Analysis of Activity Data from Smartphones

Jamieson Brynes, Somerville College

May 16, 2017

FINAL HONOUR SCHOOL OF
ENG / EEM (delete as appropriate)

DECLARATION OF AUTHORSHIP

You should complete this certificate. It should be bound into your fourth year project report, immediately after your title page. Three copies of the report should be submitted to the Chairman of examiners for your Honour School, c/o Clerk of the Schools, examination Schools, High Street, Oxford.

Name (in capitals): ……

College (in capitals): …… **Supervisor:** ……

Title of project (in capitals): ……

Page count (excluding risk and COSHH assessments): ……

Please tick to confirm the following:

I have read and understood the University's disciplinary regulations concerning conduct in examinations and, in particular, the regulations on plagiarism (*The University Student Handbook. The Proctors' and Assessors' Memorandum, Section 8.8*; available at https://www.ox.ac.uk/students/academic/student-handbook) ☐

I have read and understood the Education Committee's information and guidance on academic good practice and plagiarism at https://www.ox.ac.uk/students/academic/guidance/skills. ☐

The project report I am submitting is entirely my own work except where otherwise indicated. ☐

It has not been submitted, either partially or in full, for another Honour School or qualification of this University (except where the Special Regulations for the subject permit this), or for a qualification at any other institution. ☐

I have clearly indicated the presence of all material I have quoted from other sources, including any diagrams, charts, tables or graphs. ☐

I have clearly indicated the presence of all paraphrased material with appropriate references. ☐

I have acknowledged appropriately any assistance I have received in addition to that provided by my supervisor. ☐

I have not copied from the work of any other candidate. ☐

I have not used the services of any agency providing specimen, model or ghostwritten work in the preparation of this project report. (See also section 2.4 of Statute XI on University Discipline under which members of the University are prohibited from providing material of this nature for candidates in examinations at this University or elsewhere: http://www.admin.ox.ac.uk/statutes/352-051a.shtml.) ☐

The project report does not exceed 50 pages (including all diagrams, photographs, references and appendices). ☐

I agree to retain an electronic copy of this work until the publication of my final examination result, except where submission in hand-written format is permitted. ☐

I agree to make any such electronic copy available to the examiners should it be necessary to confirm my word count or to check for plagiarism. ☐

Candidate's signature: …….. **Date:** ……..

Acknowledgements

In addition to the guidance and advice offered to me by my supervisor Lionel Tarassenko, I would also like to acknowledge the contributions of Carmelo Velardo and Dario Salvi to this project. They gave me their time and advice as well as agreeing to walk around the Institute of Biomedical Engineering with pink straps on their legs and feet to aid me in data collection. These three outstanding individuals were more than generous and their contributions were invaluable to the success of this project.

Abstract

Chronic diseases affect large sections of the general population and incur large costs to the healthcare system, to the tune of $1.4 trillion per year in the United States. Patients with chronic diseases require regular evaluation through the collection and analysis of data. This project set out to evaluate the information that could be extracted from accelerometer data for people living with chronic diseases. Algorithms were developed for step counting using the accelerometer within a smartphone device and for sleep detection using a wrist-mounted accelerometer. These algorithms derived two metrics of interest to healthcare professionals, the level of activity and sleep quality. Since smartphones and wearables have high penetration in the population and continue to proliferate, healthcare professionals would be able to utilize these algorithms to enable remote data collection.

The step-counting algorithm is modular and based on the Windowed Peak Detection design. A ground-truth device was designed to enable rapid data collection for optimization and validation of the step-counting algorithm. A database was created from the data recordings acquired using this ground-truth device. The database was used to optimize the step counting algorithm resulting in a median step counting accuracy of 96.8% across the entire database. Optimization for specific scenarios was also performed giving high accuracy, up to 99% in some cases.

The sleep-detection algorithm is based on classification with logistic regression. A publicly available database was utilized for the optimization of this algorithm. A filter with a hard-limiter after was used to process the logistic regression output giving results that more closely resemble sleep patterns. This methodology gives a median total time asleep error of 10.9%. This improves upon previous attempts on the same database by approximately 10%.

The source code for both algorithms will be released along with the step-counting database that was created during the project. The high accuracy achieved by the step counting algorithm means that it is ready to be implemented in a medical context, for example, in the six minute walk test which is a standard procedure for evaluating the capabilities of patients with congestive heart failure. The improvements achieved by the sleep detection algorithm are a step forward and the nature of the algorithm allows for further refinements and improvements.

Contents

1 Overview

This project is focused on integrating new technological developments, like smartphones and wearable devices, into medical data acquisition so that healthcare professionals have the capability to make use of large amounts of data related to physical activity and physiology to inform their decision-making when managing the health of individuals. Up until now, in order to capture data from a patient, the patient would have to visit their General Practitioner (GP) or a healthcare professional had to visit the patient; this is costly in both time and financial resources, so this project seeks to lessen some of these barriers.

By using sensors provided in smartphones and wearables, this project seeks to monitor and quantify daily activities that are strongly related to a patient's health, like walking and sleeping.

2 Motivation

2.1 Pulmonary Disease and Heart Failure

Chronic diseases are defined as health problems that require on-going management and treatment for years or decades [1]. In the UK, 17.5 million people have a chronic disease, two-thirds of these are aged 75 or older. This presents a large financial and logistic challenge as these patients with these conditions require long term care. In the US, chronic diseases affect 130 million people, which generates around $1.4 trillion a year in healthcare costs [1] or $10.77 per patient per year. Two of these diseases are Chronic Obstructive Pulmonary Disease (COPD) and heart failure.

Chronic Obstructive Pulmonary Disease and heart failure affect a person's ability to engage in physical activity and both tend affect a similar group of people [2, 3]. By monitoring a person, who is suffering from either of these diseases, for example their activity levels and quantity or quality of sleep, a healthcare professional may be able to track the person's condition and prevent a costly hospital admission.

COPD is the name for a set of lung conditions that are the source of breathing difficulties. It is a relatively common disease for middle-aged or older adults. According to the British Lung Foundation, around 4.5% of adults aged 45 or above are diagnosed with this condition [4]. Often, the person affected does not realize that they have the disease, but their breathing problems get gradually worse. The disease cannot be cured or reversed, but treatment may keep it under control so that it does not impact on daily life. Part of this treatment may be what is known as 'pulmonary rehabilitation', a specialised programme of exercise. [2].

Heart failure is a condition where the heart is unable to pump blood around the body properly, which usually occurs because the heart is too stiff or weak. According to the British Heart Foundation, around half a million people in the UK have been diagnosed with heart failure [3]. There are four classes or stages of heart failure, with a higher class signifying higher severity.

2.2 Six Minute Walk Test

As mentioned above, COPD and heart failure impact a patient's ability to engage in physical activity. These diseases have no cure and generally tend to get worse with time. In many cases, the degree of physical activity that the patient can undergo indicates the severity of the disease [5]. This fact is utilized in the application of the Six Minute Walk Test (6MWT).

The Six Minute Walk Test (6MWT) is a standardized test of functional exercise that involves walking along a flat, straight course for six minutes. The test is self-paced and there are no requirements regarding breaks. The 6MWT is often used to track therapy progress and has been proven to be sen-

sitive to common therapies [5]. Typically the variable that is measured is the distance walked in the six minutes. However, oxygen saturation and heart rate can be measured too if the proper devices are used during the test.

2.3 Smartphones and Wearables

In recent years the penetration of the smartphone has been rapid and widespread. According to the Deloitte Mobile Consumer Survey 2016 [6], 91% of people in the UK aged 18-44 own a smartphone. Additionally, Statista reports that in 2015, 50% of those surveyed aged 55-64 owned a smartphone and 18% aged 65+ owned a smartphone [7]. Clearly, the adoption of the smartphone is reaching saturation among the general population below 45.

Parallel to the increasing adoption of smartphones are the increasing capabilities of such devices. For example, the BLU Energy Diamond M Android Smartphone which is available for £39.99 on AmazonUK has a 1.3 GHz quad-core processor and 512MB RAM [8]. This is comparable to flagship devices from earlier years, for example, the Samsung Galaxy S3 released in May 2012 at a cost of $599 at launch. This device had a 1.4 GHz quad-core processor and 1GB RAM [9]. The cost to performance ratio for such devices has decreased dramatically in recent years.

Another important factor to consider is the availability of sensors in these devices. Even such low-end devices as the BLU Diamond M has a built-in accelerometer and proximity sensor [8]. Higher-end devices will have additional sensors such as a gyroscope, a barometer, or a magnetometer. The availability of sensors allows for a unique opportunity for widespread, remote data collection.

For example, the 6MWT described above can be performed by the patient using a certified application that counts the steps automatically. This is easier and less costly than having the patient attend a testing centre to record these results.

In addition, wearables have also become widespread in the last few years with big players like Apple introducing the Apple Watch in 2015 and Google debuting Android Wear in 2014. Along with these more general-purpose devices are more niche devices like those developed by FitBit or Jawbone that are aimed at fitness enthusiasts. These new devices are packed with sensors and provide another avenue for day-to-day data collection.

3 Objectives

3.1 Step Counter

In order to track 6MWT performance, a generalized step counting algorithm must be available. More modern smartphones often incorporate step counters on device, but older or cheaper devices still lack this functionality. Applications like Google Fit aim to perform a similar task, however, the details of their algorithms are unknown. In addition, there are wearable devices such as a Fitbit or Jawbone that incorporate step counters based on processing the accelerometer data. Again, the underlying principles of such devices are not published and retrieving the data is usually prohibited or impossible. The opaque nature of such devices and software means that they cannot be easily used for medical purposes. Therefore, even though commercial devices are available, there is still a clear need for a validated algorithm for medical use.

In order to reach the maximum number of potential users, a step counter algorithm needs to target smartphone devices as the penetration of these devices is much higher. Also, there are only two platforms to target in order to capture the majority of the market (iOS and Android). The algorithm should only use the accelerometer signal as its input, again to maximize the number of potential users.

As accelerometer signals are often very noisy, the algorithm must be robust to noise and capable of sufficiently high accuracy to enable meaningful analysis of the results. Additionally, the algorithm should be computationally efficient so as to run in realtime on a smartphone device.

The performance of the algorithm developed in this project will be validated against ground truth data collected by a proprietary device, described in Chapter 7. The metric of interest will be the total number of steps over a given time period.

3.2 Sleep Detection

The other part of the project is the design of an accurate sleep detector. Accelerometers represent a very accessible alternative to polysomnography (the method employed in sleep labs) that can provide an approximate estimate of sleep onset and duration. Polysomnography requires a patient to spend a night in a hospital-based sleep lab, typically for the purpose of diagnosing sleep disorders. The patient undergoing polysomnography has their electroencephalogram (EEG), oxygen levels in the blood, heart rate, breathing, eye movements, and leg movements recorded. The data is later analysed by a trained professional, who produces a report quantifying the transitions between sleep stages (deep sleep, Rapid Eye Movement (REM), awake) during the night.

The feasibility of accelerometer-driven sleep detection algorithms has been demonstrated with

consumer products such as those sold by Fitbit. The algorithms and methodology in these commercial trackers are unknown and the raw data is made inaccessible by the manufacturers. To overcome these limitations, a sleep detection algorithm will be developed that operates on the data acquired with a wrist-mounted accelerometer. A wearable device was chosen for this aspect of the project as it can easily be attached to the user prior to sleep, unlike a smartphone which cannot be guaranteed to be in the bedroom during the night.

The sleep detection algorithm will need to identify the sleep onset time, the time at which the user falls asleep, and the wakefulness onset time, the time at which the user wakes up, using the accelerometer signal provided by a wrist-mounted device.

In order to develop and validate the algorithm, a database produced by the Embedded Sensing Systems Group at TU Darmstadt will be used [10]. This database contains both the accelerometer traces and annotated polysomnography for 42 sleep lab patients.

4 Literature Review

4.1 Step Counting

4.1.1 Mechanics of Walking

The act of walking described in Naqvi, et. al. [11] is as follows:

1. At the beginning of the step, the planted foot is pushed backwards into the floor.
2. Static friction opposes this force. This provides the driving force forward.
3. At the end of the step the stepping foot is placed on the floor and pushes forward into the floor.
4. Again, static friction opposes this force, giving rise to a backwards force.

An example of raw accelerometer data recorded during walking is shown below in Figure 4.1. The act of walking gives rise to periodic activity with a period corresponding to a single step.

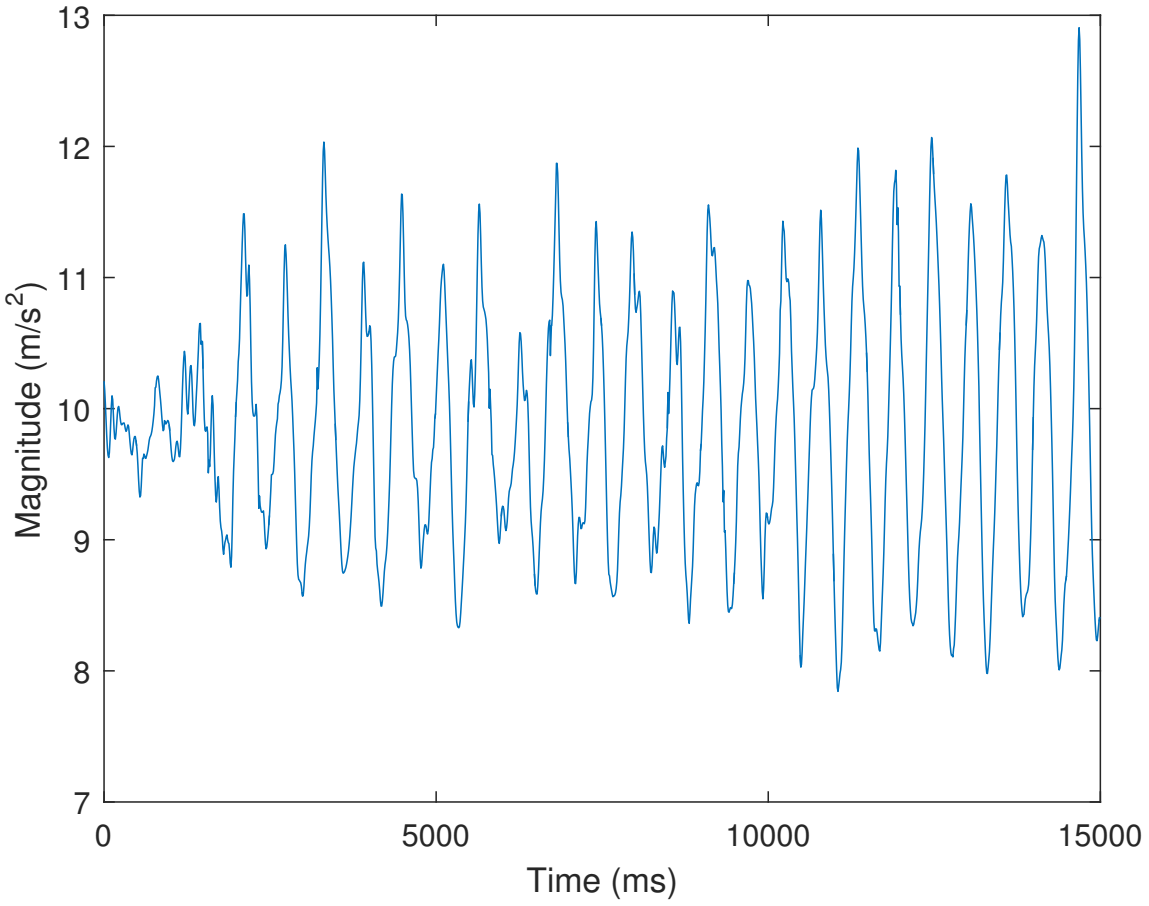

Figure 4.1: Example of an accelerometer signal recorded during a period of walking.

