Multidimensional Sensing Techniques Assignment 1 – Step counter

Md Mahbub Islam – ID: 2202978

https://github.com/it-teaching-abo-akademi/assignment-1---step-counting-Md-Mahbub-Islam/

Introduction

I will be using an accelerometer sensor with MATLAB on my smartphone. After reading the data from the sensor, I will use MATLAB to shape the data structure such that it can be used by a python application.

Then I would like to visualize and parse the data to see any patterns that may exist. From the data I count and graph, I will be able to see how many steps I took. There will be two kind of function. One is a simple step counter function which just counts the number of times the acceleration values goes above a certain threshold. Second is a more sophisticated function, which uses dynamic thresholding. This means that the threshold is not fixed, but rather it is based on the previous values of acceleration. This should give a more accurate result, but it will be more complicated to code.

Data Collection

Data is first collected from MATLAB. Using mobile sensors.

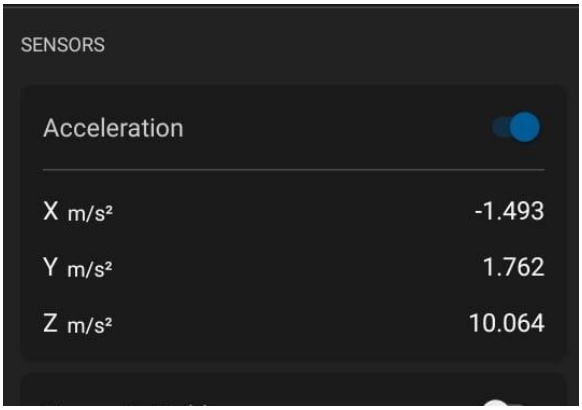


Figure 1: Matlab sensors

After doing necessary movement the data is saved automatically to MATLAB drive from where it is downloaded.

We load it with

load(“name.mat”)

Change into a matrix format

temp = [[1: length(Acceleration.Timestamp)]', Acceleration{:, :}]

csvwrite("out.csv", temp)

Gives us a data like this.

	A	B	C	D	E
1	1	-0.11055	2.8403	8.3556	
2	2	-0.27344	2.8355	9.424	
3	3	-0.48664	2.7709	10.044	
4	4	-0.58007	2.7038	10.147	
5	5	-0.38843	2.7565	9.8863	
6	6	0.10744	2.8667	9.6515	
7	7	-0.04587	2.881	9.5006	
8	8	-0.31177	2.8307	9.0287	
9	9	-0.41238	2.8188	8.8371	
10	10	-0.51539	2.8858	8.8826	
11	11	-0.38843	2.8331	8.8371	

Now It is ready to be manipulated in python

I used several data. For simplicity I used walk1.csv data where only walk data was available.

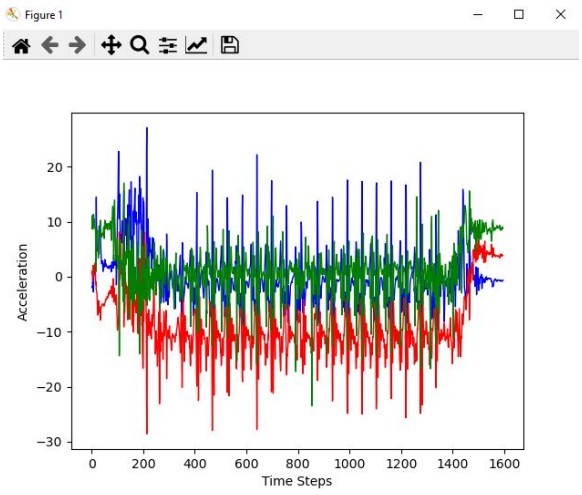
I read the data using this function

```
#Function to read the data from the log file
def read_data(filename):
    timestamps = []
    x_array = []
    y_array = []
    z_array = []
    with open(filename) as f:
        for line in f:
            data = line.split(',')
            timestamps.append(int(data[0]))
            x_array.append(float(data[1]))
            y_array.append(float(data[2]))
            z_array.append(float(data[3]))

    return timestamps, x_array, y_array, z_array
```

Which returns 4 lists with all the necessary data that we need.

The following part is not necessary for the exercise but I immediately plot the data to get a general feel of how the data looks like.



It shows data in all three axis. Next check data function basically checks if all list have same number of element. In this case it was, no error. On to next step.

