PLANETARY NAVIGATION ACTIVITY RECOGNITION USING WEARABLE ACCELEROMETER DATA

by

WEN SONG

B.S., Beijing Jiaotong University, 2011

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical & Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2013

Approved by:

Major Professor
Steve Warren

Copyright

Abstract

Activity recognition can be an important part of human health awareness. Many benefits can be generated from the recognition results, including knowledge of activity intensity as it relates to wellness over time. Various activity-recognition techniques have been presented in the literature, though most address simple activity-data collection and off-line analysis. More sophisticated real-time identification is less often addressed. Therefore, it is promising to consider the combination of current off-line, activity-detection methods with wearable, embedded tools in order to create a real-time wireless human activity recognition system with improved accuracy.

Different from previous work on activity recognition, the goal of this effort is to focus on specific activities that an astronaut may encounter during a mission. Planetary navigation field test (PNFT) tasks are designed to meet this need. The approach used by the KSU team is to pre-record data on the ground in normal earth gravity and seek signal features that can be used to identify, and even predict, fatigue associated with these activities. The eventual goal is to then assess/predict the condition of an astronaut in a reduced-gravity environment using these predetermined rules.

Several classic machine learning algorithms, including the k-Nearest Neighbor, Naïve Bayes, C4.5 Decision Tree, and Support Vector Machine approaches, were applied to these data to identify recognition algorithms suitable for real-time application. Graphical user interfaces (GUIs) were designed for both MATLAB and LabVIEW environments to facilitate recording and data analysis. Training data for the machine learning algorithms were recorded while subjects performed each activity, and then these identification approaches were applied to new data sets with an identification accuracy of around 86%. Early results indicate that a single three-axis accelerometer is sufficient to identify the occurrence of a given PNFT activity.

A custom, embedded acceleration monitoring system employing ZigBee transmission is under development for future real-time activity recognition studies. A different GUI has been implemented for this system, which uses an on-line algorithm that will seek to identify activity at a refresh rate of 1 Hz.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to express my great appreciation to Dr. Steve Warren for supporting and advising me throughout my master's program. I am grateful for the help from the NASA project team led by Dr. Barstow and his students, Carl Ade and Ryan Broxterman, from the KSU Kinesiology department. I express my thanks to Dana Gude for her collaborative efforts on this project, to Devon Krenzel for his work on the design of the prototype accelerometer board, and to David Huddleston at the KSU Electronics Design Laboratory for his excellent soldering work. Advice given by Dr. Das and Dr. Hsu on machine learning algorithms has also been a great help in this research. Most importantly, I offer special thanks to my parents for providing unconditional support and encouragement to me on this academic endeavor from thousands of miles away.

Chapter 1 - Introduction

Activity Monitoring

Activity monitoring has been an area of interest for many years [1-3]. It is widely applied in different applications such as medical care, where patients are monitored by medical staff as they perform a series of rehabilitation activities. Patients with special needs, such as the elderly or people with autism or Parkinson's syndrome, are good candidates for such monitoring approaches because they can be physically unstable [4]. In addition, activity monitoring can give medical staff a better idea of what the patients are experiencing and help them formulate better treatments. Treatment effectiveness and functional improvement can also be measured from follow-on patient activities.

Many people care about their health and their daily activities, and it would be helpful for them to keep track of what they do every day and how long they spend on each activity [5]. Tools that calculate calorie expenditure based on activity intensity are useful for this purpose and can be used to address health problems such as obesity and diabetes that result from a lack of activity. Lifestyle feedback presents a means to prod people to get off the couch and start to move.

Astronauts, in the context of this thesis, are potentially other major users for activity monitoring technologies. Numerous intra-vehicular activities (IVAs) and extravehicular activities (EVAs) can be physically taxing, often requiring astronauts to perform while wearing cumbersome space suits [6, 7]. Information about their health status, their physical strength, and how they feel is limited and often comes from self reporting. Effects of reduced gravity environments on the human body are still largely unknown and must be quantified.

In many application scenarios that employ activity monitoring, traditional monitoring methods require staff to stay with the person to be monitored. A staff member observes and records behaviors and activities as they occur. Although this method has a certain level of accuracy, it is not widely applicable or scalable due to several drawbacks. First, it involves substantial human labor. Constant attention is required from those who monitor these subjects, especially when the activity is irregular and changes quickly. Most of the time, the staff to subject ratio is one to one unless the group of people to be monitored is gathered in a certain area

and performs similarly. Cost is a leading factor that prevents this approach from being widespread. Second, the accuracy and consistency of the monitoring is not guaranteed. Absence of staff is a common cause for incorrect recordings of activity timing and duration. Staff may also be unprepared when events need to be recorded. Third, the data often come in large quantities, making them difficult to record and to transfer from paper to paper. Transcribing manual recordings to electronic form requires tedious additional work and can add errors.

One more recent approach uses video cameras and their support computers to do the activity detection and monitoring [8, 9]. This method is helpful because it is unobtrusive and requires minimal contact between the subject and the staff. Some research efforts have attempted to use machine learning to analyze subject motion and automatically determine the behavior of the people on these videos. However, these methods do not yet provide sufficient accuracy, and monitoring is restricted to small areas [10].

The limitations noted above call for new methods to record and assess activities. Context awareness computing is a rapidly developing field, and the development of micro-electromechanical systems (MEMS) accelerometers and wireless embedded systems can help expand the popularity and feasibility of wireless human condition monitoring systems [11]. Consistent with that theme, acceleration monitoring and analysis systems can be designed to address the problems experienced with traditional activity monitoring methods. A layout for such a system is depicted in Fig. 1.1.

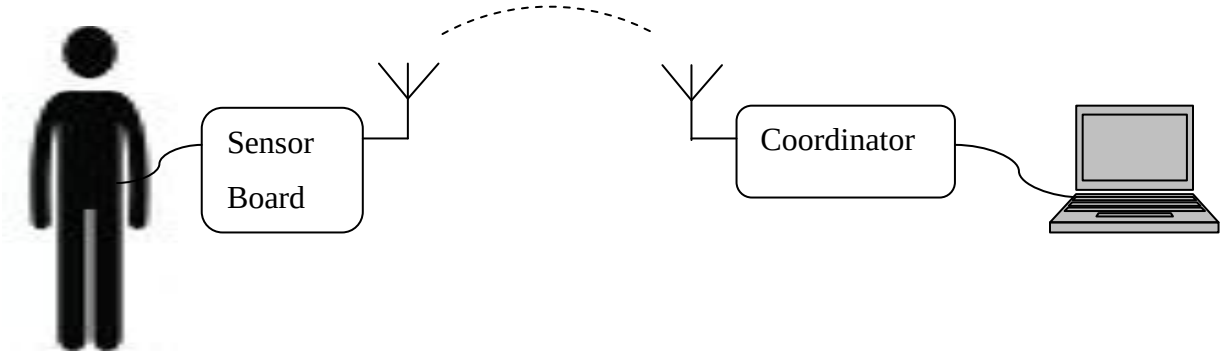

Figure 1.1 System layout for a wireless activity monitoring system.

In this system layout, a sensor board is attached to the body and gathers health data. The sensor board then preprocesses the data to get useful information, which is then transmitted wirelessly to a coordinator connected to a computer that stores these data automatically. Such a

system replaces the cost of human labor and can be used 24 hours a day in various situations, including medical monitoring for senior citizens. For personal training, for example, this approach can be more sensible when compared to some current commercial products, since it can provide more detailed information to track the training process and would not require the user to enter the activity he or she is doing.

In terms of the work presented in this thesis, we apply activity recognition to tasks similar to those that astronauts would perform when working in space. By monitoring an astronaut's behavior, researchers could have a better idea of the work intensity experienced by the astronaut, the type of work the astronaut is performing, and possibly their readiness to complete upcoming tasks, in order to avoid or respond to problematic situations. This might involve adjusting the work load according to the astronaut's performance history so that the tasks assigned to each astronaut match their physical condition and have the best chance to be successfully executed.

Focus Areas for an Activity Recognition System

A good activity recognition system should address the following issues, which relate to the physical boundaries of the system and the related problem areas.

Activity Types

Numerous human activities take place every day, like sleeping, sitting, walking and eating. These activities are divided into two main categories: static activities and dynamic activities [12]. During a static activity, the whole body tends to stay in one position without moving vigorously. Some examples are standing, sitting, lying down face-up (supine) and lying down face-down. A dynamic activity requires body movement, whether from a part of the body (e.g., arm or leg) or the entire body. Some examples are typing on a keyboard, drinking water, walking, running and climbing stairs.

Many activity recognition efforts have investigated different types of activity, including daily work activities [13-15] and activities associated with medical conditions [4]. Most of these recognition processes address specific daily tasks. For the work presented here, the activities of interest are more oriented to special movement patterns that mimic the potential behaviors in and around a space station. Details of these activities are addressed in Chapter 2.

Monitoring Methods

Activity identification can involve three kinds of monitoring approaches: video monitoring, body-attached sensors, and object-attached sensors, all of which have advantages and disadvantages. A video monitor provides a good assessment and record in the presence of people, where humans do the recognition by watching the video and manually categorizing who is in the video and what he/she is doing. Ambient light, camera placement, lens angle, resolution, and what a person wears affect the ability to correctly classify their activities. Sophisticated algorithms have been developed to detect if a person exists in the video. Related research focuses on detecting falls to the ground. The appearance of certain behaviors varies between people, often preventing accurate detection. In addition, it is unrealistic to place a video camera on the body or to set cameras in many places to capture a large area, so a video-based approach to activity monitoring is best suited for restricted indoor activities.

Body-attached sensors attract wide research attention [16][17], as they offer portability and potential ease of use. Sensors can be placed on different body parts, where popular places include the waist, thigh and hip [18]. Movement patterns measured at these locations are similar between different people, and small sensor sizes ensure the ability to carry these sensors for long periods of time.

Object-attached sensors are novel in terms of sensor placement [19], where sensors are intended to be attached to many nearby objects so that users are not troubled by the need to wear the sensors on their bodies. For example, the authors of [19] present the idea of putting sensors on telephones to monitor when people are using these telephones. This idea correlates with the idea of an "Internet of things," which would require huge wireless network coverage. However, some basic activities such as walking and running are not easily associated with physical, static items that can serve as sensor hosts. Besides, when motion is detected by such an object, it would be almost impossible to identify which person is performing the activity unless the person also has some kind of identification tag attached to him or her.

Sensor Types

Many sensors are appropriate for human monitoring, including accelerometers, gyrometers, electrocardiographs, electomyographs, etc. [20, 21]. Previous research shows that biological signals such as heart rate are not good classification signals for two reasons [18]. First,

the parameter range differs significantly between people. Take heart rate as an example. Some individuals may have a very fast heart rate even if they are not doing anything but sitting on a chair, whereas some well-trained athletes can maintain a slow heart rate when doing mild activities like jogging. Second, these signals can have a large temporal delay, meaning a signal does not change instantly as different activities take place. For example, a person who starts running has a heart rate increase for a while, but after he stops, the heart rate could stay high even though he has finished running and sits down. Even though health parameters are of great importance when monitoring, e.g., heart and muscle condition, they are not particularly usefor for activity recognition [22].

Sensor Placement

Sensor placement must effectively support the identification of different body movements [13, 14]. Previous studies have investigated many places for these sensors, such as the chest, wrist, waist, hip, thigh, and ankle. Both individual sensors and multiple-sensor configurations have been analyzed. In general, an increase in the number of sensors helps to raise the accuracy of these techniques. For the astronaut context presented here, the limited space in the space suit prevents us from placing sensors at will, so the minimum number of sensors that can provide data to accurately recognize each activity is desired. Here, we start with a single sensor.

Feature Selection

Well-chosen features provide distinct characteristics of a certain activity [2, 17]. Ideally, with only a single feature, an activity can be identified from all others exclusively [14]. However, this optimal feature does not exist. Instead, in many cases, the definition of good quality means that when a certain feature is applied to different activities, it can help to put these different activities into clusters, where within each cluster, the instances of other activities are as minimal as possible.

Time-domain, frequency-domain and wavelet features have been studied previously for this purpose [4, 23, 24]. Different feature selection methods are utilized to determine which features contribute the most and which are redundant. Features are activity dependent, which suggests that for different activities, different combinations of features are needed. In this work, we therefore consider a wide range of features and rank them according to their usefulness.

Algorithms

In an activity-classification context, an algorithm is a tool to distinguish activities from one another [25]. Two types of algorithms are generally used to determine activity: threshold-based classification and machine-learning-based classification.

Threshold-based classification utilizes a hierarchical structure to form a decision tree. At each node, the threshold is based on a value that is manually picked. This kind of classification has been useful with real time activity recognition scenarios, and the method is straightforward and easy to understand. Its implementation does not require significant programming experience, and it can be readily applied to various fields. Usually, the decision tree is designed by the researcher to reflect the fundamental features of the different activities. However, it suffers from accuracy, and given the multi-dimensional decision space that arises with this method, intuition and observations from experience are difficult to apply given the typical three-dimensional space within which humans normally function. Results from such a decision tree can seem random.

The other type is machine-learning-based classification. The concept is to use programming algorithms to find the optimal approach to distinguish different activities. This method has been well studied in recent years [15]. Many parametric and non-parametric methods are proposed to tackle different situations [26-28].

We considered both types of algorithms in order to understand which one is more effective for PNFT activity identification.

Chapter 2 - Planetary Activity Definition and Data Collection

Motivation

A team consisting of three KSU departments (Kinesiology; Electrical & Computer Engineering; and Mechanical & Nuclear Engineering) is engaged in a three-year effort funded by NASA entitled "Standardized 'Pre-Flight' Exercise Tests to Predict Performance During Extravehicular Activities in a Lunar Environment."[1] One primary purpose of this effort is to better understand fatigue as it relates to the types of extravehicular activities (EVAs) that an astronaut would be expected to complete as a team member on a space station. Further, the investigators seek to identify physiologic mechanisms that could act as precursors for task failure, or means to predict the inability of an astronaut to complete a task. Recent efforts in support of these goals involve the design of a set of exercises, or field tests, that mimic the types of EVAs that astronauts would be expected to perform [29-31]. Electromyographic (EMG) data analyses traditionally play an important role in the determination of fatigue in such applications. Such EMG data, coupled with heart rate, acceleration, and metabolic data (oxygen consumption extracted via inspiration/expiration masks) have formed the centerpiece of the team's data analysis strategy when assessing movement and fatigue. Late in the first year of this effort, team members began a more careful assessment of accelerometer data as an activity parameter, not only with the thought of using accelerometer data to indicate activity type and duration, but also as a potential means to supplement (or serve as a surrogate to) EMG data in the context of fatigue analysis. This chapter summarizes these field test activities and the efforts to collect accelerometer data from subjects that engaged in these field tests.

Activity Definition

Six Planetary Navigation Field Test (PNFT) activities were designed to simulate EVAs that might be performed by astronauts housed in a space station [32, 33],.These six tasks,

[1] "Standardized 'Pre-Flight' Exercise Tests to Predict Performance During Extravehicular Activities in a Lunar Environment," Research and Technology Development to Support Crew Health and Performance in Space Exploration Missions, NASA Human Research Program, Exploration Systems Mission Directorate, Johnson Space Center, Houston, TX, 7/1/2010–6/30/2013.

described in Table 2.1, serve as the standard activities, or classes, that need to be recognized by the classification algorithms investigated during this study. These tasks differ from regular daily activities and involve different movement patterns and intensity. The activities are depicted in Figure 2.1 through Figure 2.6. Each activity is designed to mimic a task an astronaut would perform during an EVA and also quantify a necessary area of functional ability. For example, the Step-Entry Maneuver simulates the actions an astronaut would take when they enter or exit the space station through a small opening. Bending of the body and jumping is required when moving. The Agility Cones test an astronaut's ability to rapidly change direction.

Table 2.1 Planetary Navigation Field Test Activities

1	Ladder Climb	Subjects ascend a 12-foot ladder, walk across the top of the platform, and descend on the other side
2	Agility Cones	Subjects move forward and backward through six cones, always facing forward
3	Stair Climb	Subjects climb a set of stairs and then descend the stairs backwards
4	Horizontal Climb	Subjects climb horizontally along a wall using hand and foot grips
5	Equipment Lift	Subjects lift two 10 lb and two 20 lb equipment boxes from waist- to eye-level and from ground- to waist-level, respectively, then lower them in reverse order to the starting position
6	Step-Entry Maneuver	Subjects move laterally and periodically step over ropes and duck under poles to simulate stepping over and under a hatch entry

Some of these activities share similarities. The ladder climb and stair climb both involve climbing movements and rely on leg muscles, but the stair climb requires a slight forward movement. Such similarities create complexity when trying to distinguish one activity from another. Since these tests are intended to evaluate astronauts' abilities and fatigue, they are intentionally designed to be intense. Energy expenditures may be larger than with most daily activities but similar between the PNFT activities themselves.

Figure 2.1 Ladder climb field test.

Figure 2.2 Agility cones field test.

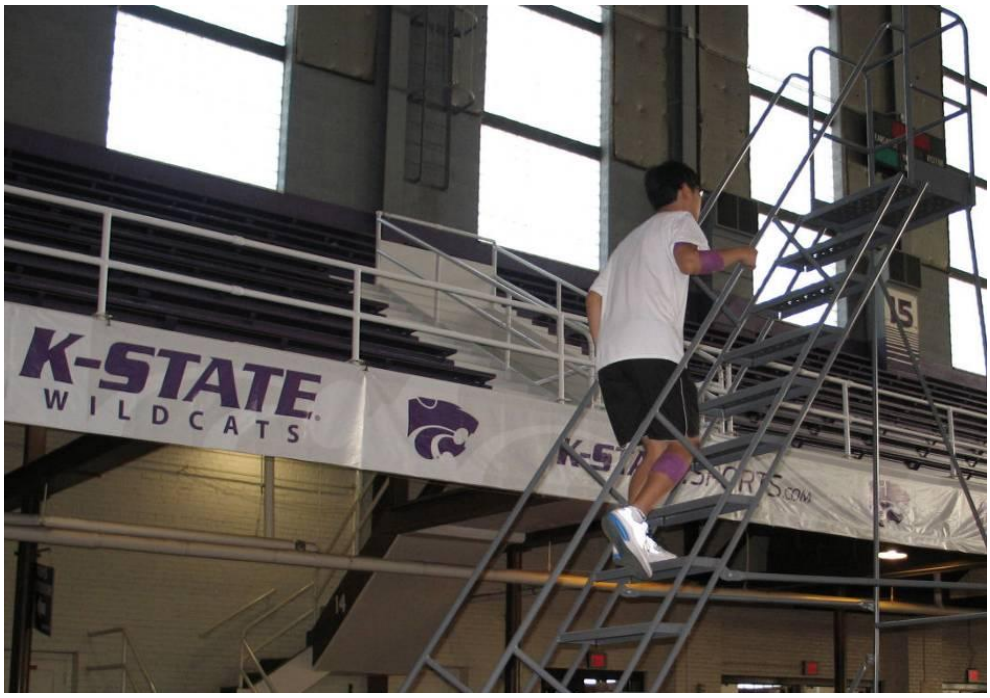

Figure 2.3 Stair climb field test.

