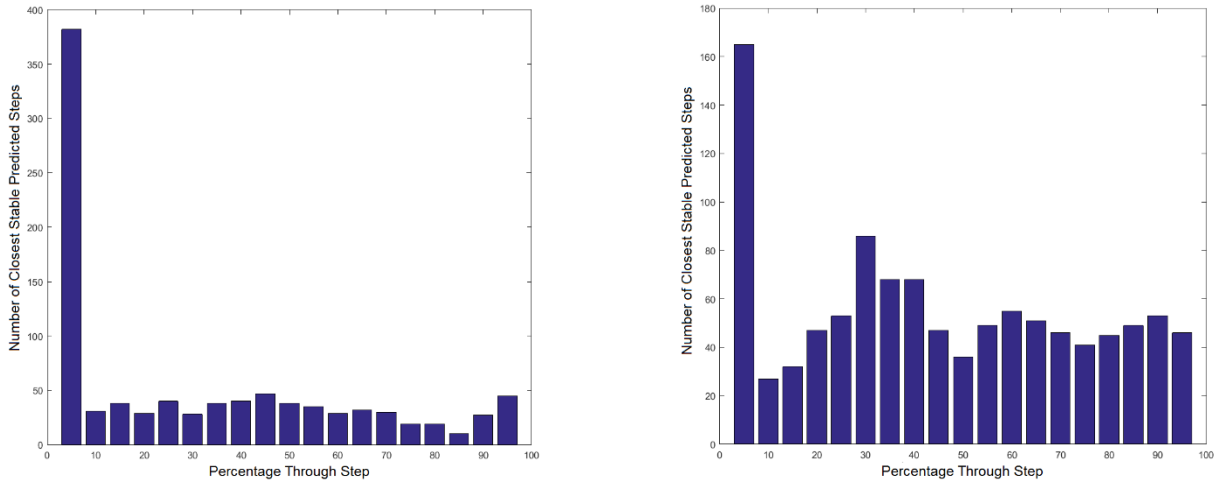

Figure 22: Percentage through Steps until Stable Aggregate Prediction for Best Case (left) and the Realistic Case (right)

These results from Figures 20 and 21 are further supported by understanding the percentage throughout the predicted step for when the step is stable and at which point the final aggregate prediction does not fluctuate. Figure 22 shows how the realistic case has a significantly greater amount of steps that do not arrive at a stable prediction for the first 5% of the stepping stride. As a result, other step classifications are considered throughout the IP process where the fluctuation in stable prediction occurs. In the best prediction accuracy case, there is minimal amount of fluctuation beyond the 5% step completion, in which many instances only a single step is considered for winning the overall linear step distance prediction.

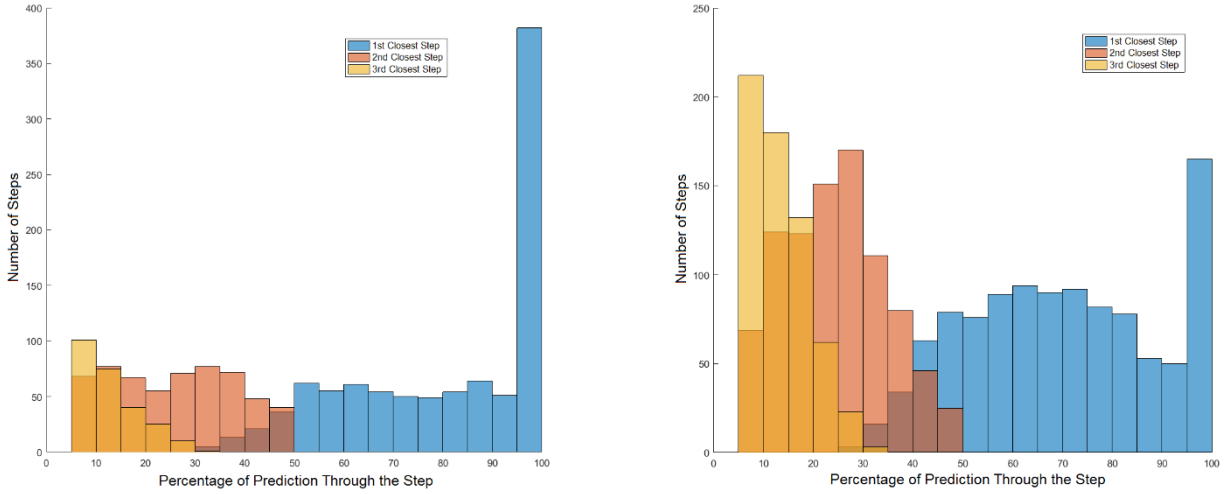

Figure 23: Percentage of 3 Most Frequent Interval Predictions for Best Case (left) and Realistic Case (right)

It was necessary to further analyze the interval step prediction frequencies and understand the influence of the aggregate prediction by the 2nd and 3rd place runner-up steps that were considered throughout the IP process, shown by Figure 23. Once the data for each incoming step has been fully processed, the system makes a prediction for each of the windows in the step where a record of all predicted window intervals is recorded. Of course the step with the most IP votes is considered as the best prediction choice, however; the second and third most voted results are compared to this majority vote. From further analysis of the two test cases in Figure 23, it was found that the best case often is able to make a stable overall prediction of the linear step distance on the very soon into the start of the window IPs. The 1st most voted step choice, represented by the blue bars, have a greater number of steps in the 100% range because many of the steps in both the best and realistic cases are predicted consistently without fluctuation. The same prediction is repeated for

each interval window prediction, thus resulting in no second or third step consideration. The instances which do consider other steps throughout the IPs are shown by the orange and yellow bars. In the best case test, the 2nd and 3rd most voted predictions do not share a significant portion of the vote for most of the step. In contrast, the realistic case has twice as few steps predicted as the best case for the first window prediction. Many more step IPs consider include and 2nd and 3rd most voted step in the realistic case. Furthermore, these runner-up steps account for a larger percentage of the vote, with an increase the 10%-40% ranges. This supports the reasoning behind the unstable fluctuation shown by the realistic case in Figure 22.

In our example, the "first step" would have a point in the bin at p1%, and the "second step" would have a point in the bin at p2%. In this example these were the only two possible predictions, so he "third step" would have a point in the bin at 0%. The 0% bin is not shown in the graph. Figure 24 also shows the prediction percentage difference between the chosen prediction step and the second most frequent runner-up step. What the results in this figure implies is that the second most likely predicted steps in the realistic case are much stronger contender and considered a larger percentage of the time throughout the step.

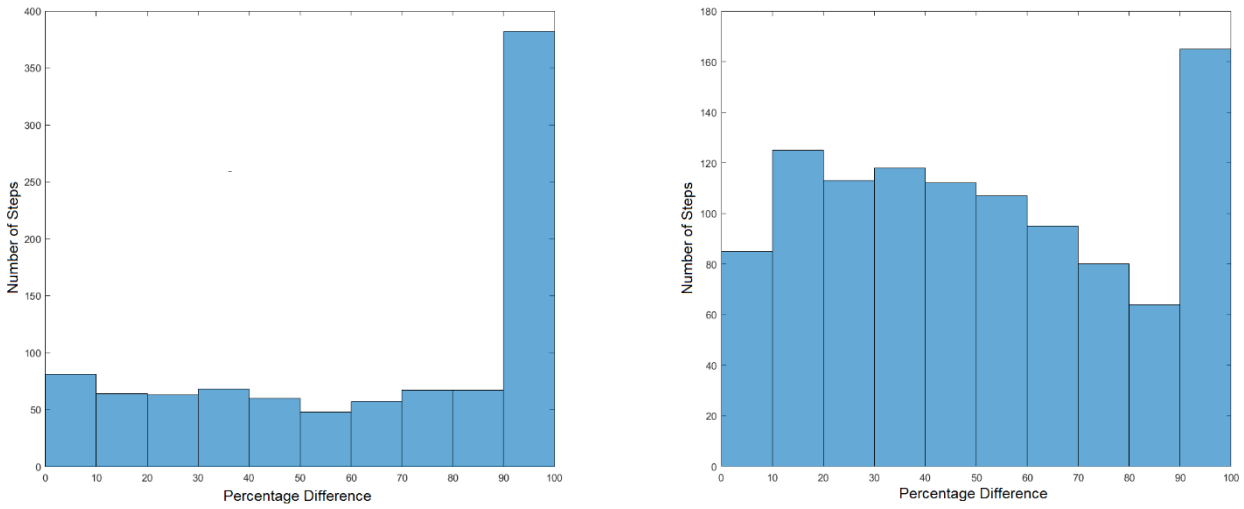

Figure 24: Percentage Difference of Interval Prediction Occurrence between the Best Predicted Step and 2nd Most Frequent Predicted Step for Best Case (left) and Realistic Case (right)

It is necessary to note that the test case of including average acceleration and velocity components of the window in the feature space performs worse than when these features are not included, shown by Table 1. Although the prediction performance has worse accuracy

overall, the case where these average acceleration and velocity features are included tend to make a stable prediction throughout the step much sooner, shown by Figure 25. In either case, an important designated outcome is lacking, which may or may not be application driven depending on its implementation. If a stable prediction is needed more in advance for each step, then the case which forfeits overall prediction accuracy is a more suitable feature space for use. However, since prediction is necessary ahead of time for determining intent of the user, then this is likely a more desirable choice so long as the overall prediction accuracy is acceptable.

