Step Counting Using In-built Sensors in Smartphones

Ayush Anand

Third year Undergraduate, Computer Science and Engineering, Indian Institute of Technology, Gandhinagar.
ayush_anand@iitgn.ac.in

In this paper, we look at how we can use the inbuilt sensors in smartphones to estimate the step count of the user. We use the accelerometer sensor for the same, although the same task could be achieved using other sensors. We try to find the periodicity in these sensor values and try to estimate the step count, either manually or using advanced filtering through autocorrelation. We then compare our results with the ground truth and other baselines, and look at how different factors like orientation, location and frequency affect the step count.

Additional Keywords and Phrases: Sensors, Ubiquitous Computing, Feature Sensitivity, Frequency sensitivity, Smartphones

1 INTRODUCTION

Step count is an important measure of physical activity provided by many well-known smartphone applications. These apps try to reduce the errors in measurement due to factors like continuously changing orientation and location. In this paper, we will try to use the inbuilt sensors of the smartphone to measure the step count of user by exploring certain algorithms, and compare the results with the step count provided by the step counting apps, and the ground truth.

Studies have shown that the linear acceleration values along different axes changes continuously while the user is walking. We can use the trend in these changes of acceleration values to estimate the step count. The accelerometer sensor of the smartphone gives us changes in linear acceleration along all the three axes X, Y and Z. However, we will try to perform all calculations on the Z axis data as some related works have shown that the values along X might not be reliable as it is affected by even a slight shaking/abrupt hand motions on the user's part. We consider one regular fluctuation of the Z axis to represent one step.

Thus, our task boils down to finding the periodicity in the Z-acceleration values (we will look at some other sensors in the later part of the paper). We can do this by using **Normalized Auto-correlation**. We have performed all calculations in MATLAB and used the MATLAB mobile app to collect the sensor data.

2. RELATED WORK

There has been some very promising work in this field. Some papers use a threshold setting; wherein the idea is to count a step for every time the sensor data exceeds a predefined threshold. Another group of paper is based on peak detection, and uses certain filters to narrow down the peaks of the sensor data to only ones that represent a potential step, and then one step is counted for every valley between two peaks. In paper [1], the authors have used autocorrelation to count steps, and our approach is based on similar lines.

3. DATA COLLECTION

We performed the experiment to collect sensory data on 5 participants, which consisted of both male and female candidates. The procedure was as follows: The participant walks a pre-decided number of steps (ranging between 10-15), and is directed to start and stop the sensor reading exactly when he starts and stops. A stopwatch records the time for which the user walked, and is later used in combination with average user

walking speed to form a weak baseline. The sensor data is automatically uploaded to MATLAB drive and is used for analysis later. The participant is then directed to hold the phone in different positions as well as in different orientations, and the corresponding readings are taken. Finally, the sampling rate is also experimented with.

4. DATA ANALYSIS

We will look at the data analysis for different subparts of the experiments separately:

4.1 Main Experiment

In the main experiment, the participant holds the phone in a vertical orientation, a few feet away from his body in an upright manner. This is demonstrated in the figure 1 below. The sensor used is the accelerometer. In figure 2, we show the plot of sensor data vs time. The data is obtained at the highest sampling rate initially, which is 100 Hz. We can see the plot is very irregular and we need to do some tuning to distinguish the peaks and valleys.

This experiment was performed with 6 different participants the dataset for which can be found here. At first, we apply a fairly simple approach, and just count the different peaks for the normalized plot using MATLAB's inbuilt *findpeaks* function. To normalize the sensor values, we calculate the magnitude of accelerometer data to convert them into a scalar. We then subtract the mean from the data to remove any constant effects like gravity. Figure 3 then shows the plot obtained with peaks marked in red triangles, and we count these to obtain the number of steps. We exclude peaks that are very close to each other.

Figure 1: Participant walking with phone held in a vertical orientation at some distance from the body