As we have seen the accelerometer data can be in any of the 3 spatial coordinate system thus a way is needed to find out acceleration regardless of phone orientation and direction.

A normalization function is helpfully given in the code that gives use the magnitude.

$$O_i = (\sum_{n=1}^{N} A_{i,n}{}^2)^{1/2}$$

the root of the squared sum of the elements of each row [1]

```
#magnitude array calculation
m_arr = []
for i, x in enumerate(x_array):
    m_arr.append(magnitude(x_array[i],y_array[i],z_array[i]))
```

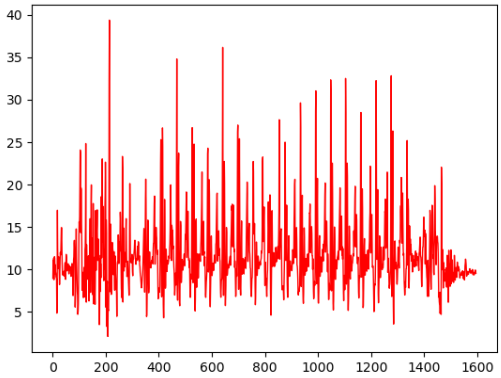


Figure 2: Magnitude graph

We find a magnitude data like this. It is 1 dimensional and we can ignore phone orientation which is better.

Now mean of all the values is calculated to get a threshold point.

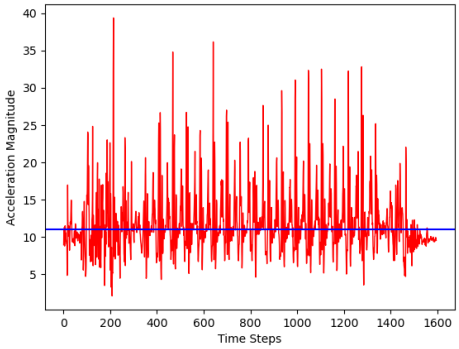


Figure 3: Threshold line through M

Blue line denotes the mean that was used as threshold point. Since a spike goes through this line twice at least so we count only when the spike is going up but not going down.

Next data is prepared for a step vs magnitude graph. Using mean as threshold.

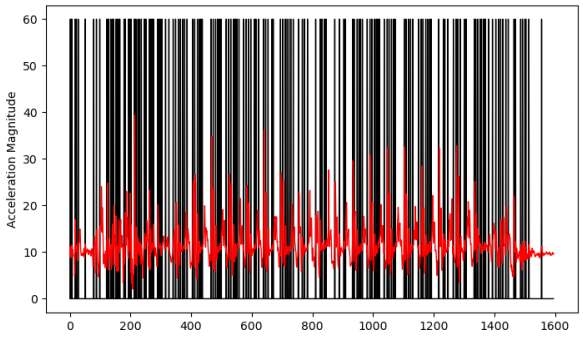


Figure 4: Steps and Magnitude

We get this final graph. Which is messy and probably over counted steps and added random noises as step. So I increase the threshold by x2 which gives me this

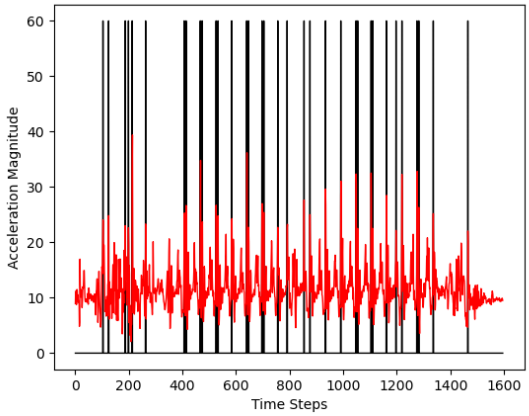


Figure 5: Steps and Magnitude rev.

This seems much more reasonable however there is better way to get the threshold as we shall see later.

