MCEN90032 Sensor Systems Workshop 1 Project Report

Jarod Fozhun Tang

1175923

Contents

Introduction:

A pedometer is a small, portable device which counts the number of steps taken by a person, with modern iterations of pedometers typically using internal gyroscopes, accelerometers and Global Positioning System (GPS) signals to track such steps (wonderopolis.org, n.d.). In recent years, pedometers have emerged as a popular, convenient and cost-effective tool to monitor physical activity levels and maintain optimum health. The purpose of this report is to develop a simple pedometer by using the multi-axis accelerometer pre-equipped in everyday smartphones and processing the accelerometer data using MATLAB's inbuilt functions, with the aim of developing an understanding of signal processing through a combination of theoretical analysis and the observation of experimental results.

Methodology:

Part 1: Obtaining Walking Data

Appropriate walking data was obtained using the application MATLAB mobile. Using the sensor function in MATLAB mobile, a sampling frequency of 10Hz was chosen. This sampling rate is appropriate as activities such as walking is a relatively smooth and continuous activity, it's unlikely that there will be many sudden changes in acceleration. Furthermore, according to the Nyquist-Shannon theorem, the sampling rate must at least be twice as large as the frequency component of the signal being observed or in other words, at least larger than the Nyquist frequency in order to accurately reconstruct a continuous time signal from its samples. The chosen sampling rate must fulfill this condition to avoid aliasing, as this would otherwise cause the high-frequency components of the signal to be misrepresented. Naturally, a higher sampling rate would produce a more comprehensive signal, with the additional benefit of improved noise immunity. However, due to resource constraints such as storage space and processing power, 10Hz is considered sufficient enough to capture the correct number of steps.

Walking data was then obtained using two different methods. This resulted in two differing datasets that will be referred to as Dataset 1 and Dataset 2.

Dataset 1 was obtained by putting the phone in the observer's pocket and having them walk in a straight line for a predetermined amount of time. To calibrate the sensor for dataset 1, the walker would firstly stand stationary with the phone in their pocket to collect some initial "zero readings" before walking in a straight line for a predetermined amount of time.

Dataset 2 was obtained by having the observer hold the phone in their hand aligned with the three axes of the accelerometer with respective to their own frame, with the longest side of the phone being pressed firmly against the walker's stomach to minimise noise. A similar calibration process to dataset 1 is performed, where the walker would firstly stand still before walking, to gather some initial "zero readings". The z-axis of the phone was kept vertical whilst the x-axis was positive in the direction of walking, with the orientation of the phone remaining unchanged for the duration of the straight line walk. The subsequent data taken using both methods was then uploaded to MATLAB

Drive, where the data can later be accessed locally by a computer to perform further analysis.

Part 2: Processing and Characterising the Collected Acceleration Signals

Data Visualisation

Once the data has been collected, the raw acceleration data is processed by firstly, calculating the norm of the acceleration signal. The norm of the acceleration signal is calculated as shown below, where a_x, a_y, a_z is the acceleration of X, Y and Z respectively. A plot of a_x, a_y, a_z is depicted below.

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Equation 1: Norm of the Acceleration Signal

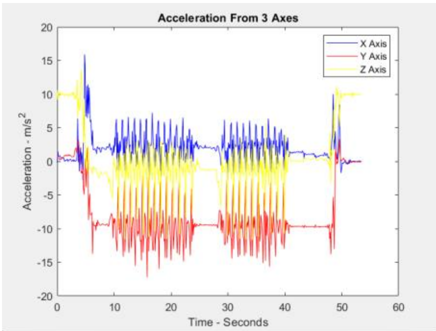

Figure 1: Acceleration of 3 Axes

This process ensures that movements that result in extreme variations in overall acceleration, such as steps taken whilst walking, irrespective of device orientation, will be detected. The mean of the norm of acceleration is then subtracted from itself. This process removes any constant effects such as gravity and centres the data about zero. This data is then plotted against time to produce a graph depicting the acceleration of the device against time (s), prior to the use of any filtering techniques.

Frequency Analysis

Acceleration data is often contaminated by noise from various sources. Therefore, to properly analyse the walking data, it must be isolated from the dataset using a Discrete Fourier Transform (DFT). A DFT, takes in a discrete time signal as an input, and produces a discrete set of frequency components which represents the amplitudes and phases of the sinusoidal signals that constitute the original function. Therefore, this simplifies the frequency analysis process for the observer, as the initial signal is decomposed into its constituent signals, separating the walking signal from the noise produced whilst recording the data. In this instance, the walking signal would be the highest peak, since the acceleration produced from walking would be the most prevalent signal in this dataset.