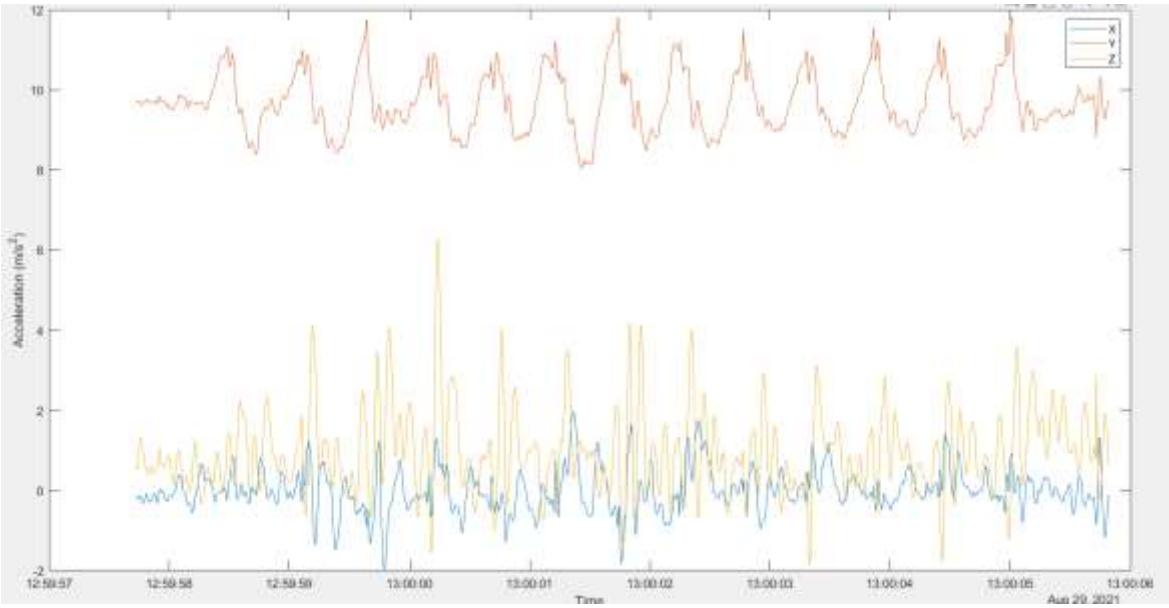

Figure 2: Raw sensor data of accelerometer vs time plot

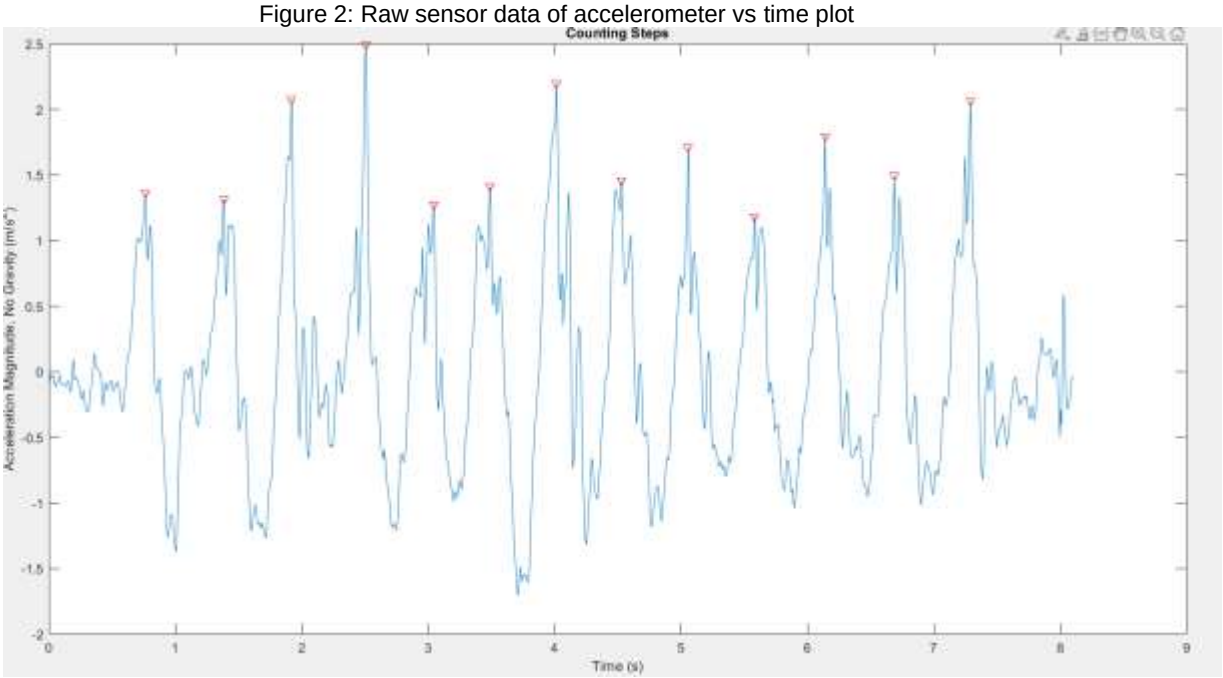

Figure 3: Normalized accelerometer sensor data with peaks marked in red

In the next section, we show an advanced filtering method that improves on this one; to find the peaks we find the periodicity of the curve using auto-correlation. But before that, we will first describe other follow up experiments we performed.

4.2 Location Sensitivity

To get a better idea of how accelerometer data varies with location of the mobile phone on the body, we asked the participant to hold the phone in the same orientation, but place it inside his/her trouser pockets. The steps followed to analyze the data are exactly the same as in the last experiment. However, we will see in the evaluation section that this experiment has larger RMSE due to prominent human errors of taking the phone out of pocket and then turning off sensor.