Figure 2.4 Horizontal climb field test.

Figure 2.5 Equipment lift field test.

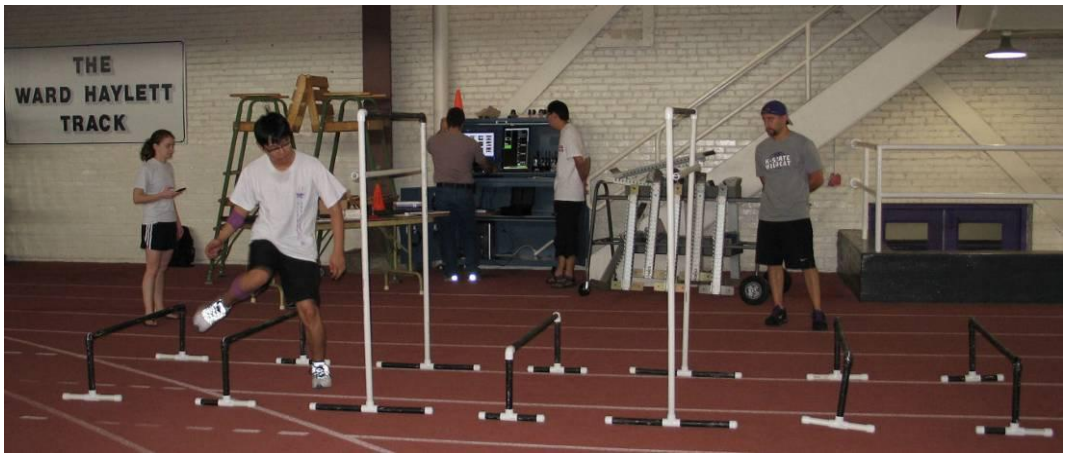

Figure 2.6 Step-entry maneuver field test..

In addition to the above tasks, three regular activities are also included in the tests. They are standing, walking and running [34] – see Table 2.2. These tasks serve three purposes. First, since they are daily-life activities, the classification accuracy for these three activities can be compared to the classification accuracy for the PNFT activities. Second, these activities are less intensive, yet they differ greatly in character from each other, so they also serve as a stepping stone when dealing with real-time activity recognition. In addition, they're helpful since during

the PNFT, subject need to move from one station to next station by running or walking, so they're the transition and if the process is continuous, they need to be identified too.

Table 2.2 Other standard activities included in the tests.

7	Stand	Subjects stand still
8	Walk	Subjects casually walk around a big circle at normal pace
9	Jog	Subjects jog around a big circle at normal pace

Accelerometer-Based Sensing and the Zephyr BioHarness

This study focuses on motion and position sensors. Sensors belonging to this category include accelerometers, gyrometers, magnetic sensors and GPS sensors. A magnetic sensor utilizes the natural magnetic field of the earth, and a GPS sensor reports latitude and longitude of a location. Both of these will be useless when an activity is performed in space. This work therefore migrated toward accelerometer- and gyrometer-based devices. To keep the system simple, a three-axis accelerometer was chosen to be the sensor for activity recognition. It measures acceleration in three directions: vertical, sagittal, and lateral. An accelerometer is regarded as the best motion indicator in previous papers and has been well-studied [35-37]. Accelerometers are excellent activity indicators for these reasons:

1. Different activities exhibit different motion directions, intensities, and patterns; these are reflected in the acceleration data.
2. Acceleration reflects instant motion.
3. An accelerometer does not require the sensor to be firmly fixed to the skin. This makes an accelerometer an ideal sensor when skin contact is prohibited or difficult (e.g., when an astronaut wears a space suit). In fact, some efforts require a loosely placed sensor, as in a pocket, where the orientation of the sensor may change and a calibration routine is periodically required [38].
4. The sampling rate is not high. In the case of human motion, acceleration data are not required to be sampled at even hundreds of hertz, since body motion cannot achieve these frequencies naturally. In the literature, sampling frequencies of 25, 50, and 128 Hz are adopted, and these systems are capable of classification. In contrast, an EMG signal which reflects muscle electrical activity requires the sampling frequency to be

as high as 1000 Hz to fully capture this behavior. This low-frequency trait is favorable, since fewer transmission data mean lower power consumption.

The first activity recognition experiment used the Zephyr BioHarness [39], shown in Figure 2.7. The BioHarness is a wireless health monitoring device that can record various signals, such as three-axis acceleration, heat rate, posture, skin temperature, and ECG. The accelerometer samples at 50 Hz, and the maximum measurable acceleration is 8 g. The device is attached to the chest using an elastic strap, and the sensor is on the left side of body. Bluetooth is the wireless data transmission protocol. It ensures a 100 m transmission range.

Figure 2.7 Zephyr BioHarness

Data Collection

Data were collected over two time intervals using volunteers recruited on campus. This work was performed with the oversight of the KSU Human Studies Board under protocol 5466.

First Data Collection Effort

The first data collection effort focused on four subjects. Two subjects finished all six activities, while the other two subjects finished five activities, leaving the horizontal climb out. For each task, the duration was less than one minute. The Zephyr BioHarness was attached to the left side of each subject's chest, and the tasks were performed in a supervised environment inside Ahearn field house on the KSU campus [31, 40]. During a normal field test of this type, each task is performed once per cycle, and the duration of that task is short – often a couple of seconds. In addition, after finishing an activity, a subject is required to run for a short distance to the next task station; this ambiguous task separation adds an additional burden to the classification process. Therefore, new training sets of data for each activity were acquired separately, with one

person performing each task repeatedly for about 1 minute. Each piece of the data is clearly labeled by activity and used for classification and identification.

Second Data Collection Effort

The second data collection effort was conducted to gather more data from different people and to record the data in a free-form manner. Ermes et al. [41] noticed that, when applying a supervised-data-trained classifier to unsupervised data, the accuracy decreased by 17%, while the mix of two sets of data helped to maintain a higher classification accuracy. Data from five people (four male and one female) were collected. Each person repeated each task for two minutes. Instructions were given to each subject before each task due to the uncommon nature of these tasks. However, no specific instructions were given regarding speed, posture, etc. For example, subjects were asked to jog at a "comfortable" speed. When stair climbing, they had the freedom to put their hand on the side rail. Subjects were given time to rest in-between tasks so that they could return to a rested state, since short-time fatigue may cause these behavioral motions to differ. Throughout the process, subjects were informed to keep a relative moderate heart rate and to stay comfortable during the session.

A Zephyr BioHarness was used and placed at the same location as in the first data collection session. Along with Zephyr BioHarness, six wireless sensors from a Delsys Trigno system [42] were placed on different muscle groups. The Delsys Trigno is a surface EMG system that records EMG signals and acceleration. EMG sensors are attached to the body using an adhesive film and a medical wrapper. The sampling rate of the sensor was 1 kHz for both the EMG and three-axis acceleration signals. Due to symmetry, all sensors were placed on the left side of body. The sensor locations and target muscle groups are listed in Table 2.3.

Table 2.3 Delsys Trigno sensor locations and target muscle groups.

1	Chest – Pectoralis Major
2	Antebrachium – Flexor Digitorum Superficialis
3	Brachium – Biceps Brachii
4	Shoulder – Deltoid
5	Thigh – Vastus Lateralis
6	Leg – Gastrocnemius

Chapter 3 - Feature Extraction

A single sample of acceleration data does not reflect an activity, as an instant acceleration may exist in any type of activity. Instead, a continuous series of data gathered in a time window presents more facets of an activity. Within the window, some information elements, called features in a machine learning context, can be generated, such as the average value or the rate at which a signal changes. The set of features is used to determine which activity the current activity is likely to be. For this work, time windows are selected to be two seconds long, with a sliding-window overlap of 50%.

As mentioned in Chapter 1, there are two types of features that can be extracted from the raw data: time-domain and frequency-domain features. Wavelets have also been introduced as a new type of feature [43]. However, recent studies indicate that wavelets may not perform as well as frequency-domain features [23], so wavelet features are not included in this feature space. This work presents 25 time-domain and frequency-domain features as an initial effort. The utility of each feature is not studied here, but rather will be examined later.

Time Domain Features

Time domain features include mean, variance, correlation, covariance, signal magnitude area (SMA) and signal magnitude vector (SMV). The **mean** value represents the average acceleration on each axis over the window slice. In a rough sense, it is used to describe the DC component of the signal and is calculated as

$$E_i = \frac{\sum_t a_i}{T} \tag{3.1}$$

where a_i is an acceleration sample and T is the length of the time window. Results for the three axes are depicted as box plots in Figure 3.1. In the plot, the box is centered at the mean of the data, and its edges represent the 25^{th} and 75^{th} percentiles. The whisker is extended to 1.5 times the interquartile range (IQR). Instances outside of the whisker are indicated using a red plus sign. The values are grouped by activity type from 1 to 9.

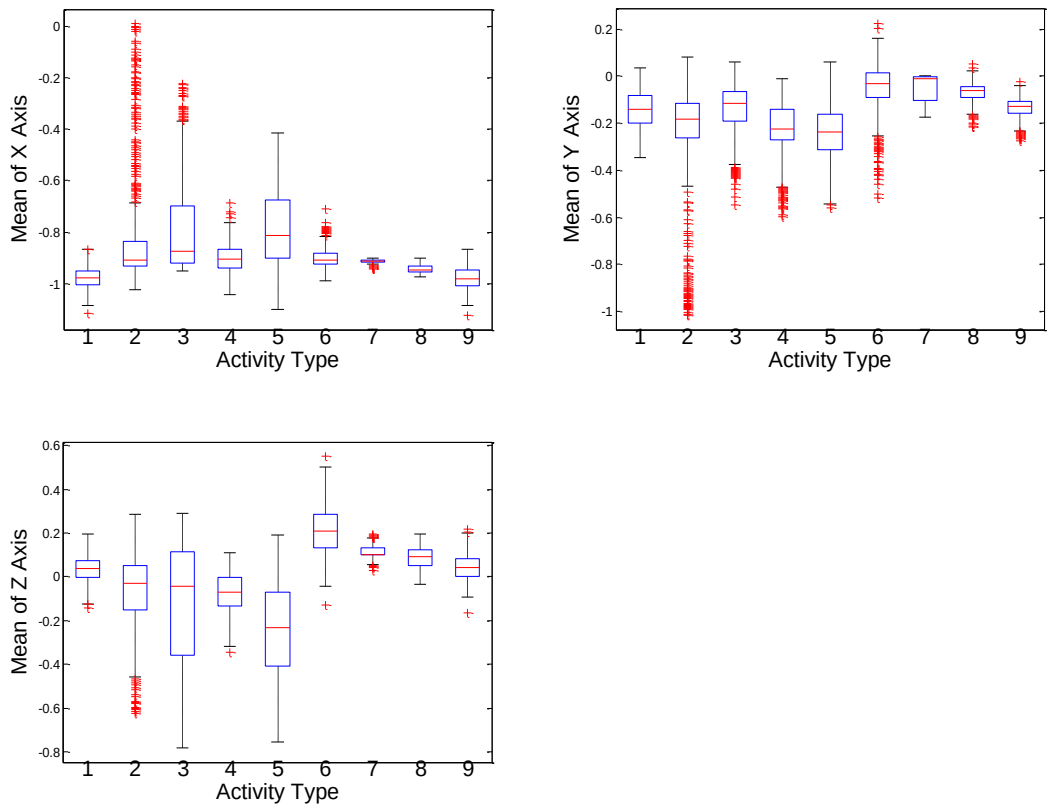

Figure 3.1 Mean acceleration values for the three accelerometer axes.

The **variance** feature is the second moment of the signal. It measures how widely the signal deviates from its mean value. In terms of acceleration, it indicates how rapidly the signal changes and how far it deviates from the mean value. A dynamic activity has larger variance than a more static activity. It is calculated as

$$V_i = \frac{\sum_t a_i^2 - E_i}{T} \tag{3.2}$$

Figure 3.2 delineates the variance on the three axes. Some higher-order moments like skewness and kurtosis can describe the shape of the data in more detail.

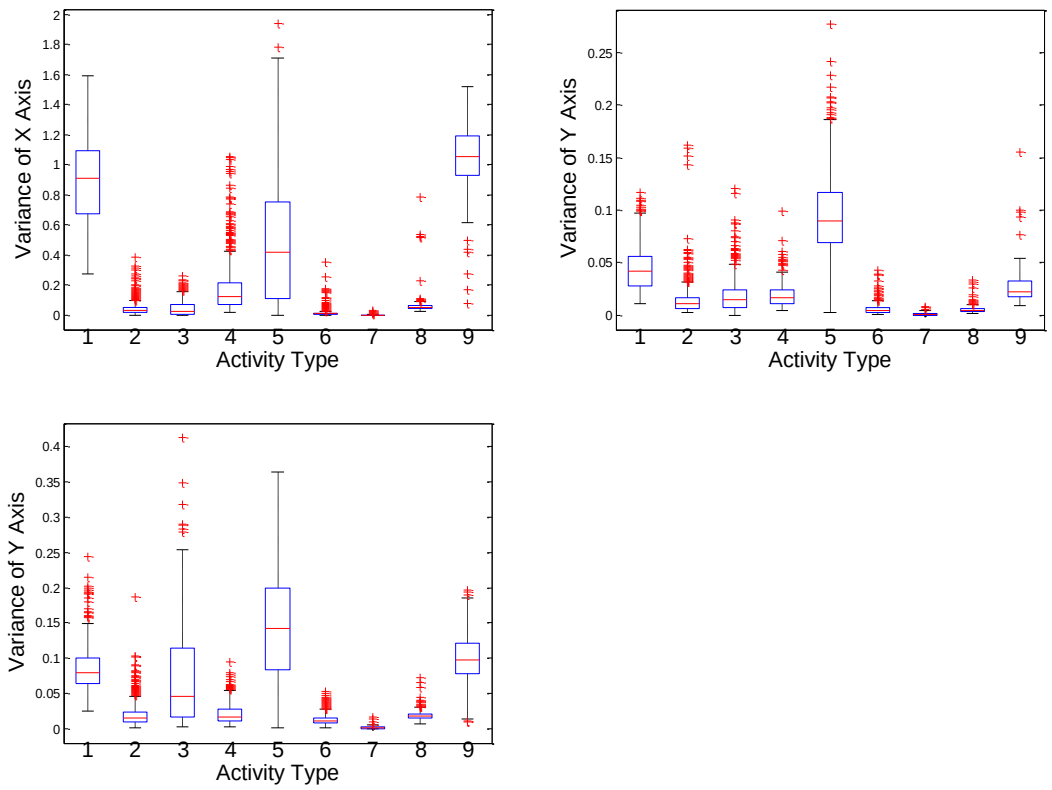

Figure 3.2 Variance on the three acceleroimeter axes.

The **correlation** coefficient denotes the linear relationship between two signals and spans the range [0,1]. The correlation coefficient is large when one signal is linearly related to another signal. Otherwise, it is a small number. When an activity contains concurrent movement in two directions, the correlation between the accelerations on these two axes may be a large value. This can help to separate activities with motion in multiple directions from those with only single-axis movement. The correlation coefficient for discrete signals from two axes is

$$E_{ij} = \frac{\sum_{t} \left(\frac{a_i - E_i}{\sigma_i}\right)\left(\frac{a_j - E_j}{\sigma_j}\right)}{T} \tag{3.3}$$

Correlation coefficients for three axis combinations are illustrated in Figure 3.3.

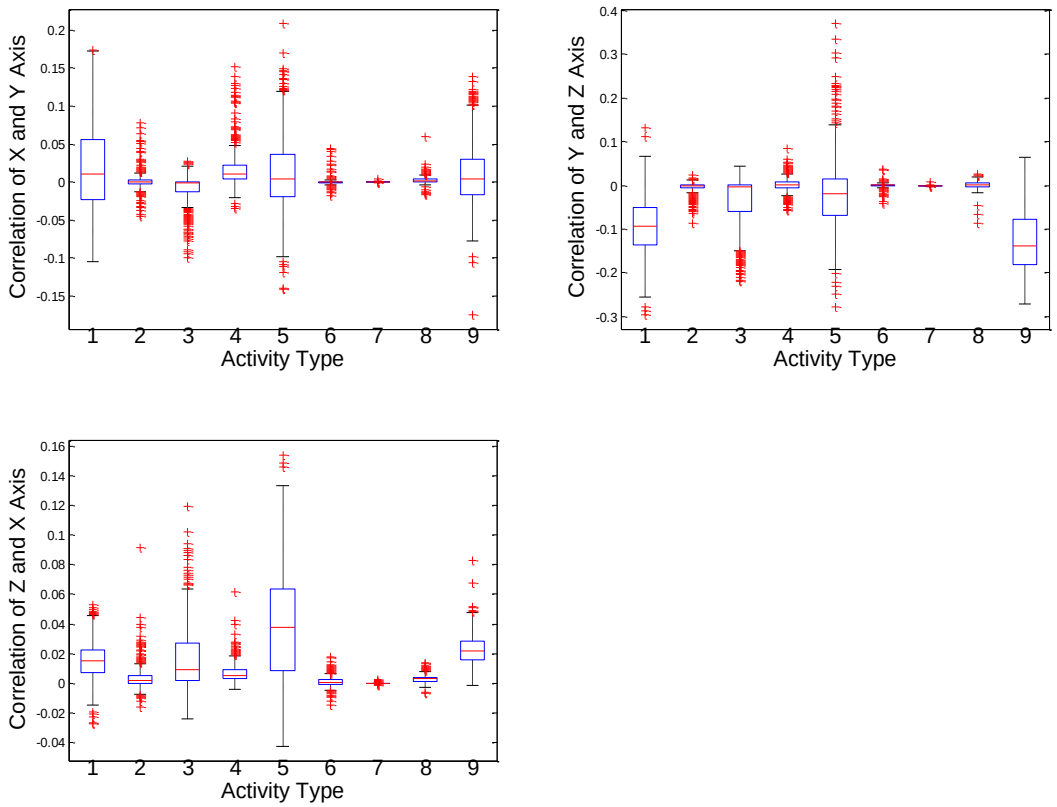

Figure 3.3 Correlation coefficients for three axis combinations.