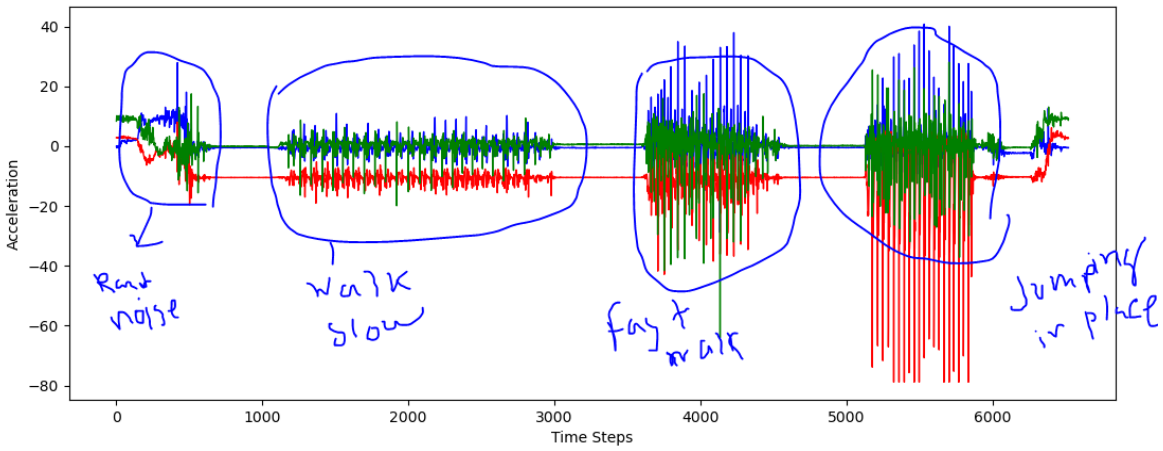
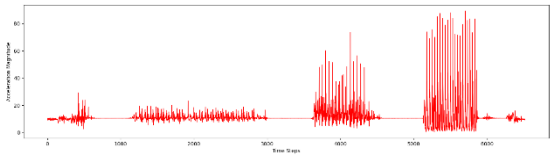


Figure 6: Data with various type of movements

The next data is bit more complex and It has various types of movement. The first and last part is just noise that generated when phone was put in pocket. Next bits have following movements “Slow Walk”, “Fast Walk”, “Jumping” in between are resting periods.

Data is different so a simple threshold may not work. What we can do is find the peaks of all the graph. Also we can use filter to smooth out the data



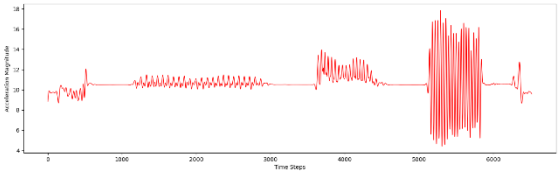


Figure 7: Low pass filter

As we can see it’s much more smoother when passed through the low pass filter. I used scipy library to do it quickly.

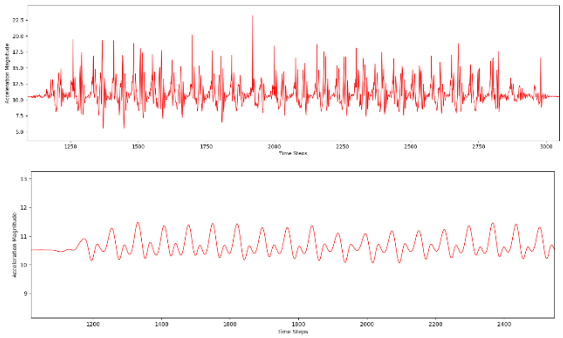


Figure 8: Low pass filter zoomed

Zoomed up on slow walk area the option is even cleared.

Tangent line when the peak is reached is 0. We can count everytime it reaches zero. I used a python library to find the peaks above a certain value. This is better than fixed threshold.

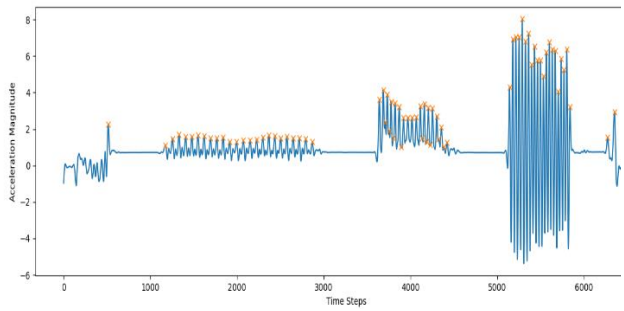


Figure 9: Dynamic Peak regardless of movement type.

Conclusion:

Step counting is difficult and more refinement is needed. From this I got ideas about how motion capture devices that can work for games and animation can be build. I learned how filters can be a very helpful tool to get useful information from noisy data.