Figure 4

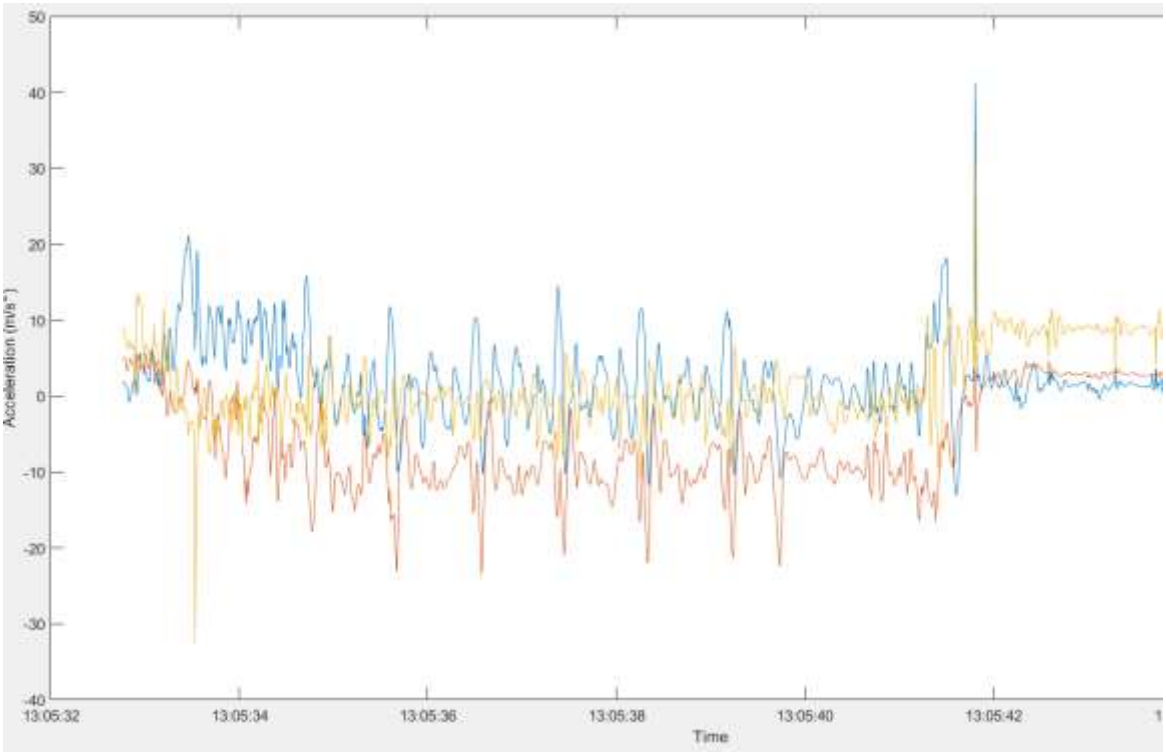

Figure 5: Raw sensor data of accelerometer vs time plot

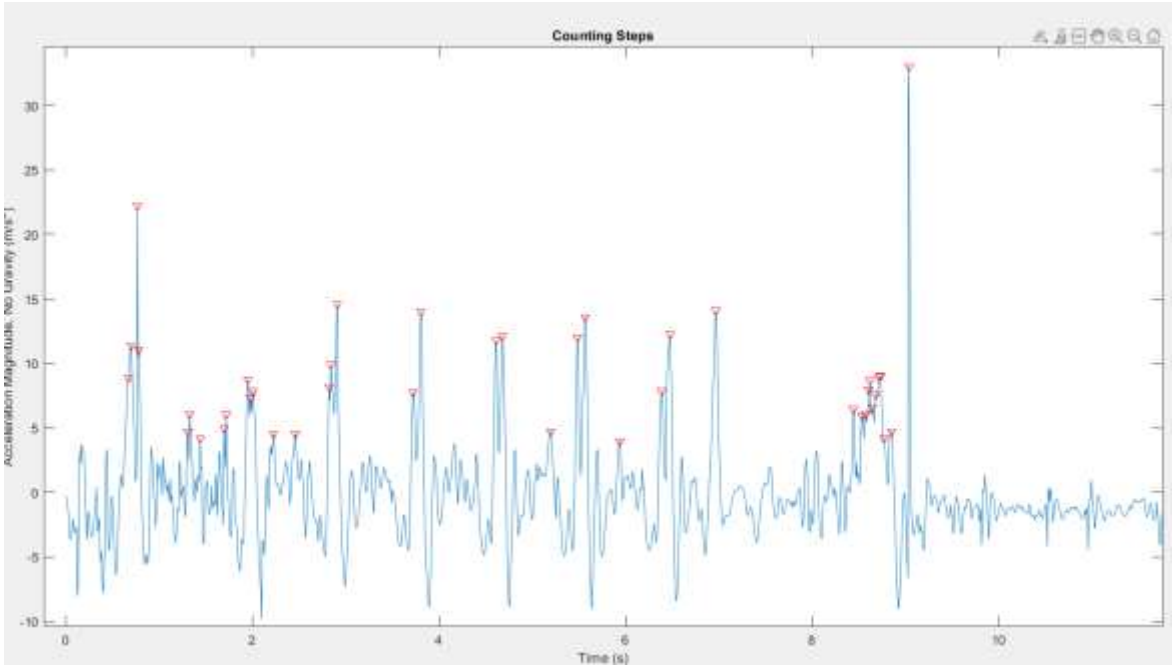

Figure 6: Normalized accelerometer sensor data with peaks marked in red

4.3 Orientation Sensitivity

The next part of the experiment involved changing the orientation of the phone while keeping it at a constant location on the body. The rest of the experiment was exactly same as before, wherein the participant walks a certain number of pre-decided steps with the sensor on. The following figures show how the experiment is conducted, and how the different plots look like.