4.1.2 Previous Attempts

There have been a number of attempts to develop step counting algorithms with only an accelerometer signal.

In Navqi, et. al. [11] a relationship between walking speed and the magnitude of the accelerometer signal is derived and used in devising an algorithm that is based on thresholding the magnitude of the signal. The algorithm is then tested on 5 samples ranging from 16 to 44 steps and achieves an average accuracy of 96.6% where the accuracy is defined as:

$$a = 100(1 - \frac{|s_a - s_{gt}|}{s_{gt}}), \tag{4.1}$$

where where a is the accuracy expressed as a percentage, s_a is the number of steps extracted by the algorithm and s_{gt} is the number of steps from the ground-truth data. The collection protocol is not noted besides the fact that the signal is collected by a device fixed at the centre of mass of the subject.

In Palshikar [12], the author describes a methodology of peak detection in time-series. The author describes the general flow of the method in which each point in the series is scored on how peaky it is. A search for local peaks using the standard deviation and mean of the signal is then performed. This methodology forms the basis for part of the step counting algorithm described later.

In Brajdic and Harde [13], the authors test a multitude of types of algorithms for both walk detection and step counting. The focus of this project is step counting, so walk detection algorithms will be ignored. The authors note that many algorithms involve thresholding a property of the signal, especially when the sensor is placed on the foot. However, it is mentioned that choosing an optimal threshold is difficult for different users, surfaces, or shoes. Other algorithms use peak detection or zero crossings to count steps. In the frequency domain, some algorithms look for high magnitudes of frequencies related to walking speeds to search for periods of walking and hence calculate fractional steps.

Fractional steps are defined as the dominant perceived walking frequency multiplied by the length of this period. Frequency estimation is carried out via the Short Term Fourier Transform or Wavelet Transforms. An equivalent option to these transforms is to use the autocorrelation function in the time domain. Large magnitudes indicate high periodic activity at the lags for which they occur. All of these algorithms suffer from the problem that they will be triggered by any movement with a similar periodicity to walking. A more complex algorithm involves non-linear template matching with dynamic time warping. A generic template can be generated offline and then correlated in real time with the

signal. Brajdic and Harde [13] also explored machine learning based techniques, with commonly used features such as mean, variance, energy, entropy, correlation between axes, and Fast Fourier Transform (FFT) coefficients. They note that previous attempts used classifiers such as neural networks, Gaussian mixture models, or support vector machines. From this list of techniques, Brajdic and Harde [13] chose 9 to evaluate. These 9 were: windowed peak detection, mean crossing counts, normalized autocorrelation, dynamic time warping, short term fourier transform, continuous wavelet transform, discrete wavelet transform, hidden markov model, and k-means clustering.

In order to evaluate all of these algorithms, the authors collected 130 data recordings from 27 subjects with a Galaxy Nexus GT-I9250 across a variety of scenarios: in hand, in a pocket, in a back pocket, or in a handbag. A video recording was taken of each session for later validation.

With the data recordings and these algorithms, the authors concluded that the windowed peak detection and the continuous wavelet transform methods worked the best, with a median error of 1.3% for each. Note that this was achieved with 80 of the 130 data recordings that had 'reliable ground truth step counts'. The authors also note that the placement of the phone had little effect on the step counting performance.

4.2 Sleep Detection

The literature on sleep detection with a wrist-mounted accelerometer is much sparser as wearable devices are a relatively new invention.

In Borazio et al. [14] the authors built a custom device to be worn on the wrist with an accelerometer that samples at 100Hz. They collected a dataset containing 409 hours of sleep lab data, with each data recording comprising a night's sleep from one patient. The patient underwent polysomnography to provide ground truth data. The authors proposed a simple algorithm that thresholds the acceleration signal based on the standard deviation of the magnitude. If the last 100 points (1 second of recording) exceeds a threshold, then the user is marked as not sleeping. The authors compared their method against the well-known Oakley [15] and Cole [16] algorithms that use activity counts to determine wakefulness, as well as against the ground-truth polysomnography.

The results show that the novel approach marginally outperforms the Oakley and Cole algorithms (79.83% vs 74.94% and 73.75% respectively). The authors note that not surprisingly all the algorithms fail when the patient is awake but not moving. The dataset collected has been made publicly available and will be used to test algorithms developed in this project.

In van Hees et al. [17], the authors attempt to classify sleep using the arm-angle of the patient. The authors state that sleep is characterized by a period of low frequency of changes in arm-angle. The methodology consists of calculating the arm angle using rolling medians of the x, y, and z com-

ponents of acceleration and then assessing changes in arm angle between 5-second epochs. If there was no change larger than 5° in 5 minutes, then these were classified as sustained inactivity. The accelerometer data was collected along with a self-reported sleep log of onset and offset. The authors report that overall there was moderate agreement between the results of the analysis of accelerometer data and self-reported sleep duration.

Part I

Step Counter Algorithm

5 Overview

This section covers the development and evaluation of the step counter algorithm. The algorithm aims to extract the occurrences of steps from the raw accelerometer data recorded on a smartphone. Effectively, the problem is one of peak detection in a noisy signal.

The algorithm developed for this project was adapted from the windowed peak detection algorithm described by Palshakir [12].

The algorithm is split into five stages, each responsible for a particular function. The data flows from left to right in Figure 5.1 below. All stages have an input data stream and an output data stream, with the exception of the final stage. Each of the five stages will be described in detail in the following sections. Note that throughout the description there will be options or parameters for each stage; this is intentional, as this allows optimization routines to be run on the algorithm to determine the best set of parameters for overall performance.

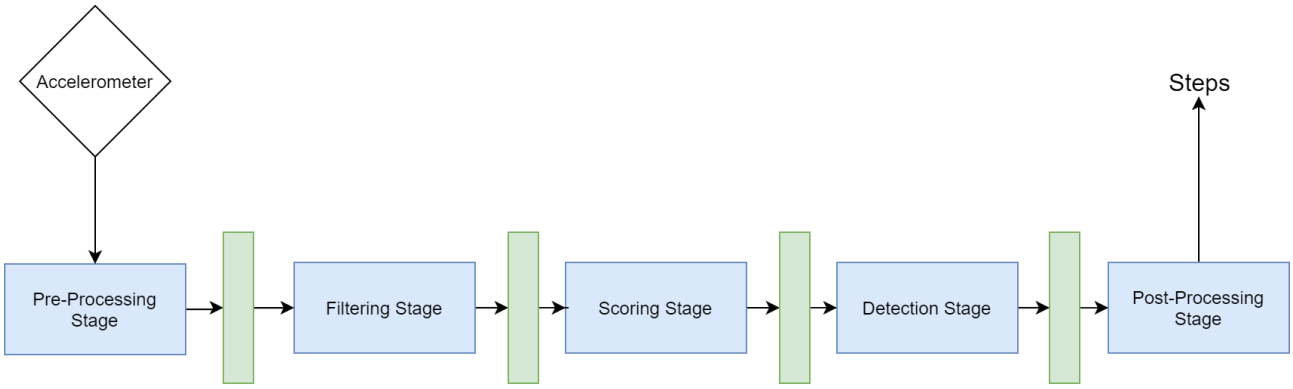

Figure 5.1: Block diagram of the step counter algorithm.

6 Algorithm Description

6.1 Pre-Processing Stage

The Pre-Processing Stage is responsible for two functions:

1. Formatting the data received from the accelerometer into a usable format.

2. Ensuring a constant sampling frequency by means of linear interpolation.

The project uses a tri-axial accelerometer, however the step-counting algorithm is concerned with the magnitude rather than any single directional component because the physical orientation of the device is unknown. The time stamps of the samples should also be adjusted appropriately so that the first sample received from the user initiating the algorithm is at $t = 0$. The time stamps are provided in nanoseconds and are not given in standard UTC format, but as the time since system boot. The equations for these operations are simple and are as follows:

$$m = \sqrt{a_x^2 + a_y^2 + a_z^2}, \tag{6.1}$$

$$t_{i,adjusted} = \frac{t_i - t_0}{t_s}, \tag{6.2}$$

where m is the magnitude of the acceleration signal, a_x is the acceleration in the x direction, a_y is the acceleration in the y direction, a_z is the acceleration in the z direction, $t_{i,adjusted}$ is the adjusted time stamp for the i-th sample, t_i is the time stamp for the i-th sample, t_0 is the time stamp of the initial sample, and t_s is the time-scaling factor. For example, converting from nanoseconds to milliseconds, $t_s = 10^6$.

Each sample from the accelerometer is then inserted into a simple data structure and is appended to an internal buffer of size 2. When this buffer is full, the two points in the buffer are linearly interpolated. Although the developer can specify a desired sample rate for the accelerometer, there is no guarantee that the accelerometer will be sampled at this rate, hence interpolation is needed. An example of this is shown below in Figure 6.1 where time between samples is plotted against samples. For the subsequent filtering stage, we need to ensure that the data is sampled at a constant rate.

The desired sample rate determines the values of the linear interpolation equation below:

$$value = \frac{y_1 - y_0}{t_1 - t_0} t_{int} + y_0, \tag{6.3}$$

where the two points of the original data stream are given by: (t_1, y_1) and (t_0, y_0), and the time

between each equally sampled value is given by t_{int}.

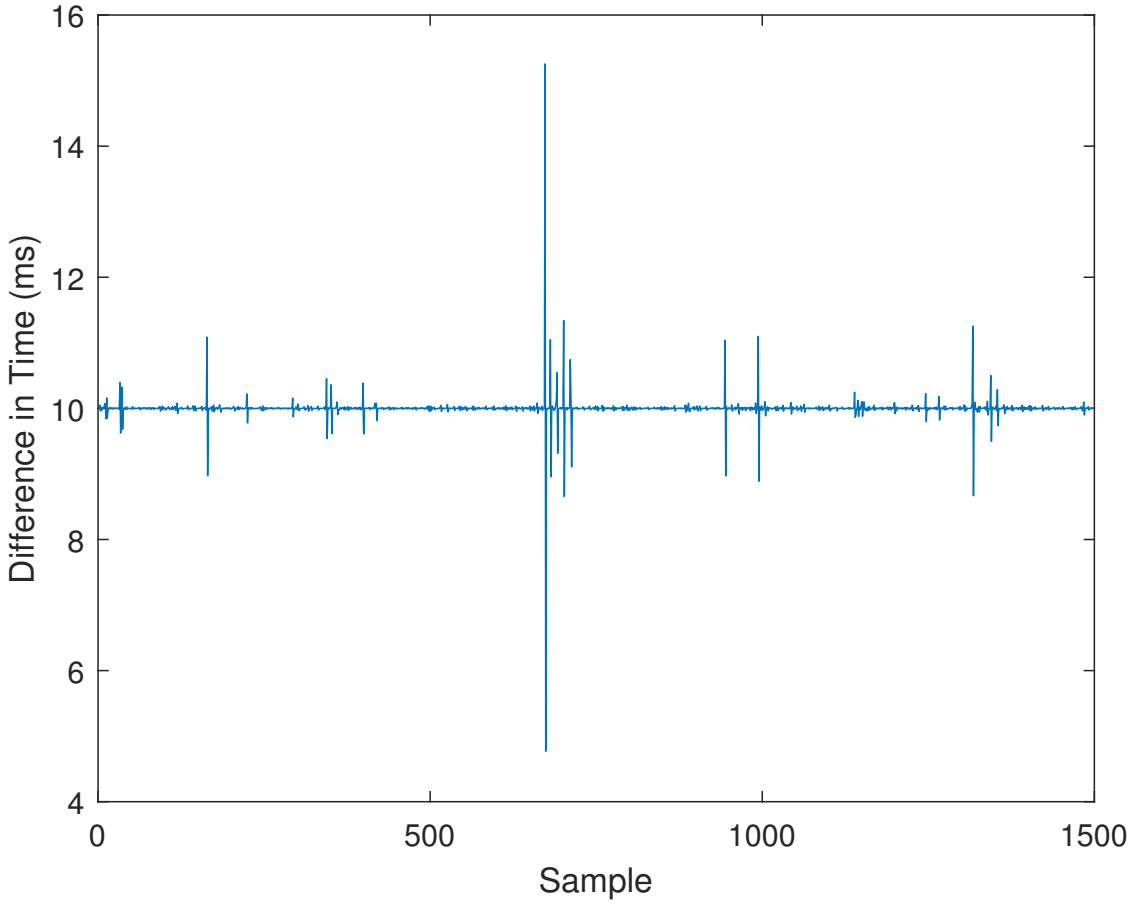

Figure 6.1: Time differences between samples for a 100Hz sampled signal. Note that the non-constant sampling times leads to the requirement for interpolation.

6.2 Filtering Stage

The accelerometer attempts to measure the acceleration forces acting upon the sensor, and in particular the motion of the sensor. However, this sensor is subject to noise from a variety of sources (mechanical, electrical, thermal, etc.). The function of the filtering stage is to smooth the signal by removing as much of the high-frequency noise from the accelerometer time series as possible.

The filter required is a simple finite impulse response (FIR), low-pass digital filter. In order to capture a variety of walking speeds, the cutoff frequency for this filter will be around 3 Hz. This should be sufficient to capture the walking of even the speediest walkers, as this would translate to an pace of $5.4mph$ according to the ratio of $2000steps = 1mile$ as given by the American College of Sports Medicine [18]. Note that the average walking pace for pedestrians aged 14 to 64 was found to be $3.1mph$ in a study by Knoblauch, et. al. [19].

An example of the raw accelerometer signal (after interpolation) and the filtered signal is shown below in Figure 6.2.

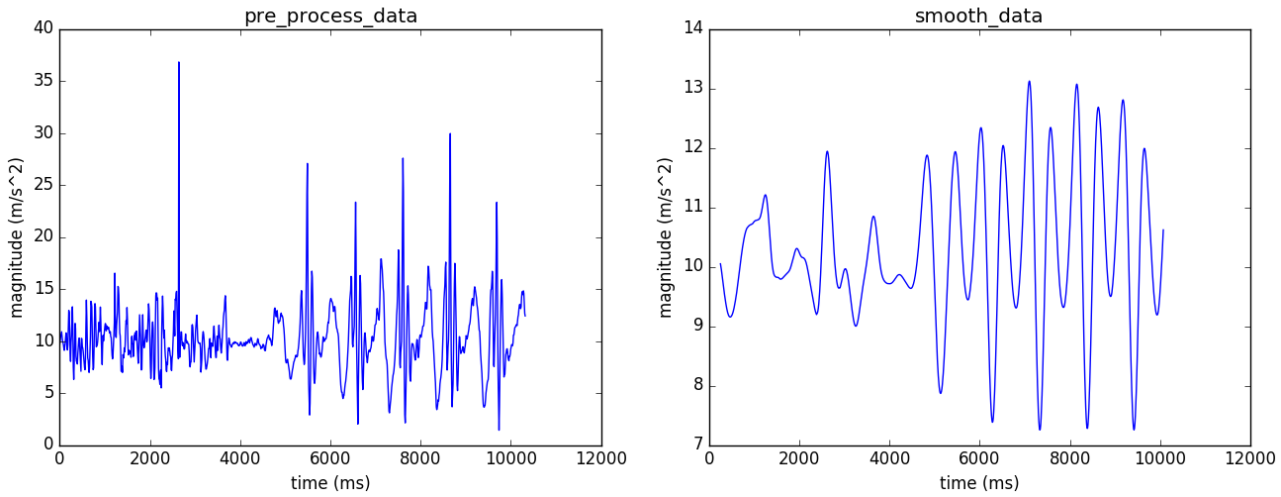

Figure 6.2: Example of a signal being filtered. Filter used: Gaussian Filter with $N = 51$ and $\sigma = 0.35$. Note the large peaks due to noise are filtered out entirely around t = 2300ms, 5500ms, 6500ms, 7750ms, and 9750ms.

A number of filters were implemented for performance testing.