Similarly, the **covariance** of signals from two axes is

$$Cov_{ij} = \frac{\sum_{t}(a_i - E_i)(a_j - E_j)}{T}. \tag{3.4}$$

Covariance values for three axis combinations are shown in Figure 3.4.

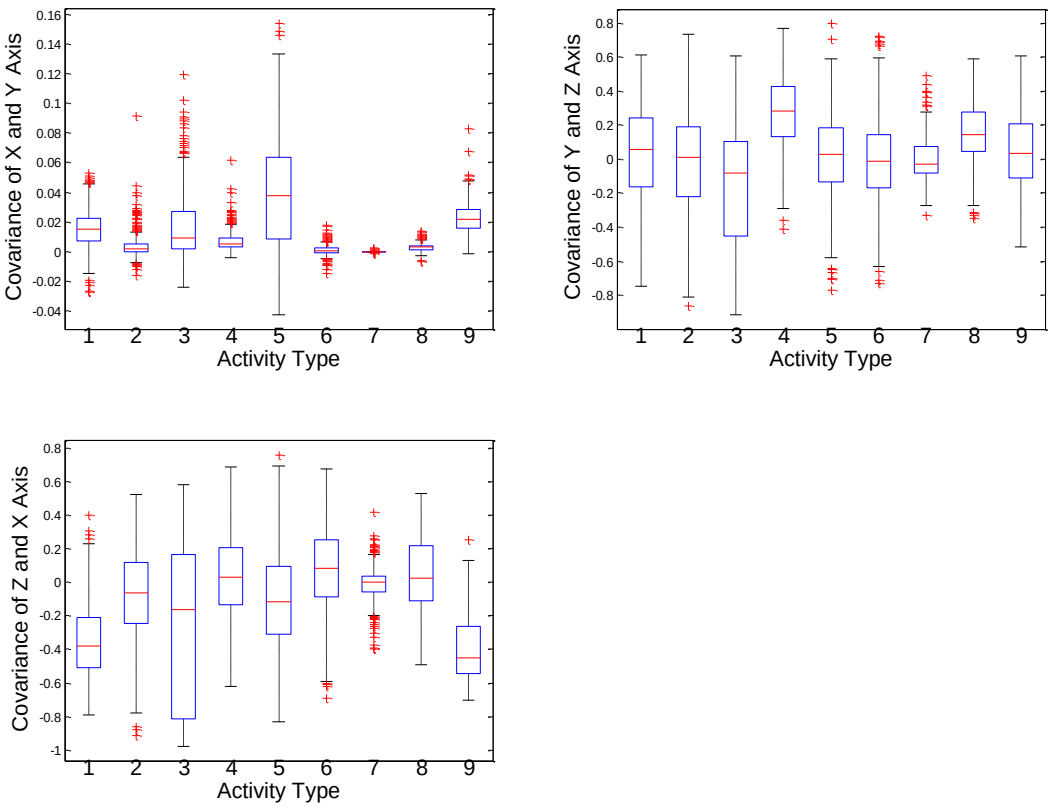

Figure 3.4 Covariance values for three axis combinations.

Aside from single-axis features, some time-domain features consider the combination of accelerations from three axes. Based on the calculation method, they can be categorized as first-order and second-order calculations.

A first-order calculation is basically the sum of the absolute value of the data obtained on each axis. The average of the summed acceleration in a time window is called the **Signal Magnitude Area** (SMA) in the literature [44] and is defined as

$$SMA = \frac{1}{T}\left(\sum_{t=1}^{T}\left|a_x(t)\right| + \sum_{t=1}^{T}\left|a_y(t)\right| + \sum_{t=1}^{T}\left|a_z(t)\right|\right). \tag{3.5}$$

The SMA has mostly been used during real-time classification to replace computationally costly calculation of frequency-domain features, due to the similarities found between SMA and power [38]. The results for these field tests are depicted in Figure 3.5.

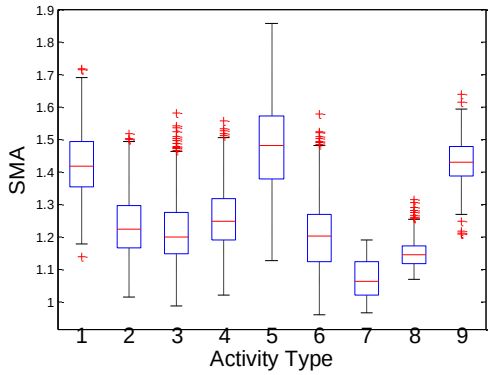

Figure 3.5 SMA values for the different activities.

The second-order calculation is the Euclidian distance of acceleration on three axes. It is called **movement intensity (MI)** [24] and **signal magnitude vector (SMV)** [44]. For consistency, it is called SMV here:

$$SMV = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2} \tag{3.6}$$

The **mean and variance of the SMV** are calculated in the same manner as in the previous definition. Since SMA is the simple summation of the magnitudes of the accelerations, SMA weighs them equally. SMV emphasizes the weight of the larger acceleration by the power of two. Figure 3.6 displays the result for these field tests.

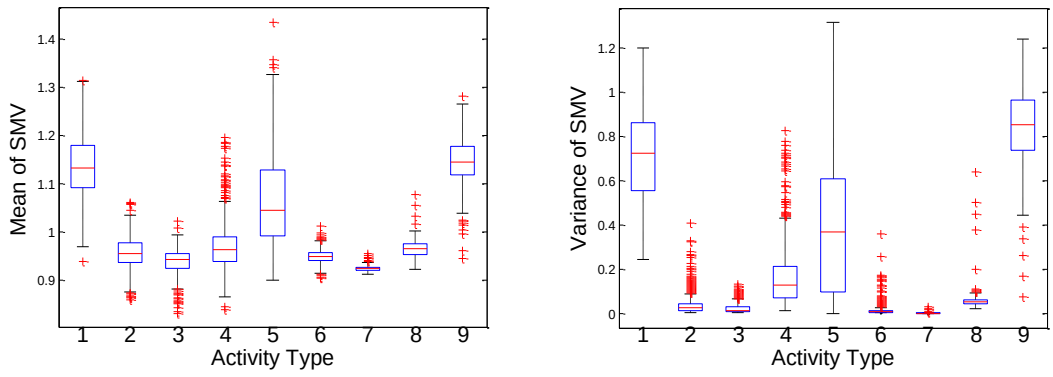

Figure 3.6 Mean and variance of the SMV.

Frequency Domain Features

Frequency-domain features include energy, spectral entropy, dominant frequency and average energy. To extract a frequency-domain feature, the signal needs to be processed with a Fourier Transform to calculate its frequency-domain spectra. A Fast Fourier Transform (FFT) is applied to each time window to accomplish this transformation. One thing to note is that, since the time window is relatively short, the resultant spectrum may be altered due to the finite signal duration. To compensate for this drawback, the original data are multiplied by a Hamming window function, illustrated in Figure 3.7. The window attenuates the signal on the edges of each interval and accentuates the data toward the middle of each interval.

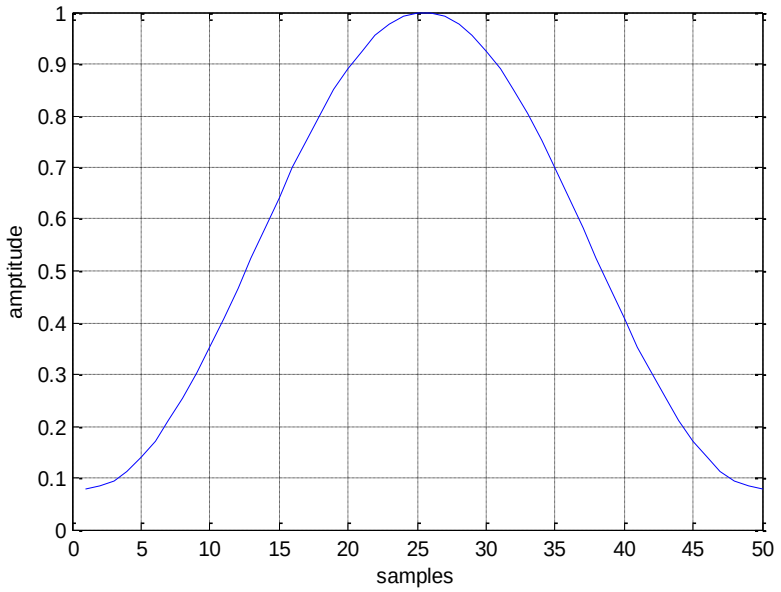

Figure 3.7 Hamming window with a width of 50 samples.

Energy is calculated to determine activity intensity. For a highly intense activity, such as running or stair climbing, more energy would be consumed by the body when compared to an activity such as sitting or lying down. Since the accelerometer was attached to the chest of each subject, only torso movement can be assessed with a reasonable level of fidelity, so limb movement is not considered unless it transmits energy to the whole body. This is one limitation of using a single accelerometer. However, in tight surroundings like a spacesuit, the use of a single accelerometer reduces the subject burden and is more realizable.

Spectral energy is obtained by calculating the Power Spectral Density (PSD) of the spectrum. A one-sided PSD is calculated using the formula

$$p(i, j) = k\left|s(i, j)\right|^2 \tag{3.7}$$

where s is the short-time FFT of the signal and coefficient $k = \dfrac{2}{f_s \sum_{n=1}^{L} \left|w(n)\right|^2}$, where f_s is the sampling frequency (50 Hz in this study). At 0 Hz and the Nyquist frequency (25 Hz), the factor of 2 in the numerator is replaced by 1 since, on the boundary, only a single FFT coefficient exists. The vector $w(n)$ holds the weights for the Hamming window, and L is the (integer) window length. Energy is the sum of the PSD across all the frequencies calculated with the FFT:

$$E = \sum p(i, j)\,. \tag{3.8}$$

It is depicted in Figure 3.8.

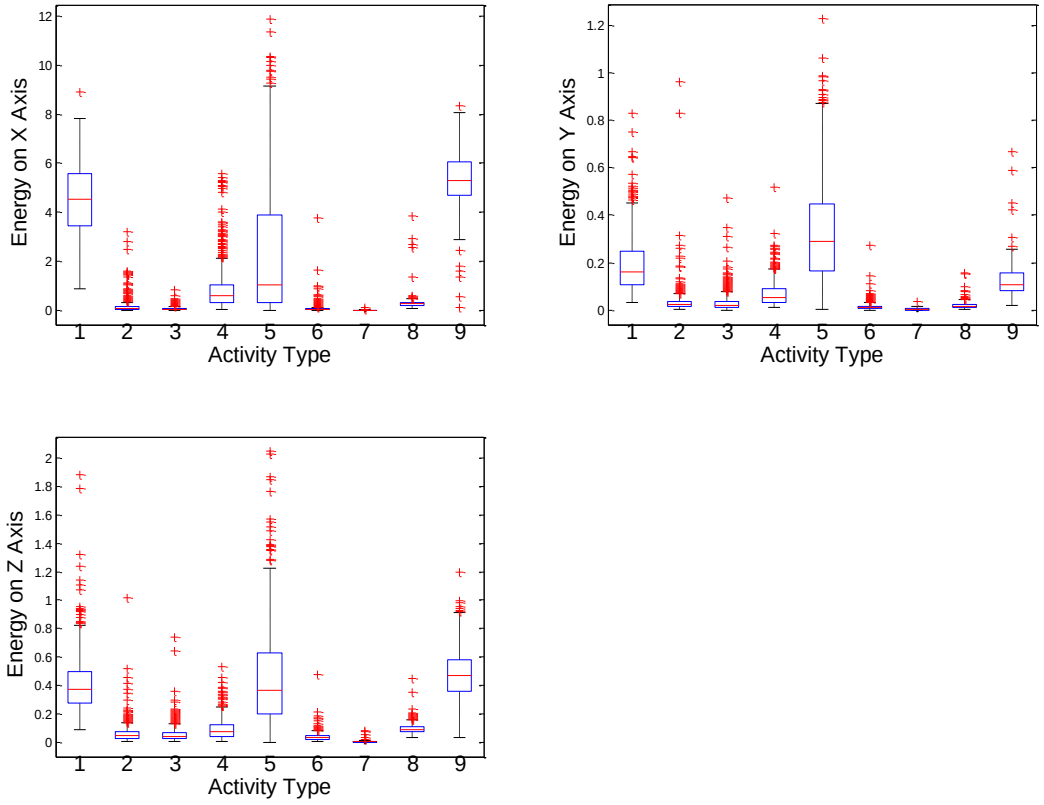

Figure 3.8 Energy for each of the three axes.

Spectral entropy is used to describe the distribution of the frequency-domain signal. Information entropy as proposed by Claude Shannon in 1948 was originally used to indicate the predictability of a series of information [45]. Easily predictable information contains less entropy, while an information stream that seems more random carries higher entropy. This concept maps to physical entropy, which describes the state of randomness for matter in the universe. Information entropy is calculated as

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x), \tag{3.9}$$

where $P(x)$ is the probability of obtaining each element. For a frequency-domain spectrum, the spectral entropy can determine whether a type of activity contains similar energy exertion across the spectrum or just focuses on particular frequencies. In other words, it can distinguish activities with rhythmic pace from those with irregular movement. The same formula is applied when calculating spectral entropy, where *p(x)* is the PSD of each frequency divided by the total energy.

$$P(x) = \frac{p(x)}{E} \tag{3.10}$$

Spectral entropy on three axes is shown in Fig 3.9.

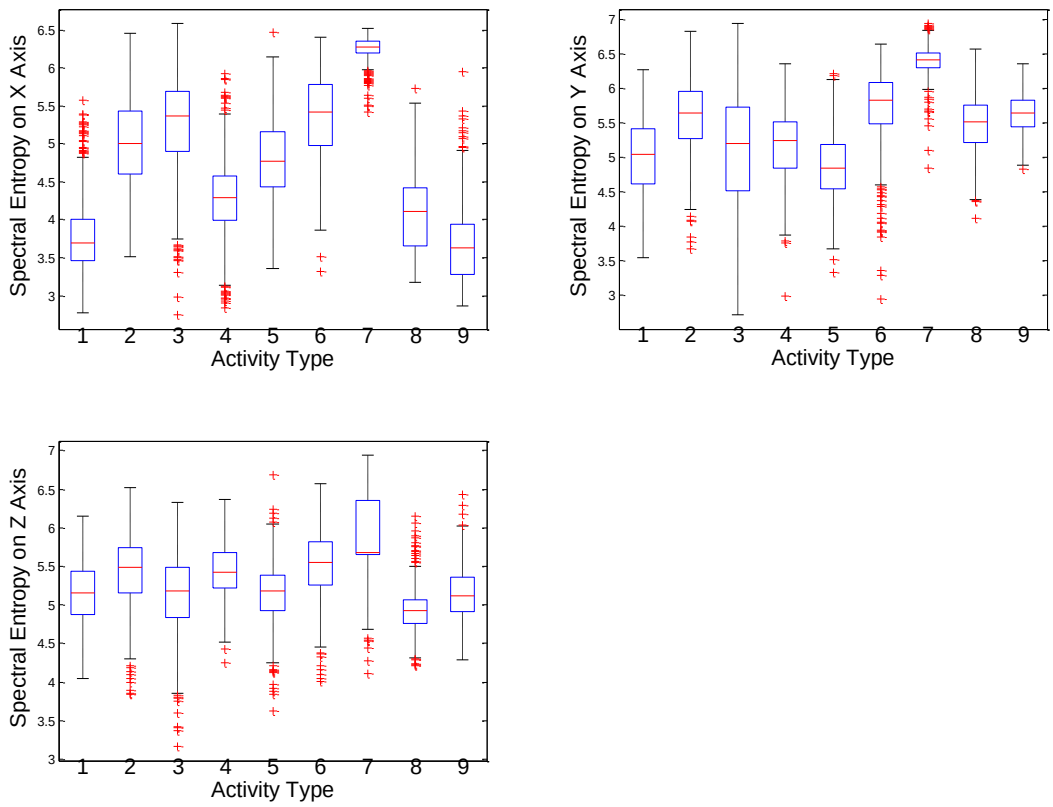

Figure 3.9 Spectral entropy on three axes.

The **dominant frequency** is the frequency at which the PSD magnitude has the maximum value. It is the fundamental frequency for the signal. In periodic activities like running, the dominant frequency usually resides in the frequency range of ~[0.5,2.5] Hz and is calculated as

$$F = \arg\max_{f}(P(f)) \tag{3.11}$$

The box plots in Figure 3.10 indicate that for most activities, the dominant frequency is around 0 Hz. This is due to the large DC offset in the time-domain data.

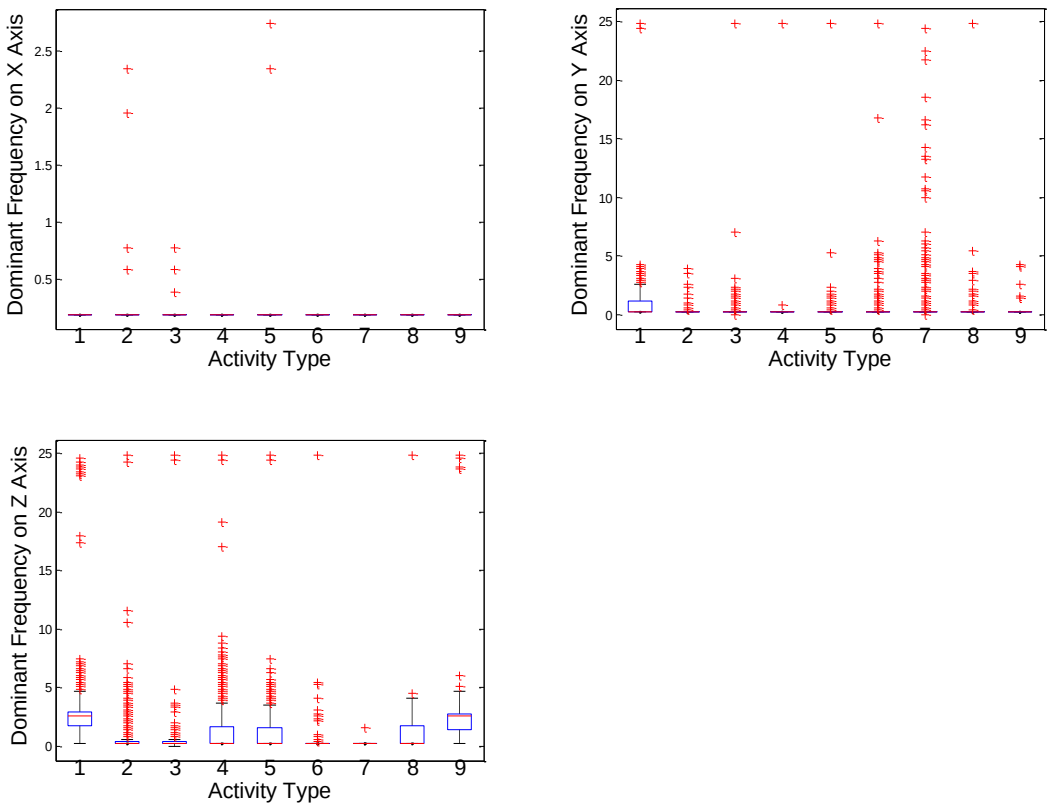

Figure 3.10 Dominant frequency for the acceleration data on the three axes.