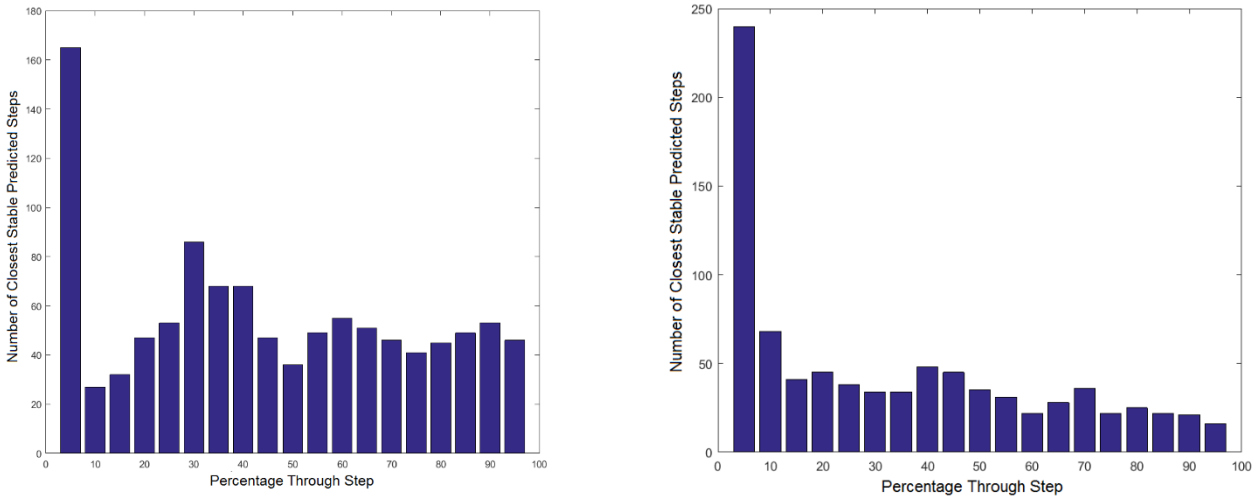

Figure 25: Percentage through Steps until Stable Aggregate Prediction for Realistic Case (left) and Realistic Case with Added ACA and ACV Features (right)

Lastly, it was desirable to understand the relationship between the linear step distance prediction error and the linear distance of the entire step itself. Figure 26 highlights this relationship by showing a clear trend in higher step distance prediction error compared to its stride distance. This result is fairly expected as most of the steps performed in the experimental trials were performed at an average and consistent walking pace. Therefore, there are fewer step instances in which shorter strides and shuffled stepping occur that the prediction model may rely on. Common occurrences for shorter steps are when the user stops walking, performs a turn around, transitions from different walking modes, and encounters an obstacle. Had the focus of this experiment been on predicting irregular and shuffled stepping, then more data would be present and the prediction model would be expected to perform better for these types of stepping behavior. That is not to say that the prediction model algorithm in this work cannot handle such stepping behavior, but merely

that the data collected was focused primarily on flat, stair, and sloped walking profiles with normal stepping behavior. More data for smaller steps were not considered as it would drastically increase the data set size for the machine learning prediction model and changed performance. Since more step classification types would be considered, a greater number of measured step mode exemplars would be needed in order to distinguish the larger variety of steps considered.

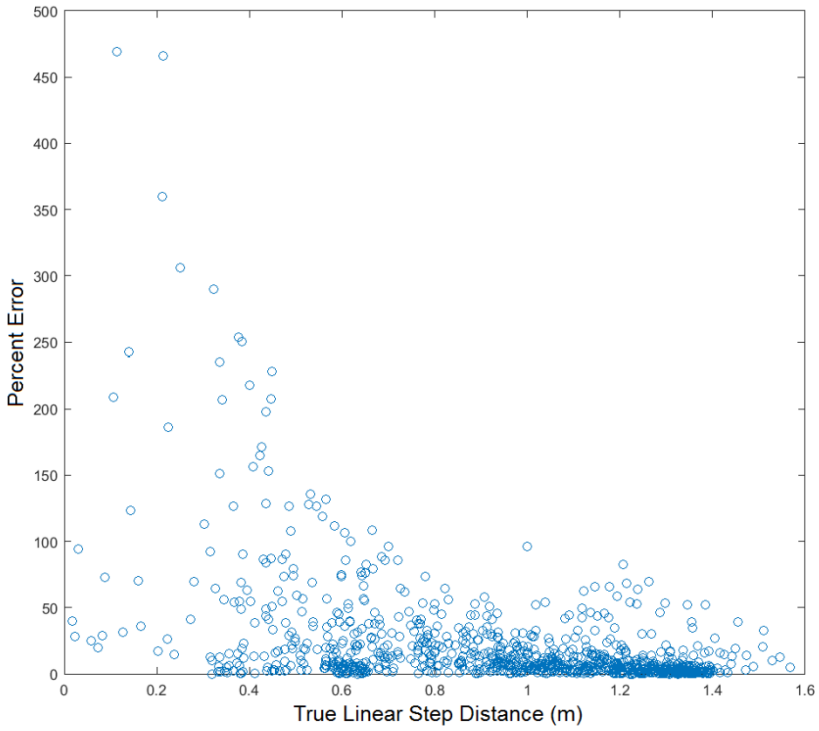

Figure 26: *Prediction Error with Respect to the True Linear Step Distance being Predicted for the Best Case Test*

The results from all of the combined walking modes and step types shows promise under realistic and best case circumstances. Although the best case feature space dimensionality is very large and computationally demanding, it is able to perform well. The realistic case without the acceleration and velocity physical features also provides acceptable results which accurately classify the step, even though minimal characterization is needed. Further investigation of the individual walking modes is necessary in order to better understand more regarding their contribution to the successful predictions of the fully combined results.

The feasibility of real-time performance is considered for the best case feature set test scenario. On average, it can take anywhere from 0.5 to 1.5 seconds to predict a step, depending on its speed and stride. The full data set of over 1000 steps was used with a window size of 20. Within the analysis, Matlab is used to calculate the total elapsed time that each interval prediction requires on average, and the total time needed to process all predictions until the end of the step when the foot is planted. A 2013 MacBook Pro with an i7 Core was the hardware used for this testing. It was found that the average interval prediction processing time for the best case feature set is 0.0082 seconds for 209 IP windows, and the full step processing time resulted in taking 1.71 seconds, on average. The time required for prediction is not fast enough for real-time. A subsampled prediction rate can be employed, however; it is expected that the prediction may lose accuracy. For instance, a new prediction can be made after collecting 5 new data measurements instead of after every new collected data measurement. The realistic case feature set under the same test metrics predicted step IPs on average at 0.0015 seconds and the full step time at 0.32 seconds for 209 IP windows, on average. The realistic case elapsed prediction times are acceptable for real-time prediction, but of course do not yield as great of accuracy as the best case. The prediction time elapsed for the best case is astonishingly five times slower than that of the realistic case, which reveals that the larger feature space hinders the speed performance of the predictions. These elapsed times could potentially be much smaller if the system's hardware and software elements were more optimized.

5.3 Individual Walking Mode Analysis

Testing for each flat, stair, and sloped ground profile walking modes were performed in order to gain a better understanding of the natural elements of interest traveled in this experiment. Each mode of walking is separated and analyzed individually to fully expose its standalone classification accuracies and their influence to the classification of the data set as a whole. The full data set of roughly 1060 steps is used with a window size of 20, and the best case scenario feature set with RA and RV were used. It is found in this work that the flat linear step distance predictions have the highest prediction success rate with roughly 93% of the flat walking linear step distance predictions having no more than 25% error, and roughly 75% of the predictions having no higher than 10% error. The features used in this test include raw acceleration and raw velocity without averaged components. These results,

shown in Figure 27, for the flat walking have the highest prediction accuracies throughout this work. This is expected as the step predictions only pertain to one single type of step classification, even though the linear step distances vary. The errors for the 2nd and 3rd place flat walking steps are slightly higher compared to the most frequently predicted step.

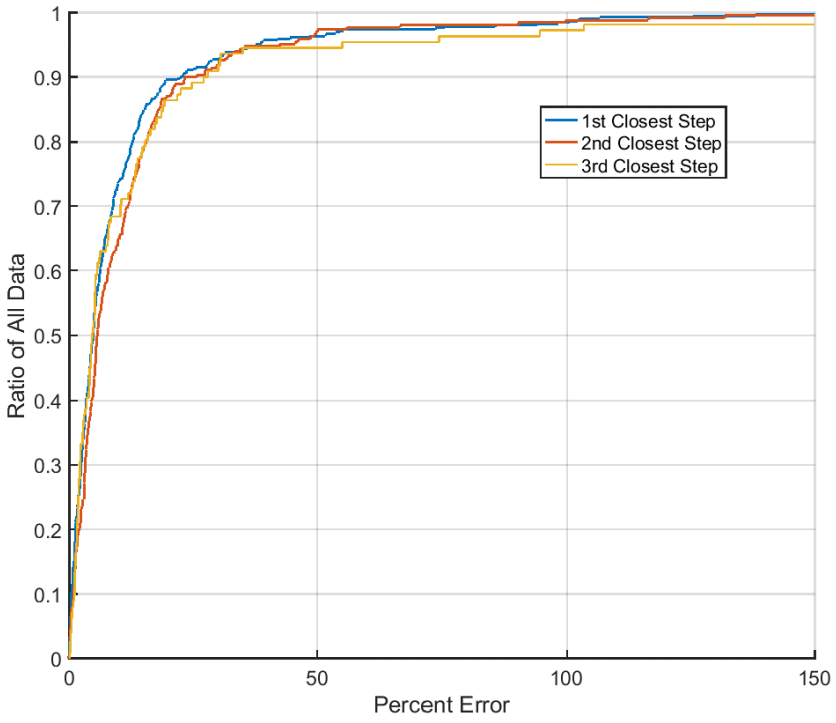

Figure 27: *Cumulative Distribution Function Plot for the 3 Most Frequently Predicted Steps for Flat Walking Only*

For further analysis, the normal distribution for the 3 most frequent IPs for each over all prediction is shown in Figure 28. The overall total number of the 2nd and 3rd most frequent step predictions is shown to be much less than the most frequently occurring prediction, meaning that many of the steps were assigned its best stable distance prediction at the very beginning of the process. Compared to even the 2nd most frequent prediction, the step error distribution is spread out across to higher errors noticeably more than for the 3rd most frequent prediction. There is some difference between the 1st and 2nd most frequent prediction with there being a fewer total number of steps. Figure 28 shows that the 2nd most frequent prediction has more steps with errors about the -25% and 25% region, where the distribution curve is shown to be wider.