6.2.1 Moving Average

This is a very simple filter, each point in the filter window being weighted equally such that:

$$m_k = \frac{1}{N}, \tag{6.4}$$

where m_k is the k^{th} filter coefficient, and N is the length of the filter. The frequency response of this filter is shown below in Figure 6.3.

6.2.2 Gaussian Filter

This is a more complex filter, using a Gaussian as the filter coefficients. This filter attempts to suppress the side lobes in the frequency domain of the moving-average filter by having a smoother transition to the stop-band. The weights of the filter are given by:

$$g_k = \exp(-\frac{1}{2}(\frac{k-\nu}{\sigma\nu})^2), \tag{6.5}$$

where $\nu = \frac{N-1}{2}$, g_k is the k^{th} coefficient of the filter, N is the length of the filter window, and σ is a parameter defining the standard deviation of the Gaussian. Note that the standard deviation also scales with the length of the filter. The frequency response of this filter is shown below in Figure 6.3.

6.2.3 Hann Filter

This is a similar filter the Gaussian filter, which also attempts to smooth out the response by suppressing the side lobes of the moving-average filter. The weights of the filter are derived from the Hann window [CIT] and are as follows:

$$h_k = \frac{1}{2}(1 - \cos(\frac{2\pi k}{N-1})) \tag{6.6}$$

where h_k is the k^{th} filter coefficient and N is the length of the filter window. The frequency response of this filter is shown below in Figure 6.3.

6.2.4 Kaiser-Bessel Filter

This is the most complex filter design, which combines the ideal filter response (note a sinc function in the time domain) and a Bessel window to achieve the desired response. The calculation of the coefficients is as follows [20]:

First, calculate the window shape parameter α.

$$\alpha = \begin{cases} 0.1102(A - 8.7) & A \geq 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & 21 \leq A \geq 50 \\ 0 & A \leq 21 \end{cases}, \tag{6.7}$$

where A is the desired attenuation at the cutoff frequency.

Then calculate the coefficients of the Kaiser-Bessel window:

$$w_k = \frac{I_0(\alpha\sqrt{1 - (\frac{k-N_p}{N_p})^2})}{I_0(\alpha)}, \tag{6.8}$$

where w_k is the k^{th} coefficient of the window, α is the window shape parameter, N_p is the midpoint of the filter, $N_p = \frac{N-1}{2}$ where N is the length of the filter, and I_0 is the 0^{th} order Bessel function of the first kind.

Then calculate the coefficients of the ideal filter response:

$$i_k = \frac{\sin(2\pi k \frac{F_c}{F_s})}{\pi k}, \tag{6.9}$$

where i_k is the k^{th} coefficient of the ideal filter response, F_c is the desired cutoff frequency and F_s is the sampling frequency.

Finally, compute the coefficients of the filter with:

$$b_k = w_k i_k, \tag{6.10}$$

where b_k is the k^{th} coefficient of the Kaiser-Bessel filter response. The frequency response of this filter is shown below in Figure 6.3.

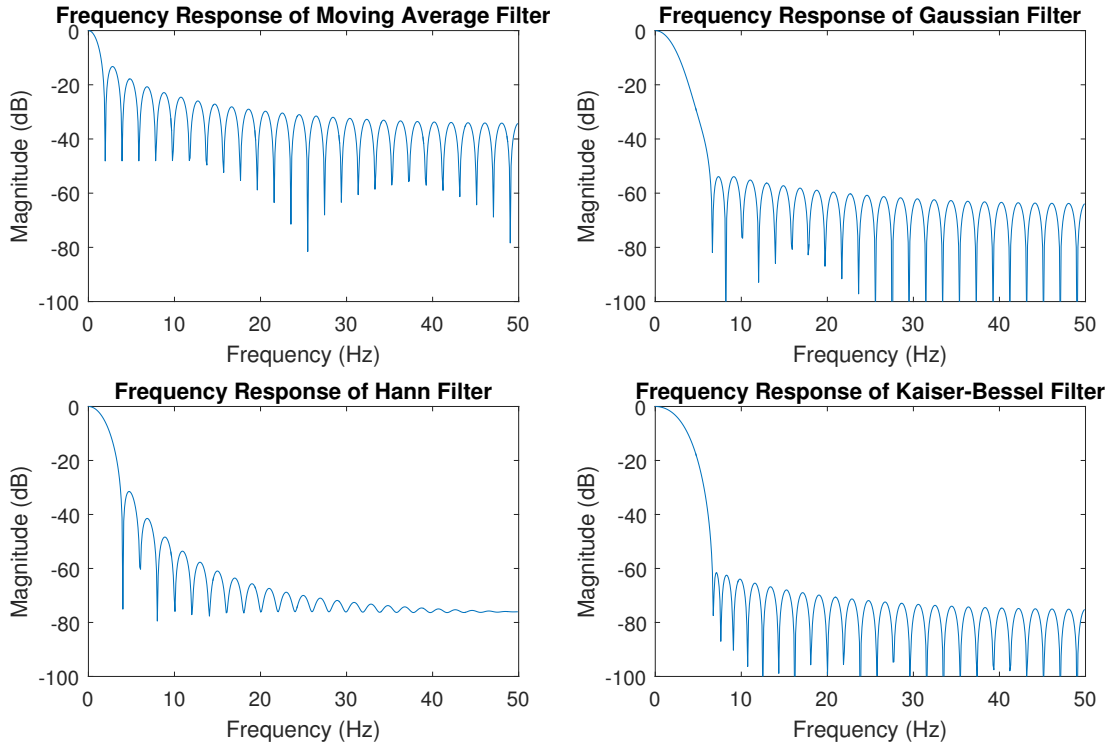

Figure 6.3: Frequency response of the four filters compared. Starting at top-left, working clockwise: Moving Average filter with $N = 51$, Gaussian filter with $N = 51$ and $\sigma = 0.35$, Kaiser-Bessel filter with $N = 51$, $A = 60dB$, $F_c = 3Hz$, and $F_s = 100Hz$, and Hann filter with $N = 51$

6.3 Scoring Stage

The function of the scoring stage is to evaluate the 'peakiness' of given point. The result of this stage should increase the magnitude of any peak, making them more obvious to the subsequent peak detector.

A few methods were evaluated. These are detailed below.

6.3.1 Maximum Difference

This method uses the local neighbours of the point in question to determine how peaky the point is. It considers the N points to the left and determines the maximum difference between the point in question and those points. It does the same for the N points to the right, and then averages the two maximum differences as the result.

This operation results in enhanced peaks. An example of the output from the scoring stage using Maximum Difference is shown below in Figure 6.4.

The equation describing this behaviour is given by:

$$x = \frac{\max\limits_{k}(x_i - x_{i-k}) + \max\limits_{k}(x_i - x_{i+k})}{2}, \tag{6.11}$$

where i is the point under consideration, and x_n is the value of the signal at the n^{th} sample.

6.3.2 Mean Difference

This method is similar to the Maximum Difference method, except that instead of taking the maximum of the difference to the left and the right of the point in question, the Mean Difference method takes the mean of all the differences. The effect is similar to the Maximum Difference method, but smaller in magnitude. It also preserves the overall shape of the waveform.

An example of the output from the scoring stage using Mean Difference is shown below in Figure 6.4.

The equation describing this behaviour is given by:

$$x = \frac{\sum_{k=-N,k\neq i}^{N}(x_i - x_{i+k})}{2N}, \tag{6.12}$$

where i is the point under consideration, x_n is the value of the signal at the n^{th} sample, and N is the characteristic length of the Mean Difference operation.

6.3.3 Modified Pan-Tompkins Scoring

This method is a derivative from the well-known algorithm by Pan and Tompkins [21] that is used for peak detection in electrocardiogram (ECG) waveforms.

The original algorithm had four main steps: digital band-pass filter, signal differentiation, squaring of the signal, then a moving integration window to reconstitute the signal.

From this baseline, a modified algorithm was developed:

- Locally zero-mean the data within a window of size N.
- Set all the data points that are less than 0 to zero.
- Square the data to amplify any large peaks.

The second step ensures that when the data is squared, only peaks get amplified. Without it, the algorithm would detect both peaks and troughs.

An example of the output from the scoring stage using the Modified Pan-Tompkins method is shown below in Figure 6.4.

6.4 Detection Stage

The next stage in the process is to identify potential candidates for peaks associated with steps. This is done using statistics to identify outliers.

As the algorithm processes the signal, it keeps track of a running mean and standard deviation. A computationally efficient way of doing this incrementally is as shown below:

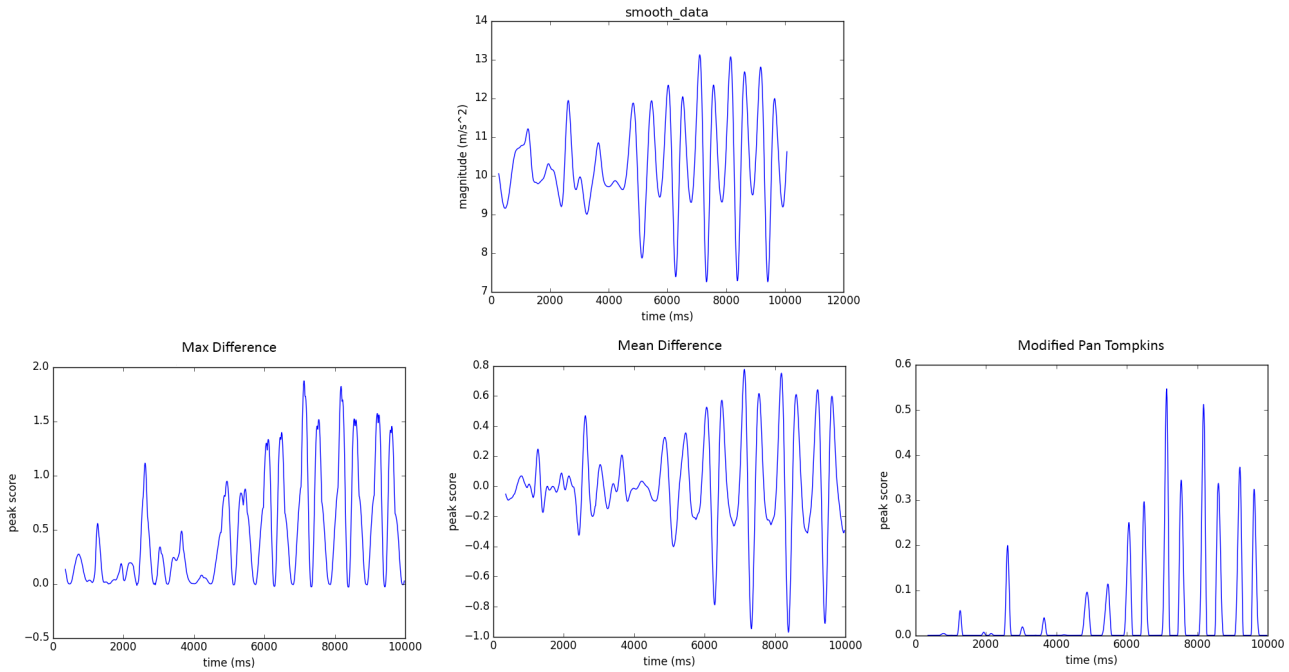

Figure 6.4: Comparison of the three scoring methods on the same filtered signal. From left to right: Maximum Difference with $N = 6$, Mean Difference with $N = 10$, and Modified Pan-Tompkins with $N = 10$.

$$\bar{x}_n = \frac{x_n + (n-1)\bar{x}_{n-1}}{n}, \tag{6.13}$$

$$\sigma_n^2 = \frac{(n-2)\sigma_{n-1}^2 + (n-1)(\bar{x}_{n-1} - \bar{x}_n)^2 + (x - \bar{x}_n)^2}{n-1}, \tag{6.14}$$

where x_n is the n^{th} sample, $\bar{x}_n$ is the running mean at the n^{th} sample, and σ_n is the running standard deviation at the n^{th} sample.

The algorithm then uses these two quantities to determine whether any given point is an outlier by thresholding. If a point is more than c current standard deviations above the current mean $\bar{x}_n$, then it is marked as a potential peak associated with a step, i.e. - if it satisfies the equation below:

$$\frac{x_n - \bar{x}_n}{\sigma_n} \geq c, \tag{6.15}$$

where c is the designated threshold.

An example of the results obtained using this method is shown below in Figure 6.5.

Note that the algorithm builds up a running mean and standard deviation after initialisation. This means that the first peak it encounters will be marked as a potential step.

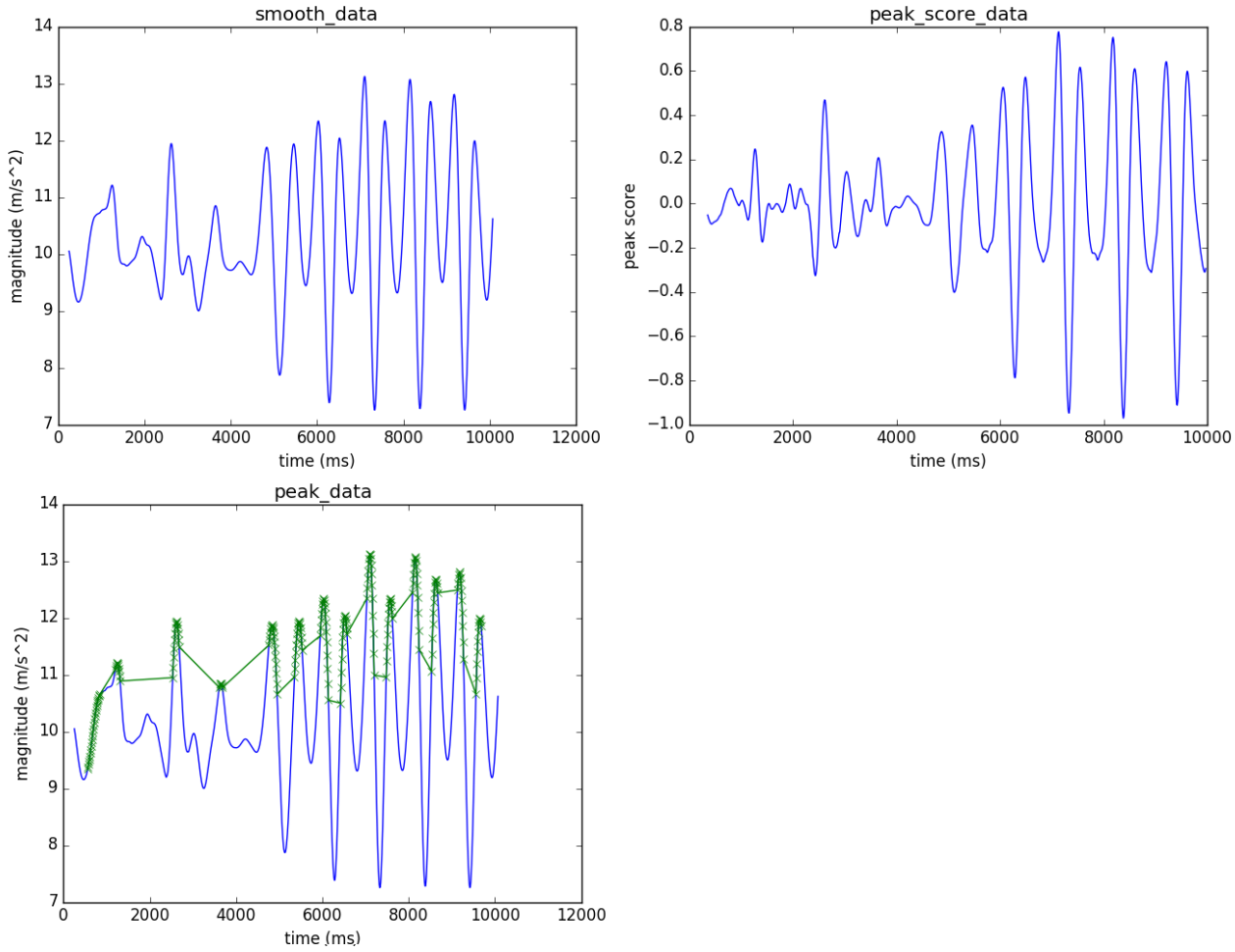

Figure 6.5: The results of the Peak Detection stage. Top Left: filtered signal, Top-Right: scored signal, Bottom-Left: filtered signal with potential peaks overlaid in green. $c = 1.2$ for the Peak Detection stage.

