Step Tracker Implementation

Matjaz Zupancic Muc

University of Ljubljana, Faculty of Computer and Information Science
Večna pot 113, SI-1000 Ljubljana, Slovenia
Email: mm1706@student.uni-lj.si

***Abstract*—In these seminar a step tracker using ESP8266 microcontroller and MPU9250 sensor was implemented. Source code: https://github.com/Matjaz12/Step-Tracker.**

I. Introduction

Step detection algorithm using acceleration signals in all three dimension is implemented. Acceleration magnitude is computed and raw signal is filtered using a first order low-pass filter and a second order butterworth filter. Step detection is performed by looking for a crossing of the signal and a dynamic threshold.

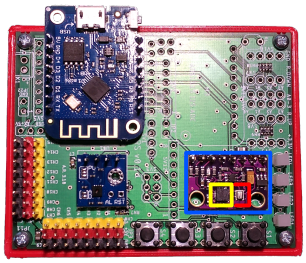

Fig. 1. ESP8266 and MPU9250 board

December 27, 2012

II. Methods

A. Acceleration readings

The accelerometer is sampled at a frequency of $f = 100Hz$. Each sample consists of three acceleration readings $\{a_x, a_y, a_z\}$. Before we start reading from the sensor we compute the offset acceleration (offset is computed by reading 1000 samples and computing the average in a steady position). We obtain values $\{a_x^o, a_y^o, a_z^o\}$. These values are than used to calibrate the sensor (Equation 1).

$$\{a_x - a_x^o, a_y - a_y^o, a_z - a_z^o\} \tag{1}$$

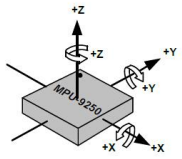

Fig. 2. Sensor axis

B. Acceleration magnitude

Sensor can be in an arbitrary position, therefore we convert the 3 signals into 1 signal by the magnitude of the vector:

$$a = \sqrt{(a_x - a_x^0)^2 + (a_y - a_y^0)^2 + (a_z - a_z^0)^2} \tag{2}$$

C. Low pass filter

1) First order low-pass filter: Accelerometers are subjects to noise from a variety of sources (mechanical, electrical, thermal, etc.). Therefore we have to remove noise at frequencies that are not related to human walking or running. [GWS+19] proposes a finite impulse response (FIR) low-pass filter with a cut-off frequency $f_c = 3Hz$. They state that this threshold still allows for a variety of walking speeds. In (Equation 2) n is a discrete step, $a_f[n-1]$ is the previous filtered acceleration, $a[n]$ is the current acceleration, $a_f[n-1]$ is the previous acceleration. The set of constants $\{\alpha,\ \beta_0,\ \beta_1\}$ has to be appropriately selected to unsure the cut-off frequency of 3Hz. To compute constants, script [x] was used. We simply have to provide the cut-off frequency f_c and a sensor sample frequency f. Fig 3. shows the filtered signal a_f and the raw signal a. We can see that noise is reduced, even when sensor moves quickly.

$$a_f[n] = \alpha \cdot a_f[n-1] + \beta_0 \cdot a[n] + \beta_1 \cdot a[n-1] \tag{3}$$

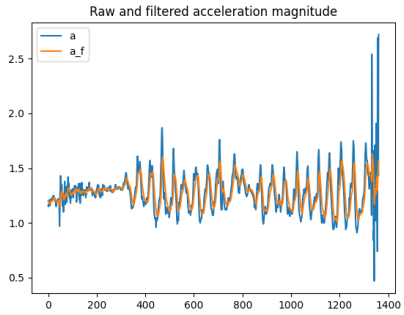

Fig. 3. Comparison of raw acceleration and low-pass filtered acceleration magnitude

2) Butterworth low-pass filter: Typically a low pass filter has wide transition band. Meaning that there exists a band of frequencies $f > f_c$ which are not canceled. The second order butterworth filter is used (Equation 4). It has a small transition band but is also relatively fast. Butterworth filter introduces more delay to the signal compared to the first order low pas filter (Fig. 4).

$$\begin{aligned} a_f[n] &= \alpha_0 \cdot a_f[n-1] + \alpha_1 \cdot a_f[n-2] \\ &+\beta_0 \cdot a[n] + \beta_1 \cdot a[n-1] + \beta_2 \cdot a[n-2] \end{aligned} \tag{4}$$

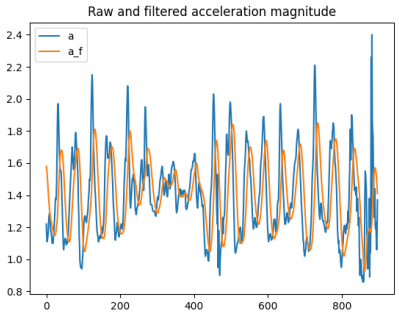

Fig. 4. Comparison of raw acceleration and butter-worth filtered acceleration magnitude

D. Step detection

In order to detect the peaks we use a method proposed in [Zha10] . Every 50 sensor readings we compute a minimal acceleration a_{min} and a maximal acceleration a_{max}, using these values we calculate the threshold a_{mean}. In order to not count noise peaks as steps, we compute a mean acceleration a_{noise}. A threshold is valid only when it is greater than a_{noise}. A step is defined as occuring when $a_{mean} > a_{noise}$ and $a_n < a_{mean} < a_{n-1}$. Fig. 5 shows the calculated threshold which is used to detect the step.

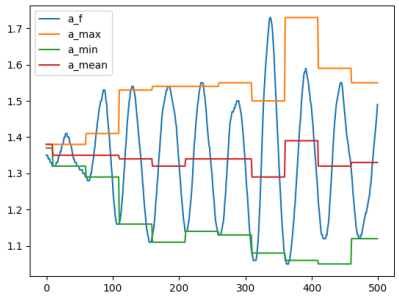

Fig. 5. min, max and mean acceleration

E. Server

Finally a simple server running on the microcontroller is implemented to visualize the number of steps taken. Server updates the step counter every 500ms. Using a browser client we can can access the UI at local-host address.

Fig. 6. Step tracker user interface

III. Results

Step tracker was tested by walking for 100 steps. At about an average walking pace, system detected 122 steps. In general the system tends to overcount the steps.The sensor was held in the upright position (such that the y-axis of the sensor is vertical). Experimenting with the tracker we realized that it also picks up rotations, this probably occures because we are computing the magnitude of the acceleration.

IV. Conclusion

We designed a simple step tracker. Tracker could be improved by using a more complex filter, which would cancel out additional noise. We could also amplifying the filtered acceleration signals (this way steps would be more pronounced). A better algorithm for peak detection could be used. One such algorithm is Z-score peak detection [vB14], but sophisticated algorithms require optimization methods to find optimal parameters. We also implemented a server to visualize the number of steps.

References

[GWS+19] Meng Gao, Haifeng Wu, Yong Shen, Xia Wang, and Yu Zeng. A peak detection algorithm adopting magnetic sensor signal for rail spike location in tamping machine. *Advances in Mechanical Engineering*, 11(11):1687814019891570, 2019.

[vB14] JPG van Brakel. Robust peak detection algorithm (using z-scores). *Stack Overflow: New York, NY, USA*, 2014.

[Zha10] Neil Zhao. Full-featured pedometer design realized with 3-axis digital accelerometer. *Analog Dialogue*, 44(06):1–5, 2010.