DT00BQ89

Multidimensional Sensing Techniques

Lab exercise 1 - Step counter

Petteri Laitinen ÅA21965

Harri Kemppainen ÅA1801751

23.9.2019

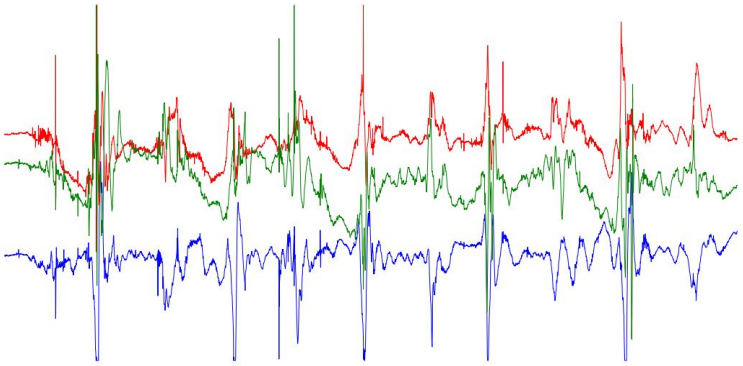
Introduction

The purpose of this experiment was to learn to collect data from sensors, visualize it and parse it. In the experiment, we used an accelerometer that was connected to an Arduino board and that further connected to a USB port of Linux operated laptop. The goal was to generate an algorithm that operates on the accelerometer data and counts number of steps when a person walks with the accelerometer in the pocket.

Classroom lab

Situation in classroom was a bit hectic. There was a constant feel of hurry and therefore we didn't really stop and think what we were doing. After attaching USB to laptop we checked from /dev/ folder which device was added ('l /dev/tty*') and changed the given script accordingly. It was the only usb device on laptop so it used port number 0 (/dev/ttyACM0). We forgot to count steps on the first recording. Second recording was appended to the previous file and was therefore unusable. On the third time we got nice and clean recording on ten plus one steps.

The time series data from classroom data is illustrated in the graph below. (copy paste from Jupyter to Google Docs does unfortunately not support legends and titles)



Signal processing methods

Once the data was uploaded to the computer, we modified the Python code we were given to visualize data and try out different ways to calculate steps.

For this, we defined function count_steps:

count_steps(timestamps, x_arr, y_arr, z_arr,option,window)

where

- inputs are timestamps, x,y,z-measurements as time series arrays,
- option is the method how to aggregate the x,y and z-measurements, and
- window is minimum number of observations between two consecutive steps

What comes to the window (W), it was used to recognize maximum of a time series. It was used in two ways:

1. to check if the current value X(t) was maximum of [X(t-W), X(t-W+1),...,X(t+W)]
    a. if this was the case, the time point t was eligible to be a peak corresponding to a step
2. to check that there was not already a recognized step right before the current one (this could happen for example if there were two consecutive equal observations).
    a. if there was a step within window backwards, the current peak was not accepted as step.

The options to process the x-, y- and z-signals were following:

1: Signal with highest variation (x or y or z)
$(u(t) \mid u \in \{x,y,z\} \wedge \forall v \in \{x,y,z\} : Var(u) \leq Var(v))$
2: Sum of signals $x(t)+y(t)+z(t)$

3: Gradient of absolute of sum of signals $\frac{d}{dt}|x(t)+y(t)+z(t)|$

4: Magnitude $\sqrt{x(t)^2+y(t)^2+z(t)^2}$

5: Absolute value of gradient of magnitude $\left|\frac{d}{dt}\sqrt{x(t)^2+y(t)^2+z(t)^2}\right|$

6: Magnitude of gradients $\sqrt{\left(\frac{dx(t)}{dt}\right)^2+\left(\frac{dy(t)}{dt}\right)^2+\left(\frac{dz(t)}{dt}\right)^2}$

7: Trapezoid integral of sum of signals $\int_{t=T}^{T+1}(x(t)+y(t)+z(t))$

The use of gradient is motivated by our interest in change of signal (At every step, muscles are used to create vertical force that is higher than gravitational force, and horizontal force that brings the walker forward).

We also tested the trapezoid integration, and realized soon that it was not a good choice here. In walking, the forces are mostly applied into certain direction, and hence the accumulated values just kept on growing.

The full code (in Jupyter python code) for signal processing is found here

Results

Results using the data recorded in lab

Our recording had walk of 11 steps.

We first tested out 7 different methods (options 1-7), and got 10 or 11 steps on all methods except the trapezoid integral that only “found” one step. An example of “derived” signal and recognized steps are given in the graph below.