Average energy considers the combination of the energy on the three axes:

$$\bar{E} = \frac{E_x + E_y + E_z}{3}. \tag{3.12}$$

Average energy is shown in Figure 3.11.

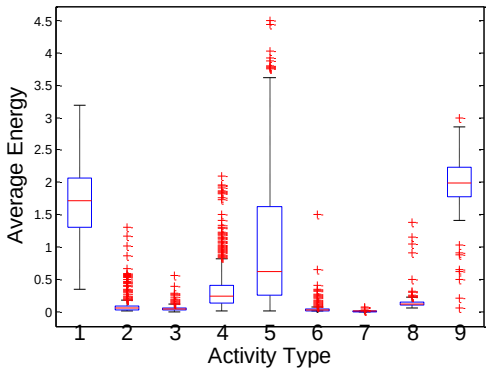

Figure 3.11 Average energy.

Chapter 4 - Feature Selection

The feature sets from the previous chapter need to be validated for their usefulness. This chapter focuses on the feature selection, or the process of choosing the most helpful features.[17]. Useless features can exist in two forms. They may be redundant and provide no new information given the existing features. Others may be irrelevant and may not help to associate data with a certain class according to a decision rule. Redundant features can add unnecessary weight to an aspect of the signal while diminishing the ability of other features to be telling, whereas irrelevant features can confuse the machine learning algorithm and may reduce the resulting accuracy. Both types of features should be eliminated from the classification process.

Feature selection is a common way to deal with this situation. It not only eliminates the useless features, but it also helps to reduce the feature space dimension. The learning algorithm may benefit from a smaller feature space dimension in several ways. First, as the feature space dimension decreases, the complexity of the machine learning process is reduced. Second, noisy feature data often lead to over-fitting, and the purification of features can reduce this tendency and control the generalization of the result. Third, a smaller feature space dimension is easier for a person to interpret.

Feature Selection Methods

In general, two kinds of feature selection methods exist: wrapper methods and filter methods. A wrapper method searches through all of the feature subsets exhaustively and compares the effectiveness of each of them, returning the subset with the best classification result [46]. This method is targeted at a certain classification approach and is subject to change when given a different algorithm. While this exhaustive approach provides the best subsets possible, the method is computationally costly. Given 25 features, the number of subsets is 2^{25} = 33,554,432. Considering the time taken to calculate the result for each subset, it is not realistic to work with such a high dimension space.

An alternative is a filter method. This method uses a proxy measure to determine the similarity between feature space subsets. The proxy measures can be various. Common measures include forward selection and backward elimination. Both algorithms use a greedy search method to select a local optimal feature and add/subtract it to/from the subset before calculating

a new error rate. This step is repeated until the error rate stops decreasing or increasing. The resulting subset is the selected feature set.

The Filter Method

The filter method was selected for feature selection in this effort. Several filter criteria need to be addressed: starting point, search organization, evaluation strategy, and stop criterion.

Starting Point

The starting point determines the beginning feature subset. In general, two approaches are widely applied, namely forward selection and backward elimination. Forward selection, as the name suggests, starts with an empty subset. As the search progresses, features are added to the subset. On the contrary, backward elimination starts with a full feature set and eliminates features one by one, where the feature that contributes the least to the accuracy is removed at each iteration.

Search Organization

An exhaustive search will find the optimum feature subset, since it searches through all of the possible subsets. However, it suffers from a large computation time since the number of subsets is 2^N, where N is the number of features. A better strategy, called a heuristic search, can be effective. This search method does not exhaustively evaluate all of the subsets but rather constructs a subset using a certain criterion, called a heuristic function. This strategy cannot guarantee that the subset it finds is the best global solution, but it does find an optimal local solution.

One of the most commonly applied search strategies is a greedy hill climb. A greedy hill climb algorithm moves through the feature space while looking for the best feature at each step. After examining the error rate of each feature in the remaining pool, the best one is selected, and if adding it to the subset reduces the error rate, then it is included in the subset. This process repeats until the error rate stops decreasing or decreases by a trivial amount. The process is shown in Figure 4.1. This method cannot backtrack, so when the error starts to increase, the algorithm terminates the loop and declares the best subset it finds. A pseudo-code description is shown below.

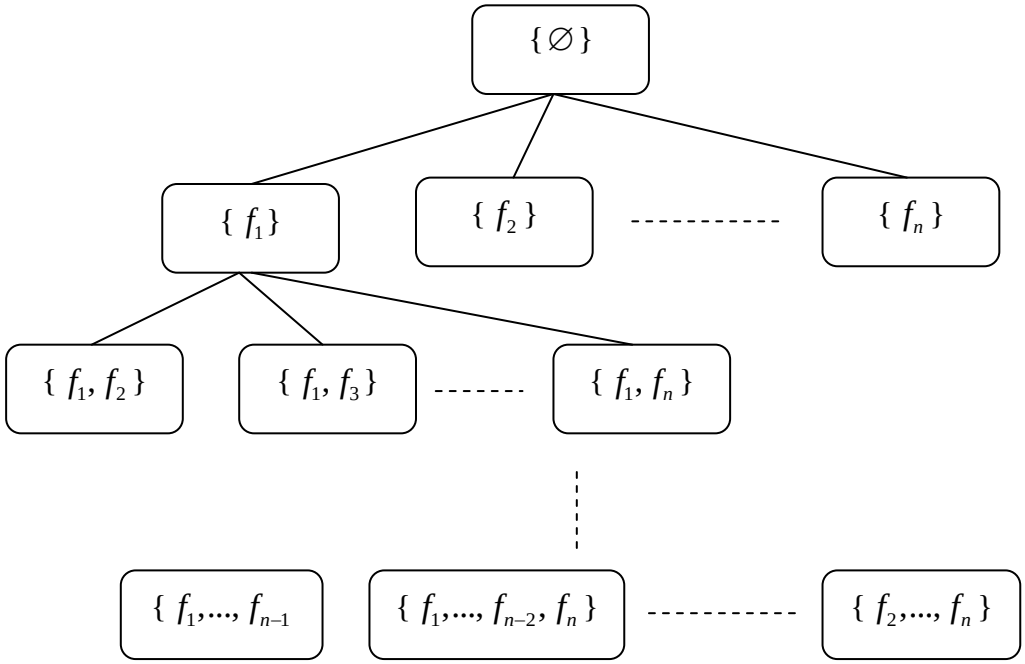

Figure 4.1 Forward selection and backward elimination feature searches.

Greedy Hill Climb Pseudocode

```
s = start state
expand s by making each possible local change
evaluate each child t of s
let s = feature t with highest evaluation e(t)
if e(s') >= e(s)
   s= s'
   repeat
else return s
```

To overcome this drawback, an improved method is proposed – a best first search algorithm. A best first search uses similar means to move through the search space. However, in this algorithm, the evaluations are stored in a bin instead of being discarded, and the algorithm

has the ability to backtrack to previous nodes when a current node does not provide an improved result for reducing the error. A pseudo-code description is listed below

Best First Search Pseudocode

```
s = start state
expand s by making each possible local change
evaluate each child t of s
let s = feature t with highest evaluation e(t) and save all
other ts to stack in ascending order
if e(s') >= e(s)
   s= s'
   find the first t in stack and repeat
else return s
```

Evaluation Strategy

As seen in the above algorithm, the evaluation function $e(x)$ is needed to judge the effectiveness of the subset. When a filter method is adopted, heuristics are used as evaluation functions. We use two evaluation methods to determine the usefulness of a feature. They are correlation-based feature selection and consistency feature selection.

Correlation-based feature selection uses a correlation evaluation function [47]. It is written as

$$M_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{4.1}$$

where M_s is the correlation between the feature subset and the activity, $\overline{r}_{cf}$ is the average correlation between the components and the activity, $\overline{r}_{ff}$ is the average inter-correlation between components, and k is the number of components. This is derived from the Pearson correlation coefficient [48], where all variables have been standardized.

In the **consistency feature selection** method, the usefulness of the features space is defined by inconsistency [49]. This measure characterizes the range of the feature values in instances of the same label. It is calculated by using the filtered version of Las Vegas Algorithm (LVF) algorithm as follows.

a. Two instances are considered inconsistent if they match except for their class labels.

b. Consider all instances with a matching feature. The inconsistency is measured by the number of non-majority labels.

c. The inconsistency rate is calculated as inconsistency counts divided by the total number of instances.

A pseudo code for the algorithm given in [49] is shown below.

LVF Pseudocode

```
for i = 1 to MAX_TRIES
     S = randomSet(seed);
     C = numOfFeatures(S);
     if(C<C_best)
          if(InconCheck(S,D)<γ)
               S_best = S; C_best=C;
               print_Current_Best(S)
     else if(C=C_best) and (InconCheck(S,D)<γ)
          print_Current_Best(S)
end for
```

Stop Criterion

For a heuristic search, a stop criterion is needed to ensure that the algorithm loop terminates properly. One stop criterion can be when the heuristic merit does not increase after more features are added to the subset. Alternatively, it can continue searching until the merit starts to decrease. In our test, we set the consecutive non-improving nodes allowed to be five. When no improvement shows up after five nodes, the search terminates.

Feature Selection Result

Using the correlation-based feature selection approach with the best first search algorithm, the following results are obtained:

For forward selection, the feature subset is:

```
[mean_x, mean_y,mean_z, variance_x, variance_y, variance_z,
covariance_yz, covariance_zx, correlation_xy, SMA, mean_SMV,
variance_SMV, energy_x, energy_y, energy_z, entropy_x,
entropy_y, entropy_z, DF_z, average energy]
```

For backward elimination, the feature subset is

```
[mean_x, mean_y, mean_z, variance_x, variance_y, variance_z,
covariance_yz, covariance_zx, correlation_xy, SMA, mean_SMV,
variance_SMV, energy_x, energy_y, energy_z, entropy_x,
entropy_y, entropy_z, DF_z, average energy]
```

These features are regarded as relevant and non-redundant features that can help to identify different classes. Notice that there are 20 features that are selected, meaning that most of the features contribute to the correctness of the classification. Both forward selection and backward elimination indicate the same result, which suggests that this method is robust and not influenced by the direction of the search. The following features were discarded:

```
[covariance_xy, correlation_yz, correlation_zx, DF_x, DF_y]
```

The dominant frequencies on the *x* and *y* axes were not selected in this subset, mainly because most of them are located at the minimum frequency. Since the DC offset is not eliminated when doing a short-time FFT, the majority of the energy gathers near 0 Hz, no matter what activity it is. Covariance and correlation are related, so for each axis only one value is chosen from the pair.

Now we use the consistency feature selection and best first search methods as another means to select features. The results from consistency evaluation are given below.

For forward selection

```
[mean_x, mean_y, mean_z, variance_x, variance_y,
covariance_xy, covariance_yz, covariance_zx,
entropy_x ,entropy_y]
```

For backward elimination

```
[correlation_y, correlation_z, MEAN_SMV, variance_SMV,
energy_y, energy_z, entropy_x, entropy_y, entropy_z, DF_y,
DF_z]
```

A big difference exists between the forward selection and backward elimination results.

The above methods can help to pick out the redundant and irrelevant features, but they do not provide information on how each feature is evaluated. Information gain can be used as a gauge to rank these features, where information gain is expressed as

$$IG(Y, X) = H(Y) - H(Y \mid X). \tag{4.2}$$

Here, *H(Y)* is the information entropy and *H(Y |X)* is the entropy of *Y* given the observation *X*. In our context, *Y* is a certain activity class and *X* is a given feature from the feature space. *H(Y |X)* is calculated using the following formula:

$$H(Y \mid X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) \log_2 (p(y \mid x)) \tag{4.3}$$

A higher information gain means a larger difference relative to entropy when provided with the feature versus without the feature. If a feature is randomly distributed, then it will not provide any information for determining a class. In that situation, *H(X)* and *H(Y |X)* are the same, and the information gain is zero. The following chart gives the information gain for each feature, ranking them in descending order.

Table 4.1 Information gain evaluation results.

Rank	Feature	Score
1	Average energy	1.61629
2	variance_SMV	1.61115
3	variance_x	1.58037
4	energy_x	1.56769
5	variance_y	1.2927
6	energy_z	1.27799
7	variance_z	1.25817
8	energy_y	1.12749
9	mean_SMV	1.1168
10	covariance_yz	1.05396
11	covariance_zx	0.95493
12	entropy_x	0.92993
13	mean_z	0.91406
14	mean_x	0.87198
15	SMA	0.8623
16	covariance_xy	0.85204
17	DF_z	0.74763
18	mean_y	0.65556
19	correlation_yz	0.55839
20	entropy_y	0.48753
21	entropy_z	0.39736
22	correlation_zx	0.31158
23	DF_y	0.22772
24	correlation_xy	0.20589
25	DF_x	0.00695

According to this listing, energy, variance and physical features like variance_SMV offer large information gain, while dominant frequency (DF_*) and correlation offer lower

information gain. Features from the time domain and the frequency domain have both high and low values, implying that both types of features are needed for accurate activity classification.

The gain-ratio-attribute ranking method was applied for comparison. The results are shown in the following table. The gain-ratio approach evaluates the worth of an attribute by measuring the gain ratio with respect to the class. It is calculated as

$$GainR(Class, Attribte) = \frac{H(Class) - H(Class \mid Attribute)}{H(Attribute)} \tag{4.4}$$

Results from the gain ratio method are shown in Table 4.2. The result is similar to the one from information gain attribute evaluation method, where energy and variance are ranked on top and correlation and dominant frequency are ranked on the bottom.

Table 4.2 Gain ratio evaluation results.

Rank	Feature	Score
1	energy_x	0.4727
2	Average energy	0.4633
3	variance_x	0.4374
4	variance_SMV	0.4244
5	variance_z	0.3948
6	mean_SMV	0.3884
7	energy_z	0.3686
8	variance_y	0.3455
9	DF_z	0.34
10	energy_y	0.3178
11	entropy_x	0.2899
12	covariance_zx	0.2781
13	covariance_yz	0.2777
14	SMA	0.2703
15	mean_z	0.2683
16	mean_x	0.2662
17	covariance_xy	0.2457
18	DF_x	0.2189
19	DF_y	0.2107
20	mean_y	0.1878
21	entropy_y	0.1808
22	correlation_yz	0.1776
23	entropy_z	0.1667
24	correlation_zx	0.121
25	correlation_xy	0.0864

Chapter 5 - Activity Classification

Machine learning plays an important role in activity recognition. Many different algorithms exist for different applications, but no one particular algorithm is generally dominant for a given application. This chapter addresses the examination of multiple algorithms to find the algorithm that works the best for these data sets and can satisfy the real-time algorithm requirement.

Machine Learning Algorithms

Machine learning algorithms can be divided into three primary types: supervised learning, unsupervised learning, and reinforced learning [25]. Supervised learning takes instances with labeled outputs as the training data. The learning algorithm compares the decision results with the labels to adjust the parameters in the decision function. Punishment is given when the decision and the actual label are different. Unsupervised learning uses unlabeled data for training. The algorithm is not informed whether the output is correct or not. The algorithm is built upon the concept that data with identical labels tend to conglomerate in some particular space. Clustering is a common idea for this type of learning, in which the number of classes is given, and the algorithm iterates until the centroids of every cluster are found. Measurements for centroids vary from scenario to scenario. Reinforced learning exists somewhere in between, where no labels in the training data are given, but the algorithm receives the correctness of each decision. The correctness is given as a Boolean variable, where correct is 1 and incorrect is 0.

Recent research suggests another type of training algorithm, referred to as semi-supervised learning [50-52] . It is a combination of supervised and unsupervised learning that utilizes a half-blind learning process. Initially, a small portion of labeled data is provided to the algorithm for training purposes. However, following the process, a large quantity of data with no labels is fed to the algorithm to help refine the classification function. This is often used when labeling data is tedious and cumbersome or labeled data are impossible to get.

For activity recognition, the ability to obtain labeled data allows us to use supervised learning. This type of learning has a faster convergence rate and provides better results. Four

different classification algorithms are compared: Naïve Bayes, k Nearest Neighbor, Decision Tree, and Support Vector Machine.

Naïve Bayes

The Naïve Bayes algorithm assumes the data have a Gaussian distribution, and the method is trained based on this assumption [53]. It utilizes the Bayes formula:

$$p(\omega_j \mid x) = \frac{p(x \mid \omega_j)\, p(\omega_j)}{p(x)} \tag{5.1}$$

where ω_j is the class and x is a feature value. It states that we can deduce the conditional probability that a feature value is part of a class given prior knowledge of the probability distribution of the feature value, the probability distribution of the class, and the conditional probability distribution of a feature value in a certain class. $P(x \mid \omega_j)$ is called likelihood, $p(\omega_j)$ is the *a priori* probability, $p(x)$ is the evidence, and $P(\omega_j \mid x)$ is the *a posteriori* probability. The formula expressed in word form is

$$a\ posteriori = \frac{likelihood \times a\ priori}{evidence} \tag{5.2}$$

For a feature space including more than one feature, the likelihood is modified into

$$p(\mathbf{x} \mid \omega_j) \tag{5.3}$$

where

$$\mathbf{x} = x_1 \bigcap x_2 \bigcap ... \bigcap x_n \tag{5.4}$$

An important assumption in Naïve Bayes classification is that features are conditionally independent from each other given the class. It simplifies the probability condition in that joint probability is zero and likelihood becomes the product of individual probabilities:

$$p(x_1 \bigcap x_2 \bigcap ... \bigcap x_n \mid \omega_j) = \prod_i p(x_i \mid \omega_j) \tag{5.5}$$

For discrete or nominal features, the probability is equal to the number of specific instances over the total number of instances. For a numeric feature, the distribution is assumed to be Gaussian (which is true for most of these data), so the distribution then can be represented with a mean and standard deviation. The probability of a given value is calculated from the probability density function:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.6}$$

where σ is the standard deviation and μ is the mean of the distribution. Here we ignore the missing feature value scenario as every feature is assigned to a value in a given instance.