Figure 7

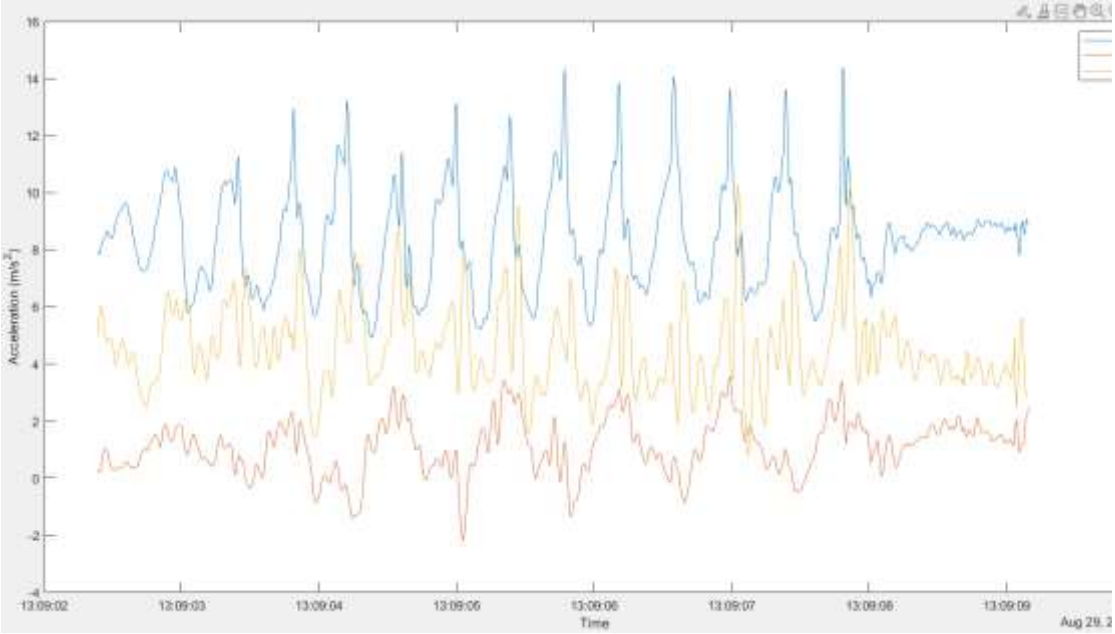

Figure 5: Raw sensor data of accelerometer vs time plot

.

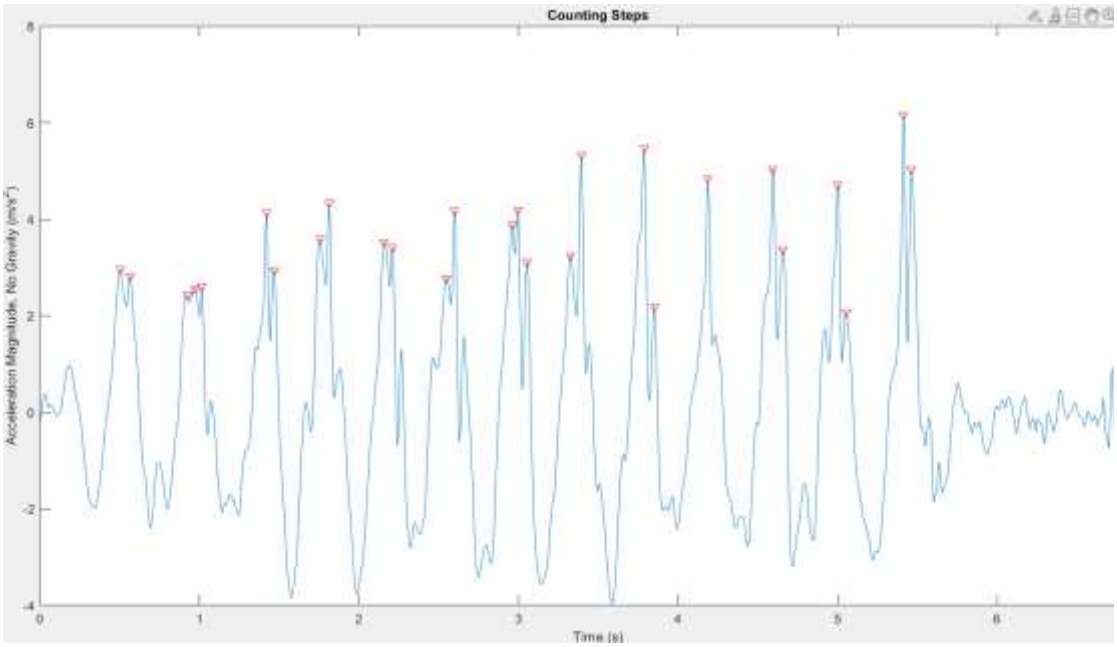

Figure 9: Normalized accelerometer sensor data with peaks marked in red

4,4 Frequency Sensitivity

In this experiment we down sample the frequency to 50 Hz and 10 Hz and take similar readings as in the main experiment. We then note the error and compare it with that in the main experiment. The plots pertaining to the same are shown below.