6.5 Post-Processing Stage

The final stage of the algorithm is to identify the true peaks from the potential peaks from the previous stage. In Figure 6.5, all of the potential peak points are clustered on the rise to the main peak, This stage removes the erroneous points and leaves only the local maximum or the true peak. This stage also makes use of the fact that the peaks associated with walking are usually periodic with a minimum delay between them. Note this delay is related to the speed of walking.

This stage slides a window of a fixed size, t_{window}, across the potential peaks and only keeps the maximum point within the window. This effectively removes all the points leading up to the peak while preserving the peak. By tuning the window size, the algorithm can ensure that for most cases no two steps will reside in this window at the same time. The window length was set to $200ms$ which corresponds to a maximum walking speed of $5steps/sec$, much faster than average walking pace.

The pseudocode for this stage is as follows:

```
int windowSize = 200; // In ms
Point currentMax;
// Iterate through the sorted list of points
for (Point candidate in points) {
    if (candidate.time - currentMax.time > windowSize) {
        // The currentMax point is now outside the window.
        yield currentMax;
        currentMax = candidate;
    } else {
        // Keep the maximum point.
        currentMax = (currentMax.value > candidate.value) ? currentMax : candidate;
    }
}
```

The result of this stage retains the main peak points while throwing away the points surrounding the peak, as intended. An example of this can be seen in 6.6.

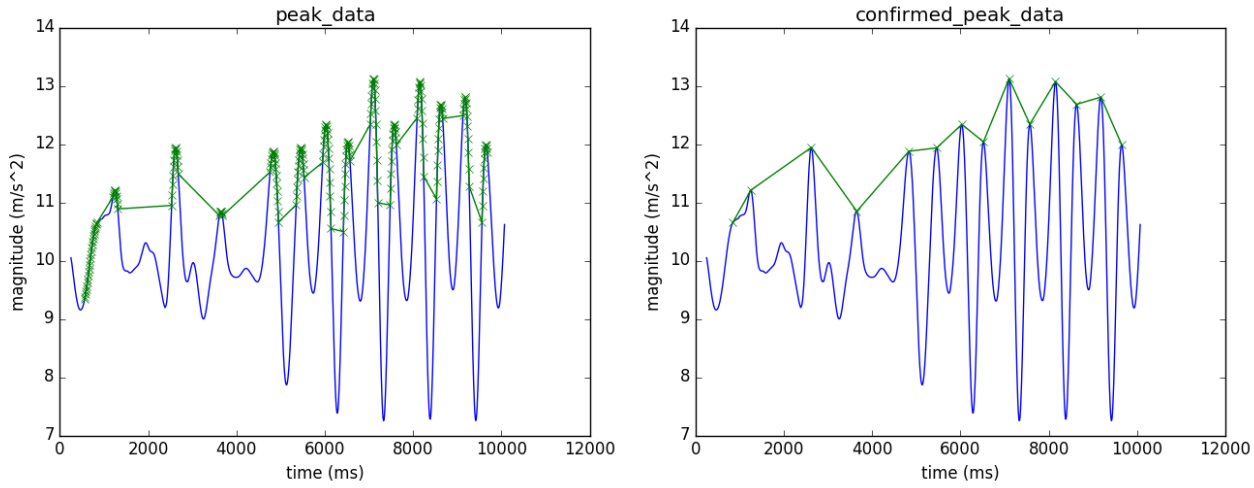

Figure 6.6: The results of the Post-Processing Stage. Output of the Detection Stage is on the left, output of the Post-Processing Stage is on the right. Note how the set of points surrounding the peaks have been filtered out.

7 Data Collection Equipment

A dataset with ground truth data must be acquired in order to optimize and validate the step-counting algorithm. Previous studies, such as [13], used video annotation to determine the time stamps of steps, but this is time intensive and prone to error. Instead, a custom device was designed and built to derive this information.

The main goal of the algorithm is the accurate counting of steps over a period of time. Hence, the ground truth device only needs to provide this metric. However, in order to open up the versatility of the dataset and enable future improvements of the algorithm, a dataset was collected that would also contain the information for identifying the steps using reference data.

7.1 Design

The ground-truth device is designed to record steps with a sensor attached to the shoe. The sensors are connected to an RFduino, which is a Bluetooth-enabled Arduino microcontroller board. The RFduino polls the sensor for information and broadcasts it over Bluetooth Low Energy to an Android smartphone, as discussed later.

A first iteration of the device used push buttons mounted on straps as shown in Figure 7.2. These push buttons are connected to the RFduino with a simple pull-down resistor, such that logic level 0 indicates that the push button is up and the user's foot is up and logic level 1 indicates that the push button is down and the user's foot is down. A picture of the push button mounted on the straps can be seen below in Figure 7.1.

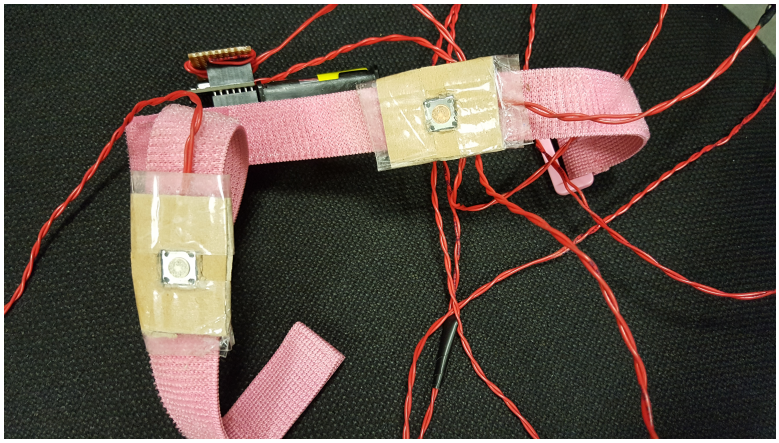

Figure 7.1: A picture of the original ground truth device design. Note the push button in the middle of the pink strap. The contact area was reinforced with cardboard in an attempt to increase robustness.

Unfortunately, this first iteration of the device suffered from low robustness. The device was accurate when functional, but the push buttons broke due to excessive force. The pins of the push buttons also broke a number of times as they were folded flat against the strap due to the weight of the user.

Figure 7.2: A picture of the original ground truth device in use. Notice the placement of the straps on the user's shoes such that the push button is positioned roughly under the ball of the foot.

The second iteration of the design abandoned the push buttons entirely and relied instead on having a pressure-sensitive resistor positioned under the heel of the user inside the shoe. To ensure that this resistor was tolerant and sensitive to the relatively high pressures in this use case, the device used a material called velostat as the variable resistor. Velostat, used primarily as a packaging material, is "made of polymeric foil impregnated with carbon black to make it electrically conductive" [22]. The resistance is reduced when pressure is applied, hence a high current flow corresponds to the foot being down on the ground. The new device was made by taping conductive thread to either side of the velostat, completing the circuit. A picture of the new device can be seen in Figure 7.3.

The firmware on the RFduino uses a moving threshold to determine the transition between the two states. The threshold for each sensor is updated by retaining the minimum and maximum values. Every 5 seconds, if the difference between these two exceeds a set minimum difference, the minimum and maximum are penalised by a factor proportional to the difference and the threshold is set to the midpoint between the minimum and maximum. This penalisation technique ensures that the minimum

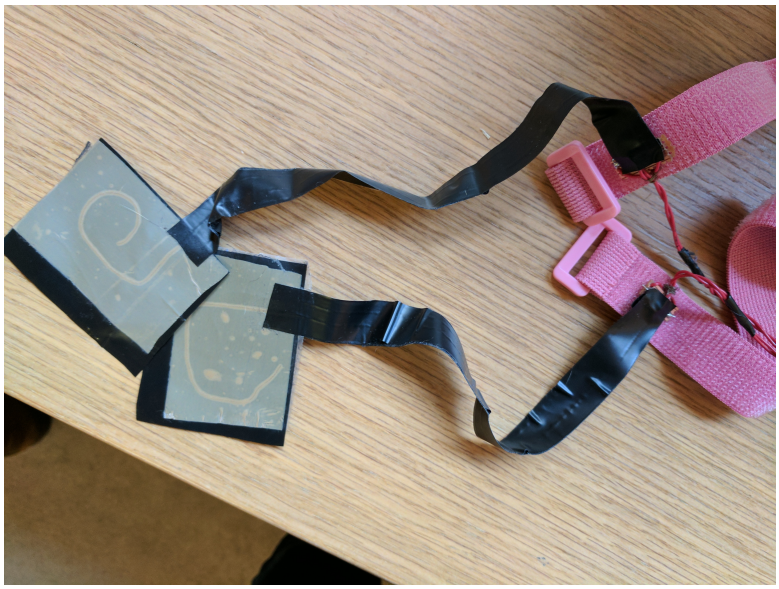

Figure 7.3: The new ground truth device. Note the conductive thread visible through the tape and the electrical tape used to keep the two threads electrical isolated. The straps are now wrapped around the user's ankles.

and maximum are local, so that minimum and maximum values that were set long ago are penalised until a local value becomes relevant. A graph representing the raw data from the sensors plotted together with the thresholds against time is shown below in Figure 7.4.

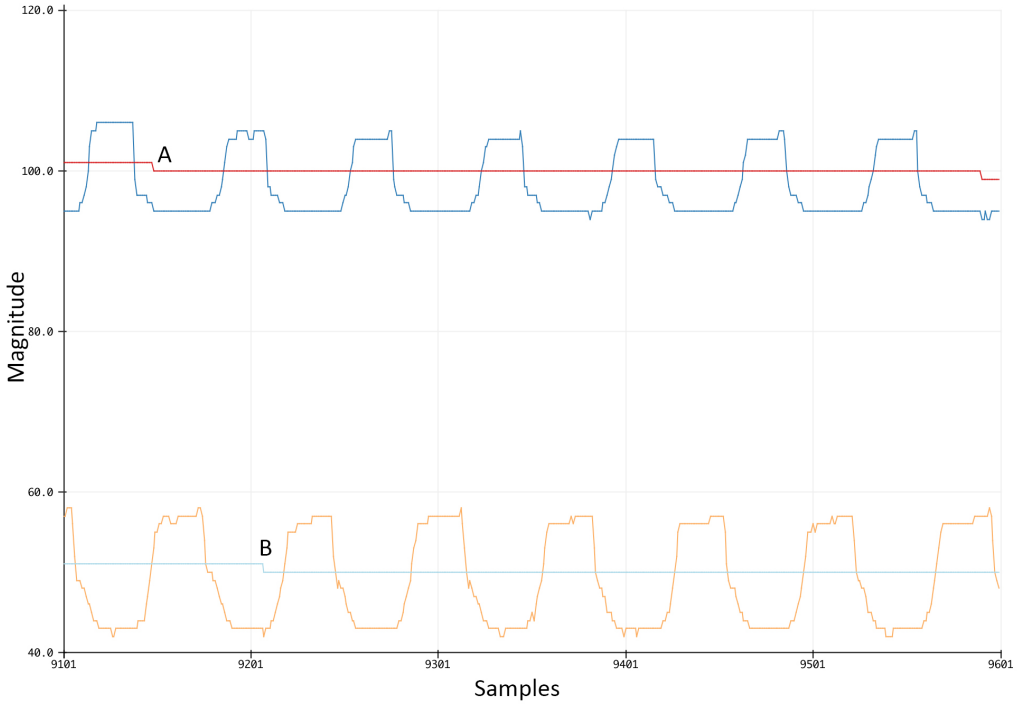

Figure 7.4: The raw data captured by the new ground truth device. Note how the steps are clearly discernible and the moving threshold correctly differentiates between the two states. The difference in magnitude between the left-foot and right-foot signal is explained by the natural variation in velostat pads. Note at labeled points A and B the threshold is being recalculated.

7.2 Android Application

To capture the accelerometer data and the ground truth simultaneously, an Android application had to be written to accommodate this functionality. The specifications were:

- Capture accelerometer data at a high sample rate (at least 100Hz) and save to a file.
- Connect via Bluetooth Low Energy to the ground-truth device, decode the signal, and save the data to a file with synchronized timestamps with respect to the accelerometer data.
- Provide a simple method of retrieving this data off the device.
- Provide a simple UI for the user to capture all of this data efficiently.

The application was designed such that the data would be zipped and compressed together and uploaded to a remote server via HTTP POST. This enabled the user to record multiple data recordings after one another without needing to manually retrieve the files off the device, thus expediting the data collection phase.

7.3 Data Collection

A dataset was collected that aimed to capture as many scenarios as possible. The dataset featured three participants with different heights and weights, 3 smartphones (a Google Pixel, a Samsung Galaxy S6, and a Google Nexus 5 device) to ensure that similar devices performed similarly, and 6 phone 'positions': in a hand, in a front pocket, in a back pocket, in an armband, in a shoulder purse, and in a neck pouch on a lanyard. In order to ensure generality, recordings were made on two different surfaces: hard floor (stone or linoleum) and carpeted floors. This resulted in a dataset containing 48 distinct recordings, each with approximately 2.5 minutes of accelerometer and ground-truth data.

In addition, 3 recordings were collected from consenting patients participating in a clinical pilot study without the ground truth device data for post-optimization validation. The total steps, in these 3 instances, were hand counted by a trained healthcare professional present during the recording.

8 Step-Counting Algorithm Optimization

Optimization of the algorithm can be described as a maximization problem for which the problem domain is defined by the parameter space. The only way of getting close to the true maximum is to run an exhaustive grid search across the parameter space. Each stage may have parameters associated with it, and these parameters may change depending on the stage selection (choice of filter, scoring technique).

For the running time of the optimization routine to be kept within reasonable limits, the parameter search will need to be fairly coarse to limit the number of parameter permutations. The final set can be seen in Tables 8.1, 8.2, 8.3. Note that the first stage, Pre-Processing, has been held constant at $t_{scale} = 10^6$ and $f_{int} = 100Hz$. The Post-Processing stage has also been held constant at $t_{window} = 200ms$.

Table 8.1: The parameter set for the Filtering Stage.

Parameter	Minimum	Step	Maximum
Moving Average Filter			
Window Size (N)	13	8	53
Hann Filter			
Window Size (N)	13	8	53
Gaussian Filter			
Window Size (N)	13	8	53
Standard Deviation (σ)	0.35	Fixed	0.35
Kaiser-Bessel Filter			
Window Size (N)	13	8	53
Sampling Frequency (f_s)	100Hz	Fixed	100Hz
Cutoff Frequency (f_c)	3Hz	Fixed	3Hz
Gain At f_c (A)	-60dB	Fixed	-60dB

Table 8.2: The parameter set for the Scoring Stage.

Parameter	Minimum	Step	Maximum
Maximum Difference			
Window Size (N)	3	8	51
Mean Difference			
Window Size (N)	3	8	51
Modified Pan Tompkins			
Window Size (N)	11	8	51
No Scoring Method			
No Parameters			

Table 8.3: The parameter set for the Detection Stage.

Parameter	Minimum	Step	Maximum
Standard Deviation Threshold (c)	1.2	0.2	1.4

These parameter variations give a total number of permutations equal to 1008. Each of these parameter sets was tested on all 48 data recordings listed in the previous section. The results were averaged across all data recordings for each permutation to give an overall value for the accuracy.

However, since there are approximately 2 hours of data to run through the algorithm, each combination of parameters takes 5 minutes to be tested on a machine with an Intel i5-6200u. If the optimization were run solely on that machine, it would take 3.5 days to complete.