For a given classification problem, the *a posteriori* probability indicates the likelihood of the instance belonging to a class for a given set of features. Therefore, the need to determine the class converts to finding the maximum conditional probability:

$$\omega = \arg\max_{j \in [1,J]} (p(\omega_j \mid x)) \tag{5.7}$$

where *j* represents the class and *J* is the total number of classes.

It is clear that *evidence* is a constant number since it is the combined probability distribution for all of the classes. Therefore it can be neglected while doing the calculation. The *a priori* probability is the likelihood that a class is present. This is a scenario-specific number and changes according to the real situation. In a daily activity classification problem, classes such as sitting and walking have a higher probability than climbing stairs or running, because they happen more often. In the case of the PNFTs, activities are performed in a sequence. Therefore the probability for each activity is roughly the same and can also be neglected.

The classification results obtained using the Naïve Bayes approach are represented as confusion matrix in Table 5.1. In this matrix, each row represents the true activity, and each column represents the classified activity (algorithm result).

Table 5.1 Confusion matrix obtained from Naive Bayes PNFT classifications.

	Classified As								
Activity	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
a-Agility Cones	**513**	0	0	12	20	0	0	0	103
b-Ladder Climb	3	**260**	37	36	7	240	0	66	0
c-Equipment Lift	1	42	**333**	7	11	238	4	2	0
d-Step Entry	28	150	1	**344**	10	8	0	71	0
e-Stair Climb	36	51	32	14	**501**	1	1	0	2
f-Horizontal Climb	2	22	7	9	0	**601**	5	10	0
g-Stand	118	0	0	0	0	6	**393**	1	0
h-Walk	3	5	0	9	0	8	0	**633**	0
i-Run	60	1	0	4	2	0	0	0	**602**

Accuracy is calculated as the ratio of the trace of the matrix to the sum of the entire matrix.

$$Accuracy = \frac{tr(C)}{\Sigma c_{ij}} = \frac{\Sigma c_{ii}}{\Sigma c_{ij}} \tag{5.8}$$

Based on the recognition results, the overall accuracy obtained from the Naïve Bayes classifier is 73.51%.

K Nearest Neighbor

The k Nearest Neighbor method uses a "lazy" approach to the problem, where the decision making process is delayed until the data that need to be classified are present. The program pre-stores a group of labeled data in memory. When the new data arrive, the program calculates the distance of the coming data relative to the existing data, finds the closest *k* instances, and labels the data with the majority classes in the *k* instances, where *k* is an integer ranging from 1 to *N*. This kind of classification process is also called instance based learning (IBL) because it makes a decision based on direct feature information from the instances instead of generalizing a set of rules or a decision tree.

Distance can be evaluated in various ways. Two common metrics are the Manhattan distance and the Euclidean distance. The Manhattan (a.k.a., taxicab) distance between two points is the sum of the absolute differences of their coordinates:

$$d_1(p,q) = \sum_i |p_i - q_i| \tag{5.9}$$

The Euclidean distance is the distance in Euclidean space. It equals the square root of the sum of the squared differences:

$$d_2(p,q) = \sqrt{\sum_i (p_i - q_i)^2} \tag{5.10}$$

The square root process is often ignored due to the computation time required. This does not affect the results when comparing these distances.

An issue emerges when the distances are not equal among all of the features. Features with large differences will dominate the distance-based decisions, and those with small differences may not show enough impact on the equation. To equalize the effect of each feature, normalization is applied to the feature:

$$x = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{5.11}$$

This restricts the range of each feature to [0, 1].

Table 5.2 illustrates the effect of *k* in the kNN classifier. As *k* increases, the accuracy does not have a significant change. Staying above 85%, the accuracy of the kNN method is higher than when using the Naïve Bayes classifier. The highest accuracy, 87.51%, is achieved when $k = 3$. From a hyperplane perspective, the kNN approach is more flexible because it can formulate complex non-linear hyperplanes that divide the feature space. In a situation where the data are not necessarily conditionally independent or irredundant, that helps to define a better evaluation surface. The confusion matrix resulting from the kNN process is depicted in Table 5.3.

Although the kNN method seems to be an excellent candidate for activity recognition, there are two problems associated with this classifier when applied to a real-time situation. Since

no decision is made until the input data are received, it requires much memory to store all of the training data, as they are kept in the original form. Second, the classification process requires one instance to be compared with all the instances in order to find the nearest neighbor, so the calculations are numerous.

Table 5.2 Recognition accuracy for the kNN approach as a function of k number.

N	Accuracy (%)
1	87.07
2	86.51
3	87.51
4	87.33
5	87.13
6	87.34
7	87.40
8	87.12
9	86.86
10	86.86

Table 5.3 Confusion matrix obtained from k Nearest Neighbor PNFT classifications

	Classified As								
Activity	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
a-Agility Cone	**593**	0	0	5	4	0	0	0	46
b-Ladder Climb	0	**515**	29	46	2	24	2	31	0
c-Equipment Lift	0	40	**538**	4	2	42	3	9	0
d-Step Entry	3	24	1	**571**	1	1	0	11	0
e-Stair Climb	30	10	2	9	**584**	0	2	1	0
f-Horizontal Climb	0	61	54	1	1	**502**	24	13	0
g-Stand	0	56	69	0	0	3	**390**	0	0
h-Walk	4	17	2	1	1	3	0	**630**	0
i-Run	10	0	1	4	0	0	0	1	**653**

Support Vector Machine

The concept of a Support Vector Machine (SVM) is similar to that of a linear machine learning algorithm [54]. The difference is that it can be applied to non-linear datasets and separate them out using a hyperplane. The method seeks to raise the feature space to a higher dimension, allowing the nonlinear dataset to be linearly separated. This process is called a kernel trick and the mapping function is called a kernel function. Table 5.4 shows the confusion matrix obtained after applying the SVM approach. The overall accuracy for this method is 85.82%.

Table 5.4 Confusion matrix obtained from support vector machine classifications.

	Classified As								
Activity	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
a-Agility Cone	**591**	0	0	4	6	0	0	1	46
b-Ladder Climb	3	**446**	46	61	6	41	3	43	0
c-Equipment Lift	0	49	**443**	20	8	96	20	2	0
d-Step Entry	15	70	3	**510**	3	0	0	11	0
e-Stair Climb	32	12	15	1	**574**	0	2	0	2
f-Horizontal Climb	1	34	27	3	0	**551**	16	24	0
g-Stand	0	0	2	0	0	9	**507**	0	0
h-Walk	2	30	4	3	0	5	0	**613**	1
i-Run	15	0	0	4	3	1	0	1	**645**

C4.5 Decision Tree

C4.5 is a form of decision tree algorithm which forms the decision making process by taking a tree-shaped route [47]. Each node of the tree corresponds to a feature, and the branches are different values of the node. A leaf of the tree is an activity. It is not required that each feature be used only once or each activity be labeled only once, so there can be multiple leafs with the same activity label and different nodes that use the same feature as a judgment. When judging an activity, the algorithm starts at the root of the tree and migrates from node to node until it reaches a leaf at the bottom of the tree.

C4.5 uses a greedy approach to determine the feature attached to the node for each step. Specifically, it starts with the gain ratio criterion to select the feature for the root and forms the sub-trees repetitively. The process terminates when the entire instance sets belong to the same activity. Most of the time, this criterion is difficult to meet, and the tree splitting continues for a long time. Over-fitting is also an issue, where a slight change in the value of a feature can lead to the prediction of an activity on the wrong leaf. To address this issue, an alternative approach is to terminate when the purity of each leaf reaches a threshold.

For a complex feature space, the tree expansion can be large. Training the C4.5 algorithm with the PNFT feature data results in a tree with 533 nodes and 277 leaves. The overall accuracy of this approach when applied to these PNFT data is 86% percent. The related confusion matrix is contained in Table 5.5.

Table 5.5 Confusion matrix obtained from C4.5 decision tree classifications.

	Classified As								
Activity	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
a-Agility Cone	**564**	2	0	9	16	0	0	0	57
b-Ladder Climb	2	**479**	55	45	7	34	0	25	2
c-Equipment Lift	0	57	**499**	7	12	62	1	0	0
d-Step Entry	10	67	2	**516**	10	1	0	4	2
e-Stair Climb	27	10	17	6	**576**	0	1	0	1
f-Horizontal Climb	1	35	59	3	2	**546**	1	9	0
g-Stand	0	1	2	0	0	6	**509**	0	0
h-Walk	2	21	4	3	0	4	0	**622**	2
i-Run	35	2	0	1	5	2	0	0	**624**

Figure 5.1 depicts a sub-tree of the overall decision tree for the PNFT accelerometer data. Each node is labeled with its corresponding features, and the numbers used for the dividing points are attached to their respective branches. At the end of some branches, a leaf notes the activity label and the ratio of correct to incorrect instances.

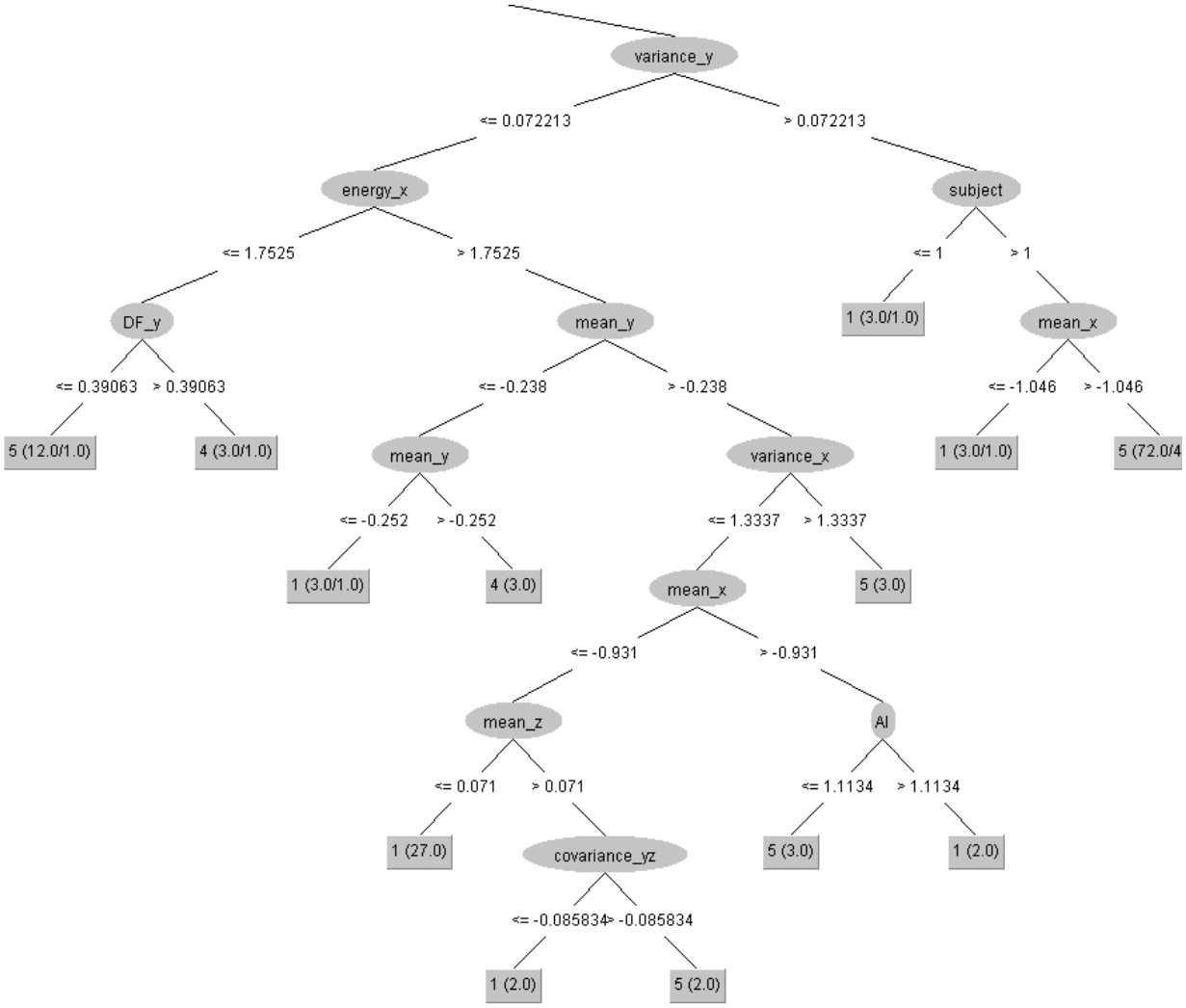

Figure 5.1 Sub-tree within the overall decision tree for the PNFT features.

Chapter 6 - Graphical User Interface Design

Data transmitted from the Zephyr BioHarness are collected by a computer connected to a Bluetooth dongle and then displayed on a graphical user interface (GUI). The motivation to develop such a GUI was driven by the fact that the original data collection tool bundled with the BioHarness provides limited functionality that does not well-support the data collection and visualization needs of the NASA project. Further, a custom user interface can incorporate data obtained from additional sensors in the future. Two versions of GUIs were developed (in MATLAB and in LabVIEW) – one is used for typical Zephyr BioHarness data collection sessions, and the other supports more sophisticated data acquisition and analysis.

Transmission Protocol

Zephyr defined a standard data frame to facilitate data transmission. All of the commands to and from the Zephyr unit are encapsulated in a known message format. This message includes a three-byte header, followed by a data payload with a variable length, followed by two terminating bytes. The length of the data payload ranges from 0 to 128 bytes.

The first byte in the header is the start of text (STX) byte, and it is always designated as 0x02. The next byte is the message ID (Msg ID), which uniquely identifies the type of message. After receiving a data transmission, we use this byte to sort the message into different data process subroutines. The next byte declares the number of bytes in the data payload. It is a hex number ranging from 0x00 to 0x80. The actual data payload is specified in the "communications link specification" in the Zephyr documentation. An 8-bit CRC follows the data payload with the polynomial being 0x8C. The last byte is an end of text (ETX) byte. It is a constant hex number of 0x03. The packet form is presented in Figure 6.1.

For every command request, whether from the wearable device or from the computer, there is a response message from the receiver. The frame format of the response is similar to the request. However, the message ID is different from the corresponding request-message ID. Also, the last byte in the response is acknowledge/not acknowledge (ACK/NAK) instead. If a message is correctly received, an acknowledgement of 0x06 is in the last byte. If a message is not correctly received, an unacknowledgement of 0x15 is appended.

Notice that this data frame format does not contain an address, because the communication is intended to be point-to-point: a Zephyr BioHarness can only communicate with a single data logger. However, multiple Zephyr devices can communicate concurrently with that same data logger. In other words, the Zephyr tools support a star configuration network.

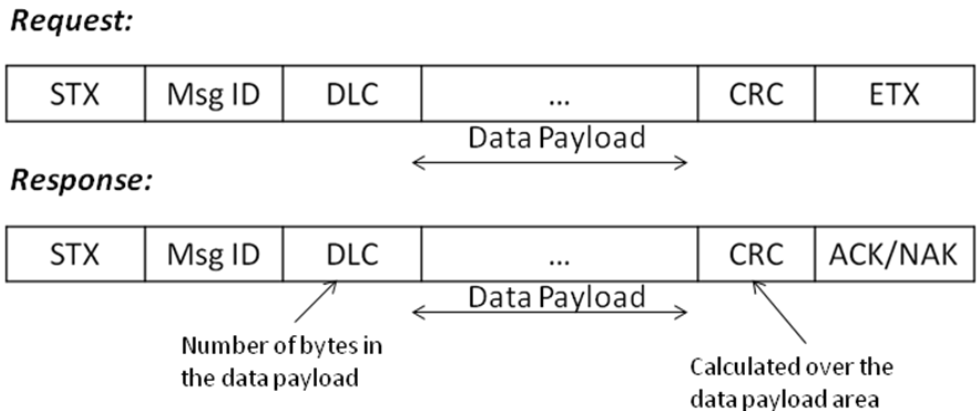

Figure 6.1 Zephyr BioHarness transmitting data packet format.

Data packages are encapsulated in Bluetooth packets before being transmitted (see Figure 6.2). More than one communication packet may exist within one Bluetooth packet. The transmission scheduling of these Bluetooth packets is discontinuous and determined by the Bluetooth protocol. The Bluetooth encapsulation and decapsulation is processed by the Bluetooth IC chip on the Zephyr unit and the Bluetooth dongle connected the computer. Therefore, when sending or receiving from a custom program, Bluetooth packet encapsulation is not required. Instead, data are aligned according to their communication packet format and sent via a serial port.

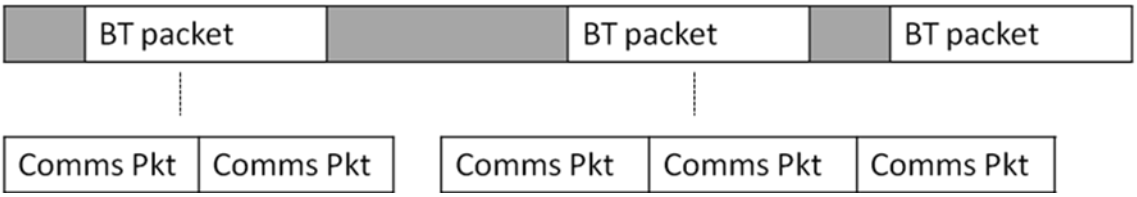

Figure 6.2 Data packet encapsulation process.

Data Frame

A list of messages used in the GUI program is presented in Table 6.1. The enable and disable message for every packet are similar. The data packet consists of 1 byte that represents transmission state: 1 stands for enabled and 0 stands for disabled. A message ID is used to identify which packet to enable or disable. The response from the device has no data payload, and consequently the DLC byte is 0.