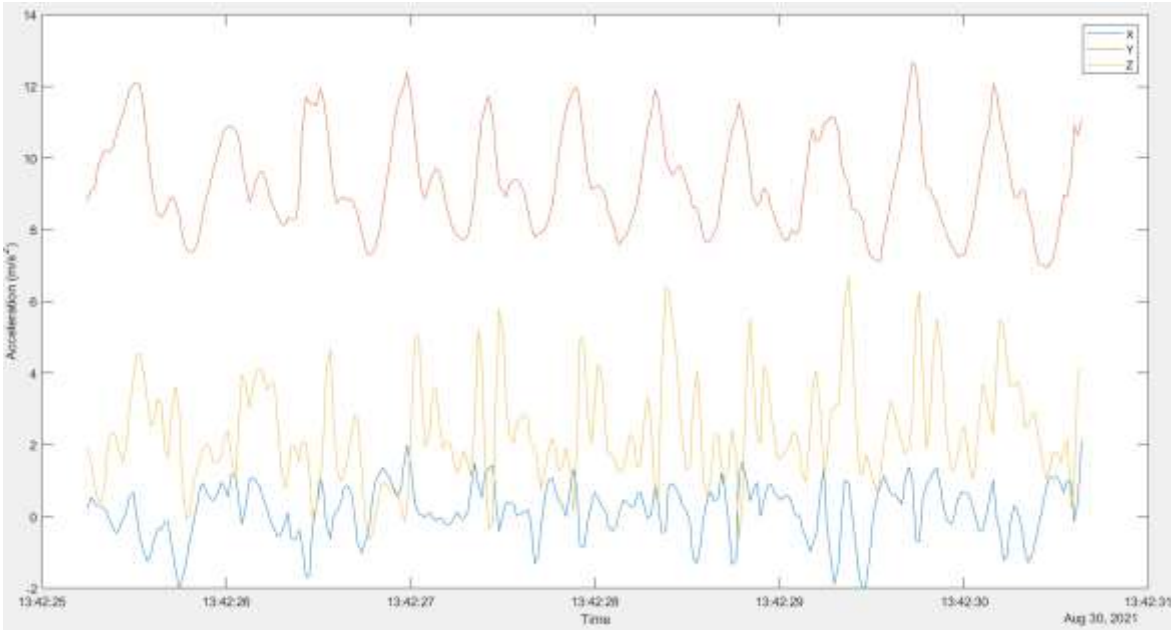

Figure 10: Raw sensor data of accelerometer vs time plot

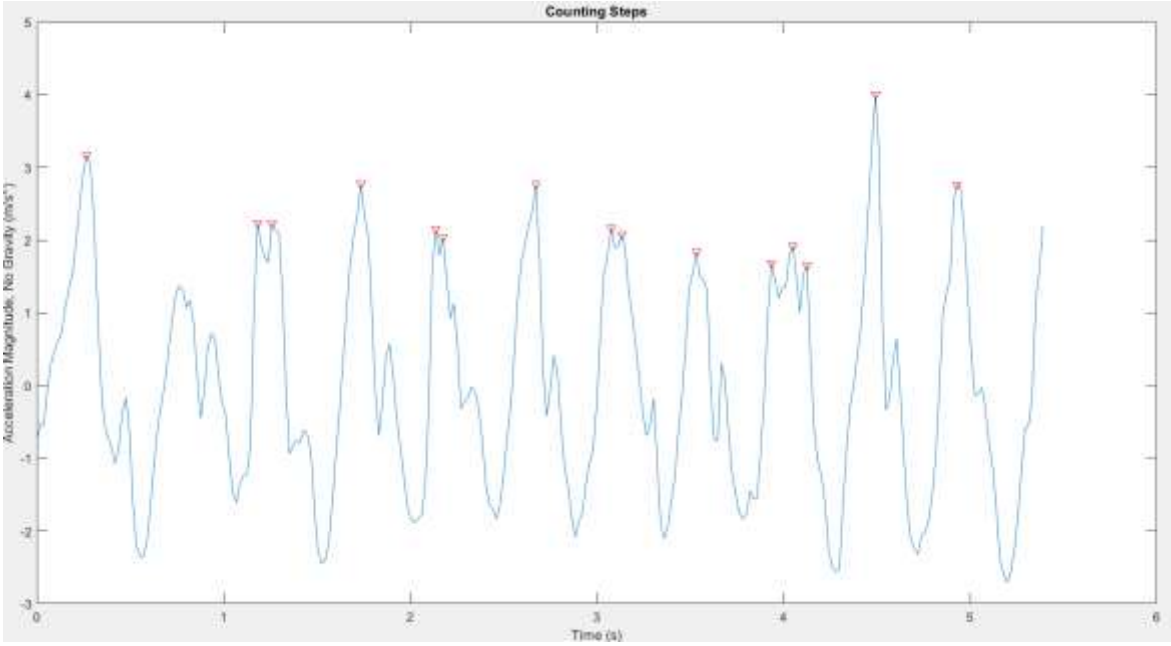

Figure 11: Normalized accelerometer sensor data with peaks marked in red

4.5 Feature Sensitivity

In this part of the experiment, we perform an ablation study by varying different hyperparameters like Minimum peak height, minimum peak prominence and so on and report the error. The following figures show the variation in the peaks detected by the *findPeaks* function at different values of these hyperparameters.

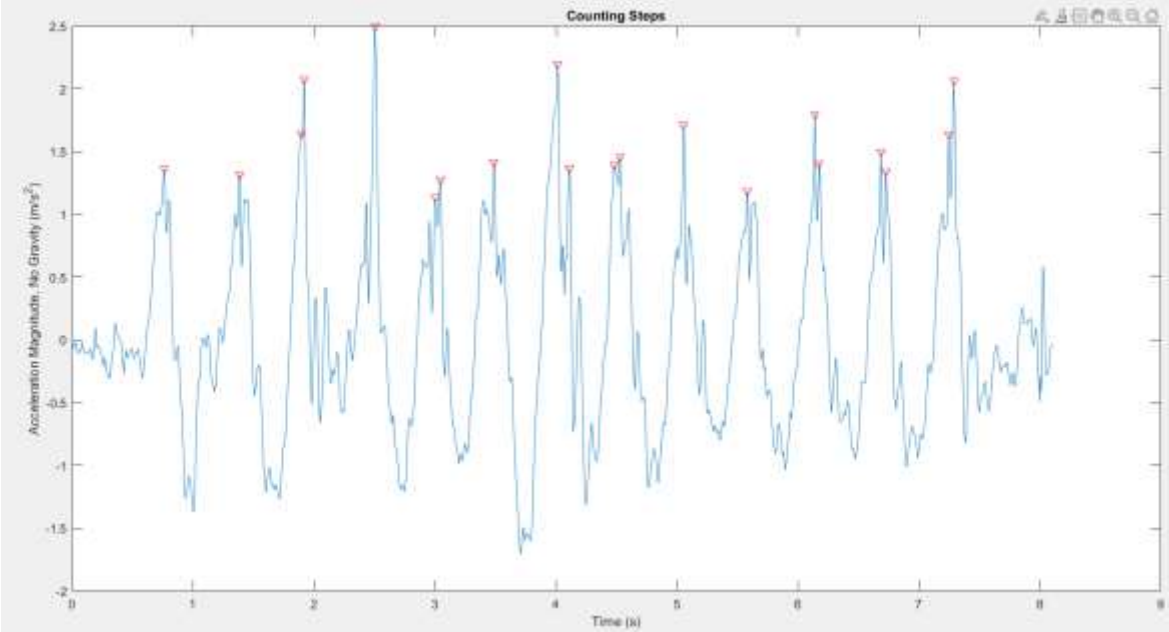

Fig 12: Minimum Peak height=1.5*standard deviation (Nmag)