The MATLAB function *fft()* is used to perform DFT on inputted dataset. After the Discrete Fourier Transform is applied, the zero-frequency component is shifted to the centre of the spectrum using the MATLAB function *fftshift()*. This process shifts the symmetrical distribution leftwards and only leaves the positive half of the initial distribution in view for the observer. The largest peak and its respective frequency of the distribution is stored in the variables *highest_val* and *highest_val_freq* for later use. The resulting amplitude spectrums of dataset 1 and 2 are then plotted against frequency (Hz). The position of the variables *highest_val* and *highest_val_freq* relative to their respective single sided amplitude spectrum is marked on the figures below as a star.

Processing the Signal Using Digital Filter

From figures 3 and 4, through the various peaks depicted (other measurements besides steps), it can be observed that there is noise present in both dataset. Therefore, a 2nd order Butterworth bandpass filter is suitable for this application. A Butterworth filter has a frequency response which is maximally flat in the passband whilst having a zero roll off response in the stop band, however this comes at the expense of having a relatively wide transition band. This type of frequency response preserves the signals within the range of the passband whilst ideally attenuating signals outside of the cutoff frequencies. Since the filter is of 2nd order, it will subsequently have a roll off rate of 40db/decade. Higher order filters of a similar nature will have a steeper transition band and naturally, better attenuate signals that fall outside of the passband. However, an increase in the order of the filter will demand more storage and additionally, will reduce filter accuracy due to a loss of information. Figure 1 below visualises the roll off response as the order of the filter increases.

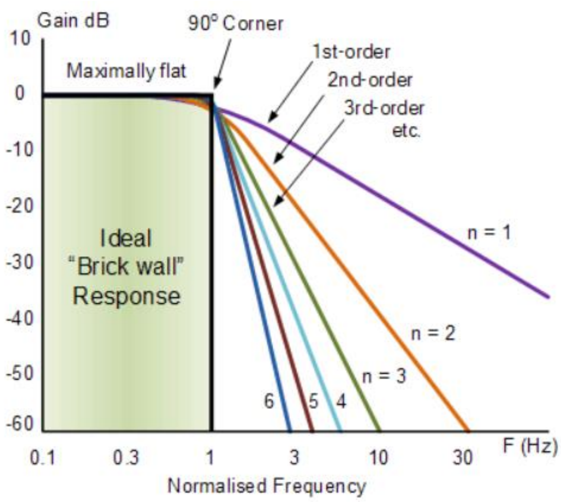

Figure 2: Visualisation of roll off steepness (Electronics Tutorials, 2013)

Since the walking cadence for each person will be different, the cutoff frequencies are calculated relative to inputted data, in order to make the pedometer algorithm as dynamic and as intelligent as possible. The calculation of the lower and upper bound of the of the cutoff frequencies is depicted below, where w_1 (Hz) and w_2 (Hz) are the lower and upper bound respectively, f (Hz) is the walking frequency determined from the data and α is a scalar which controls how wide the passband is. A suitable value for α is determined after implementing more test data. The value f can be found by iterating through the single-sided amplitude dataset and finding the frequency, f which corresponds to the highest peak.

$$w_1 = f - \alpha \times f$$

Equation 2: Lower Bound of Cutoff Frequency

$$w_2 = f + \alpha \times f$$

Equation 3: Upper Bound of Cutoff Frequency

Part 3: Building the Pedometer Using the Accelerometer Measurements

Using test data, the number of steps taken is calculated using the inbuilt function below, where the variable y is the filtered data.

[pks,locs] = findpeaks(y, 'MINPEAKHEIGHT' ,minPeakHeight);

The variable *minPeakHeight* ensures that only local maxima with a minimum height above $minPeakHeight = std(y) - \frac{std(y)}{\beta}$ is counted as step, where β is tuned experimentally. The filtered versions of dataset 1 and 2 as well as their respective filtered amplitude spectrums are then plotted for further analysis and the counted steps are marked with a red triangle.