The solution to this problem was to build a distributed computing platform such that the computing load could be spread across multiple machines. The backend server was built on Flask [23], a web microframework for Python, and served over gunicorn [24], a Python WSGI (Web Server Gateway Interface) HTTP webserver. The data were transmitted using JSON (JavaScript Object Notation). The backend also stored the results in a PostgreSQL (Structured Query Language) database [25] such that the results could be queried easily.

Each machine had a local implementation of the algorithm and a copy of the dataset. The machine queried the backend for a new set of parameters, and fetched the JSON of these with an HTTP GET request. Once the computation was completed, the machine appended the results to the JSON file and returned it via an HTTP POST request to the backend. A new set of parameters was then requested, until all permutations had been exhausted.

In this instance, Google Cloud Compute [26] was used to instantiate 8 machines to run the algorithm, such that the runtime for the exhaustive search was cut down to around 11 hours.

9 Results

Note that in this chapter, all of the statistics given are in terms of accuracy defined in the Literature Review (Section 4.1).

9.1 Variability in Surface and Phone

Our first investigation is concerned with the differences between floor surfaces and phone devices.

9.1.1 Variability Surface

There are 4 recordings from the Google Pixel on each of the surfaces listed: hard (stone or linoleum) and carpet. All of these recordings were carried out with the phone in hand. To check the variation between these surfaces, the algorithm parameters are held constant and the average score is checked for each surface. The results of these tests can be seen below in Table 9.1. Note that the parameters used were those of the optimal set, as detailed in the next section.

Table 9.1: The results for each of the recordings according to surface.

Recording	Ground Truth (steps)	Algorithm Output (steps)	Accuracy
Hard Floor			
1	292	298	97.9%
2	267	271	98.1%
3	323	290	89.7%
4	322	307	95.3%
Average	-	-	95.3%
Carpet			
1	243	245	99.1%
2	255	263	96.9%
3	206	246	80.6%
4	195	257	68.2%
Average	-	-	86.2%

While there is a noticeable difference between the accuracies obtained on hard floor versus carpet, the high variance in the carpet results means that it is not possible to establish whether there is a meaningful difference between the types of surfaces or whether Carpet Recording 4, which lowered the average significantly was an isolated instance. Further tests would be needed to confirm this hypothesis.

9.1.2 Phone

The difference between phones is expected to be negligible and this proved to be the case. Data recordings were taken with each of the three phones: Google Pixel, Google Nexus 5, and Samsung

S6. The phone was held in hand and the surface was a hard floor. As with the analysis for the different surfaces, the algorithm parameters were those of the optimal set as described below. The table of results is shown below in Table 9.2

Table 9.2: The results for each of the recordings according to the phone. Note that only 2 recordings were made with each of the Google Nexus 5 and Samsung S6.

Recording	Ground Truth (steps)	Algorithm Output (steps)	Accuracy
Google Pixel			
1	292	298	97.9%
2	267	271	98.1%
3	323	290	89.7%
4	322	307	95.3%
Average	-	-	95.3%
Google Nexus 5			
1	305	293	96.1%
2	306	298	97.3%
Average	-	-	96.8%
Samsung S6			
1	266	265	99.6%
2	265	253	95.5%
Average	-	-	97.6%

By inspection, it is clear that the choice of phone has no significant impact on accuracy.

9.2 Optimal Parameters

With the optimal set of parameters, an average accuracy of 93.7% and a median accuracy of 96.8% are obtained across the 48 data recordings. The parameters that achieve this accuracy are given in Table 9.3.

Table 9.3: The parameters for the optimal algorithm performance.

Stage	Type	Parameters
Pre-Processing	Default	$t_{scale} = 10^6$, $f_{int} = 100Hz$
Filtering	Gaussian	$N = 13$, $\sigma = 0.35$
Scoring	Mean Difference	$N = 27$
Detection	Default	$c = 1.2$
Post-Processing	Default	$t_{window} = 200ms$

The spread of the accuracy results can be seen below in Figure 9.1. This shows that the majority of the accuracy results is within the 90% to 100% range with some outliers showing significantly lower accuracy.

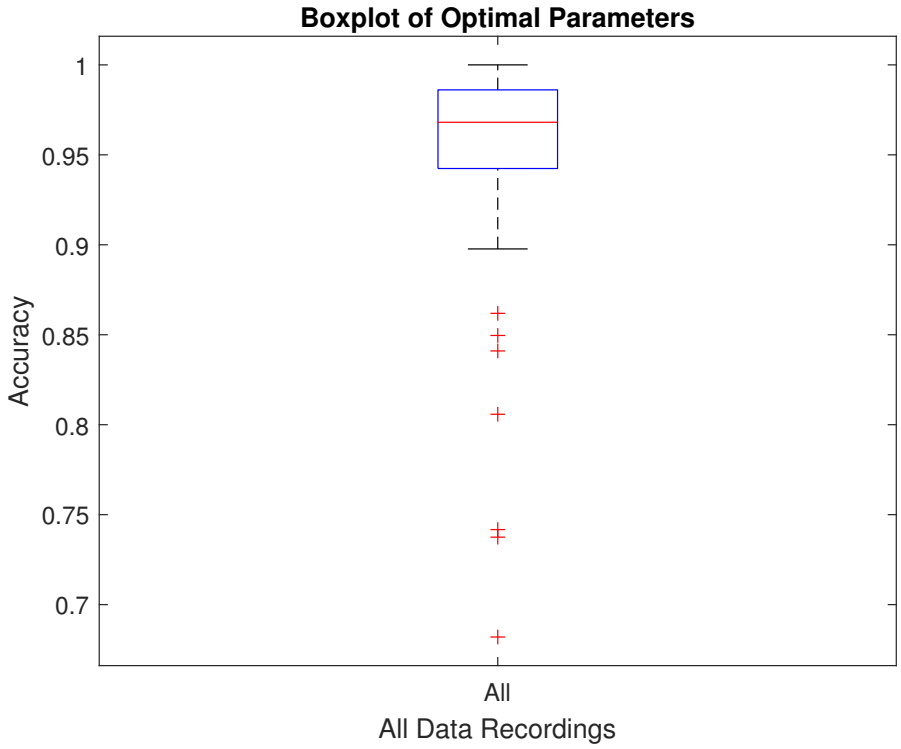

Figure 9.1: The spread of the accuracy of the optimal parameters over the 48 data recordings.

If the results are split according to the position of the phone during the recordings, the source of these outliers becomes evident. These plots can be seen below in Figure 9.2.

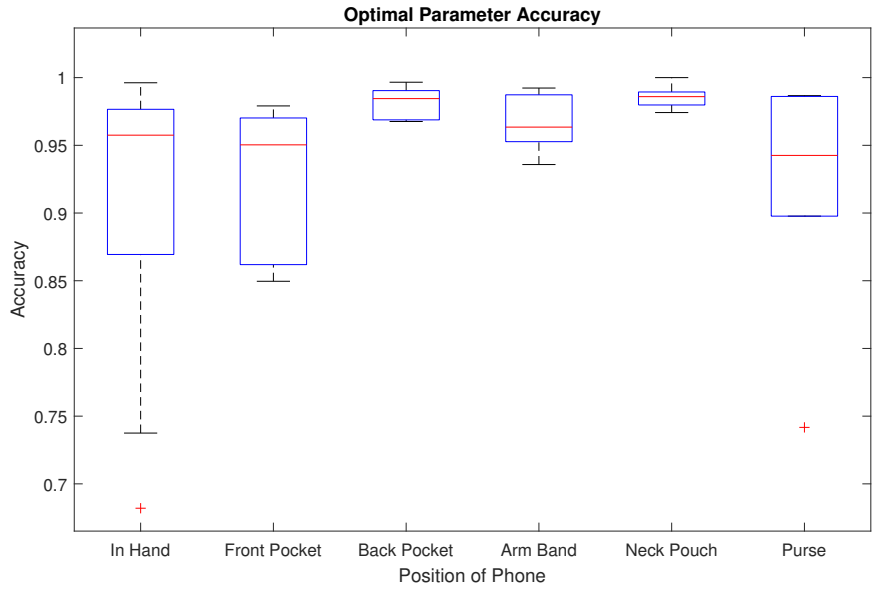

Figure 9.2: The spread of the accuracy of the optimal parameters over the 48 data recordings grouped by phone position. The most extreme outliers come primarily from the 'In Hand' position.

As can be seen from Figure 9.2, the outliers come primarily from the 'In Hand' or 'Purse' positions. This leads to the hypothesis that these positions may induce some harmonics in the walking frequency range. The source of these may be the user swinging their arm when holding the phone or the purse swinging back and forth under arm.

This grouping also reveals that the 'Back Pocket', 'Arm Band', and 'Neck Pouch' positions seem to be the best positions. It should be noted that the 'Arm Band' may be subject to the same harmonics as the 'In Hand' position; however, the relatively limited movement of the upper arm when compared to the hand limits the impact of these harmonics.

These parameters can now be validated against the three patient data recordings mentioned earlier. If the accuracies are similar to the average obtained with the 48 data recordings acquired for optimization, then it supports the idea that these parameters are near optimal.

The results of applying the algorithm with the optimal set of parameters to the patient recordings are detailed below in Table 9.4. The average accuracy is 92.3%. This is very close to the average accuracy obtained with the 48 data recordings used to optimise the algorithm and well within the median quartiles in the boxplot.

Table 9.4: The results of the optimal parameters for the three patient data recordings.

Label	Observed Number of Steps	Algorithm Output	Accuracy
Patient 1	614	654	93.5%
Patient 2	567	605	93.2%
Patient 3	682	749	90.2%

9.3 Position Specific Parameters

The investigations above revealed that a given set of parameters gives different results on different positions. This leads to the possibility of the algorithm switching between parameters given the position of the phone to give the best result. For each position, a set of optimal parameters will be determined and then specified for that position. This analysis leads to very high accuracies for some positions.

The optimal parameters for the In Hand, Front Pocket, Arm Band, Neck Pouch, and Purse positions all have the types of stage in common, so these will be reported together for brevity. The types of stage for the mentioned positions are: Moving Average for the Filtering Stage and Mean Difference for the Scoring Stage. For all of these, the Pre-Processing stage used $t_{scale} = 10^6$ and $f_{int} = 100Hz$ and the Post-Processing stage used $t_{window} = 200ms$. The rest of the parameters for these positions are given in Table 9.5.

Table 9.5: The parameters for optimal algorithm performance in the In Hand, Front Pocket, Arm Band, Neck Pouch, and Purse positions.

Position	Filtering	Scoring	Detection	Average Accuracy
In Hand	$N = 53$	$N = 11$	$c = 1.4$	92.4%
Front Pocket	$N = 29$	$N = 27$	$c = 1.2$	98.0%
Arm Band	$N = 21$	$N = 3$	$c = 1.2$	99.4%
Neck Pouch	$N = 21$	$N = 11$	$c = 1.2$	98.9%
Purse	$N = 29$	$N = 11$	$c = 1.2$	96.9%

9.3.1 Back Pocket

The best set of parameters for the 'Back Pocket' position gives an overall accuracy of 99.2%. The parameters that achieve this are given in Table 9.6.

Table 9.6: The parameters for the optimal algorithm performance in the 'Back Pocket' position.

Stage	Type	Parameters
Pre-Processing	Default	$t_{scale} = 10^6$, $f_{int} = 100Hz$
Filtering	Gaussian	$N = 21$, $\sigma = 0.35$
Scoring	No Scoring	
Detection	Default	$c = 1.2$
Post-Processing	Default	$t_{window} = 200ms$

10 Further Work

The impact of surface on the results should be further investigated. A larger sample size is required to determine whether there is any significant difference in algorithm performance between different floor surfaces.

Since the timestamps data for the actual steps data exists in the database, one direction of exploration could be to analyse how many steps the algorithm identifies correctly and how many steps the algorithm misses. This is equivalent to a false positive/false negative analysis.

Another area that could be explored is a less coarse search for the optimal parameters. The results so far could be treated as being preliminary and a finer grained search over the parameter space could be run to yield the true maximum.

Part II

Sleep Detection

11 Overview

This section covers the development and evaluation of the sleep detection algorithm. The algorithm aims to extract features like sleep onset time, wakefulness onset time, and total time asleep from accelerometer data recorded on a wrist-mounted device.

The basis of the algorithm relies on the fact that the level of activity of a person can be determined from the magnitude of the standard deviation or the energy of the accelerometer signal. If there is no movement, the standard deviation will simply be the square root of the variance of the noise in the accelerometer sensor data (assumed to be zero-mean Gaussian). If there is a high level of movement, then there will be a high standard deviation from the mean signal. If there is a long period of low activity then this period will be defined as corresponding to sleep time. This set of rules and inferences will be the guiding principles upon which the algorithm is built.

12 Database

Algorithm development and testing described in this chapter relies on a database created by Borazio et al. at TU Darmstadt [14]. The database contains accelerometer data from 42 patients, their clinically annotated polysomnography results, and light sensor data from overnight stays in a sleep lab. The patient undergoing polysomnography in a sleep lab has their electroencephalogram, oxygen levels in the blood, heart rate, breathing, eye movement, and leg movement recorded. A trained professional, then produces a report summarising the transitions between sleep stages (deep sleep, Rapid Eye Movement (REM), awake) during the night. The accelerometer data in the Darmstadt database comes from a custom device that is mounted on the wrist of the patient. The data is logged at 100Hz. The data collected by Borazio e. al. was collected from 42 sleeping lab patients aged between 28 and 86 years old. These patients were suffering from a variety of sleeping disorders, later diagnosed as primarily sleep apnea syndrome (SAS), restless leg syndrome (RLS), or narcolepsy. In total, 45 nights of data were collected, with three patients providing 2 nights' worth of data [14].

The data provided is contained and compressed in such a way that it requires reformatting prior to the use in the algorithm described below. Each patient record has its own file (a .npy file, used for Numpy calculations in Python) which contains 7 columns: a timestamp, the runlength encoding, x-acceleration, y-acceleration, z-acceleration, light sensor value, and a ground truth value. The runlength encoding represents how many times in a row does the set of measurements that follows repeat, for example runlength encoding of 10 represents 10 identical copies of the data (x,y,z acceleration, light sensor, and ground truth) with increasing timestamps.

All of the acceleration data is stored in 8-bit unsigned integer format, from 0 to 255. According to Borazio, et. al. this maps to an acceleration value of $-4g$ to $4g$, where g is gravity. The light sensor value is not used in this project and as such will be disregarded. The ground truth data is coded as follows: 0 indicates unknown sleep state, 1-3 maps to different sleep states rated from deepest to lightest, 5 indicates random eye movement (REM) sleep, 6 indicates awake, and 7 indicates movement.

In order to format this data to fit the needs of this project, a few steps were taken:

- The x,y, and z accelerations were linearly mapped back to $[-4g, 4g]$ and the magnitude calculated as $\sqrt{x^2 + y^2 + z^2}$.
- The light sensor value was removed from the data.
- The runlength encoding was decoded.
- The data was downsampled to $10Hz$ to extend the battery life of the wearable device which will

eventually be used in the application of this algorithm.

The downsampling is a simple operation: keep one point out of every 10 after the runtime encoding decoding is complete. Decoding the runlength encoding is equally simple: duplicate the row of n times where n is the runlength encoding for that row.

12.1 Acceleration Scaling

An original value of 127 corresponds to $0g$ and thus the value of acceleration can be derived from:

$$a_n = (a_o - 127) * \frac{8 * g}{256} \tag{12.1}$$

where a_n is the new acceleration value, a_o is the original, integer value of acceleration, and $g = 9.81 \frac{m}{s^2}$ is gravity.

This approach is validated by ensuring that the magnitude of the signal is always gravity or higher. This was true for the reformatted dataset, hence this approach is appropriate.

13 Sleep-Detection Algorithm Description

The algorithm works in two stages: the classification stage and the post-classification filter. The algorithm is designed to work with data sampled at 10Hz.