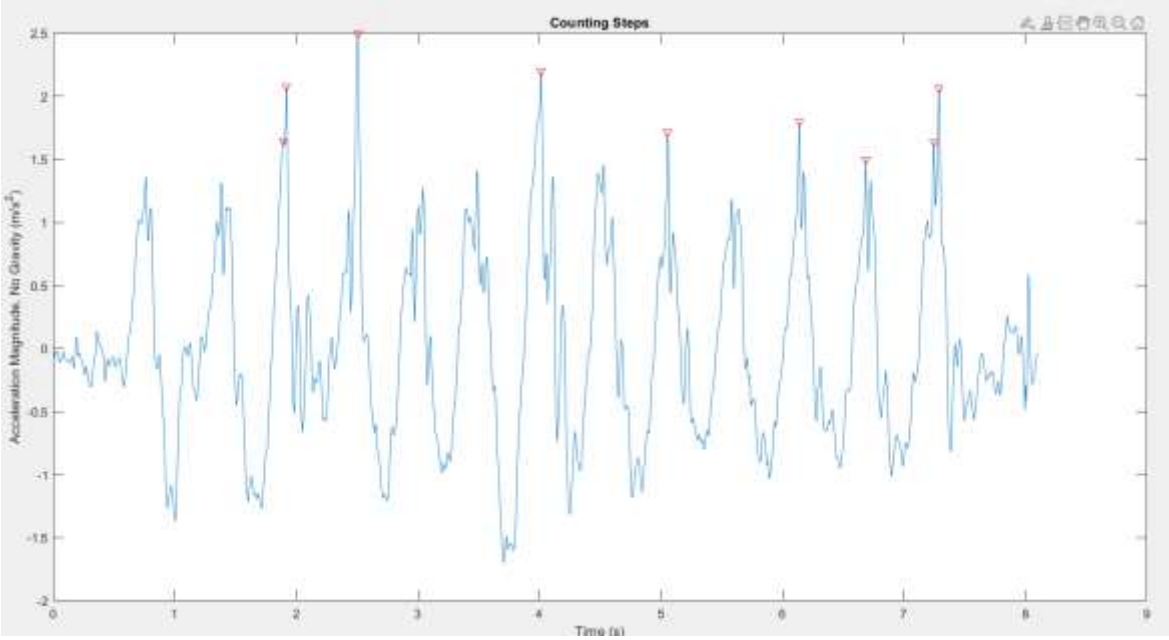

Fig 13: Minimum Peak Height= 2* Standard Deviation (Nmag)