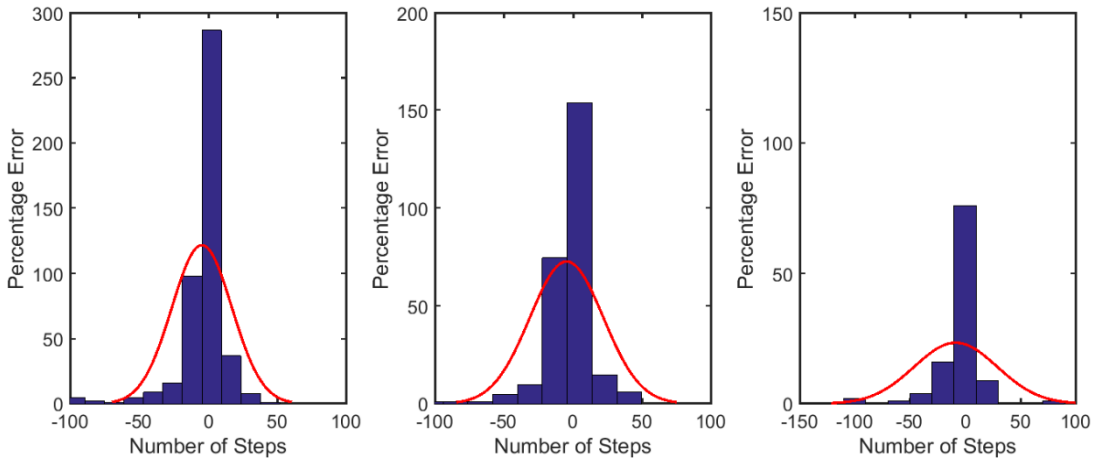

Figure 28: *Normal Distribution Plot for the Best Aggregate Step Prediction (left), 2nd Closest Step Prediction (center) and 3rd Closest Step Prediction (right)*

Figure 29 shows the quantile accuracies for the linear step distance of the flat walking mode and supports the results displayed in Figure 1. The 4th quantile representing 75% of the data step, or roughly 360 of the 480 flat walking steps, consists of 9.3% or less error. This is by no surprise dramatically more accurate compared to commingled data set test results as it pertains to only one single mode of controlled walking.

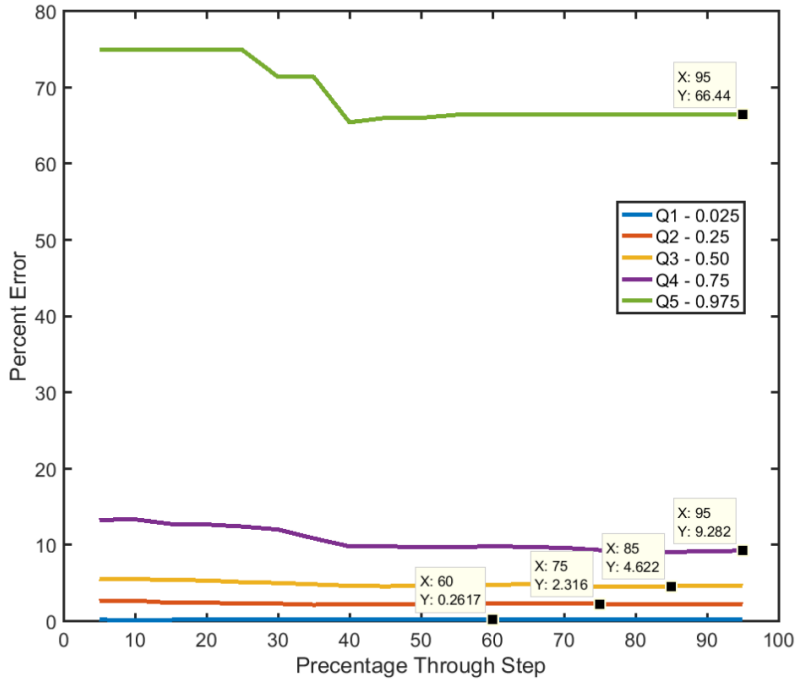

Figure 29: *Flat Walking Quantile Errors throughout the Predicted Step Test Data*

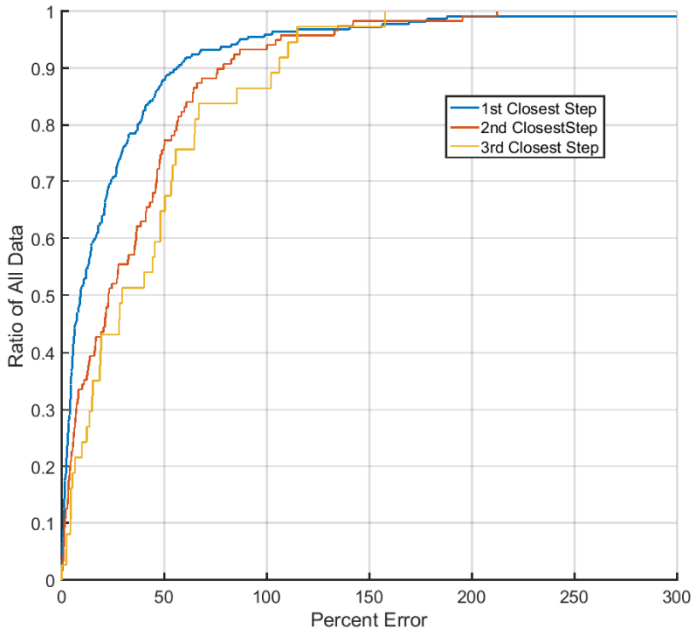

Figure 30: *Cumulative Distribution Function Plot for the 3 Most Frequently Predicted Steps for Stair Walking Only*

Similar results are presented for the stair walking and sloped walking modes by the cumulative distribution function shown in Figures 30 and 31, respectively. The stair walking mode presents larger error shown by the cumulative distribution function plots. However the quantile-error results, in Figure 32, shows to have slightly improved errors for the 2.5%, 25%, and 50% quantiles. The prediction model performs better for each mode individually than it does when all of the modes are combined. This desirable and expected performance gives promise to the full prediction model system's performance since each individual walking mode test predicts well enough to provide higher standalone accuracies. It would be of great concern if a particular walking mode performed worse than as the combined system's performance as that would indicate a significant weak point. The combined system is greatly influenced by the prediction capability of each individual walking mode, and therefore is largely impacted by the walking mode with the lowest quality of performance.

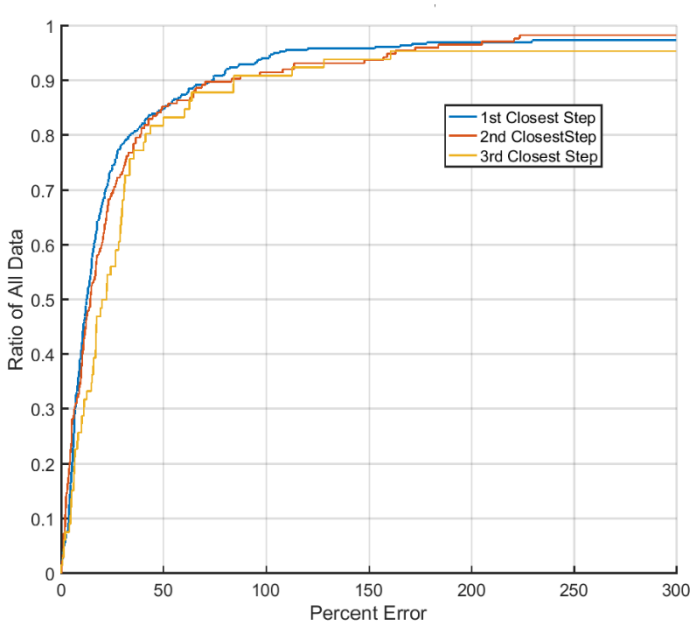

Figure 31: *Cumulative Distribution Function Plot for the 3 Most Frequently Predicted Steps for Sloped Walking Only*

The sloped walking performance shown in Figure 31 is more similar to that of the flat walking performance than the stair walking performance. The greatest walked slope angle during data collection is no steeper than 35 degrees and has a more similar terrain profile to flat ground than to stairs. For this reason the sloped walking performance results are comparable to the best performing individual mode of flat walking.

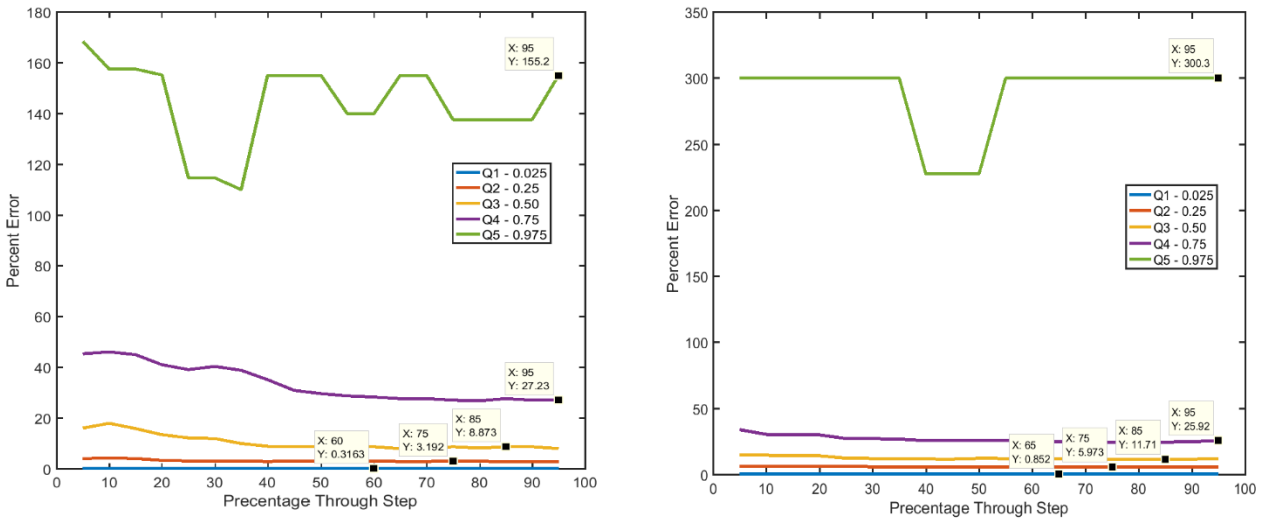

Figure 32: *Stair Walking (left) and Sloped Walking (right) Quantile Errors throughout the Predicted Step Test Data*

Included in the stair walking data set are the transitions between walking modes. For example, when the user steps from flat ground onto an inclining or declining stair, the linear step distance is sometimes larger than normal stair steps. The distance is dependent on the preparation of the foot when walking to comfortably transition from flat ground onto the stair step. These transitions are responsible for the increase in prediction error shown in Figure 30. However, Figure 32 shows that much of the error is acceptable, compared to the other walking modes and combined prediction tests, and that the transitions do not greatly influence the prediction accuracies for the majority of steps with lower error.