13.1 Pre-Classification Filter

Originally, the design included a pre-classification filter. However, this was removed as it was found to degrade overall performance. The implementation and explanations for the removal of the filter are described below.

The aim was to remove high-frequency noise and attenuate spikes from the signal. This was done with a simple moving average filter, with the following coefficients:

$$h_k = \frac{1}{N}, 1 \leq k \leq N \tag{13.1}$$

where h_k is the k^{th} coefficient of the filter and N is the length of the filter.

The frequency response of this filter can be seen below in Figure 13.1. Body movement during sleep is generally slow, so a filter removing higher frequency data is desired in this use case.

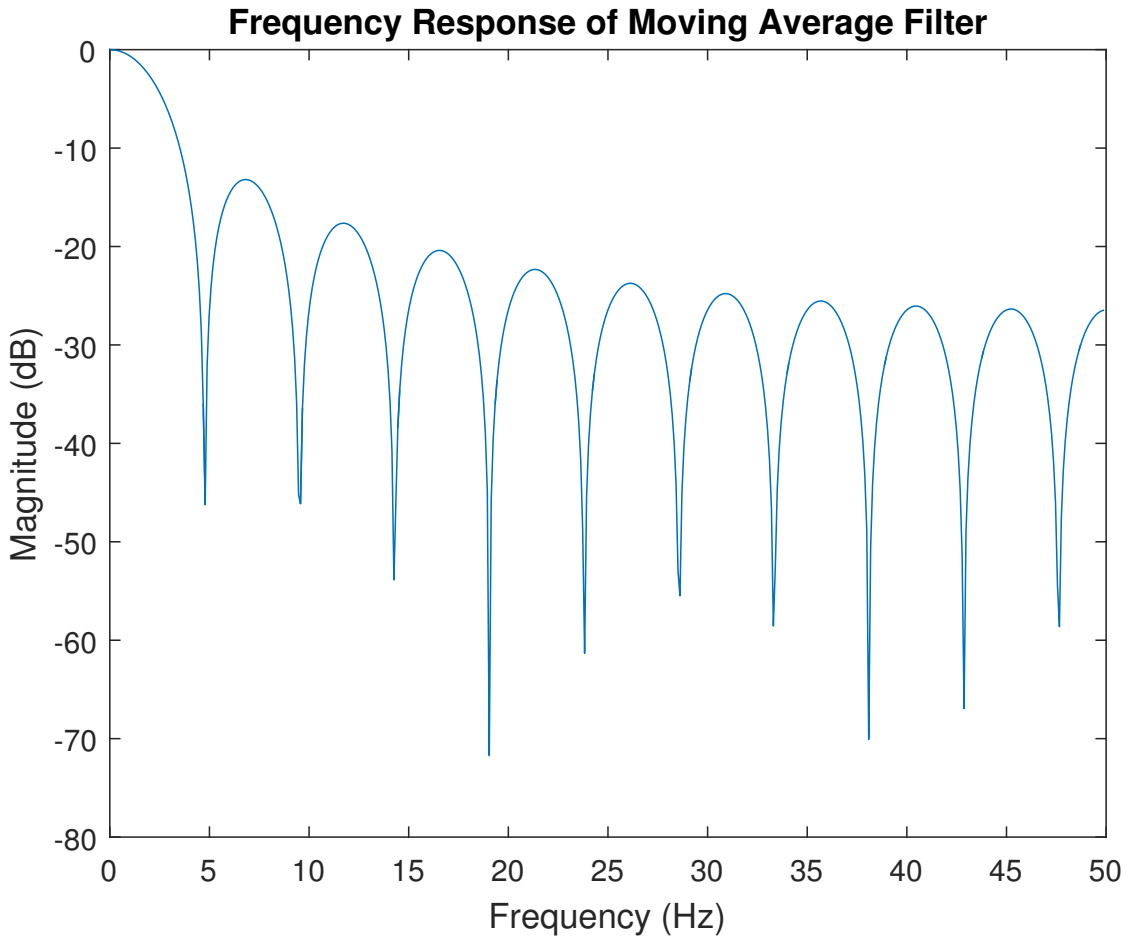

Figure 13.1: Frequency response of the pre-classification filter with a length of 21 samples.

An example of the data before filtering and after filtering can be seen below in Figure 13.2.

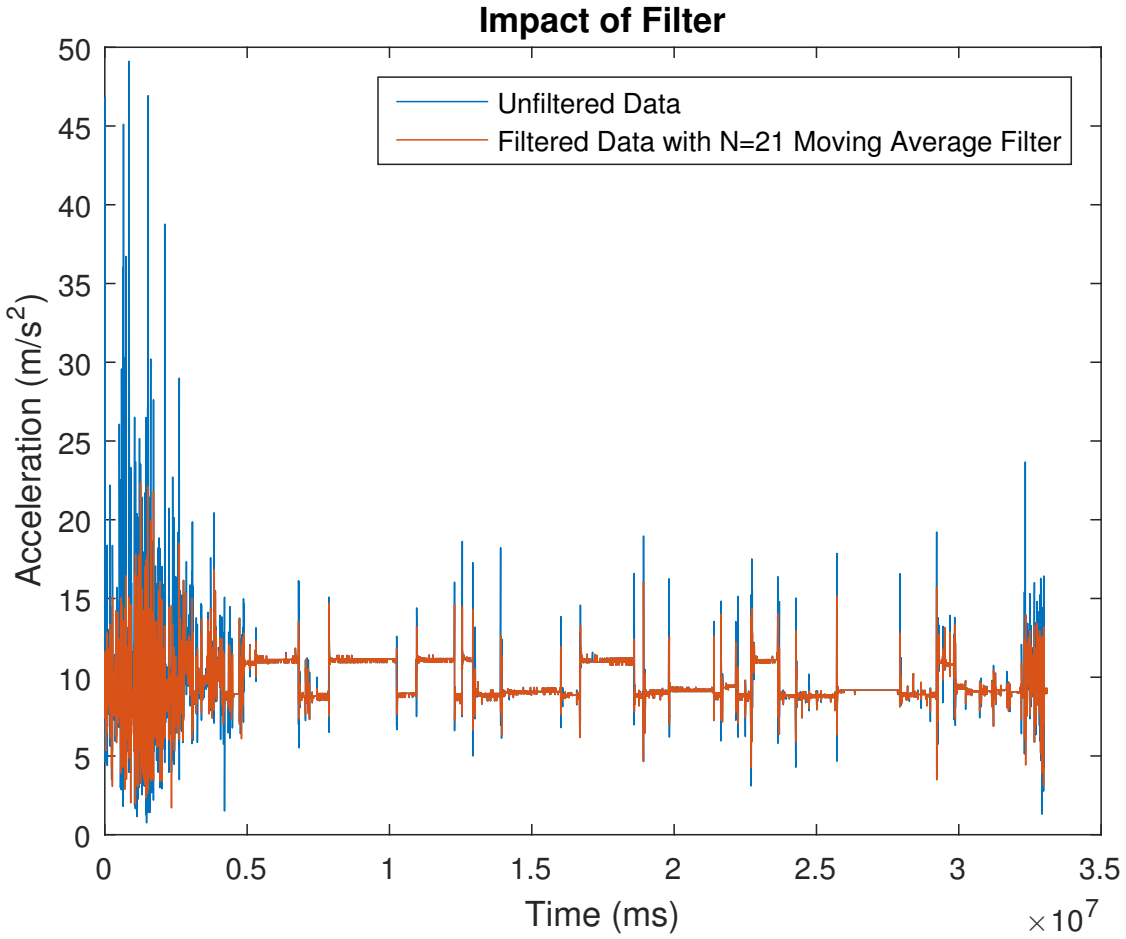

Figure 13.2: Example of data before and after filtering(blue before filter, orange after filter). Note the reduced magnitude in the active areas (areas of high magnitude.)

This graph illustrates why the filter had to be removed. It decreases the amplitude of high-activity signals thus reducing the standard deviation. This has a direct impact on the parameter that we wish to estimate. Removing the filter increased the algorithm accuracy by around 5%.

13.2 Classification Stage

The next stage of the algorithm is to predict whether a segment of the signal corresponds to the subject being asleep or awake. The signal is divided into segments 300 samples long, equivalent to a period of 30 seconds at a 10Hz sampling frequency. This period was chosen as it is the segment length used in polysomnography, hence can be considered the minimum length of a sleep episode.

13.2.1 Data Characterization

There is only one variable upon which the classification is done, so a simple analysis can be done to obtain a threshold that will act as the decision boundary between the two classes: asleep and awake. The segments are separated based on their ground truth class (from polysomnography) and the mean and standard deviation are computed for each class. The probability density functions (pdf)

can then be plotted. The decision boundary is the intersection of these two cruves. This can be seen in Figure 13.3.

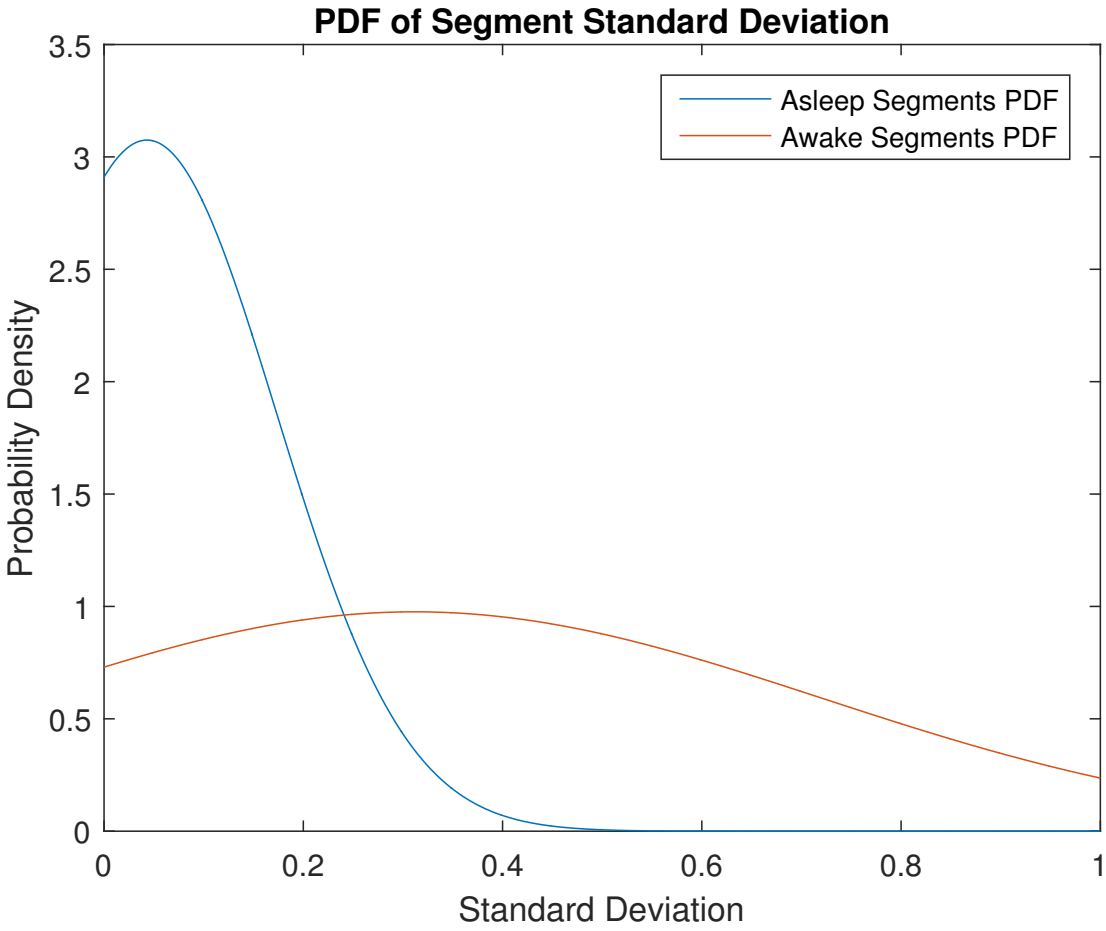

Figure 13.3: The pdfs for both the asleep segments and awake segments. The intersection occurs at $\sigma = 0.221$.

13.2.2 Classification Process

Classification is performed through logistic regression. The weights of the regression are obtained through a learning process with the following steps:

- Separate the segments into awake and asleep sets based on the ground-truth data from polysomnography.
- To ensure a balanced training set (i.e. - equal number of each class) compute the number of segments for training through $n = \frac{min(l_{awake}, l_{asleep})}{2}$ where l_{set} is the length of the set.
- Randomly select n segments from each of the asleep and awake sets. Train on this data.
- Test the learned coefficients on the remaining segments not used in training verify that the training data was not overfitted.

The logistic classifier is defined as:

$$p(\mathbf{x}, \mathbf{m}) = \frac{1}{1 + e^{-\mathbf{mx}}} \tag{13.2}$$

where $\mathbf{x}$ is the feature vector and $\mathbf{m}$ is the coefficient vector. If $p(\mathbf{x}) \geq 0.5$ then that point is classified as asleep (or 1). Otherwise is classified as awake (or 0).

To allow for the possibility of including more than one feature, the algorithm was implemented using a feature vector. When there is only one feature (the standard deviation of the accelerometer data), the feature vector contains 2 elements: a constant (to serve as a bias), and the aforementioned standard deviation. This is defined as follows:

$$\mathbf{x} = \begin{bmatrix} 1 & \sigma \end{bmatrix} \tag{13.3}$$

To train the classifier, a loss function needs to be minimized. To derive this, start with the likelihood of the coefficients given the data:

$$\mathcal{L}(\mathbf{m}) = \prod_{i=1}^{n} p(y_i | \mathbf{x_i}, \mathbf{m}) \tag{13.4}$$

where $\mathcal{L}(m)$ is the likelihood of the coefficients, y_i is the ground truth of the i^th data point.

If the log-likelihood is considered, then this expression can be expanded to:

$$\log \mathcal{L}(\mathbf{m}) = \sum_{i=1}^{n} (y_i \log p(\mathbf{x_i}, \mathbf{m}) + (1 - y_i) \log (1 - p(\mathbf{x_i}, \mathbf{m}))) \tag{13.5}$$

where the first term under the summation represents the contribution to the likelihood when $y_i = 1$ and the second term represents the contribution when $y_i = 0$.

If the log-likelihood is differentiated with respect to $\mathbf{m}$ then it can be shown that the differential is given by:

$$\frac{\partial}{\partial \mathbf{m}} \log \mathcal{L}(\mathbf{m}) = \sum_{i=1}^{n} (y_i p(\mathbf{x_i}, \mathbf{m}) \mathbf{x_i} - (1 - y_i)(1 - p(\mathbf{x_i}, \mathbf{m})) \mathbf{x_i}) \tag{13.6}$$

Unfortunately, this expression has no closed form solution if set to zero. Logistic regression can only be optimized using methods like gradient descent. The premise of gradient descent is simple: calculate the gradient at a point and then move either up or down the gradient, for maximization and minimization respectively. The downside of this method is that there is no guarantee of finding the global maximum, merely a local maximum.

In this project, the training and testing was performed by a Python package called scikit. The

coefficients yielded are given by:

$$\mathbf{m} = \begin{bmatrix} 0.8321 & -8.1804 \end{bmatrix} \tag{13.7}$$

where the first coefficient is related to the constant and the second coefficient is related to the standard deviation. For the training dataset, the final accuracy with these coefficients was 72.9% and for the testing dataset it was 71.2% which shows the data was not overfitted during training.

The decision boundary for the logisitic classifier is shown in Figure 13.4. The boundary occurs at $\sigma = 0.11$.