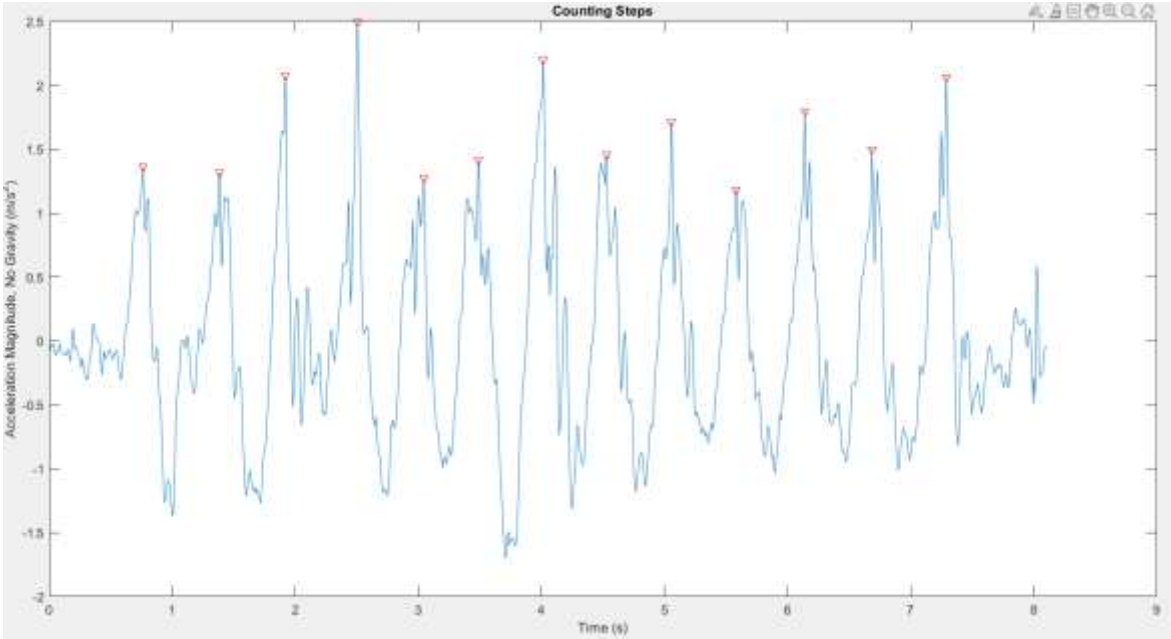

Fig 14: Minimum Peak Prominence=1.5

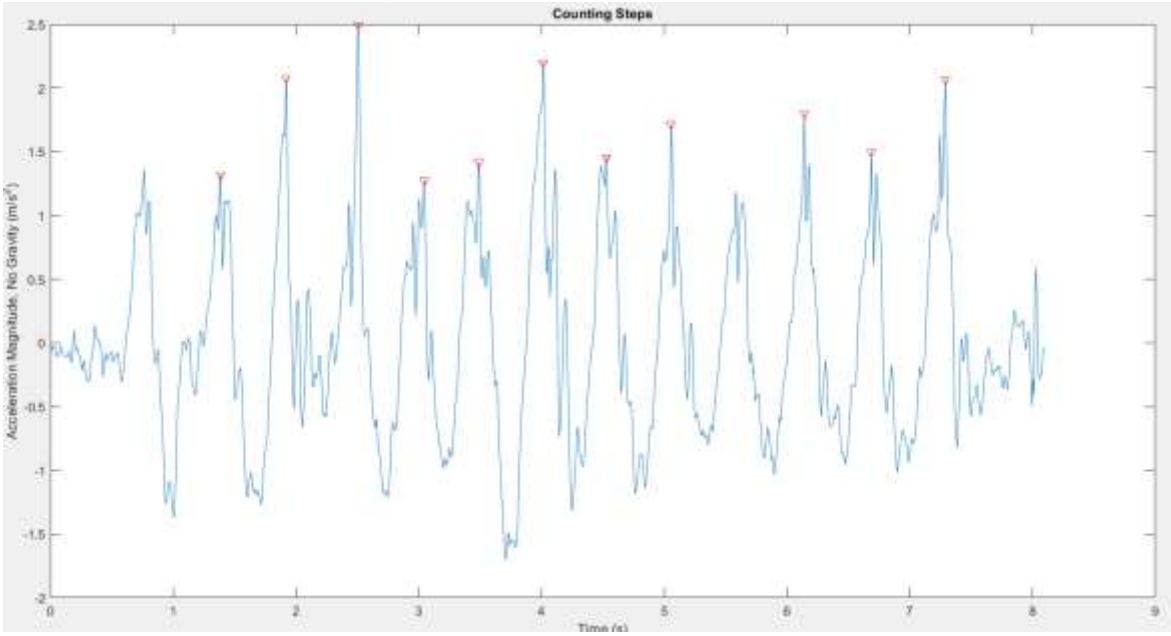

Fig 15: Minimum Peak Prominence =2

We will look at how the error varies with different hyperparameters in the Evaluation section.

5. ADVANCED FILTERING:

So far, we have looked at a relatively simple method to count steps, which is to look at the peaks manually and decide how many steps have been walked. In this section, we explore an advanced filtering method using autocorrelation, as proposed in this paper. The basic idea is to find the periodicity in the sensor data using autocorrelation, and then we can a step is counted for every (*Time elapsed)/2* samples. The following sections describe the algorithm and the code in more detail.

5.1 Algorithm

The following is the algorithm used for the method described above

ALGORITHM
X= Acceleration.X
Y=Acceleration.Y
Z=Acceleration.Z
Magnitude of Acceleration=Sq rt(X^2+Y^2+Z^2)
Normalized magnitude=Acceleration-Mean of Acceleration
Find autocorrelation
Find the short and the long period
Number of steps =Time elapsed*2/Long period

5.2 COMPUTER CODE

Here is the MATLAB code for the above algorithm

```
load('Siddhi_13_9.34_16(1).mat');
tt=9.34;
x=Acceleration.X;
y=Acceleration.Y;
z=Acceleration.Z;

mag = sqrt(sum(x.^2 + y.^2 + z.^2, 2));
Nmag = mag - mean(mag);
fs=100;
[autocor, lags] = xcorr(Nmag,fs,'coeff');

plot(lags/fs,autocor);
xlabel('Lag')
ylabel('Autocorrelation');
axis([-21 21 -0.4 1.1])

[pksh,lcsh] = findpeaks(autocor);
short = mean(diff(lcsh))/fs;

[pklg,lclg] =
findpeaks(autocor,'MinPeakDistance',ceil(short)*fs,'MinPeakheight',std(Nmag));
long = mean(lclg)/fs;

step_count=2*tt/long;
```

6 EVALUATIONS

We carried out an extensive evaluation of our experiments and compared the results with

1) The ground truth, which is the number of steps the participant walked according to him/her
2) Weak Baseline: The average human walking speed in steps/sec multiplied by the time recorded for which the user walked
3) Strong Baseline: The step count shown according to the step counting application on the phone.