Though the results show that the individual mode prediction performances are sufficient, the question of misclassification is raised in regard to the predictions when the steps for each mode are commingled. After manual analysis, it was found that very few steps are misclassified as an incorrect modes. The most common misclassification occurs between the stair and slope modes. A few examples are shown in Figure 33 where misclassification occurs.

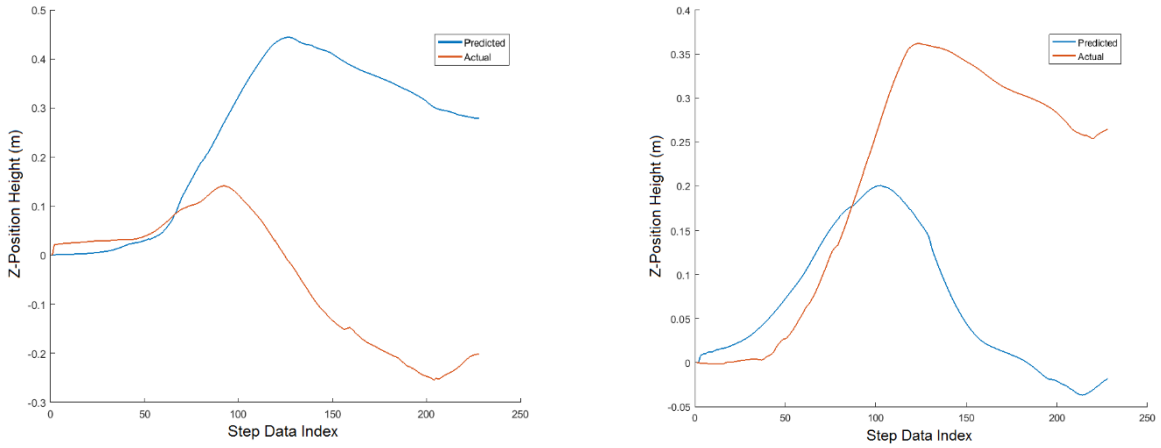

Figure 33: *Profiles of Sloped Step Misclassification (left) and Stair Step Misclassification (right)*

The nature of the individual walking modes are relatively similar, which makes the prediction model susceptible misclassification. Out of 120 manually assessed steps, only 3 were found to have been misclassified between stair and slope steps, and only 2 were misclassified with flat steps. The error rate of less than 4.2% for this misclassification is fortunately very low and does not threaten the overall accuracy of the prediction model when all walking step mode types are combined. Since sloped walking behavior is somewhat in between the behaviors of the flat walking and stair walking modes, it is understandable that

the algorithm may misclassify a few less distinguishable instances. A few instances of the step mode misclassifications are presented in Figures 34 and 35. Although misclassification between walking mode types does occur, it is uncommon and it may be detected or avoided through simple future algorithm improvements.

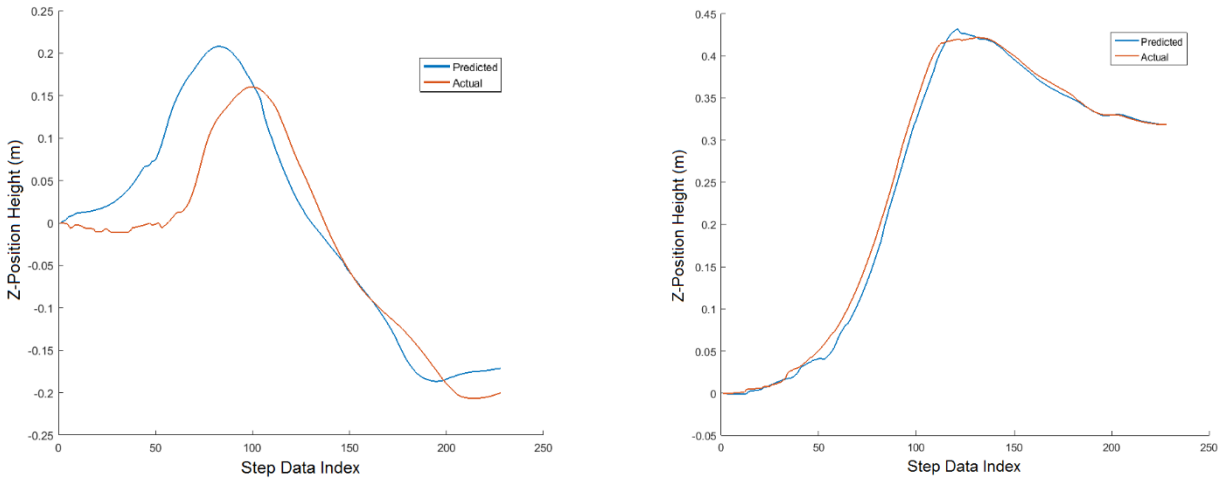

Figure 34: *Profiles of Correct Sloped Step Classification (left) and Correct Stair Step Classification (right)*

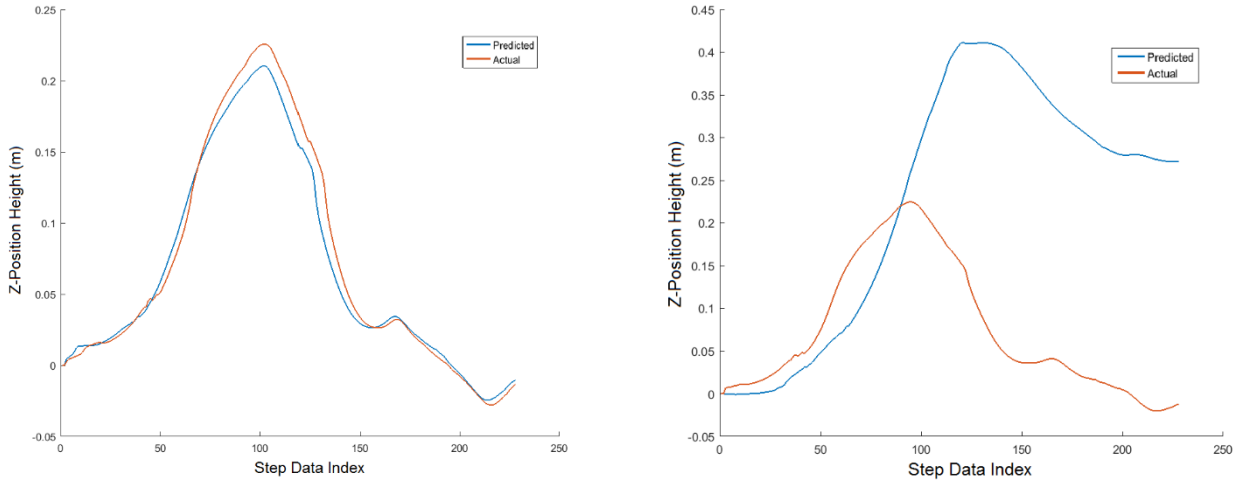

Figure 35: *Profiles of Correct Flat Step Classification (left) and Flat Step Misclassification (right)*

Chapter 6 Conclusions and Future Work

6.1 Conclusions

In the span of this research, an algorithm has been developed and tested for the prediction of natural human gait stepping behavior for determining future linear step distances. The machine learning implementation included in this work has presented not only a method for this purpose, but also a novel approach to predicting future behaviors to provide insight into planned walking and locomotion. A Matlab software implementation, analysis and full documentation has been included in this literature which describes the full methodology of this research. The in-depth details and explanation it covers allows for the adaptation of this work to other research applications and systems alike.

This work is largely desired and motivated by worldwide academic interest for improving exoskeleton technology to better assist human locomotion. The findings and work performed in this research is meant to greatly benefit and progress future efforts in developing advanced wearable robotic systems that can be implemented commercially for human needs. The prediction model framework developed in this work is presented as a tool for researchers and students to apply to future work. It is also desired to serve as an experimental lesson that highlight all failures and successful findings in the performed work. The experimental results detailed in Chapter 5 provide insights of a unique prediction method for determining future step positions and use related works provided in Chapter 2 to offer an alternative approach which does rely on heavily complex parametric algorithms that require extensive processing power. Through this approach, a low overhead system with accurate and quick prediction may be applied to applications which require specific information enough time in advance for critical decision making.

Although there are many advantageous, the developed KNN model used for prediction has several limiting factors which make its application in real-time situations debatable. The primary concern for the KNN is its inability to predict quickly given large data sets. This in theory limits the number of different instances or states which the algorithm can predict. A large number of exemplars are extremely desirable to improve prediction accuracy and

dependency. However, this large amount of data demands more data point comparisons and computation within the primary routine of the KNN algorithm. For this reason, the possible use cases for this system is limited to less complex systems which do not require large datasets to accurately represent the nature of the model's behaviors. A less detrimental but still notable limitation of the prediction model is preprocessing of the collected data. The current cleaning of the measured data is not suitable for real-time application. Though, the careful design of a data filter should be more then capable of correcting the real-time measurements with little to no dependency on the other recorded series data.

For optimal results from this system, it is recommended to use it with terrain which is not too complex and that does not have too many different states. Uneven terrain with a larger number of identifiable footstep states introduces greater complexity into the system which requires more information to be provided for accurate prediction results. This conflicts with the large data set limitation and will hinder real-time performance. Additionally, it is recommended to use a reduced sampling frequency for higher acquisition rates. Though the Xsens system used in this work operates on 240 Hz, it is not necessary to use all recorded measurements and subsampling may be adopted.

In conclusion, the machine learning prediction model framework developed in this work leaves a lot of room for expansion and improvement. Future developments in the growing field of wearable robotics will likely focus largely on improving accuracy and providing more advanced systems for human-assisted actuation and sensing. The industry standard electromechanical event triggered technology of today still struggles with gauging a user's intent and initiating action for providing desired system performance ahead of time. Machine learning capability offer great robustness and implementation which can be integrated in many applications and generalized across different system architectures regardless of their specific requirements. The presented work plays an integral role of scratching the surface for the development of sophisticated computer software of such applicable nature. Increasingly intelligent systems have been a sought after reality for researchers and industry for decades, and the demand has since proven to continuously grow toward greater achievement.

6.2 Future Work

The findings in this work show promising potential for improving future applications of exoskeleton technologies. Implementation of machine learning in wearable human co-dependent applications is becoming an increasing interest among academic research and continuous to push toward being the industry standard.