Results

Dataset 1

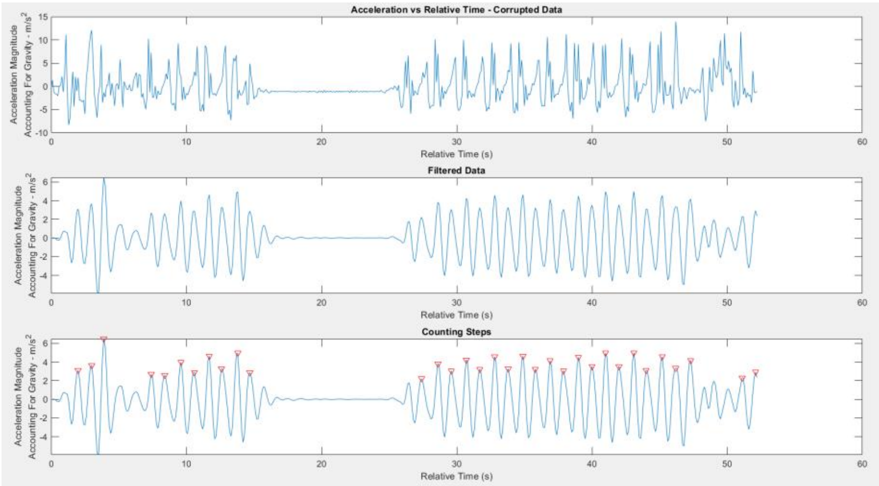

Figure 3: Time Domain Plot of Dataset 1

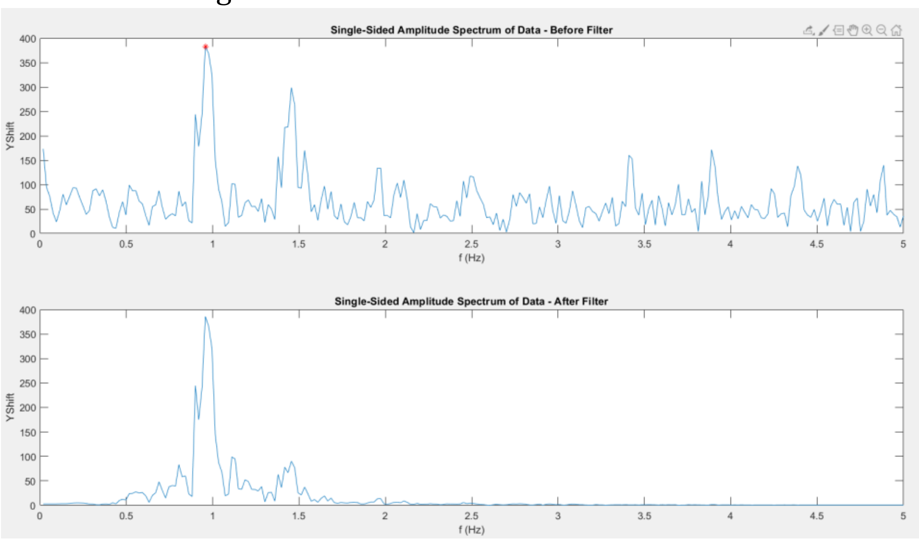

Figure 4: Single Sided Amplitude Plots of Dataset 1

From observing the graphs depicted in figure 2, the amplitude of the filtered data has been greatly reduced, when compared to the corrupted data. Additionally, some features such as the sharp edges of the peaks are absent in the filtered data. Therefore, it can be assumed these features are the result of noise, as they have been attenuated via the implemented filter. Furthermore, it can be observed that for consecutive steps, there is a pattern where peak height will alternate from larger to smaller. This can be attributed to the fact that the phone was placed in the walker's right leg pocket, meaning that the signal from the right leg will be more pronounced compared to the signal produced by the left leg.

From figure 3, a similar effect can be observed, where the surrounding peaks about the largest peak that can be observed in the single-sided amplitude spectrum before the filter are absent in the below graph. This indicates that the filter has properly attenuated the frequencies outside the defined passband.

For the dataset used to produce figures 2 and 3, the algorithm calculated the total number of steps counted being 33, whilst the true value is 30 steps. A reason for this inflated result is that the sensor counts the procedure of the walker moving their phone from and into their pocket as a step. To account for this, the observer would have to subtract the incorrectly counted steps from the initial total, using the graphs produced. Additionally, when the sensor was turned on, pauses were taken before and after walking to ensure that the walking acceleration could be differentiated from acceleration caused by moving the phone from and into the pocket. In this instance, there were 3 additional steps that were counted from the walker putting the phone in their pocket, and 2 additional step that as counted when the walker removed the phone from their pocket. By accounting for this discrepancy, the measured number of steps becomes 28. Below is table with additional training data, obtained using the steps outlined for dataset 1 in the methodology. The overall accuracy of the pedometer when dataset 1 is the input is 97.6%.