The complete analysis of the RMSE and results obtained for all the experiments can be found in the github repository for this paper. In general, we noticed that our algorithm performs better in comparison to the weak baseline, and is close but not quite at par with the strong baseline. Also, changing the orientation and location

affect the results, since we are not performing a data pre-processing step (excluding the normalized magnitude step) to take into account these factors.

Some general observations that we made are: 1) Results are better in general when sampled at a lower frequency 2) Results are better when the Minimum Peak height is taken to be standard deviation of Normalized magnitude of acceleration and 3) Although in the evaluations we see that just counting the peaks manually might give a slightly better results as compared to using autocorrelation, autocorrelation does not require manual effort and hence is a better solution.

CONCLUSION:

In this paper we looked at how to utilize sensor data from smartphone to estimate step count of the user. We looked at a simple method to achieve this, which is to manually count the peaks in the sensor data, and then also looked at an Advanced filtering method using Autocorrelation. We compared our results with the step count recorded by the step counting app on the smartphone and the weak and strong baselines mentioned above. Finally, we varied the conditions of the experiment and also performed an ablation study.

REFERENCES

[1] Rai, Anshul, Krishna Kant Chintalapudi, Venkata N. Padmanabhan, and Rijurekha Sen. "Zee: Zero-effort crowdsourcing for indoor localization." In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pp. 293-304. 2012

[2] Pan, Meng-Shiuan, and Hsueh-Wei Lin. "A step counting algorithm for smartphone users: Design and implementation." *IEEE Sensors Journal* 15, no. 4 (2014): 2296-2305.

[3] https://in.mathworks.com/help/signal/ug/find-periodicity-using-autocorrelation.html

[4] https://in.mathworks.com/help/matlabmobile_android/ug/counting-steps-by-capturing-acceleration-data.html

[5] Brajdic, Agata, and Robert Harle. "Walk detection and step counting on unconstrained smartphones." In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 225-234. 2013