6.2.1 System Improvements

Experimentation with the machine learning algorithm and Xsens IMU sensor package has revealed several areas for improvement. The machine learning framework developed in this research is not optimized for commercial application settings. Much of the academic and theoretical development completed in this work could be improved and further expanded upon in order to create a sophisticated real-time system.

Two primary limitations in this work which may be greatly improved include the hardware and software. Matlab was used on a 2013 laptop computer using an Intel® core i5 processor for the data process and analysis. The setup was sufficient for the work performed in this research but it would in most cases fail to meet real-time requirements in an actual application setting. A computer system with fast enough processing specifications would greatly improve the processing performance of the system. Although an improved hardware unit would greatly benefit this work, it goes without saying that the software of choice plays just as large of an importance, if not more, in the overall processing performance of the system. Matlab offers a suitable environment for testing data analytics, providing reliable algorithms developed by professional programmers, and interfacing for developing prototype software routines. However, there are better options for developing more advanced software runtime applications which may be highly optimized for system performance. The drawback to this approach is that making these performance changes would require a complete software rewrite, a project that could be feasibly accomplished only by a highly skilled software developer and researcher. The overall cost of such a project in terms of labor time could be tremendous, but worth the benefit nonetheless.

As mentioned previously in Chapter 3 and the conclusion, the machine learning algorithm used in this work has its processing limitations when dealing with larger training data sets. For each exemplar, a comparison is made in order to determine if it is a nearest neighbor fit for consideration in the probability prediction formula. Faster and more complex

algorithms would make for a better choice in regards to prediction performance in real-time because they can perform with the same speed regardless of exemplar set size. The crucial requirement for these more advanced algorithms is having acceptable performance and accuracy of predictions, like the KNN machine learning algorithm used in this work. A more sophisticated algorithm is desirable for future work which may involve implementation into various real-time industry applications.

An appealing improvement for this system is the organization of data by binning similar terrain exemplars. Through the use of other sensors, it can be known ahead of time if there is any difference in the surrounding ground profile which the user may approach. In such cases, it would be significantly less time consuming to only consider data instances that relate to the terrain profile which a user is currently walking. By organizing the data in bins, it reduces the number of comparisons and computation the KNN algorithm must perform in order to provide a prediction. Since the KNN algorithm performs slowly when testing new data with large data sets, the binning of multiple sets and KNN predictor models per terrain instance would be extremely advantageous.

The Xsens sensor package is a well-developed technology with many great benefits, as is. Though the debatable accuracy of the data, discussed in Chapter 4, has been accepted and overall not important in this work, it is an improvement which could greatly benefit future work concerning IMU-based motion prediction. Additionally, in several cases of the experimental trials, some of the sensor units would shift during walking, and in one case the lower leg unit completely detached. Such improvements to the Xsens system were not necessary to complete this work, however; they could provide great benefit to future works which depend on greater accuracy and sensor stability.

6.2.2 Future Experiments

The Virginia Tech ARL has many goals in sight for future work which aim to improve human walking and exoskeleton locomotion. The research performed in this work supports a number of future experiments that are of interest to the ARL. Utilizing the knowledge of a user's intent and algorithmic future prediction opens up many doors for improving the performance of electromechanical exoskeleton systems. The primary future work of interest that stems from the work in this literature is the prediction of robotic actuators and physical state on a user's behalf. By knowing the location of the user's foot placement ahead of time, the robotic system is afforded the opportunity to adjust motor torques and actuation in-line

to a human operators actions. In correctly doing so, the user may theoretically walk effortlessly as the system acknowledges and fully supports the wearer's intent during motion. This level of confidence of a user's intent may allow such a system with enough data and understanding to act on behalf of the wearer, making the performance semi-autonomous. Full autonomy of the system is a desirable goal but comes with much greater capacity to understand the user's intention and decision making.

It would be beneficial to include multiple IMU sensors along each leg for the predictive modeling of next linear step distance. As mentioned previously, measurements from the other IMU sensors along the shin, thigh, and hip plays a prominent role in the estimation of velocity and position. However, the collected data from these IMUs are not used in the current machine learning prediction model. The inclusion of data from these IMU sensors may provide better features which are more characteristic of the step behavior. Additionally, based on the physical models of walking and my experience gained from this research, it is likely that the following features may be useful: distance from the foot take-off position to the current position in the step, and the joint angle about the ankle from the foot and shin IMUs. It is crucial to consider that system complexity and real-time complication is likely occur as more data processing and usage of various data features will demand greater processing power. Furthermore, other measurement systems may not include several linked IMU sensors which could make such an approach impractical. Regardless of these issues, the addition of other IMU data features for prediction may greatly improve the model accuracy for predicting next linear step distance.

The ARL has an increasing interest for the use of machine learning applications in much of the exoskeleton research performed. This research effort is one of the first done in support of the ARL research group and has offered much insight into the possibilities of analyzing assistive robotics behavior and predicting actions in the future. Ongoing projects in the ARL do not yet include such integration and influence of machine learning, but it is seen from this work how they also may be able to benefit from similar findings. Several current studies of heavy focus by students within the ARL center on the development of novel actuators and support systems for upper and lower body appendages. Incorporation of a machine learning subsystem level could offer actuation assistance by providing event recognition and virtual realization of a user's intent. Ultimately, the incorporation of machine learning applications similar to the work in this study may be included in nearly all applications of assistive robotics, and may one day be streamlined such that it is customary in modern day exoskeleton technology.

Bibliography

[1] R. Begg and J. Kamruzzaman, “A machine learning approach for automated recognition of movement patterns using basic , kinetic and kinematic gait data,” *J. Biomech.*, no. 38, pp. 401–408, 2005.

[2] A. E. Patla and J. N. Vickers, “How far ahead do we look when required to step on specific locations in the travel path during locomotion?,” *Exp. Brain Res.*, pp. 133–138, 2003.

[3] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, “Activity Recognition from Accelerometer Data,” pp. 1541–1546, 2005.

[4] A. Mannini and A. M. Sabatini, “Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers,” *MDPI Open Access - Sensors*, vol. 10, pp. 1154–1175, 2010.

[5] A. M. Dollar and H. Herr, “Lower Extremity Exoskeletons and Active Orthoses: Challenges and State-of-the-Art,” vol. 24, no. 1, pp. 144–158, 2008.

[6] G. Barbareschi, R. Richards, M. Thornton, T. Carlson, and C. Holloway, “Statically vs dynamically balanced gait: Analysis of a robotic exoskeleton compared with a human,” *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, pp. 6728–6731, 2015.

[7] A. B. Zoss, H. Kazerooni, and A. Chu, “Biomechanical Design of the Berkeley Lower Extremity Exoskeleton (BLEEX),” *IEEE/ASME Trans. Mechatronics*, vol. 11, no. 2, pp. 128–138, 2006.

[8] J. S. Matthis and B. R. Fajen, “Visual Control of Foot Placement When Walking Over Complex Terrain,” *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 40, no. 1, pp. 106–115, 2014.

[9] A. T. Asbeck, S. M. M. De Rossi, I. Galiana, Y. Ding, and C. J. Walsh, “Stronger , Smarter , Softer : Next Generation Wearable Robots,” pp. 1–11, 2014.

[10] R. J. Farris, H. A. Quintero, and M. Goldfarb, “Performance Evaluation of a Lower Limb Exoskeleton for Stair Ascent and Descent with Paraplegia,” in *34th Annual International Conference of the IEEE EMBS*, 2012, pp. 1908–1911.

[11] H. Banaee, M. U. Ahmed, and A. Loutfi, “Data Mining for Wearable Sensors in Health Monitoring Systems: A Review of Recent Trends and Challenges,” *MDPI Open Access - Sensors*, vol. 13, pp. 17472–17500, 2013.

[12] M. Liu, D. Wang, and H. Huang, “Development of an Environment-Aware Locomotion Mode Recognition System for Powered Lower Limb Prostheses,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 24, no. 4, pp. 434–443, 2016.

[13] R. K. Begg, M. Palaniswami, and B. Owen, “Support Vector Machines for Automated Gait Classification,” *IEEE Trans. Biomed. Eng.*, vol. 52, no. 5, pp. 828–838, 2005.

[14] M. Zhang and A. A. Sawchuk, “A Feature Selection-Based Framework for Human Activity Recognition Using Wearable Multimodal Sensors,” 2011.

[15] A. Akl, C. Feng, and S. Valaee, “A Novel Accelerometer-Based Gesture Recognition System,” *IEEE Trans. Signal Process.*, vol. 59, no. 12, pp. 6197–6205, 2011.

[16] T. Oates, L. Firoiu, and P. R. Cohen, “Clustering Time Series with Hidden Markov Models and Dynamic Time Warping,” 1999.

[17] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, and Y. Amirat, “Physical Human Activity Recognition Using Wearable Sensors,” *MDPI Open Access - Sensors*, vol. 15, pp. 31314–31338, 2015.

[18] C. Randell and H. Muller, “Context Awareness by Analysing Accelerometer Data,” 2000.
[19] L. Peterson, “K-nearest neighbor,” *Scholarpedia*, 2009. .
[20] T. Srivastava, “Introduction to k-nearest neighbors : Simplified,” *Analytics Vidhya*, 2014. .
[21] S. Thirumuruganathan, “A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm,” 2010. .
[22] D. Roetenberg, H. Luinge, and P. Slycke, “Xsens MVN : Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors,” *Tech. Rep.*, vol. 3, pp. 1–7, 2013.
[23] J. Zhang, A. C. Novak, B. Brouwer, and Q. Li, “Concurrent validation of Xsens MVN measurement of lower limb joint angular kinematics,” *Physiol. Meas.*, vol. 34, no. 8, pp. 63–69, 2013.