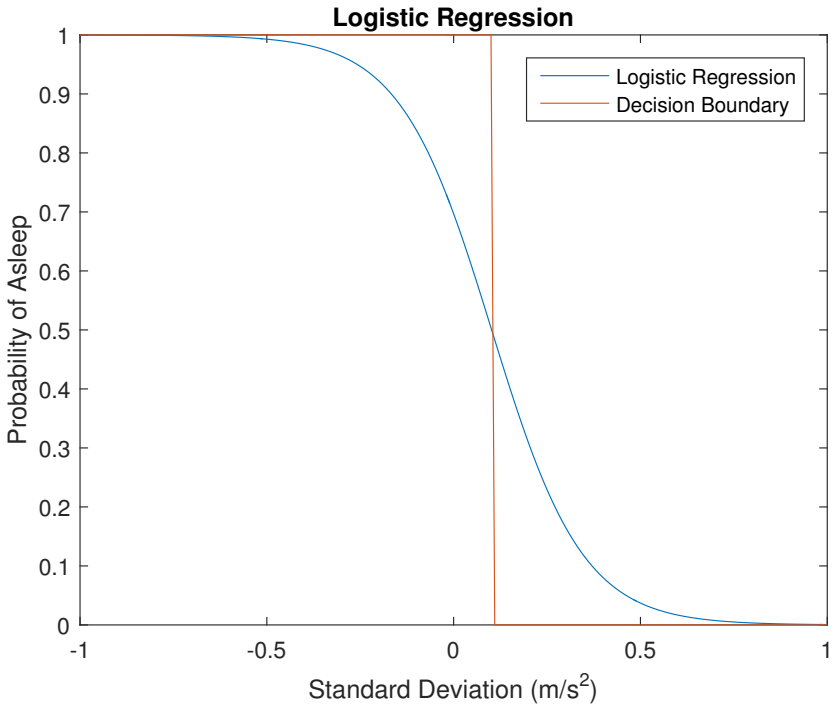

Figure 13.4: A plot of the standard deviation of a signal segment against the probability of the subject being asleep. The step function represents the decision boundary based on $p(x) = 0.5$.

An example of the classified data can be seen below in Figure 13.5. Recall that the output from the logistic regression provides the binary classification: if $p(x) > 0.5$ then the midpoint of the 30-second segment is assigned to class 1 (asleep), otherwise it is assigned to class 0 (awake).

13.3 Post-Prediction Filter

The next and final stage is a post-classification filter. It is reasonable to assume that for our purposes, the minimum sleep time detectable is approximately 5 minutes. The filter works as a hard-limiter, that is to say, if the output of the filter is above some threshold, the output gets rounded up to 1.0 and all other values get rounded down. This is to maintain the 0-1 classification inherent in the problem. Note that this is an additional hard-limiter to the hard-limiter inherent in the logistic regression classifier.

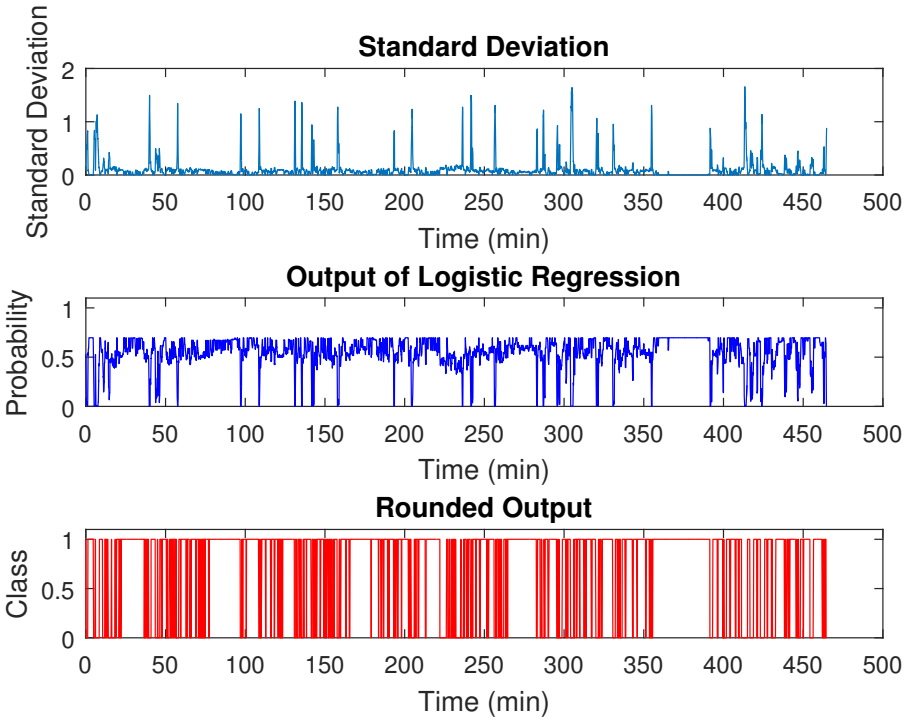

Figure 13.5: An example of the logistic regression classifying the data. From top to bottom: local standard deviation, logistic regression probability, and rounded output.

For any sample, a window is centered on that sample with k points to the left and k points to the right of this point. The filter value is calculated as follows:

$$f_n = \sum_{i=-k}^{k} h_i x_{n+i} \tag{13.8}$$

where f_n is the filter value at the n^{th} point, h_i is the i^{th} filter coefficient and x_{n+i} is the $(n+i)_{th}$ point. If f_n exceeds t, a chosen threshold, then the n^{th} point is marked as corresponding to asleep. k is set to 1500 for this filter, deriving from the 5 minute minimum.

Two windows were considered for this filter: a moving average window (flat) and a Gaussian window. With the moving average window each point inside the window is valued equally such that $h_i = \frac{1}{2k+1}$. With the Gaussian windows points closer to the point under consideration are weighted more heavily than those farther away such that $h_i = \frac{1}{\sqrt{2\pi\sigma_f}} e^{\frac{-(i-k)^2}{2\sigma_f^2}}$ where σ_f is the standard deviation of the filter.

The threshold, t was varied for both filters from 0.5 to 1 in 0.1 increments and the standard deviation, σ, for the Gaussian filter was varied from 1000 to 3000 in 500 increments. After this coarse search was done, the optimization was repeated for thresholds around the maximum from the previous step in increments of 0.01. The σ values were chosen to align with the filter length of 3001, ensuring that

all values have a non-negligible weight, as filter coefficients at points outside of the $\begin{bmatrix} -2\sigma & 2\sigma \end{bmatrix}$ range are effectively given zero weighting.

The best performing filter was the Gaussian with $t = 0.580$ and $\sigma = 2500$. An example of this post-classification filter in action is shown below in Figure 13.6.

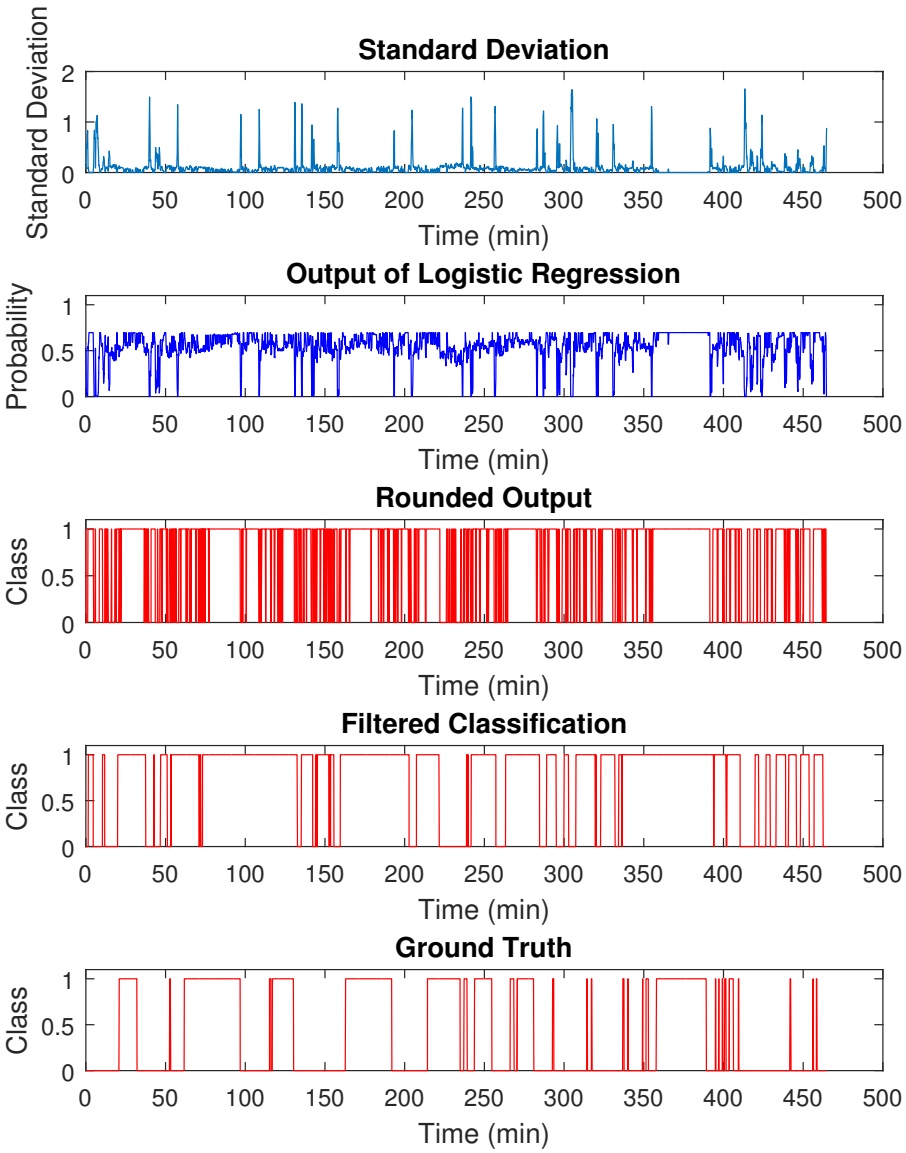

Figure 13.6: An example of the classified output being filtered. The filter used was a Gaussian filter with length 3001, standard deviation of 2500 and threshold 0.58. From top to bottom: local standard deviation, logistic regression probability, rounded output, filtered classification, and ground truth.

14 Results

The results presented here are the result of running the algorithm described above, with no pre-classification filter, over 45 of the 46 data records in the Darmstadt database. One of the records was rejected as there was very little sleep in the record and the data was inconsistent throughout.

An example of the algorithm classification in action can be seen in Figure 14.1. Note that a state of 1 indicates sleep, a state of 0 indicates wakefulness.

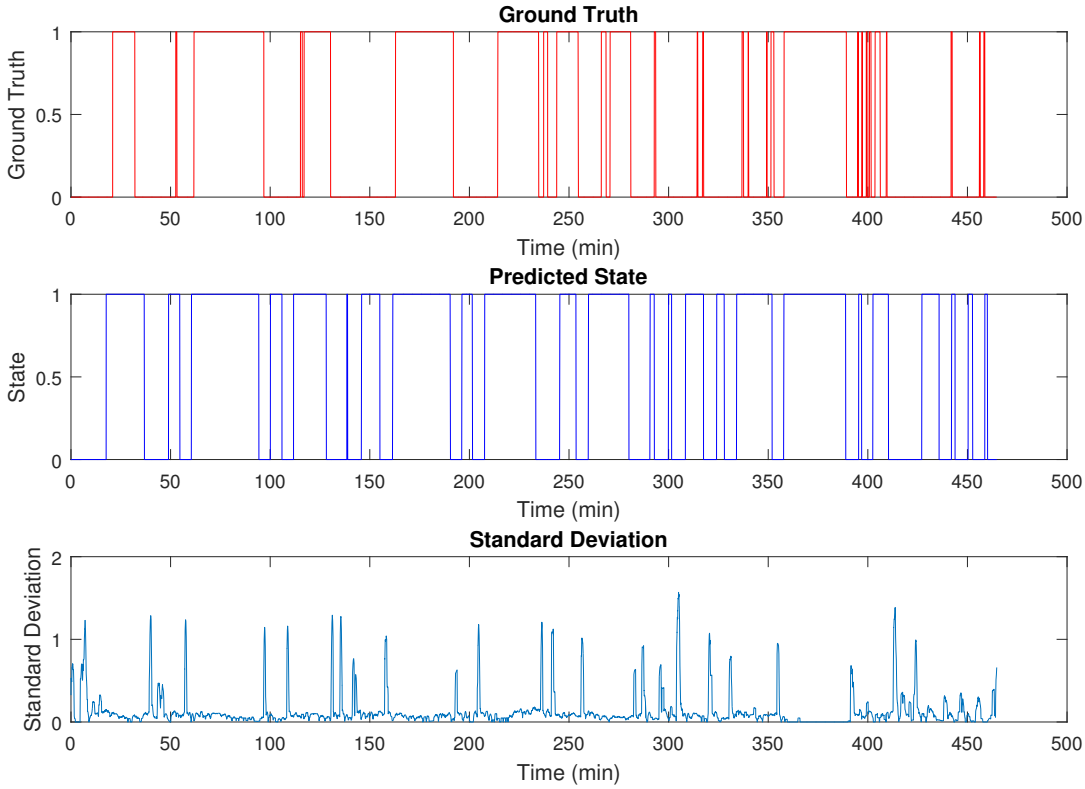

Figure 14.1: Example of the classification algorithm. From top to bottom: ground truth, classification output, standard deviation.

14.1 Statistics

The performance of the algorithm is assessed using 3 metrics: the Total Time Asleep (TTA), the total amount of sleeping time in the record, the Sleep Onset Time (SOT), the timestamp of the first incidence of sleep, and the Wakefulness Onset Time (WOT), the timestamp of the last incidence of sleep. The latter two parameters are representative of the start and end of a night's sleep.

For the algorithm described above, the following results are obtained.

Table 14.1: Results for the sleep detection algorithm.

Metric	Mean Error	Median Error
Total Time Asleep (%)	22.64%	10.93%
Sleep Onset Time (minutes)	25.35 min	19.37 min
Wakefulness Onset Time (minutes)	19.00 min	8.48 min

In comparison, the algorithm presented performs better than the algorithm described in Borazio, et. al. [14] which achieved 21.17% median error for Total Time Asleep (TTA). The paper from Borazio et al. makes no mention of Sleep Onset Time (SOT) or Wakefulness Onset Time (WOT) statistics, so there cannot be a comparison here.

From the results, it is clear that all the error statistics have outliers far from the median since $e_{mean} > e_{median}$ for all the statistics. This is confirmed by the boxplots of the errors shown Figures 14.2 and 14.3. For example, for the boxplot of TTA there are a number of outliers that skew the mean away from the median, but the bulk of the errors is found between 5% and 32% error.

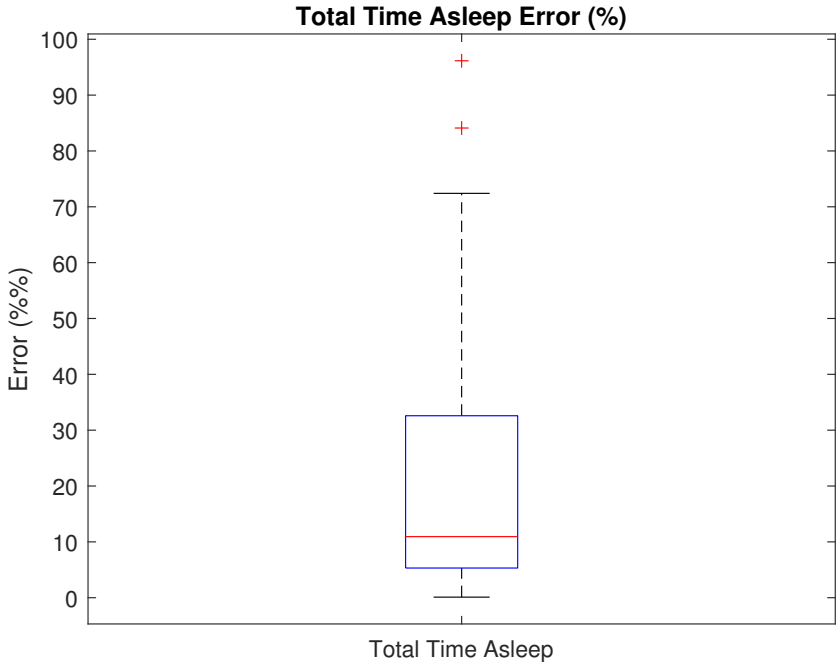

Figure 14.2: Boxplot of errors for the Total Time Asleep. Median is at 11.89%.