Table 1: Dataset 1 Training Data

Filename	Measured Number of Steps	True Number of Steps	Accuracy
`sample_dataset1.mat`	19	18	94.7%
`accel_dataset1run1.mat`	30	30	100%
`dataset1take1.mat`	48	50	98%

Dataset 2

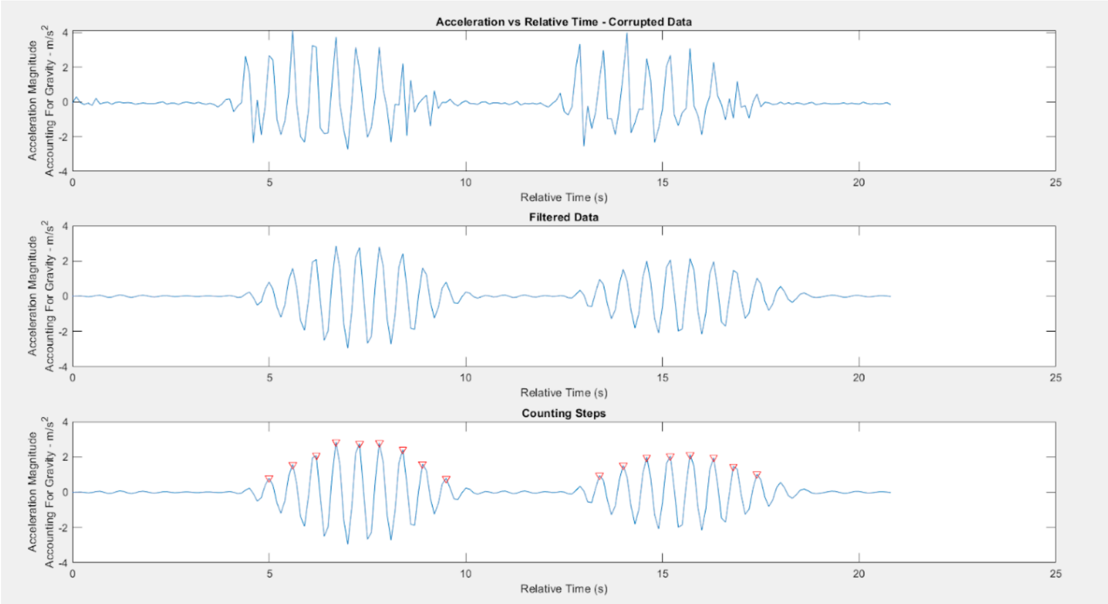

Figure 5: Time Domain Plot of Dataset 2

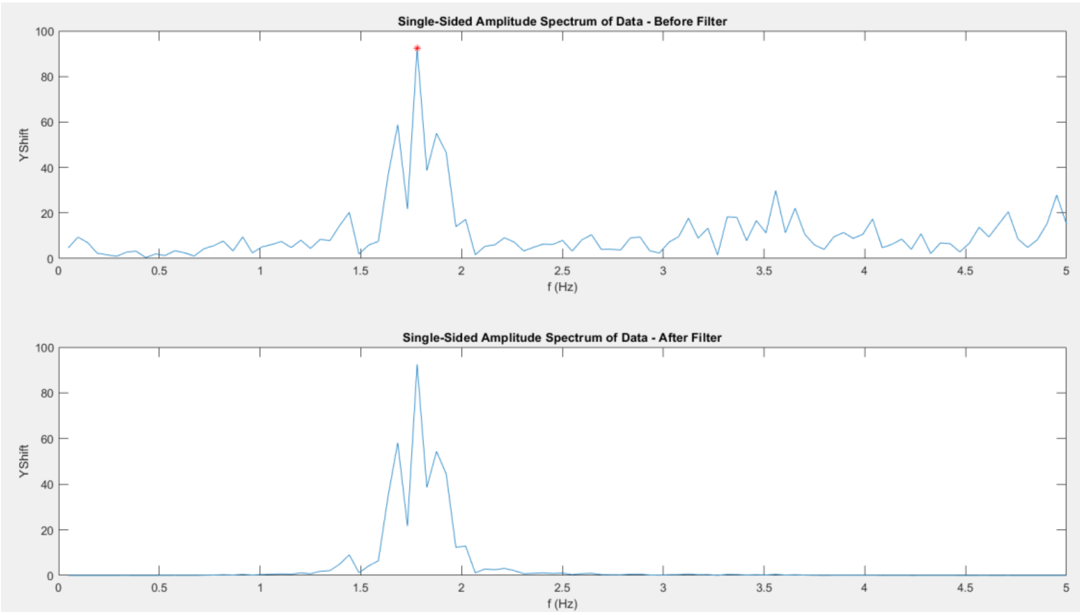

Figure 6: Single-Sided Amplitude Spectrum of Dataset 2

Comparing the filtered and unfiltered data depicted in figure 4, it is observed that the sharp edges which constituted the peaks in the corrupted data are absent in the filtered data. Additionally, frequencies which fall outside of the passband bounds are heavily attenuated by the implemented filter, as observed in figure 5. The algorithm for at least this instance of dataset 2 calculates the correct number of steps taken, being 18 steps. Below is table containing additional training data. The overall accuracy of the pedometer based on this set of training data is 97.6%.