Appendix

load_mvnx.m

```
function mvnx = load_mvnx(filename)
% mvnx = load_mvnx(filename)
% loads a mvnx file
% filename is name of file (including path)
% mvnx is result struct containing all data of the mvnx file
% Xsens Technologies BV 28-05-2015

%% check filename
if isempty(strfind(filename,'mvnx'))
    filename = [filename '.mvnx'];
end
if ~exist(filename,'file')
    error([mfilename ':xsens:filename'],['No file with filename: ' filename ', file is not present or file
has wrong format (function only reads .mvnx)'])
end
%% read data
fid = fopen(filename, 'r', 'n', 'UTF-8');
content = char(fread(fid,'char')');
fclose(fid);
%% get data into a cell array, one cell per line
lines = find(content == 10);
cellContent = cell(1,length(lines));

hWaitbar = waitbar(0, 'MVNX import: importing data...');
for n=1:length(lines)
    if n==1
        cellContent{n} = content(1:lines(n)-1);
    else
        cellContent{n} = content(lines(n-1)+1:lines(n)-1);
    end
end
% look for comment lines and remove them
cellContent(cellfun(@(x) x(1)=='-' || (x(1)=='<' && (x(2)=='?' || x(2)=='!')),cellContent)) = [];

% get start of text and clean up some
index = cellfun(@(x) find(x==60,1),cellContent);
openandclose = cellfun(@(x) sum(x==60)==2,cellContent); % lines with have an opening and
closing statement in one line
cellContent = cellfun(@(x) x(find(x==60):end), cellContent,'UniformOutput',false);

%% get start and end words
```

```
n = 0;clear value name
iWord = 0;
word = cell(1,length(cellContent));
wordindex = cell(1,length(cellContent));
wordvalue = cell(1,length(cellContent));
wordfields = cell(1,length(cellContent));
while n < length(cellContent)
    n=n+1;
    line = cellContent{n};oneword = false;
    hooks = find(line == 62);
    iWord = iWord+1;
    wordindex{iWord} = index(n);
    if any(line == 32) && ~openandclose(n)
        if ~isempty(hooks) && hooks(1) < find(line==32,1)
            word{iWord} = line(2:hooks(1)-1);
            iLine = hooks(1)+1;
        else
            word{iWord} = line(2:find(line==32,1)-1);
            iLine = find(line==32,1)+1;
        end
    elseif ~isempty(hooks) && length(line)>=8 && strcmp('comment',line(2:8))
        % add exception for comment
        word{iWord} = line(2:hooks(1)-1);
        iLine = hooks(1)+1;
    elseif openandclose(n)
        word{iWord} = line(2:find(line==62,1)-1);
        iLine = find(line==62,1)+1;
    else
        word{iWord} = line(2:end-1);
        oneword = true;
    end
    if word{iWord}(1) ~= '/'
        if ~oneword && ~openandclose(n)
            k = find(line == 34);
            k = reshape(k,2,length(k)/2)';
            l = [iLine find(line(iLine:end) == 61)+iLine-2];
            fieldname = cell(1,length(l)-1); value = cell(1,length(l)-1);
            if ~isempty(k)
                for il=1:size(k,1)
                    fieldname{il} = line(iLine:find(line(iLine:end) == 61,1)+iLine-2);
                    if size(k,1) > 1 && il < size(k,1)
                        a = strfind(line(iLine:end),''' ')+iLine+1;
                        iLine = a(1);
                    end
                    value{il} = line(k(il,1)+1:k(il,2)-1);
                end
            else
                value = []; fieldname =[];
                value = line(find(line == 62,1)+1:end);
            end
        elseif ~oneword && openandclose(n)
```

```
            value = []; fieldname =[];
            value = line(find(line == 62,1)+1:find(line==60,1,'last')-1);
        else
            value = NaN;fieldname = [];
        end
        wordvalue{iWord} = value;
        wordfields{iWord} = fieldname;
    end
end
isendword = cellfun(@(x) x(1) == '/',word);
endwords = cellfun(@(x) x(2:end),word(isendword),'UniformOutput',false);
kindofendwords = unique(endwords);
placeoffirststartword = zeros(1,length(kindofendwords));
placeofallendwords = cell(1,length(kindofendwords));
placeofallstartwords = cell(1,length(kindofendwords));
for n=1:length(kindofendwords)
    lengthWord = length(cell2mat(kindofendwords(n)));
    placeoffirststartword(n) = find(strncmp(word,kindofendwords(n),lengthWord),1);
    placeofallstartwords{n} = find(strncmp(word,kindofendwords(n),lengthWord));
    placeofallendwords{n} = find(strncmp(word,['/' kindofendwords{n}],lengthWord+1));
end
[a b] = sort(placeoffirststartword);
startwords = kindofendwords(b);
for n=1:length(startwords)
    obj.(startwords{n}).number = length(placeofallstartwords{n});
    obj.(startwords{n}).index = index(a(n));
    obj.(startwords{n}).count = 0;
end

%% get values
for n=1:length(wordvalue)
    if mod(n,5000) == 0
        waitbar((n/length(wordvalue))/2, hWaitbar);
    end

    if iscell(wordvalue{n})
        if length(wordvalue{n}) == 1
            B = [];
            try
                B = str2num(wordvalue{n}{1});
            end
            if ~isempty(B)
                wordvalue{n} = B;
            else
                wordvalue{n} = wordvalue{n}{1};
            end
        else
            for m=1:length(wordvalue{n})
                try
                    B = str2num(wordvalue{n}{m});
                    if ~isempty(B)
```

```
                    wordvalue{n}{m} = B;
                end
            end
        end
    end
else
    try
        B = str2num(wordvalue{n});
        if ~isempty(B)
            wordvalue{n} = B;
        end
    end
end
end
%% put everything in struct
firstFramesFound = false;
nFramesToBeFound = 5;
nFramesFound = 0;
nFrames = length(cell2mat(placeofallendwords(strcmp(kindofendwords,'frame'))));
lastwordisendword = false;lastendword = '';
superstruct = struct;name = cell(1,max(index)+1);namecounter = ones(1,max(index));endplace
= inf(1,max(index));
for iWord=1:length(word)
    if mod(iWord,5000) == 0
        waitbar(iWord/length(word) + 0.5, hWaitbar);
    end
    if word{iWord}(1)=='/'
        startword = word{iWord}(2:end);
        if any(strcmp(startword,name))
            name(find(strcmp(startword,name),1):end) = [];
        end
        if lastwordisendword
            obj.(lastendword).count = 0;
        end
        lastendword = startword;
        lastwordisendword = true;
    else
        lastwordisendword = false;
        try
            if ~isempty(wordfields{iWord})
                wordfields{iWord} = cellfun(@(x) x(x~=32), wordfields{iWord},'UniformOutput',false);
            end
            word(iWord) = checkText(word(iWord));
            wordfields{iWord} = checkText(wordfields{iWord});

            if any(strncmp(startwords,word{iWord},length(startwords)))
                if strcmp(word{iWord},'frame')
                    nFramesFound = nFramesFound + 1;
                    if ~firstFramesFound && nFramesFound >= nFramesToBeFound
                        firstFramesFound = true;
                        % preallocate frames in superstruct
```

```
                % Assume remainder of the file contains frames,
                % with exception of security code at the end
                frames(1:nFrames,1) = superstruct.mvnx.subject.frames.frame(end);
                frames(1:nFramesFound-1) =
superstruct.mvnx.subject.frames.frame(1:nFramesFound-1);
            end
            if firstFramesFound
                fields = wordfields{iWord};
                for il = 1:length(fields)
                    frames(nFramesFound).(fields{il}) = wordvalue{iWord}{il};
                end
            end
        end
        obj.(word{iWord}).count = obj.(word{iWord}).count +1;
        name{index(iWord)} = word{iWord};
        namecounter(index(iWord)) = obj.(word{iWord}).count;
        if ~firstFramesFound && (iscell(wordvalue{iWord}) || ~isempty(wordvalue{iWord}) &&
all(~isnan(wordvalue{iWord})))
            superstruct =
setvalue(superstruct,index(iWord),wordvalue{iWord},name(1:index(iWord)),wordfields{iWord},na
mecounter);
        end
    elseif iWord > 1 && strcmp(word{iWord},word{iWord-1})
        name{index(iWord)} = word{iWord};
        namecounter(index(iWord)) = namecounter(index(iWord))+1;
        superstruct =
setvalue(superstruct,index(iWord),wordvalue{iWord},name(1:index(iWord)),wordfields{iWord},na
mecounter);
    else
        name{index(iWord)} = word{iWord};
        namecounter(index(iWord)) = 1;
        if firstFramesFound && ~strcmp(word{iWord},'securityCode')
            frames(nFramesFound).(word{iWord}) = wordvalue{iWord};
        else
            superstruct =
setvalue(superstruct,index(iWord),wordvalue{iWord},name(1:index(iWord)),wordfields{iWord},na
mecounter);
        end
    end
  catch e
    disp(getReport(e))
  end
 end
end
if firstFramesFound
 superstruct.mvnx.subject.frames.frame = frames';
 if obj.mvnx.number > nFramesFound
  superstruct.mvnx.subject.frames.frame(nFramesFound+1:end) = [];
 end
end
```

```
%% get output
mvnx = superstruct.mvnx;

delete(hWaitbar);
end
function superstruct = setvalue(superstruct,index,value,name,fields,namecounter)
%superstruct = setvalue(superstruct,index,value,name,fields,namecounter)
% add values to a very big struct
if ~isempty(fields) && (~iscell(fields) || ~isempty(fields{1}))
   if length(fields) == 1
      name(end+1) = fields;
   else
      name{end+1} = fields;
   end
   index = index+1;
end
switch index
   case 1
      if ~iscell(name{1})
         superstruct.(name{1}) = value;
      else
         for il = 1:length(name{1})
            superstruct.(name{1}{il}) = value{il};
         end
      end
   case 2
      if ~iscell(name{2})
         superstruct.(name{1})(namecounter(1)).(name{2}) = value;
      else
         for il = 1:length(name{2})
            superstruct.(name{1})(namecounter(1)).(name{2}{il}) = value{il};
         end
      end
   case 3
      if ~iscell(name{3})
         superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})...
            = value;
      else
         for il = 1:length(name{3})
            superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3}{il}) =
value{il};
         end
      end
   case 4
      if ~iscell(name{4})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...
            .(name{4}) = value;
      else
         for il = 1:length(name{4})
```

```
superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
..
                .(name{4}{il}) = value{il};
            end
        end
    case 5
        if ~iscell(name{5})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...
                .(name{4})(namecounter(4)).(name{5}) = value;
        else
            for il = 1:length(name{5})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...
                .(name{4})(namecounter(4)).(name{5}{il}) = value{il};
            end
        end
    case 6
        if ~iscell(name{6})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...
                .(name{4})(namecounter(4)).(name{5})(namecounter(5)).(name{6}) = value;
        else
            for il = 1:length(name{6})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...
                .(name{4})(namecounter(4)).(name{5})(namecounter(5)).(name{6}{il}) = value{il};
            end
        end
    case 7
        if ~iscell(name{7})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...
                .(name{4})(namecounter(4)).(name{5})(namecounter(5)).(name{6})(namecounter(6))...
                .(name{7}) = value;
        else
            for il = 1:length(name{7})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...

.(name{4})(namecounter(4)).(name{5})(namecounter(5)).(name{6})(namecounter(6))...
                .(name{7}{il}) = value{il};
            end
        end
```

```
    case 8
        if ~iscell(name{8})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...
                .(name{4})(namecounter(4)).(name{5})(namecounter(5)).(name{6})(namecounter(6))...
                .(name{7})(namecounter(7)).(name{8}) = value;
        else
            for il = 1:length(name{7})

superstruct.(name{1})(namecounter(1)).(name{2})(namecounter(2)).(name{3})(namecounter(3)).
...

.(name{4})(namecounter(4)).(name{5})(namecounter(5)).(name{6})(namecounter(6))...
                    .(name{7})(namecounter(7)).(name{8}{il}) = value{il};
            end
        end
end
end

function word = checkText(word)
%word = checkText(word)
% make sure the field name is just text and allowed symbols.
if ~isempty(word)
    if ~iscell(word)
        error('should get cells')
    end
    for i=1:length(word)
        if length(word{i})>1
            word{i}(word{i}=='!') = [];
            word{i}(word{i}==':') = '_';
            if word{i}(end) == '/'
                word{i}(end) = [];
            end
            while (word{i}(1) < 65 || word{i}(1) > 122 || (word{i}(1)> 90 && word{i}(1) < 97))
                word{i}(1)=[];
            end
        end
    end
end
end
```