Table 6.1 Messages used in Zephyr BioHarness communication.

Message ID	**Message**	**Description**
0x14	Set General Data Packet Transmit State	Enable/Disable packet
0x16	Set ECG Waveform Packet Transmit State	Enable/Disable packet
0x1E	Set Accelerometer Packet Transmit State	Enable/Disable packet
0x20	General Data (streaming) Packet	No ACK required
0x22	ECG Waveform (streaming) Packet	No ACK required
0x25	Accelerometer Data (streaming) Packet	No ACK required
0xA4	Set BT Link configuration	

Table 6.2 Request message for Set General Data Packet.

Byte/Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Field**
0	STX								STX
1	0x14								Msg ID
2	1								DLC
3	Transmission State								Payload
4	CRC								CRC
5	ETX								ETX

Table 6.3 Response message for Set General Data Packet.

Byte/Bit	7	6	5	4	3	2	1	0	Field
0	STX								STX
1	0x14								Msg ID
2	0								DLC
3	CRC								CRC
4	ACK								ACK/NAK

Data packet lengths vary from packet to packet because these data can have different lengths, sampling rates, and transmission rates. For the Zephyr Boharness, the sampling rate and transmission rate are different, since one data packet can contain multiple data samples, e.g., ECG and multiple-axis acceleration data. In addition, the data packets for ECG and acceleration signals use a compact format that wraps data to reduce the transmission burden. These are explained momentarily. Every data packet also contains a time stamp, facilitating data synchronization from different streams and devices. The following list show the sampling rate and transmission rate of each data packet.

Table 6.4 Sampling rate and transmission rate of Zephyr data packets.

Data packet	Sampling rate	Transmission rate
General data packet	-	1000 ms
ECG data packet	4 ms	252 ms
Acceleration data packet	50 ms	400 ms

General Data Packet

A general data packet is a comprehensive message that combines all of the vital sign signals and device condition information. It is sent once every second. Data represented in the packet include timestamp, heart rate, respiration rate, skin temperature, posture, and battery information. Heart rate is generated from an ECG signal processed by an internal QRS detection algorithm and reflects the range of [25, 240] bpm. Posture indicates front-to-back body tilt in degrees with respect to the vertical axis. Forward is a positive degree and backward is a negative degree. Respiration rate is roughly characterized by breaths per minute. Skin temperature

measures the temperature where the Zephyr unit is attached. Note that for every signal, at least two bytes of data exist. The concatenation of these bytes is required to interpret the data. The detailed data packet format for the data payload in a general data packet is listed in Table 6.5.

Table 6.5 General data packet format for the Zephyr BioHarness.

Byte/Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Field**
0	STX								STX
1	0x20								Msg ID
2	53								DLC
3	Sequence Number (0…255)								Payload
4	Timestamp – Year (LS Byte)								
5	Timestamp – Year (MS Byte)								
6	Timestamp – Month								
7	Timestamp – Day								
8	Timestamp – Milliseconds of day (LS Byte)								
9	:								
10	:								
11	Timestamp – Milliseconds of day (MS Byte)								
12	Heart Rate (0…240) – LS Byte								
13	Heart Rate (0…240) – MS Byte								
14	Respiration Rate (0…70) – LS Byte								
15	Respiration Rate (0…70) – MS Byte								
16	Skin Temperature (0…60) – LS Byte								
17	Skin Temperature (0…60) – MS Byte								
18	Posture (-180…180) – LS Byte								
19	Posture (-180…180) – MS Byte								
20	VMU (0…16) – LS Byte								
21	VMU(0…16) – MS Byte								
22	Peak Acceleration (0…16) – LS Byte								
23	Peak Acceleration (0…16) – MS Byte								

24	Battery Voltage	
26	Breathing Wave Amplitude	
28	ECG Amplitude	
30	ECG Noise	
32	Vertical Axis Acceleration Min	
34	Vertical Axis Acceleration Peak	
36	Lateral Axis Acceleration Min	
38	Lateral Axis Acceleration Peak	
40	Sagittal Axis Acceleration Min	
42	Sagittal Axis Acceleration Peak	
44	Zephyr System Channel	
46	GSR	
48	*unused*	
50	*unused*	
52	ROG	
53	ALARM	
54	Battery Status(see 6.2 below)	
55	Button/Worn(see 6.2 below)	
56	CRC	CRC
57	ETX	ETX

ECG signal

The data packet format for the raw ECG waveform is laid out in Table 6.6. The sampling rate is 250 Hz and the precision is 10 bits. It takes more than one byte to contain one sample. Instead of using two bytes of data to express one sample, leaving the rest of the bits to be zero (unused), the Zephyr unit utilizes a compact form to pack the data continuously. The wrapping pattern is described in Table 6.7. Each ECG snippet contains 63 samples. Every sample needs to be reconstructed after being received by the remote terminal.

Table 6.6 ECG data packet format.

Byte/Bit	7	6	5	4	3	2	1	0	Field
0	STX								STX
1	0x22								Msg ID
2	88								DLC
3	Sequence Number (0…255)								Payload
4	Timestamp – Year (LS Byte)								
5	Timestamp – Year (MS Byte)								
6	Timestamp – Month								
7	Timestamp – Day								
8	Timestamp – Milliseconds of day (LS Byte)								
9	:								
10	:								
11	Timestamp – Milliseconds of day (MS Byte)								
12	ECG Waveform Data (63 Samples)								
91	CRC								CRC
92	ETX								ETX

Table 6.7 Compact payload format used with the ECG data packets.

Byte/Bit	7	6	5	4	3	2	1	0
0								Bit 0
1							Bit 0	Bit 9
2						Bit 0	Bit 9	
3					Bit 0	Bit 9		
4	Bit 9							
5	As Byte 0 (pattern repeats every 5 bytes).							

Acceleration signal

Since the acceleration data precision is 10 bits, the acceleration data packets utilize a similar data wrapping format as the ECG data packets, except thethree-channel data are packed alternatively. Table 6.8 notes the acceleration data packing format and Table 6.9 notes the wrapping pattern.

Table 6.8 Acceleration data packet format.

Byte/Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Field**
0	STX								STX
1	0x25								Msg ID
2	84								DLC
3	Sequence Number (0…255)								Payload
4	Timestamp – Year (LS Byte)								
5	Timestamp – Year (MS Byte)								
6	Timestamp – Month								
7	Timestamp – Day								
8	Timestamp – Milliseconds of day (LS Byte)								
9	:								
10	:								
11	Timestamp – Milliseconds of day (MS Byte)								
12	Accelerometer Data (20 Sample Sets)								
87	CRC								CRC
88	ETX								ETX

Table 6.9 Compact data format in the acceleration data packets.

Byte/Bit	7	6	5	4	3	2	1	0
0	X-Bit 0							
1	Y-Bit 0						X-Bit 9	
2	Z-Bit 0					Y-Bit 9		
3	X-Bit 0				Z-Bit 9			
4	X-Bit 9							
5	Y-Bit 0							
6	Z-Bit 0						Y-Bit 9	
7	X-Bit 0					Z-Bit 9		
8	Y-Bit 0				X-Bit 9			
9	Y-Bit 9							
10	Z-Bit 0							
11	X-Bit 0						Z-Bit 9	
12	Y-Bit 0					X-Bit 9		
13	Z-Bit 0				Y-Bit 9			
14	Z-Bit 9							
15	As Byte 0 (pattern repeats every 15 bytes).							

Life Sign Signal

A reception timeout mechanism is adopted in the Zephyr communication link to detect communication loss and recovery. The timeout is mutual, meaning the device sends the life sign signal periodically to the remote terminal, and the remote terminal sends the life sign signal to the device; both sides can recognize whether the other is alive. The timeout period is adjustable through a BT link configuration message (0xA4). In the situation where either terminal fails to receive a message from the other side over a pre-defined timeout period, it will close the connection and stop sending further messages. In this situation, the computer will try to reconstruct the channel to the Zephyr unit. The Zephyr unit, however, will wait for the reconnection from the computer. The default timeout period for the Zephyr unit is threes second, where it is 10 seconds for the computer. In the GUI, they are both set to be zero seconds,

meaning that no timeout is available for either sides. This simplifies the data transmission and reception process.

MATLAB GUI

The initial Graphical User Interface (GUI) to receive and display BioHarness data was created in MATLAB and is displayed in Figure 6.3. This GUI receives general packet data as input through the computer serial port and parses it accordingly. Information such as heart rate, respiration rate, skin temperature, and posture are displayed on the interface. The instantaneous values are shown in numeric form, and a line plot displays the 60-second trend for each signal. An adjustable threshold is given for each signal that acts as a warning line. When a parameter exceeds that threshold, the GUI will display a red light; otherwise the light stays green. The right panel contains a timer that displays the current time synced with the Zephyr unit. Once the "START" button is clicked, the program commands the Zephyr unit to transmit general packet data, and the interface begins to store these data into a text file. In the mean time, the START button will change to a "STOP" function, and another click on the button closes the link. An "Events" bar lists a series of typical activities, which can help the researcher keep track of the duration of each activity. By clicking the "Mark" button, the selected event and time will be put on the "Events History" panel in list form. Meanwhile, the event is saved in the output text file.

In the program, request messages are pre-stored as a constant vector and are called when the recording session starts. The data extraction algorithm is shown below.

MATLAB GUI Data Extraction Pseudocode

```
while( not stop)
{
    if( available byte size is 58)
    {

        read from buffer;
        extract time( hour: min: second);
        extract heart rate;
```

```
            extract respiration rate;
            extract skin temperature;
            extract posture;
            determine the threshold;
            write to text file;
            update plot;
            empty buffer;
        }
}
```

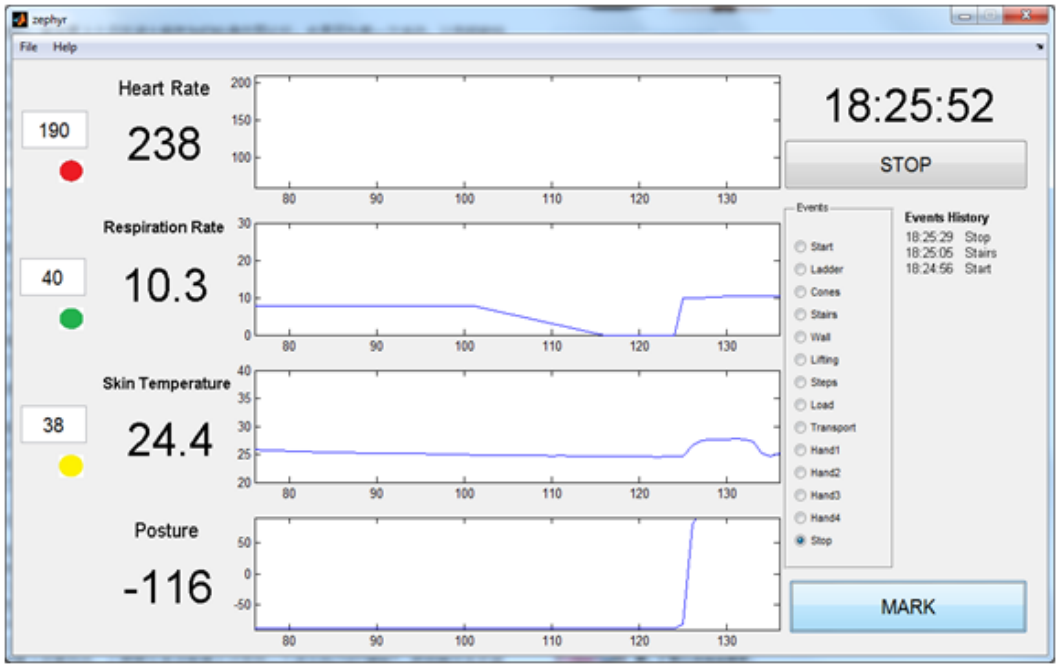

Figure 6.3 MATLAB GUI for the Zephyr BioHarness.

LabVIEW Interface

The second version of the GUI was implemented in LabVIEW (see Figure 6.4), where the goal was to create an interface that could integrate data from multiple sensors on an extended sensor network and combine them on one screen. LabVIEW virtual instruments (VIs) can be created using a graphical programming approach, where functional blocks are connected by lines that indicate data flow. This approach makes it easy to gather data and do simple processing on

these data using the pre-built function blocks, but sophisticated functions and algorithms that are not part of the LabVIEW distribution can be a challenge.

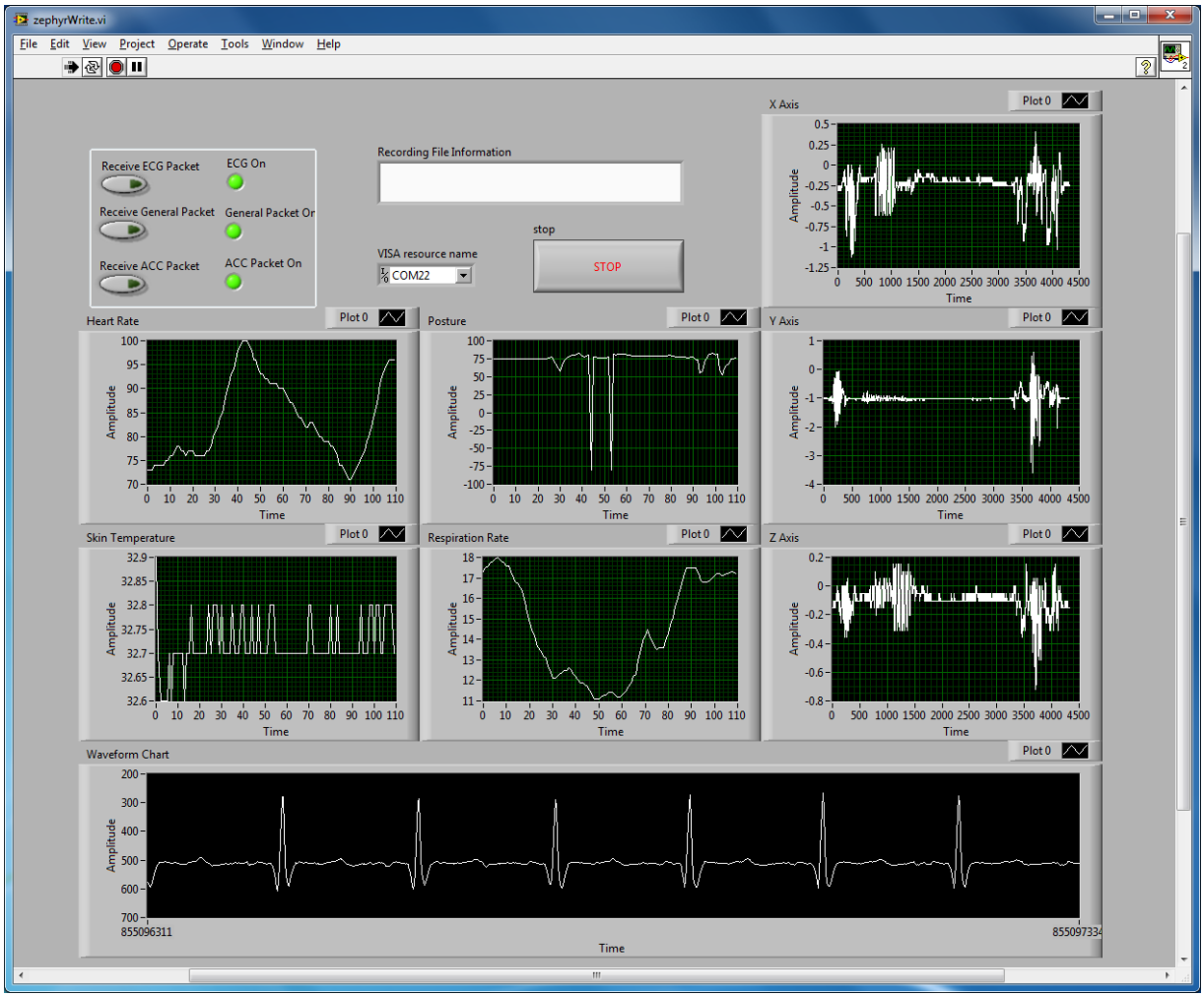

Figure 6.4 LabVIEW GUI for the Zephyr BioHarness.

As noted earlier, the update rate of each data stream is different and so is the length of each different packet. This requires a mechanism to handle , and correctly time align, diverse input data. One solution is to read the header of the buffer load first and then read the rest of the bytes according to the header information. Details are included in the flow diagram in Figure 6.5. Once more than three bytes are in the buffer, the program assumes there is at least a full size message in the buffer. The program reads the first three bytes, which contain the message ID and data payload. Then, the data payload plus two end bytes are read, and depending on the message ID, the payload portion is guided to the corresponding subroutine for processing. The process

iterates 100 times every second. This mechanism helps to correctly sort and display disparate data. The updated version of the GUI has the following features:

- **Display of various vital signs.** The four plots in the center display heart rate, skin temperature, respiration rate, and posture from the general packet data, respectively. A continuous ECG waveform is shown on the bottom, and three axes of acceleration data are displayed on the right side. The update rate of each plot is the same as the transmission rate of each data packet.
- **Independent signal control. Since** different data packages may be used in different scenarios, the GUI provides the ability to turn on and off the transmission of each data packet. In contrast, the original GUI from the company can only display general data packets and has no control of the provision of each signal. On the LabVIEW interface, the availability of each signal can be separately controlled using three buttons in the upper left corner. The LED light next to each button indicates whether the transmission of the particular data packet is enabled.
- **Signal recording and text file export.** A text file is created at the start of the program to save the data received from the GUI. Since data sampling rates vary, multiple files are required to export more than one type of data.