Table 2: Dataset 2 Training Data

Filename	Measured Number of Steps	True Number of Steps	Accuracy
sample_dataset2.mat	17	17	100%
dataset2_take2.mat	48	50	96%
dataset2_take3.mat	58	60	96.7%

Discussion

An overall accuracy of 97.6% for dataset 1 and 97.6% for dataset 2 demonstrates that for at least this dataset, the accuracy of the pedometer is generally accurate within 5%. Although it is recorded that both datasets are equally as accurate, it is still reasonable to assume dataset 2 should be more accurate. A reason for this is that dataset 1 will be contaminated by more sources of acceleration, such as phone swaying in the walker's pocket after a step is taken.

Additionally, the process of removing and placing the phone in the walker's pocket is detected as a step, as mentioned when discussing the results pertaining to dataset 1. To compensate for this, the observer will have to manually subtract the extra steps from the count total, adding a manual step to the overall process.

Limitations and Performance

The algorithm generally performed well against the training data however, a major shortcoming uncovered when developing the pedometer is properly calibrating the algorithm.

So far, the pedometer is only calibrated according to training data, however issues may arise if another walker's steps were to be counted. For example, if the walker were to take short, shuffling steps, the peaks of the walking signal may fall below the defined threshold to be counted as step, leading to a lower step count. Steps have been taken in development to ensure that the algorithm is as dynamic as possible, to account for these differences such as having the height threshold and cutoff frequencies (equations 2 and 3) dependent on the data parsed in. Additionally, the pedometer has only calibrated for steps taken in a straight line. This should be accounted for so long as equation 1 is utilised, however this has yet to be tested.

Dataset 1 also presented some issues with processing. As mentioned previously, extra steps were counted when moving the phone in and out of the pocket. This required the extra steps to be removed at the observer's discretion.

If possible, using datasets with step counts significantly higher than what has been used so far (e.g. 1000-10000 steps) would be beneficial in determining the pedometer's reliability as small step count datasets are more prone to be heavily influenced by random fluctuations in acceleration. Using different walkers would also be helpful.

Conclusion

A pedometer was developed using the multi-axis accelerometer pre-equipped in everyday smartphones and processing the accelerometer data using MATLAB's inbuilt functions. The developed pedometer works as attended at least according to the training data and returns the counted steps with a reasonable accuracy. However, further testing is required to fully gauge the pedometer's accuracy.

Bibliography

1. au.mathworks.com. (n.d.). *Butterworth filter design - MATLAB butter - MathWorks Australia*. [online] Available at: https://au.mathworks.com/help/signal/ref/butter.html.

2. au.mathworks.com. (n.d.). *Fast Fourier transform - MATLAB fft - MathWorks Australia*. [online] Available at: https://au.mathworks.com/help/matlab/ref/fft.html.

3. au.mathworks.com. (n.d.). *Find local maxima - MATLAB findpeaks - MathWorks Australia*. [online] Available at: https://au.mathworks.com/help/signal/ref/findpeaks.html.

4. au.mathworks.com. (n.d.). *Shift zero-frequency component to center of spectrum - MATLAB fftshift - MathWorks Australia*. [online] Available at: https://au.mathworks.com/help/matlab/ref/fftshift.html.

5. Electronics Tutorials (2013). *Butterworth Filter Design and Low Pass Butterworth Filters*. [online] Basic Electronics Tutorials. Available at: https://www.electronics-tutorials.ws/filter/filter_8.html.

6. WhatIs.com. (n.d.). *What is Nyquist Theorem? - Definition from WhatIs.com*. [online] Available at: https://www.techtarget.com/whatis/definition/Nyquist-Theorem.

7. wonderopolis.org. (n.d.). *How Does a Pedometer Work? | Wonderopolis*. [online] Available at: https://wonderopolis.org/wonder/how-does-a-pedometer-work [Accessed 3 Apr. 2023].