load_xsens_data.m

```
function [tree, segmentData, sensorData, segmentCount, nSamples] =
load_xsens_data(filename)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

% load data
tree = load_mvnx(filename);
% read some basic data from the file
%mvnxVersion = tree;
%fileComments = tree.subject.comment;
%read some basic properties of the subject;
%frameRate = tree.subject.frameRate;
%suitLabel = tree.subject.label;
%originalFilename = tree.subject.originalFilename;
%recDate = tree.subject.recDate;
segmentCount = tree.subject.segmentCount;
%retrieve sensor labels
%creates a struct with sensor data
if isfield(tree.subject,'sensors') && isstruct(tree.subject.sensors)
    sensorData = tree.subject.sensors.sensor;
else
    sensorData = [];
end
%retrieve segment labels
%creates a struct with segment definitions
if isfield(tree.subject,'segments') && isstruct(tree.subject.segments)
    segmentData = tree.subject.segments.segment;
else
    segmentData = [];
end
%retrieve the data frames from the subject
nSamples = length(tree.subject.frames.frame);

end
```

stepPredictionKFoldStatistics.m

```
set(0,'defaultaxesfontsize',14)
set(0,'defaulttextfontsize',14)
set(0,'defaultaxeslinewidth',1.5)
set(0,'defaultAxesFontName','Arial')
set(0,'defaultTextFontName','Arial')

filename1 =
'C:\Users\Bryan\Documents\VirginiaTech\Research\ResearchCode\research_ongoing\xsens_da
ta_collection\Bryan_Walkcourse';
if(~exist('testRoot','var') || strcmp(loadedFile1, filename1) == 0)
   [testRoot, segmentData, sensorData, segmentCount, nSamples1] =
load_xsens_data(filename1);
   startIter1 = 1;
   while testRoot.subject.frames.frame(startIter1+1).time < 1
      startIter1 = startIter1+1;
   end
   samplesVec1 = startIter1:nSamples1;
   nSamples1 = length(samplesVec1);
   loadedFile1 = filename1;
end

segIndexR = 18; % Right Foot
segIndexL = 22; % Left Foot
segDataR = zeros(nSamples1,9);
segDataL = zeros(nSamples1,9);
if isfield(testRoot.subject.frames.frame(1), 'position')
   for i=samplesVec1(1):samplesVec1(end)
      idx=1+i-samplesVec1(1);
      tVec(idx) = testRoot.subject.frames.frame(i).time;

      if length(testRoot.subject.frames.frame(i).acceleration) >= 3+(segIndexR-1)*3
         segDataR(idx,1:3) = testRoot.subject.frames.frame(i).acceleration(1+(segIndexR-
1)*3:3+(segIndexR-1)*3);
         segDataR(idx,4:6) = testRoot.subject.frames.frame(i).velocity(1+(segIndexR-
1)*3:3+(segIndexR-1)*3);
         segDataR(idx,7:9) = testRoot.subject.frames.frame(i).position(1+(segIndexR-
1)*3:3+(segIndexR-1)*3);
      else
         segDataR(idx,:) = zeros(1,9);
      end
      if length(testRoot.subject.frames.frame(i).acceleration) >= 3+(segIndexL-1)*3
         segDataL(idx,1:3) = testRoot.subject.frames.frame(i).acceleration(1+(segIndexL-
1)*3:3+(segIndexL-1)*3);
         segDataL(idx,4:6) = testRoot.subject.frames.frame(i).velocity(1+(segIndexL-
1)*3:3+(segIndexL-1)*3);
         segDataL(idx,7:9) = testRoot.subject.frames.frame(i).position(1+(segIndexL-
1)*3:3+(segIndexL-1)*3);
      else
```

```
            segDataL(idx,:) = zeros(1,9);
        end
    end
end

N = nSamples1;
[sIdxRAll, eIdxRAll] = scanSteps(segDataR(1:N,:), 60);
[sIdxLAll, eIdxLAll] = scanSteps(segDataL(1:N,:), 60);

sampleFactor = 1.0;
subsampleStepsR = randsample(1:length(sIdxRAll),round(length(sIdxRAll)*sampleFactor));
sIdxR = sIdxRAll(subsampleStepsR);
eIdxR = eIdxRAll(subsampleStepsR);
subsampleStepsL = randsample(1:length(sIdxLAll),round(length(sIdxLAll)*sampleFactor));
sIdxL = sIdxLAll(subsampleStepsL);
eIdxL = eIdxLAll(subsampleStepsL);

length(sIdxR)
length(sIdxL)

winSize = 20;
interpSize = round(mean([eIdxR-sIdxR+1 eIdxL-sIdxL+1]));
[featVecR, distVecR, timeVecR, dataVecR, ~] = knnGetStepData3(testRoot, segDataR, sIdxR,
eIdxR, interpSize, tVec, winSize);
[featVecL, distVecL, timeVecL, dataVecL, interpSize2] = knnGetStepData3(testRoot, segDataL,
sIdxL, eIdxL, interpSize, tVec, winSize);

featVec = [featVecR; featVecL];
distVec = [distVecR; distVecL];
dataVec = [dataVecR; dataVecL];

%% K-Fold Statistical Analysis
percentSpace = 20;
ls = linspace(1,interpSize2,20);
percentVec = [];
percentDistVec = [];
percentStop = [];
distError = [];
stepErrors = [];
predictions = [];
actualDists = [];

K=10;
[~, bl, ~] = unique(distVec);
indices = crossvalind('Kfold', bl, K);
idx = [];
for k=1:length(indices)
    idx = [idx; ones(interpSize2,1)*indices(k)];
end

for k=1:K
```

```
testIdx = (idx == k);
testLen = sum(indices == k);
trainIdx = ~testIdx;
trainFeatSet = featVec(trainIdx, :);
trainDistSet = distVec(trainIdx);
testFeatSet = featVec(testIdx, :);
testDistSet = distVec(testIdx);

trueLabels = indices ~= k;
labels = zeros(sum(indices~=k)*interpSize2,1);
it = 1;
for t=1:length(trueLabels)
    if(trueLabels(t))
        labels(((it-1)*interpSize2)+1:it*interpSize2) = t*ones(1,interpSize2);
        it = it + 1;
    end
end

tic;
mdl_KNN = fitcknn(trainFeatSet, labels, 'NumNeighbors', 8);
predKNN = predict(mdl_KNN, testFeatSet);
toc
size(testFeatSet)
testLen

kFoldRunActualDists = [];
[u1,u2,~] = unique(labels);
for i=1:testLen
    M = mode(predKNN(((i-1)*interpSize2)+1:i*interpSize2));
    mIdx = find(M==u1);
    stepPred = trainDistSet((u2(mIdx(1))));
    stepAct = testDistSet(((i-1)*interpSize2)+1);
    predictions = [predictions; stepPred];
    actualDists = [actualDists; stepAct];
    kFoldRunActualDists = [kFoldRunActualDists; stepAct];
    stepErrors = [stepErrors; [abs(stepPred-stepAct), abs(stepPred-stepAct)/stepAct*100]];
end

for i=1:testLen
    percentAct = [];
    percentVal = [];
    prev = -1;
    percentError = [];

    for j=2:percentSpace
        M = mode(predKNN(((i-1)*interpSize2)+1:((i-1)*interpSize2+(round(ls(j))))));
        mIdx = find(M==u1);
        percentAct = [percentAct, M];
        V = trainDistSet((u2(mIdx(1))));
        percentVal = [percentVal, V];
```

```
            percentError = [percentError, abs(V-
kFoldRunActualDists(i))/kFoldRunActualDists(i)*100];

            if M ~= prev
                last = (j-1)*5;
            end
            prev = M;
        end
        percentStop = [percentStop, last];
        percentVec = [percentVec; percentAct];
        percentDistVec = [percentDistVec; percentVal];
        distError = [distError; percentError];
    end
end
interpSize2

up = unique(percentStop);
figure
bar(5:5:(size(histc(percentStop, up),2)*5),histc(percentStop, up))
ylabel('Closest Stable Prediction')
xlabel('Percentage Through Step')
title('Closest Stable Overall Prediction Vs. Percentage of Step Completion')

% Does the error rate decrease as more predictions are collected and
% assessed?
% figure
% plot(5:5:95, distError,'-')

sum(stepErrors(:,2) < 8 & stepErrors(:,1) < 0.076)/size(stepErrors,1)
sum(stepErrors(:,2) < 30 & stepErrors(:,1) < 0.15)/size(stepErrors,1)
sum(stepErrors(:,2) < 30 & stepErrors(:,1) < 0.23)/size(stepErrors,1)
sum(stepErrors(:,2) < 40 & stepErrors(:,1) < 0.31)/size(stepErrors,1)

distErrorAvg = mean(distError,1);
distErrorStd = std(distError,0,1);
distQuantile = quantile(distError,[.025 .25 .50 .75 .975]);

% Quantile Errors
figure
plot(5:5:95,distQuantile','linewidth',3)
xlabel('Precentage Through Step')
ylabel('Percent Error')
legend('Q1 - 0.025', 'Q2 - 0.25', 'Q3 - 0.50', 'Q4 - 0.75', 'Q5 - 0.975')
title('Percent Error for Quantiles Vs. Percentage Through Step')

% Step Error Vs. Step Distance
figure
pIdx = actualDists < 1.6 & stepErrors(:,2) < 500;
plot(actualDists(pIdx), stepErrors(pIdx,2), 'o')
xlabel('True Linear Step Distance (m)')
ylabel('Prediction Error (%)')
```

```
title('Predicted Step Distance Error Vs. True Step Distance')

%%
best3Steps = [];
best3Freqs = [];
for i=1:size(percentVec,1)
    a = unique(percentDistVec(i,:));
    out = histc(percentDistVec(i,:),a);
    [sa, sb] = sort(out,'descend');

    best1 = a(sb(1));
    freq1 = sa(1);
    if length(out) < 2
        best2 = 0;
        freq2 = 0;
        best3 = 0;
        freq3 = 0;
    elseif length(out) < 3
        best2 = a(sb(2));
        freq2 = sa(2);
        best3 = 0;
        freq3 = 0;
    else
        best2 = a(sb(2));
        freq2 = sa(2);
        best3 = a(sb(3));
        freq3 = sa(3);
    end

    best3Steps = [best3Steps; [best1, best2, best3]];
    best3Freqs = [best3Freqs; [freq1/size(percentDistVec,2)*100,
freq2/size(percentDistVec,2)*100, freq3/size(percentDistVec,2)*100]];
end

% This plot shows how close in the overall prediction decision the 2nd step
% is from the 1st. How close are they to being chosen? Are many about
% equal or is it fairly obvious in the decision.
figure
histogram(abs(best3Freqs(:,1)-best3Freqs(:,2)), 0:10:100)
title('Difference Between 1st and 2nd Best Step Prediction Occurrences')
xlabel('Percentage Difference')
ylabel('Number of Steps')

% This plot shows the percentage of occurrence for the 3 most frequent
% intra-steps predictions throughout the entire step prediction process
% and presents the number of steps for each percentile range.
figure
hold on
histogram(best3Freqs(:,1), 5:5:100)
histogram(best3Freqs(:,2), 5:5:100)
histogram(best3Freqs(:,3), 5:5:100)
```

```
title('Percentage of Occurrence of the 3 Most Frequent Intra-Step Predictions')
xlabel('Percentage of Prediction Throughout the Step')
ylabel('Number of Steps')
legend('1st Step', '2nd Step', '3rd Step')

distDiff1 = abs(actualDists - best3Steps(:,1))./actualDists*100;
distDiff2 = abs(actualDists - best3Steps(:,2))./actualDists*100;
distDiff3 = abs(actualDists - best3Steps(:,3))./actualDists*100;
distDiff1(best3Steps(:,1)==0)=[];
distDiff2(best3Steps(:,2)==0)=[];
distDiff3(best3Steps(:,3)==0)=[];

% This shows the normal distribution for the accuracy of the predicted
% steps for the 3 most frequently intra-step predictions
figure
title('Normal Distribution of Step Prediction Errors for the 3 Closest Step Predictions')
subplot(1,3,1)
histfit(distDiff1,20)
title('1st Step')
xlabel('Number of Steps')
ylabel('Percentage Error')
axis([-200 500 0 800])
subplot(1,3,2)
histfit(distDiff2,20)
title('2nd Closest Step')
xlabel('Number of Steps')
ylabel('Percentage Error')
axis([-500 800 0 800])
subplot(1,3,3)
histfit(distDiff3,20)
title('3rd Closest Step')
xlabel('Number of Steps')
ylabel('Percentage Error')
axis([-500 800 0 800])

figure
grid on
hold on
cdf1 = cdfplot(distDiff1);
cdf2 = cdfplot(distDiff2);
cdf3 = cdfplot(distDiff3);
set(cdf1, 'linewidth', 1.5);
set(cdf2, 'linewidth', 1.5);
set(cdf3, 'linewidth', 1.5);
axis([0 400 0 1])
title('Cumulative Distribution Function Plot for the 3 Most Frequent Intra-Step Distance
Predictions')
xlabel('Percent Error')
ylabel('Ratio of All Data')
legend('1st Step', '2nd Step', '3rd Step')
```