Another interesting phenomenon is the difference in accuracy between the SOT and WOT, as these fundamentally represent the two abilities of the algorithm: to identify sleep and wakefulness segments respectively. One explanation for the WOT being generally more accurate is that the end of sleep is usually close to the end of the recording. This limits the WOT error, whereas a similar phenomenon does not occur with the SOT.

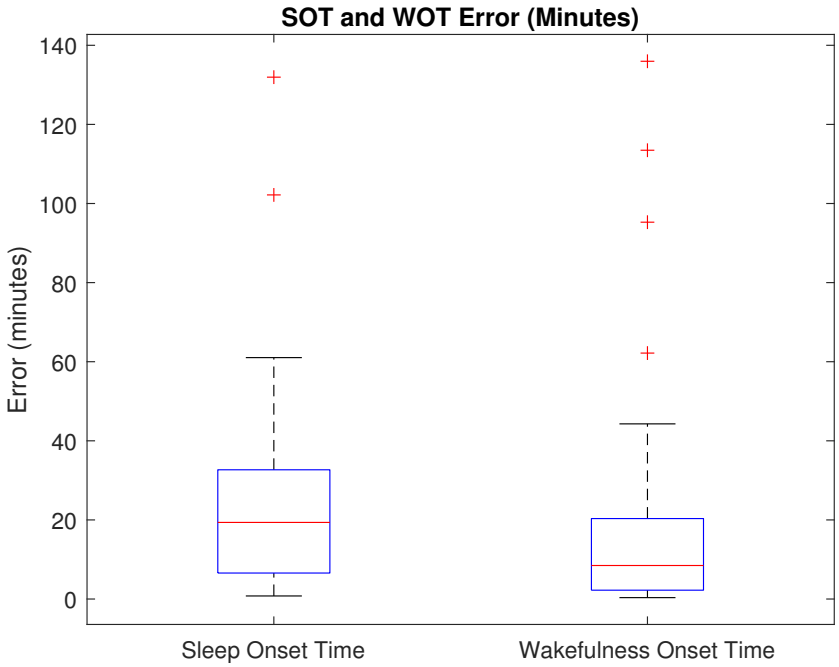

Figure 14.3: Boxplots of errors for the Sleep Onset Time and Wakefulness Onset Time. Note that the WOT is generally more accurate than the SOT.

14.2 Precision and Recall

Precision is defined as "the fraction of retrieved instances that are relevant" and recall is defined as "the fraction of relevant instances that are retrieved" [27]. Here sleep precision is the percentage of classified sleep states that are sleep states in the ground-truth data. Sleep recall is the percentage of ground-truth sleep states that are classified as sleep states. The precision and recall were calculated on a per recording basis and tabulated for the entire database. The results of this can be seen in Figure 14.4.

The first point to note is that sleep precision is generally higher than awake precision. An explanation for this is that the algorithm is more biased towards classifying a sample as awake than sleep. This means that the algorithm will only classify a sample as corresponding to sleep when it is very sure about the state, that is to say the default assumption is that the sample corresponds to awake. Another point to note is that there is a very large variance in the awake precision and recall. This means that the algorithm generally has a hard time correctly identifying awake states. The awake recall being higher than precision indicates again that the algorithm has a default assumption that any given sample corresponds to awake.

In Borazio et al. the authors behaved the opposite way, along with the Cole [16] and Oakley [15] algorithms, in that they tended to bias towards classifying a sample as belonging to the sleep state. This is evidenced by the awake recall being slightly higher in our algorithm (51% median vs. 45% for

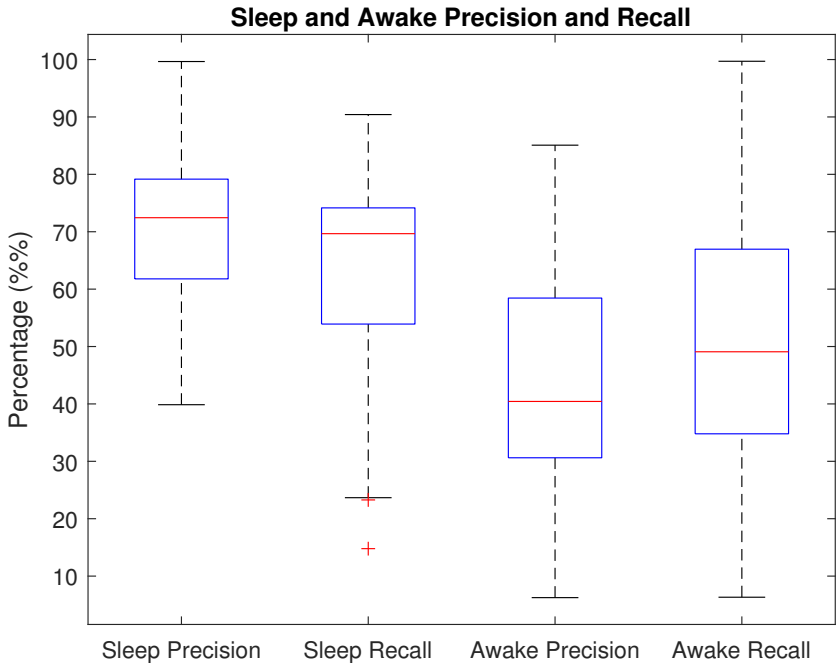

Figure 14.4: Boxplots of precision and recall for sleep and awake states. From left to right: sleep precision, sleep recall, awake precision, awake recall.

Borazio et al.). This however, comes at the cost of a reduced sleep recall (72% vs 94% for Borazio et al.).

This presents an interesting trade-off that may be made when selecting the optimal algorithm. The algorithm developed in this project gives better performance with respect to the estimation of Total Time Asleep, whereas the recall statistics for the Borazio et al. were generally better. If the application of the algorithm was purely for recording the time spent asleep each night, as may be the case when tracking the well-being of a healthy individual, then the algorithm presented in this report would be the better choice. On the other hand, if the application was geared towards identifying and tracking sleep fragmentation during the night (which reflects the condition of an individual with a chronic disease such as COPD or heart failure), an algorithm that has generally better recall statistics, like those presented by Borazio et al., would be the better choice.

15 Further Work

Although small improvements were made to the sleep detector, there is still room for significant improvement. One area that can be explored is extending the feature vector used for classification. Other features of the signal segment like the skewness or kurtosis of the distribution could be explored. The frequency content of the signal could also be investigated, in an attempt to correlate this with the sleep state. Ultimately, this approach is limited due to the simple fact that it is impossible to differentiate between the situation in which someone is awake and not moving and the situation in which someone is sleeping.

One other area of interest would be to explore extending the algorithm to include more sensors than the accelerometer, for example recording heart rate or light intensity, using these factors as additional features. The structure of the algorithm is designed to accommodate additional features as these can simply be added to the feature vector without otherwise changing the algorithm.

Part III

Conclusion

16 Conclusion

This project set out to achieve two primary goals: develop a general step counter implemented on a smartphone that uses an accelerometer as its data source and develop a sleep detection algorithm that uses a wrist mounted accelerometer as its data source.

The generalized smartphone step counter exhibits very high accuracies with a median of 96.8% across a variety of scenarios. A number of scenarios like phone in pocket, phone in hand, etc. were examined and specific parameters for these scenarios were derived with even higher levels of accuracy than for the generic parameters derived for all scenarios. The general algorithm was further validated against three data recordings from patients which resulted in similar statistics demonstrating its generalised capabilities. The step-counter algorithm could also be optimised as it was developed modularly and has an infrastructure built around it to enable rapid iteration. The database collected in the optimization and validation stage will be released publicly to provide a resource for future research.

The sleep detection algorithm was developed using an existing database. It outperformed previous algorithms evaluated on this database with respect to the accuracy of metrics like the extraction of Total Asleep Time (10% reduction in median error). Compared to previous attempts, which tended to overestimate sleep, this algorithm was more balanced in its estimation of the states (awake and asleep) which is a contributing factor to the higher accuracy. As with the step counting algorithm, the sleep detection algorithm is also built with future development in mind, as minimal changes would be needed to introduce new features into the feature vector for classification.

These two algorithms introduce new tools into the medical healthcare professional's toolkit that enable rapid and inexpensive data collection. This data can then be used to help inform care decisions. The data collection can be done at home, without the presence of medical professionals or the use of expensive equipment. For patients suffering from conditions such as chronic obstructive pulmonary disease or heart failure that require constant monitoring and evaluation, this can both simplify the patients' lives and reduce the cost to the healthcare system. The proliferation of smartphones and smart devices mean that this sort of approach can become more widespread, increasing the data available to healthcare professionals, and driving down the costs of healthcare provision.

References

[1] Oxford Biomedial Research Center. *Digital Health for Chronic Diseases Management*.

[2] *Chronic obstructive pulmonary disease (COPD)*. URL: http://www.nhs.uk/conditions/Chronic-obstructive-pulmonary-disease/Pages/Introduction.aspx.

[3] *Heart Failure*. URL: https://www.bhf.org.uk/heart-health/conditions/heart-failure.

[4] *Chronic Obstructive Pulmonary Disease Statistics*. URL: https://statistics.blf.org.uk/copd.

[5] T. Rasekaba et al. "The Six Minute Walk-Test: A Useful Metric for the Cardiopulmonary Patient". In: *Internal Medicine Journal* (2009), pp. 495–501.

[6] *Deloitte Mobile Consumer Survey 2016*. URL: https://www.deloitte.co.uk/mobileuk/assets/pdf/Deloitte-Mobile-Consumer-2016-There-is-no-place-like-phone.pdf.

[7] *UK: smartphone ownership by age from 2012-2016*. URL: https://www.statista.com/statistics/271851/smartphone-owners-in-the-united-kingdom-uk-by-age/.

[8] *BLU Energy Diamond Mini UK SIM-Free Smartphone with 3000 mAh Battery - Gold*. URL: https://www.amazon.co.uk/BLU-Diamond-SIM-Free-Smartphone-Battery/dp/B01LEXH900/ref=sr_1_3?ie=UTF8&qid=1493714738&sr=8-3&keywords=BLU+Diamond+M.

[9] *Samsung I9300 Galaxy S III*. URL: http://www.gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php.

[10] *Towards a Benchmark for Wearable Sleep Analysis with Inertial Wrist-worn Sensing Units*. URL: https://es.informatik.uni-freiburg.de/datasets/ichi2014.

[11] Najme Zehra Naqvi et al. "Step Counting Using Smartphone-Based Accelerometer". In: *International Journal on Computer Science and Engineering* 4 (2012).

[12] Girish Palshikar. "Simple Algorithms for Peak Detection in Time-Series". In: (2009).

[13] Agata Brajdic and Robert Harle. "Walk Detection and Step Counting on Unconstrained Smartphones". In: *UbiComp - Session: Sport and Fitness*. 2013.

[14] Marko Borazio et al. "Towards Benchmarked Sleep Detection with Intertial Wrist-worn Sensing Units". In: *IEEE International Conference on Healthcare Informatics*. 2014.

[15] Oakley. "Validation with Polysomnography of the Sleepwatch Sleep Wake Scoring Algorithm used by the Actiwatch Activity Monitoring System". In: (1997).

[16] Cole et al. "Automatic Sleep/Wake Identification from Wrist Activity". In: (1992).

[17] Vincent vanHees et al. “A Novel, Open-Access Method to Assess Sleep Duration Using a Wrist-Worn Accelerometer”. In: (2015).

[18] Werner et al. “One-Mile Step Count at Walking and Running Speeds”. In: *ACSM’s Health and Fitness Journal* 12 (2008).

[19] Ronald Knoblauch, Martin Pietrucha, and Marsha Nitzburg. “Field Studies of Pedestrian Walking Speed and Start-Up Time”. In: *Journal of the Transportation Research Board* 1538 (1996).

[20] A. R. Collins. *Digital Filter Design*. URL: http://www.arc.id.au/FilterDesign.html.

[21] Jiapu Pan and Willis Tompkins. “A Real-Time QRS Detection Algorithm”. In: *IEEE Transactions on Biomedical Engineering* BME-32 (1985).

[22] *Velostat*. URL: https://en.wikipedia.org/wiki/Velostat.

[23] *Flask Microframework*. URL: http://flask.pocoo.org/.

[24] *Gunicorn*. URL: http://gunicorn.org/.

[25] *PostgreSQL Database*. URL: https://www.postgresql.org/.

[26] *Google Cloud Compute*. URL: https://cloud.google.com/compute/.

[27] *Precision and Recall*. URL: https://en.wikipedia.org/wiki/Precision_and_recall.

4YP Risk Assessment 2016

Description of 4YP task or aspect being risk assessed here: *(Read the Guidance Notes before completing this form)* Creating Pressure Sensor Circuit with Arduino Microcontroller for Step Counting		4YP Project Number: 11060
Site, Building & Room Number: Thom Building, 5th Floor, Electronics Lab	Approx size of equipment/apparatus used or built (in metres): Height: 0.2m Width: 0.2m Length: 1m	Photo provided? NO
Assessment undertaken by: Jamieson Brynes	Signed: Jamieson Brynes	Date:1/11/2016
Assessment Supervisor: L. TARASSENKO	Signed: L. Tarassenko	Date: 9/11/16

Assessing the Risk*

You can do this for each hazard as follows:

- Consequences: Decide how severe the outcome for each hazard would be if something went wrong (i.e. what are the Consequences?) Death would be "Severe", a minor cut to a finger could be regarded as "Insignificant".
- Likelihood: How likely are these Consequences to actually happen? Highly likely? Remotely likely, or somewhere in between?
- Risk Rating: Start at the left of the coloured Matrix. On your chosen Consequences row, read across until you are in the correct Likelihood column for the hazard in question. For example, an outcome with Severe consequences but with a Low probability of actually happening equates to a Medium risk overall. In this case "Medium" is what should be written in the Risk.

RISK MATRIX

CONSEQUENCES	LIKELIHOOD (or probability)			
	High	Medium	Low	Remote
Severe	High	High	Medium	Low
Moderate	High	Medium	Medium/Low	Effectively Zero
Insignificant	Medium/Low	Low	Low	Effectively Zero
Negligible	Effectively Zero	Effectively Zero	Effectively Zero	Effectively Zero

Hazard *(potential for harm)*	Persons at Risk	Risk Controls In Place *(existing safety precautions)*	Risk*	Further Actions Needed to Reduce Risk
Potential burns as soldering Iron is extremely hot	Person soldering	1. Mounts available for placing the irons when not using them. 2. Wear heat proof gloves when soldering. 3. Use clamps or pliers to hold components when soldering to increase distance between the iron and the user. 4. Turn off iron when not in use.	Medium	Ensure the heat proof gloves are available.
Fumes from soldering can potentially be harmful	Those in close proximity to the soldering action	1. Use lead-free solder 2. Maintain a good distance from the soldering action to minimize the fumes inhaled 3. If in an enclosed area, use fume extraction 4. If possible, use an open area to allow for	Low	Check on availability of fume extraction systems

Hazard *(potential for harm)*	Persons at Risk	Risk Controls In Place *(existing safety precautions)*	Risk*	Further Actions Needed to Reduce Risk
		the dispersion of the fumes		
240 Vac electrical hazard	People in the lab	1. Regular testing and inspection of the soldering units 2. Switches closely available for emergency shutoff of the units	Low	Check PAT label for validity