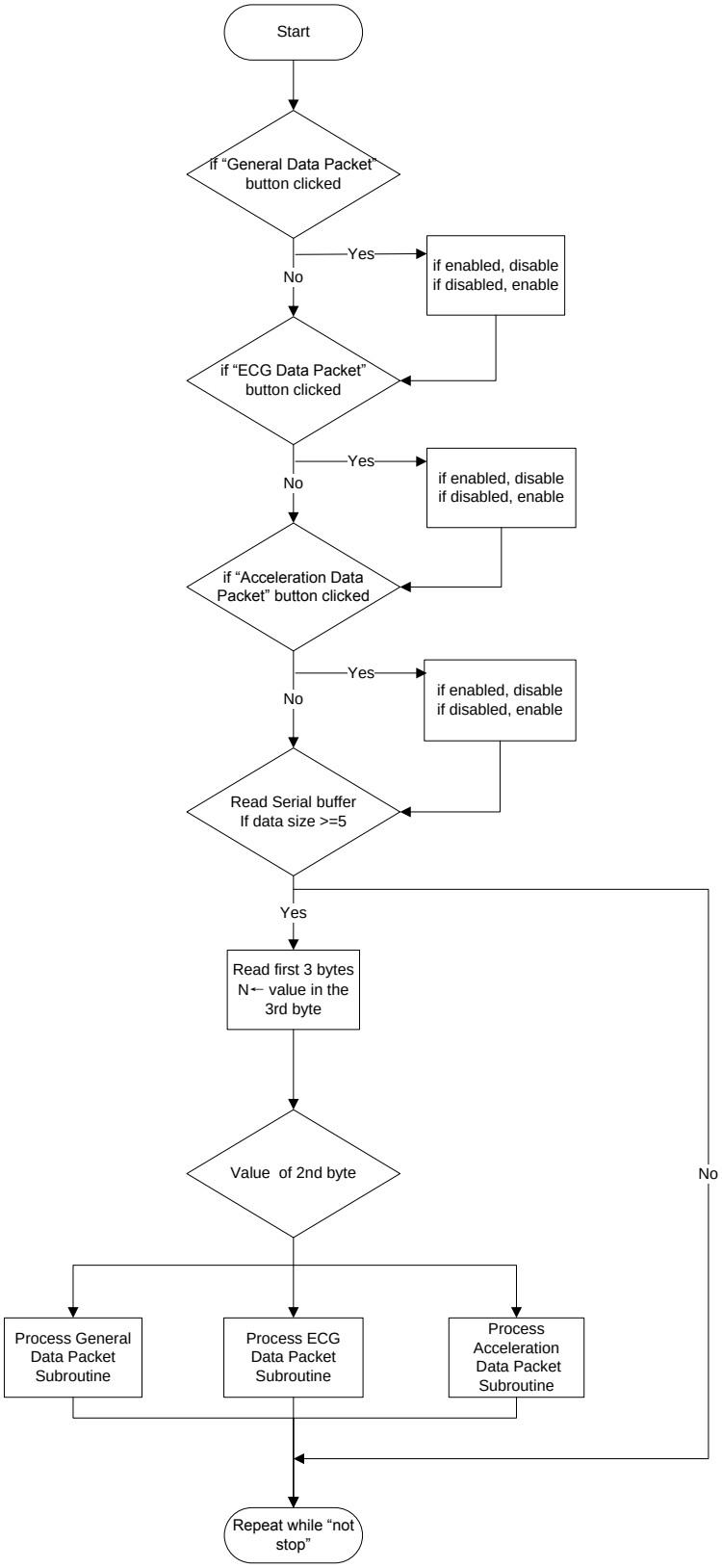

Figure 6.5 LabVIEW program flow chart.

Chapter 7 - Custom Board Design

One important goal for this research, moving forward, will be the ability to place a mobile accelerometer device on a subject and have the device classify activity type in real-time through the use of on-board classification algorithms. To that end, a custom board has been applied to gather early data for this work. The wireless hardware was developed by Devon Krenzel (KSU) for recent Cerner-funded research work focused on designing hardware and the affiliated firmware algorithms to predict the occurrence of slips and falls [55]. The custom board provides high-quality accelerometer data and has physical features well-suited to this PNFT work. A brief discussion of the board and design follow.

Wireless Accelerometer Hardware

The system is essentially a small wireless motion sensing board – see Figure 7.1. It is intended to be attached to the body and record acceleration and rotation via the on-board accelerometer and gyrometer sensing hardware. A JN 5148 Jennic microcontroller coordinates all tasks, such as data acquisition, data storage, ZigBee wireless signal transmission, and power management. Power for the system comes from either a portable lithium-ion button cell battery or from a USB serial cable, depending on the connectivity with a host/computer. The accelerometer/gyrometer chip uses a serial peripheral interface (SPI) bus to communicate with the microcontroller. Two flash chips help to store large quantities of data on-board. Wireless communication is realized through interactions with another ZigBee-enabled microcontroller module that connects to the computer USB port. The overall board size is 54x39x19 mm.

Figure 7.1 Custom accelerometer hardware.

Microcontroller and Transceiver

The Jennic JN 5148 is an ultra-low-power wireless microcontroller. The microcontroller integrates a 32-bit RISC processor with a 2.4 GHz IEEE 802.15.4 (ZigBee) compliant transceiver, 192 kB of ROM, 96 kB of RAM, and abundant analog and digital peripherals. IEEE 802.15.4 supports transmission distances up to 100 m. When the subject performs activities out of that range, the raw data and processed feature data can be saved locally on the flash modules. Alternatively, an app could be developed for mobile phones that support newer versions of the Android OS (more specifically, the Android systems that support USB host mode) for uninterrupted data transmission.

Accelerometer

An MCA3000 D01 MEMS three-axis digital accelerometer was selected for this design due to its small size and low power consumption. The supply voltage can be 1.7 V to 3.6 V, and power consumption varies according to the different sampling rates. At a sampling rate of 100 Hz, the chip consumes 50 uA of current. The acceleration range can be chosen from ±2 g or ±8 g with 8-bit resolution. Since some intense activity involves accelerations larger than 2 g, the ±8 g range is chosen to prevent sensor saturation. The least significant bit contributes 71 mg. In other words, one g is divided into 14 steps.

The sensor does not provide the calibration function to the user. Instead, each sensor is factory calibrated before shipping. It has been seen, in one sensor being tested, that the measured acceleration value does not reflect the actual acceleration, and this cannot be resolved except by replacing the chip.

This device also supports a detector mode, where the sensor stays dormant until a certain level of motion is detected. The motion can be a free fall or simple motion. Once the motion is detected, the device sends out an interrupt flag to notify the microcontroller. This function can be incorporated into the device in the future.

The accelerometer module communicates with the microcontroller through a serial peripheral interface (SPI) bus. The bus requires 4 channels between the master (the microcontroller) and the slave (the sensor). The master supplies the serial clock signal on the SCLK channel, since the data are transmitted synchronously according to the clock signal. Common clock frequencies exist in the range of 1 to 100 MHz. The MOSI (Master Out Slave In)

and MISO (Master In Slave Out) channels support actual data transmission. These two channels ensure that the protocol works in full duplex mode. The final channel, SS (slave select), provides a method for a master to select from several slaves that are connected to it. When there are multiple SPI slaves communicating with the same master, the SCLK, MOSI, and MISO channels are connected in series, and the SS channels are connected to the master in parallel. The clock and data signals are sent to every slave, but only the slave being selected with its SS pulled down to zero reacts to the data coming in. For a cooperative slave situation, a more complex daisy chain topology may be applied. No communication between sensors is required, so a simple direct connection is utilized.

Chapter 8 - Firmware development

A protocol named JenNet is used in the firmware development. JenNet is a proprietary protocol that builds on IEEE 802.15.4. Similar to ZigBee PRO, self-repairing is supported when the network fails. Both star and tree topologies can be operated under the protocol. An API library, named Jenie API, is provided for application layer programming. Three types of nodes incorporate three distinct programs, as the functionality of these nodes varies. The node types are a coordinator, which is the hub in the topology, an end device, which is the leaf node in the topology, and a router, which connects coordinator and end devices. Notice that in simple applications, an end device can be linked to a coordinator directly without a router. Only one coordinator at a time can exist in the network topology.

Like all 802.15.4 protocols, the network operates in the 2400 MHz radio frequency band. The frequencies span from 2405 MHz to 2480 MHz and are divided into 16 channels (channel 11 through channel 26). The data rate is 250 kbps, more than adequate for acceleration data transmission. When setting up the channel, the program automatically assigns the channel with the least detected activity for use. With a standard transmission module, the output power is 0 dBm and can reach a range of 200 m in an open environment, suitable for current planetary field tests. If a longer distance is desired in future applications, the system can be replaced by a high-power module with a 15 dBm output power. Alternatively, routers can be set in between to relay the data.

An integrated peripheral (IP) API library and low-power-radio frequency (LPRF) board application programming interface (API) are available to control the resources in the microcontroller and on the evaluation board. The IP API controls resources like analog-to-digital converters, digital-to-analog converter, UARTS, timers, pulse counters, SPIs, and external flash memory. The LPRF API controls resources like the light, humidity, and temperature sensor, the on-board LED, buttons, and an LCD screen on the controller board. Functions from three APIs are utilized to fulfill the program written for this effort.

The underlying operating system of JenNet is called a basic operating system (BOS). It is hidden from the user, and most functions (e.g., initialization) are called from a user application with callbacks.

The main function, `vJenie_CbMain()`, is called periodically. However, since this periodicity is not selectable, this function presents obstacles for the user in terms of programming and control. For this reason, most of the sensor data collection and related functions are realized using hardware event functions in **`vJenie_CbHwEvent()`**. `Timer`, an internal periodically triggered interrupt, is employed when receiving accelerometer data.

The JN5148 microcontroller incorporates two on-chip timers. The applications for such timers are numerous. A timer can be used as a counter to count the input pulses entering the designated pin. Signal generators, e.g., a PWM waveform, can be realized using this function. In our case, a timer is used as a trigger to evoke an interrupt for a certain period.

The clock used by the timer is the internal 16 MHz signal. A prescaler is an exponential factor(2^N) by which the clock frequency is divided, in order to yield a smaller number. It provides course adjustment for the original clock. A postscaler is a linear factor that allows one to fine-tune the frequency to set it to a desired value. By manipulating the prescaler and postscaler concurrently, we can obtain an interrupt frequency of 50.01 Hz, matched to the desired accelerometer sample frequency. Since the intrinsic sampling rate of the accelerometer is set to 100 Hz, we assume that every time we sample the data, new data are ready in the output registers. During each interrupt, the latest sampled data obtained from these three axes is read out by polling the corresponding register on the sensor. The SPI clock is set to 250 kHz.

After the data are stored in local memory, they are transmitted to the coordinator. Each packet is 8 bytes long, as illustrated in Figure 8.1. Byte 0 is the packet ID. There are two types of IDs: a Join ID (used to request to join the network) and a Message ID (intended to be used from a registered node to transmit actual information). The acceleration data packet uses a message ID to identify itself to the coordinator. Bytes 1 to 3 and bytes 4 to 6 are the two most recently sampled acceleration data values arranged in X, Y, Z axis order. Byte 7 is empty and is reserved for future use.

Msg ID	X axis	Y axis	Z axis	X axis	Y axis	Z axis	Reserved

Figure 8.1 ZigBee acceleration packet data format.

Sleep Mode

In a battery driven mobile device, power consumption is important. To reduce power consumption, the firmware is optimized so that unnecessary processes are eliminated in the program. At the same time, the microcontroller sleep mode is utilized to further reduce power consumption. Since the coordinator gathers information from more than one end device, and the data can arrive at any time, the a coordinator sleep mode is not permitted. Similarly, a router, if existing in the network, cannot sleep for the same reason. Fortunately, both types of nodes can be flexibly placed next to a power outlet and be powered by a DC source, so their power consumption is not a major issue.

On the other hand, end devices are usually powered by a battery, and their location of use in our case is restricted to the body. Connecting an end device to a power outlet is unrealistic. Huge power consumption by the transceiver calls for the need to sleep when the microcontroller is idle.

Signal Strength

Reliable signal transmission is crucial to the overall performance of this compact wireless sensor board. The transmission range is related to the signal strength received from the coordinator. Generally, as a signal diminishes to -100 dBm, it cannot be received by a coordinator anymore, as it is buried in noise. One way to characterize the intensity of the received power is through a Link Quality Indication (LQI) parameter. It is an 8-bit integer ranging from 0 to 255, where 255 represents the strongest signal. For the JN 5148, the detected power can be reconstructed using the equation given in the API user guide:

$$P = \frac{(7 \times LQI - 1970)}{20} \tag{8.1}$$

The range for detected power is [-9.25, -98.5] dBm. Notice that when the readout from the LQI is 0 or 255, that does not mean the measured power is -98.5 dBm or -9.25 dBm, as the actual value may be even lower than the minimum detectable level or higher than the maximum detectable level. Although the possibility of exceeding the maximum level is low, it is possible that the minimum power reception level will be reached as the distance between the sensor board and the coordinator increases.

For every packet received, the coordinator returns the detected radio signal strength. This value can be retrieved by calling **u8Api_GetLastPktLqi()** in the data event handler. During our data collection, we call the function to find the actual power detected by the coordinator. Using this function, we can determine the approximate transmission range of the sensor board.

Data Reception

Every time a data packet is received by the coordinator, a data event handler is evoked in **vJenie_CbStackDataEvent()** on the coordinator. As this handler operates, the data are analyzed and stored. When a data packet arrives, its packet ID is first analyzed. If the packet is indeed a data message, then the length is checked to see if it is eight bytes. If so, then the address of the sensor board is compared with all of the registered addresses to determine if the board has been bound with the coordinator. When the data are from a known board, they are then saved in the local arrays. At the same time, the data are also concatenated to a long string. Every 50 data packets, the coordinator sends out the serial packet to the computer. The pseudo code for the data event handler follows:

Data Event Handler Pseudocode

```
if data length >0
     if packet ID is message_ID
          if data length is 8
               search for the node with registered address
               if no address found, return
               store data to local memory
               missing data == 0
               concatenate data to serial packet string
               when count == 50
                    send the package through serial port
                    reset count
               increment count
     if packet ID is Join_ID
          process Join_ID
```

Received packets are repacked and transmitted to the computer using an RS 232 protocol. The time taken to transmit one serial packet is long enough that, even at an update rate of 50 Hz, it will jam the normal signal-receiving channel. To solve this problem, a longer form of serial packet is employed. This packet stores up to one second of data in a format shown in Figure 8.2. Each received packet contributes 4 attributes to the serial packet. The first three attributes are the acceleration values on the three axes, and the forth is the LQI from each packet. Notice that since the data are transmitted in an ASC II code format, the 8-bit unsigned integer value is converted to several bytes representing a decimal value, so one attribute does not necessarily cover only one byte, but may be as wide as three bytes. A space byte is inserted between two attributes, and an end-of-serial-packet byte is attached to the tail. Data processing and analysis are performed using the GUI.

X	Y	Z	LQI	X	…	X	Y	Z	LQI

Figure 8.2 Serial transmission data format.

Chapter 9 - GUI Design and Real-time Algorithm

Implementation of the Real-Time Algorithm

The benefit of on-board activity recognition lies in its potential to vastly reduce the information that needs to be transmitted wirelessly. For the accelerometer data alone, without any data compression, every second generates at least 150 bytes of data that need to be transmitted. By transmitting extracted features from the raw data, the data payload can be reduced to 25 bytes or less per second, 16.67% (or less) of the original, assuming that data even need to be transmitted continuously for some applications. Once the activity recognition process is performed on-board, only one byte of the result will be required to be sent to the coordinator periodically, drastically reducing the overall maximum power consumption. In an even more power-constrained scenario, a result might be transmitted to the coordinator only when an abnormal event is detected.

The implementation of the activity detection algorithm directly on the microcontroller is a challenge for several reasons.

1. While the original data consist of only integer numbers representing acceleration relative to gravity, most of the features require mathematical manipulation, which often results in a decimal number. The conventional representation of a decimal number on a computer is in floating point format. Unfortunately, that representation is unavailable on the JN5148 microcontroller – many other ultra-low-power microcontrollers share that constraint. Theoretically, this drawback can be overcome by using a floating point emulation library on, e.g., the gcc compiler, but the code transplantation work load could be significant and the computing capability of the microcontroller might still be questionable.

2. As the feature selection results in Chapter 4 indicate, frequency domain features play an important role in activity recognition. Several activities can be singularly recognized by frequency domain features, e.g., average energy. However, this requires that a Fast Fourier Transform (FFT) be performed on the raw data to acquire the frequency domain coefficients. Not only is the Fourier Transform itself computational costly, but the FFT calculations would require a floating point emulator. Given both issues, it could be a challenge to achieve the frequency-domain data processing and activity recognition algorithms on the board. That said, the idea of processing acceleration data and performing activity recognition (with frequency-domain

parameters) is not impossible. Recent microcontrollers based on ARM architectures, e.g., an STM32 cortex MCU, do support floating point calculations and have higher clock rates, making on-chip FFTs sensible.

In the context of the current project and resources, real time activity recognition is realized in another way. All raw data are initially transmitted to the coordinator, and the coordinator sends these data to a local computer. On the computer, the data are obtained through a MATLAB-based GUI, and the rest of the processing is executed in that environment.

Real-Time Naïve Bayes Classifier

As mentioned in Chapter 5, a Naïve Bayes classifier constructs a statistical model based on the training model. The classification technique uses the model to calculate the probability of each activity and pick the activity with largest probability. This helps to realize the classifier in a real-time sense, as training results can be easily stored in a microcontroller or in a program, so that no heavy computation is required during the classification process. The Naïve Bayes classifier is an effective classifier in that it has strong mathematical underpinnings and can be applied to a real-time situation with a low calculation expenditure.

When training the Naïve Bayes classifier, a Gaussian distribution model is calculated for each class. Specifically, the mean and standard deviation of the data are found. Then, the prior probability of each activity is calculated based on how likely each activity is to take place. In our situation, as we collect training data for equal times on each of 9 activities, we assume each activity has an identical likelihood to happen. After the training process, Gaussian model parameters and prior probability are stored as the classification model.

A pseudo code for the training process follows:

Naïve Bayes Training Psudocode

```
for each class
   find mean value on each feature;
   find standard deviation value on each feature;
   find a priori of each activity;
end
```

During the test, when an instance arrives, an *a posteriori* probability is calculated for each class. Then, the class with the maximum probability is assigned to the instance. The judgment is done once for every instance that arrives. Therefore, the classifier is realized in a real-time manner. The pseudo code for the testing process is explained below.

Naïve Bayes Testing Psudocode

```
for each instance
     for each class
          calculate an a posteriori probability on each class;
     end
     find the maximum probability and assign the class to the
instance
end
```

Figure 9.1 depicts one activity recognition result for a time series. The independent axis represents the activity instances, and the dependent axis represents the activity classes. Activity instances are arranged according to their original classes, shown in blue dots. Each series of blue dots indicates an activity performed by 5 subjects. Red dots represent the classified activity using a real-time Naïve Bayes approach. Overlapping dots indicate a correct recognition result. The recognition accuracy is 73.51%, matching the off-line Naïve Bayes algorithm.