knnGetStepData3.m

```
function [featVec, distVec, timeVec, dataVec, featItemSize] = knnGetStepData3(root, footData,
startArr, endArr, interpSize, ttime, winSize)

winOff = winSize-2;
featItemSize = interpSize-winOff-1;
dataVec = zeros(length(startArr),interpSize,9);
if interpSize < 3
   return
end

featVec = [];
distVec = [];
timeVec = [];
for i=1:length(startArr) % number of steps
   offSets = [getFrameItem(root.subject.frames.frame(startArr(i)).acceleration, 18), ...
      getFrameItem(root.subject.frames.frame(startArr(i)).velocity, 18), ...
      getFrameItem(root.subject.frames.frame(startArr(i)).position, 18)];

   L = endArr(i)-startArr(i)+1;
   xq = linspace(0,L,interpSize);
   for j=1:9 % number of features (acc,vel,pos)
      v = footData(startArr(i):endArr(i),j);
      vq = interp1(v,xq);
      dataVec(i,:,j) = vq - offSets(j);
      dataVec(i,isnan(dataVec(i,:,j)),j) = 0;
   end

   stepDist = sqrt((dataVec(i,end,7)-dataVec(i,1,7))^2 + ...
            (dataVec(i,end,8)-dataVec(i,1,8))^2 + ...
            (dataVec(i,end,9)-dataVec(i,1,9))^2);

   for j=2:interpSize-winOff
      Z = j+winOff;
      distComps = [];
      for z=j:Z % 4 values, differences between 5 points
         distComp = sqrt((dataVec(i,z,7)-dataVec(i,z-1,7))^2 + ...
            (dataVec(i,z,8)-dataVec(i,z-1,8))^2 + ...
            (dataVec(i,z,9)-dataVec(i,z-1,9))^2);
         distComp = round(distComp, 4);
         distComps = [distComps, distComp];
      end

      distVec = [distVec; stepDist];

      timeComp = ttime(endArr(i)+j+winOff)-ttime(startArr(i)+j-1);
      timeVec = [timeVec; timeComp];

      % Calc mean rate of change da/dt between 5 acceleration points
```

```
        accX = dataVec(i,(j-1):(j+winOff),1);
        %accMeanRateX = mean(diff(accX));
        accY = dataVec(i,(j-1):(j+winOff),2);
        %accMeanRateY = mean(diff(accY));
        accZ = dataVec(i,(j-1):(j+winOff),3);
        %accMeanRateZ = mean(diff(accZ));

        % Calc mean rate of change da/dt between 5 acceleration points
        velX = dataVec(i,(j-1):(j+winOff),4);
        velMeanRateX = mean(diff(velX));
        velY = dataVec(i,(j-1):(j+winOff),5);
        velMeanRateY = mean(diff(velY));
        velZ = dataVec(i,(j-1):(j+winOff),6);
        velMeanRateZ = mean(diff(velZ));

        totalDistComp = sqrt((dataVec(i,Z,7)-dataVec(i,j,7))^2 + ...
              (dataVec(i,Z,8)-dataVec(i,j,8))^2 + ...
              (dataVec(i,Z,9)-dataVec(i,j,9))^2);
        totalDistComp = round(totalDistComp, 4);

        % All features
        %featVec = [featVec; [timeComp, dataVec(i,Z,9)-dataVec(i,j,9), totalDistComp, distComps,
accX, accY, accZ, velX, velY, velZ]];
        %featVec = [featVec; [timeComp, dataVec(i,Z,9)-dataVec(i,j,9), totalDistComp, distComps,
accX(1:2:end), accY(1:2:end), accZ(1:2:end), velX(1:2:end), velY(1:2:end), velZ(1:2:end)]];
        featVec = [featVec; [timeComp, dataVec(i,Z,9)-dataVec(i,j,9), totalDistComp, distComps]];
    end
end

end
```

scanSteps.m

```
function [startIdxVec, endIdxVec] = scanSteps(inputData, minStep)
% INPUTS
% inputData - Nx9 acc, vel, pos data vector
% minStep - minimum data points for new step

% OUTPUTS
% startIdxVec - 1xM vector of start indices
% endIdxVec - 1xM vector of end indices

startIdxVec = [];
endIdxVec = [];

zeroAccPts = inputData(:,1) < 0.25 & inputData(:,1) > -0.25 & ...
    inputData(:,2) < 0.25 & inputData(:,2) > -0.25 & ...
    inputData(:,3) < 0.25 & inputData(:,3) > -0.25;
zeroVelPts = inputData(:,4) < 0.5 & inputData(:,4) > -0.5 & ...
    inputData(:,5) < 0.5 & inputData(:,5) > -0.5 & ...
    inputData(:,6) < 0.5 & inputData(:,6) > -0.5;

idx = find(zeroAccPts & zeroVelPts);
L = length(idx)

% Filter outliers
idxDelete = [];
for i=1:L-1
    if idx(i+1)-idx(i) > 8
        idxDelete = [idxDelete; i];
    end
end

idx(idxDelete) = [];
L = length(idx)

for i=1:L-1
    if idx(i+1)-idx(i) > minStep
        startIdxVec = [startIdxVec idx(i)];
        endIdxVec = [endIdxVec idx(i+1)];
    end
end

if length(inputData) - idx(end) > minStep
    startIdxVec = [startIdxVec idx(end)];
    endIdxVec = [endIdxVec length(inputData)];
end

% figure
% hold on
```

```
% plot(inputData(:,9))
% plot(find(zeroAccPts==1 & zeroVelPts==1), inputData(zeroAccPts & zeroVelPts,9), 'k.')
% plot(startIdxVec, inputData(startIdxVec, 9), 'bo')
% plot(endIdxVec, inputData(endIdxVec, 9), 'ro')
% a=1;
end
```