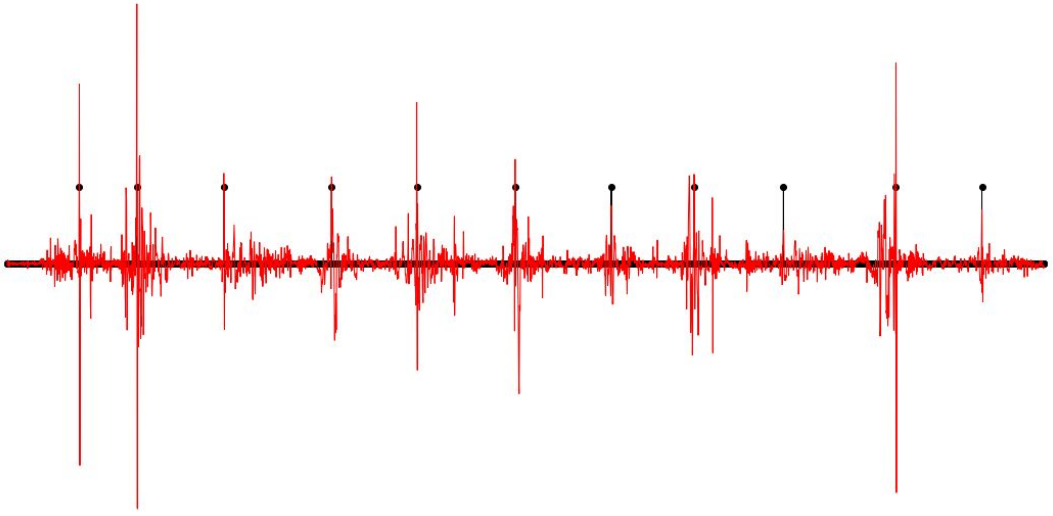


Figure: Aggregated signal from lab, and detected steps (black dots) with option 3: gradient of absolute of sum of signals

Results using the given test data

After having good results with our lab data set, we applied the methods 1-6 (method 7 failed so we did not spend time on that) to given test data sets “easy”, “medium” and “very hard”. We got quite stable results for each data set: easy 42-47 steps, medium 52-56 steps, very hard 81-92 steps.

We then tested to add some moving average element (option 8 and 9 with different moving average windows), but that did not improve the results.

Results of our algorithms are shown in the following table:

Results using minimum window between steps = 200 units		Data				Deviance (how much more steps were found in maximum compared to minimum)
		Lab data	Test data easy	Test data medium	Test data very hard	
True number of steps		11	Not known	Not known	Not known	
Method	1: Signal with highest variation	11	42	52	81	93 %
	2: Sum of signals	10	42	53	92	119 %
	3: Gradient of absolute of sum of signals	11	47	53	85	81 %
	4: Magnitude (2-norm)	11	43	56	92	114 %
	5: Absolute value of gradient of magnitude	11	43	52	87	102 %
	6: Magnitude of gradients	11	47	52	89	89 %
	7: Trapezoid integral of sum of signals	1	10	1	15	1400 %
	8: Gradient of smoothed magnitude of signals (smoothing window 25)	11	43	53	93	116 %
	8: Gradient of smoothed magnitude of signals (smoothing window 50)	11	46	58	93	102 %
	9: Smoothed abs of Gradient of smoothed magnitude of signals (smoothing window 50 both times)	8	30	43	57	90 %
	9: Smoothed abs of Gradient of smoothed magnitude of signals (first smoothing window 50, second 20)	9	33	47	70	112 %

The best model seemed to be option 3: taking the gradient of absolute of sum of signals. It found correct number of steps (11) from the lab data.
In addition, if we understood the assignment correctly, there should be equal number of steps in test data sets “easy”, “medium” and “very hard”. Option 3 estimated 47 steps for easy data, 53 for medium data and 85 for very hard data. The difference between these was smallest out of all models we tested. However, we never reached 10% error (if the number of steps really was equal in the test data sets.

Reflections

What we will do different next time:

- Read the instructions thoroughly before the lab
    - now we recorded data of roughly 10 steps, corresponding to approx 8 seconds of data. In the instructions it was told to record roughly 1 minute of walking - longer data could have made it easier to verify the algorithm
- Check the sampling rate or “calibrate” the measurement by using a stopwatch or similar
    - now we were able to measure data, but because of challenges with the Python code we were not able to visualize it in the lab room. That lead to situation where we left the lab and started building the Python code. When we were ready with the code and got the graphs, we were unsure what the measurement frequency was. That could easily have been verified by using external stopwatch during the lab.

- Be consistent using norms
    - Now we tested sums of signals, and absolute of sums, and 2-norm but never tested 1-norm properly

That said, we (in addition) learned

- How to read in data from one sensor using Arduino board, Linux computer and Python script
- How difficult it is to invent a robust solution for an easy algorithm like step detection.
- Take your time and concentrate on what you are doing. Don’t get sucked into hustle and bustle around you