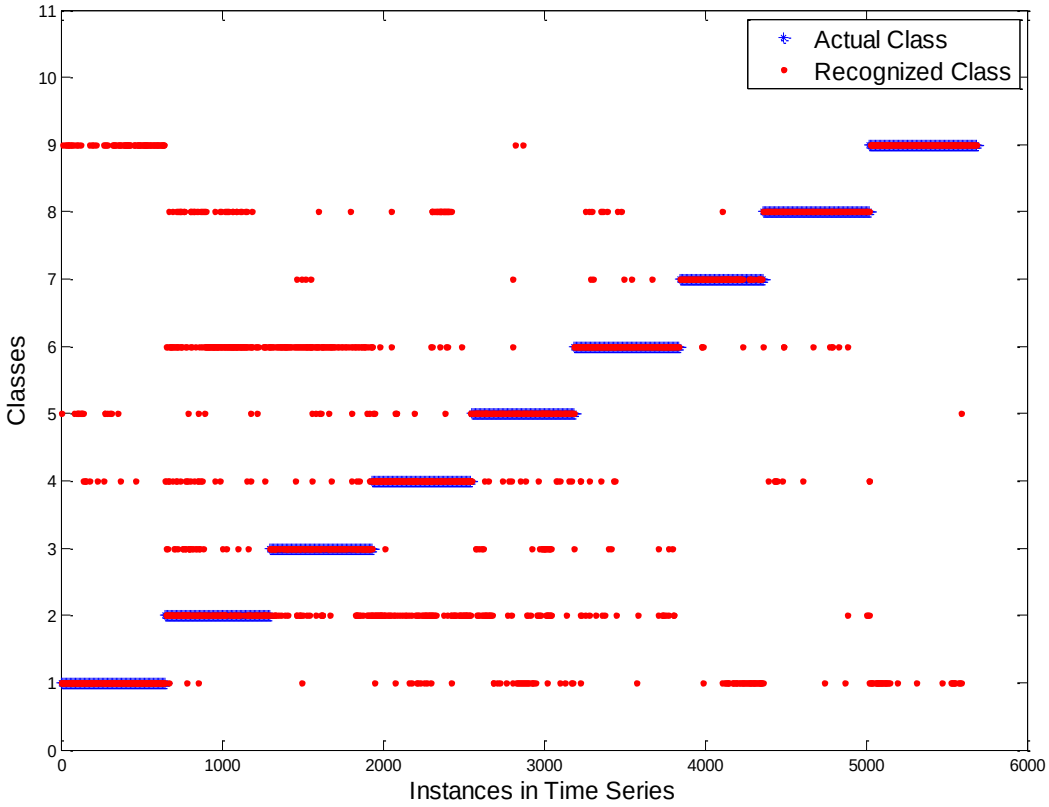

Figure 9.1 Real-time activity recognition result using Naive Bayes.

Graphic User Interface Design

The MATALAB GUI for this work is an updated version of the Zephyr BioHarness GUI presented earlier. Three rows of numeric text and plots indicate the acceleration on the three axes. The last row is used to demonstrate the signal quality for each packet. The currently recognized activity is displayed in the center of the bottom row.

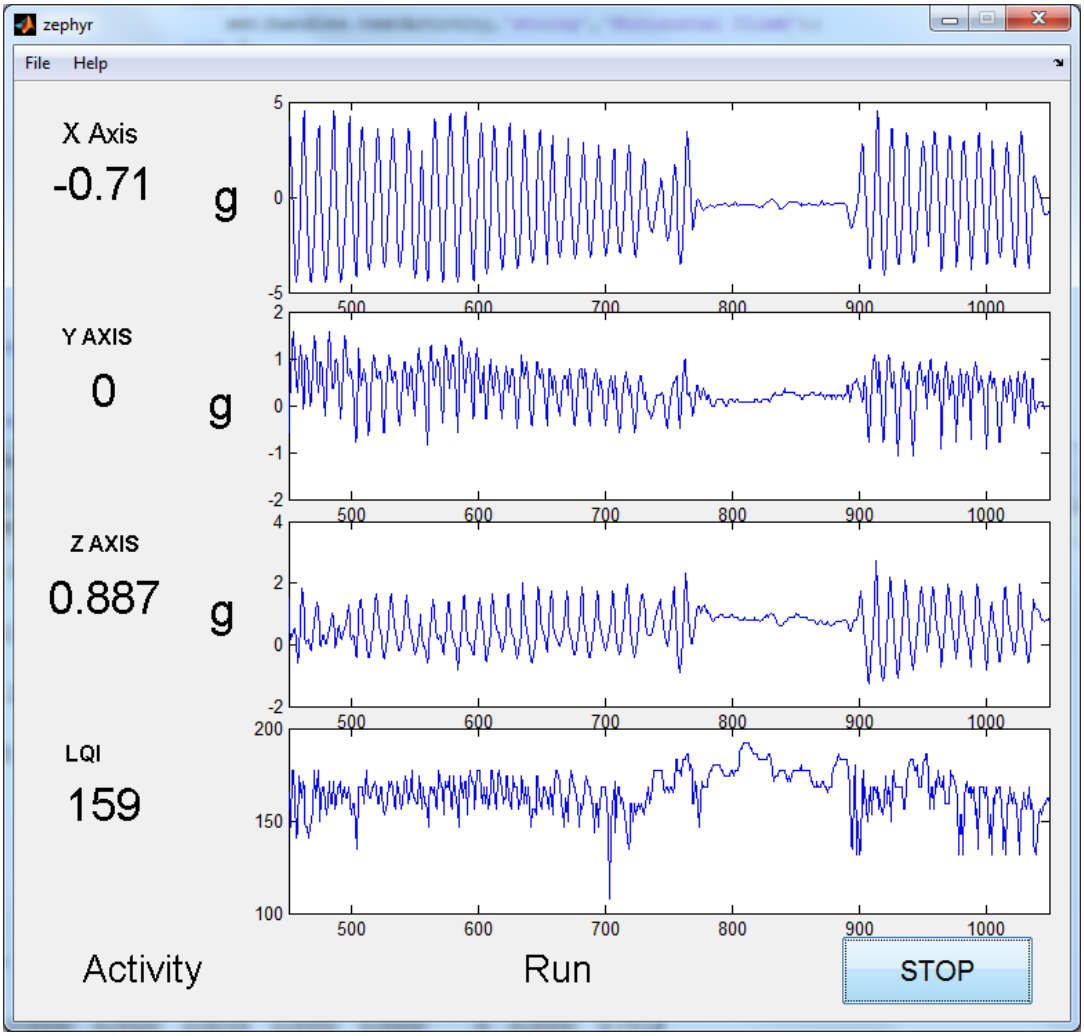

Figure 9.2 MATLAB GUI for real-time activity recognition.

Chapter 10 - Conclusions

Activity recognition has been a widely studied area that has resulted in various research products. When applied to activities that astronauts might undertake as part of an EVA, activity recognition may help researchers to better monitor mission-task progress and astronaut condition. Further, these techniques may help to predict an astronaut's potential to complete tasks based on existing data. To address the idea of activity recognition in zero- or reduced-gravity environments, six different activities were created by the KSU team. These Planetary Navigation Field Tests (PNFTs), intended to reflect typical tasks that one might perform as part of an EVA, have been used to assess both fatigue and motion.

A commercial health monitoring system, a Zephyr BioHarness, was used to acquire acceleration data and other health data during these PNFTs. The Zephyr BioHarness sensing devices are mounted on the left side of the chest – a convenient position for human movement and cardiorespiratory monitoring that does not obstruct limb movement. Two GUIs were created in MATLAB and LabVIEW to visualize BioHarness data.

A set of 25 time- and frequency-domain features were extracted from these Zephyr accelerometer data and compared in an effort to select the features that would be the most helpful for activity classification. Generally, the results indicate that both time-domain and frequency-domain features contribute to task recognition. The acceleration combinations from the three axes differ the most between the various activities and are more useful in the recognition process. Several machine learning algorithms were applied to these features, where the average recognition accuracy could be greater than 86%.

A custom wireless sensor board was also evaluated for its potential to acquire accelerometer data and identify the affiliated activities in real time. The board hardware incorporates the JenNet protocol as the wireless transmission protocol. An updated GUI in MATLAB was designed to acquire the data from this custom board and implement real-time activity recognition algorithms. The Naïve Bayes approach was selected for this initial implementation, as it offers concrete mathematical underpinnings and a straightforward classification process.

Future Work

A prototype hardware accelerometer board has been used in this recent work. Several elements of this design can be improved. The microcontroller can be upgraded to a model that supports floating point calculations so that the decision making process can be directly implemented on the microcontroller. With the upgrade, the wireless protocol may be reconsidered as new standards emerge (e.g., Bluetooth low energy) to improve range and/or consume less power. A digital accelerometer with a higher resolution may replace the current sensor to yield data with higher quantization fidelity. A physical connector with an elastic strap or stick may be added on the board so that it can be more easily attached to the body.

Regarding activity recognition, a deeper understanding of these algorithms will be necessary so that more sophisticated algorithms can be implemented and tailored to the PNFT activities. Semi-supervised training processes may be considered, since the initial training process can be accomplished off line, but the training would not stop when new data arrive. These unlabeled data may continue to help to adjust the classifier. As for the actual usage of the activity recognition system, both the sensor board system and the coordinator must be placed inside a space suit at some point. A temporary storage module needs to be implemented to record data while the astronaut is in session. In addition, the sensor board's transmission performance within the space suit is unknown.

References

[1] E. M. Tapia, S. S. Intille, and K. Larson, *Activity recognition in the home using simple and ubiquitous sensors*: Springer, 2004.

[2] R. Polana and R. Nelson, "Recognizing activities," in *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, 1994, pp. 815-818.

[3] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," in *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, 1997, pp. 90-102.

[4] L. Palmerini, L. Rocchi, S. Mellone, F. Valzania, and L. Chiari, "Feature selection for accelerometer-based posture analysis in Parkinson's disease," *Information Technology in Biomedicine, IEEE Transactions on,* vol. 15, pp. 481-490, 2011.

[5] N. Oliver and F. Flores-Mangas, "MPTrain: a mobile, music and physiology-based personal trainer," in *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, 2006, pp. 21-28.

[6] A. Frazer, B. Pitts, J. Hoffman, and D. Newman, "Astronaut Performance: Implications for Future Space Suit Design," in *IAF abstracts, 34th COSPAR Scientific Assembly,* 2002, p. 179.

[7] J. L. Rochlis and D. J. Newman, "A tactile display for international space station (ISS) extravehicular activity (EVA)," *Aviation, space, and environmental medicine,* vol. 71, p. 571, 2000.

[8] N. Robertson and I. Reid, "A general method for human activity recognition in video," *Computer Vision and Image Understanding,* vol. 104, pp. 232-248, 2006.

[9] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 22, pp. 747-757, 2000.

[10] K. Aminian, P. Robert, E. Buchser, B. Rutschmann, D. Hayoz, and M. Depairon, "Physical activity monitoring based on accelerometry: validation and comparison with video observation," *Medical & biological engineering & computing,* vol. 37, pp. 304-308, 1999.

[11] C. V. Bouten, K. T. Koekkoek, M. Verduin, R. Kodde, and J. D. Janssen, "A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity," *Biomedical Engineering, IEEE Transactions on,* vol. 44, pp. 136-147, 1997.

[12] F. Foerster and J. Fahrenberg, "Motion pattern and posture: correctly assessed by calibrated accelerometers," *Behavior research methods, instruments, & computers,* vol. 32, pp. 450-457, 2000.

[13] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim, "A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer," *Information Technology in Biomedicine, IEEE Transactions on,* vol. 14, pp. 1166-1172, 2010.

[14] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*, ed: Springer, 2004, pp. 1-17.

[15] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Proceedings of the national conference on artificial intelligence*, 2005, p. 1541.
[16] S. J. Preece, J. Y. Goulermas, L. P. Kenney, D. Howard, K. Meijer, and R. Crompton, "Activity identification using body-mounted sensors—a review of classification techniques," *Physiological measurement*, vol. 30, p. R1, 2009.
[17] T. Huynh and B. Schiele, "Analyzing features for activity recognition," in *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, 2005, pp. 159-163.
[18] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, pp. 119-128, 2006.
[19] S. Wang, J. Yang, N. Chen, X. Chen, and Q. Zhang, "Human activity recognition with user-free accelerometers in the sensor networks," in *Neural Networks and Brain, 2005. ICNN&B'05. International Conference on*, 2005, pp. 1212-1217.
[20] L. Biel, O. Pettersson, L. Philipson, and P. Wide, "ECG analysis: a new approach in human identification," *Instrumentation and Measurement, IEEE Transactions on*, vol. 50, pp. 808-812, 2001.
[21] G.-Z. Yang and M. Yacoub, *Body sensor networks* vol. 6: Springer London, 2006.
[22] D. Jabaudon, J. Sztajzel, K. Sievert, T. Landis, and R. Sztajzel, "Usefulness of ambulatory 7-day ECG monitoring for the detection of atrial fibrillation and flutter after acute stroke and transient ischemic attack," *Stroke*, vol. 35, pp. 1647-1651, 2004.
[23] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, and D. Howard, "A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data," *Biomedical Engineering, IEEE Transactions on*, vol. 56, pp. 871-879, 2009.
[24] M. Zhang and A. A. Sawchuk, "A feature selection-based framework for human activity recognition using wearable multimodal sensors," in *Proceedings of the 6th International Conference on Body Area Networks*, 2011, pp. 92-98.
[25] E. Alpaydin, *Introduction to machine learning*: The MIT Press, 2004.
[26] L. Liao, "Location-based activity recognition," University of Washington, 2006.
[27] J.-Y. Yang, J.-S. Wang, and Y.-P. Chen, "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers," *Pattern recognition letters*, vol. 29, pp. 2213-2220, 2008.
[28] A. Madabhushi and J. Aggarwal, "A bayesian approach to human activity recognition," in *Visual Surveillance, 1999. Second IEEE Workshop on,(VS'99)*, 1999, pp. 25-32.
[29] C. Ade, R. Broxterman, G. Gadbury, D. Schinstock, and S. Warren, "Standarized Exercise Test to Evaluate Planetary Mission Readiness," *HRP 2012, NASA Human Research Program Investigators' workshop*, February 14 - 16, 2012 2012.
[30] D. Gude, R. Broxterman, C. Ade, T. Barstow, T. Nelson, W. Song, *et al.*, "Automated hand-forearm ergometer data collection system," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012, pp. 2379-2382.
[31] W. Song, C. Ade, R. Broxterman, T. Barstow, T. Nelson, and S. Warren, "Activity recognition in planetary navigation field tests using classification algorithms applied to

accelerometer data," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012, pp. 1586-1589.
[32] R. Broxterman, C. Ade, GaryGadbury, D. Schinstock, S. Warren, and T. Barstow, "Physiological responses during simulated planetary field tests," *American college of Sports Medicine*, May 29-June 2 2012.
[33] C. Ade, R. Broxterman, S. Warren, R. Taylor, G. Gadbury, and T. Barstow, "Development of Standardized Exercise Tests for Predicting Planetary Task Performance," *18th IAA Humans in Space Symposium, International Academy of Astronautics*, April 11 - 15, 2011 2011.
[34] C. M. Clements, M. J. Buller, A. P. Welles, and W. J. Tharion, "Real time gait pattern classification from chest worn accelerometry during a loaded road march," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012, pp. 364-367.
[35] R. LeMoyne, T. Mastroianni, and W. Grundfest, "Wireless accelerometer iPod application for quantifying gait characteristics," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, 2011, pp. 7904-7907.
[36] M. Sipos, P. Paces, J. Rohac, and P. Nováček, "Analyses of Triaxial Accelerometer Calibration Algorithms," *Sensors Journal, IEEE*, vol. 12, pp. 1157-1165, 2012.
[37] L. Oudre, A. Lung-Yut-Fong, and P. Bianchi, "Segmentation of accelerometer signals recorded during continuous treadmill walking," in *Proceedings of the European Signal Processing Conference (EUSIPCO), Barcelona, Spain*, 2011.
[38] D. Curone, G. M. Bertolotti, A. Cristiani, E. L. Secco, and G. Magenes, "A real-time and self-calibrating algorithm based on triaxial accelerometer signals for the detection of human posture and activity," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, pp. 1098-1105, 2010.
[39] http://www.zephyr-technology.com/products/bioharness-3/.
[40] W. Song, C. Ade, R. Broxterman, T. Nelson, D. Gude, T. Barstow, *et al.*, "Classification Algorithms Applied to Accelerometer Data as a Means to Identify Subject Activities Related to Planetary Navigation Tasks," *HRP 2013, NASA Human Research Program Investigators' workshop*, Febrary 11-14, 2013 2013.
[41] M. Ermes, J. Parkka, J. Mantyjarvi, and I. Korhonen, "Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, pp. 20-26, 2008.
[42] http://www.delsys.com/products/trignowireless.html.
[43] M. Nyan, F. Tay, K. Seah, and Y. Sitoh, "Classification of gait patterns in the time–frequency domain," *Journal of biomechanics*, vol. 39, pp. 2647-2656, 2006.
[44] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, pp. 156-167, 2006.
[45] C. E. Shannon and W. Weaver, "A mathematical theory of communication," ed: American Telephone and Telegraph Company, 1948.
[46] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*: Morgan Kaufmann, 2005.
[47] M. A. Hall, "Correlation-based feature selection for machine learning," The University of Waikato, 1999.

[48] J. Lee Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician,* vol. 42, pp. 59-66, 1988.
[49] H. Liu and R. Setiono, "A probabilistic approach to feature selection-a filter solution," in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, 1996, pp. 319-327.
[50] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning* vol. 2: MIT press Cambridge, 2006.
[51] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, "Activity recognition based on semi-supervised learning," in *Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on*, 2007, pp. 469-475.
[52] M. Stikic, K. Van Laerhoven, and B. Schiele, "Exploring semi-supervised and active learning for activity recognition," in *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, 2008, pp. 81-88.
[53] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 1995, pp. 338-345.
[54] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*: Wiley-interscience, 2012.
[55] D. Krenzel, S. Warren, K. Li, B. Natarajan, and G. Singh, "Wireless slips and falls prediction system," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012, pp. 4042-4045.