Derivation of correction factors for sensor axis-specific calibration error in Autocalibration of accelerometer data studies

A solution by

Athul Mithran

October 4, 2021

1 Introduction

As a part of the interview procedure, a solution based on *Autocalibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents* by `Vincent T. van Hees` `, Zhou Fang` *et al* [1] was worked out for the problem on auto calibration as provided. The content of the paper was studied with the aim of understanding the real world applications as well as the procedures to follow so that an accurate and functional python script could be written. This problem helped to provide some insight into the future works corresponding to this position and a rough estimate of the time-commitment required. They also act as a source for aspiring PhD and master students to sort out various questions in their mind before embarking on the research journey.

This solution is separated into multiple sections and goes in detail through various challenges faced during the construction of the python script and the associated thought process and reasoning involved in overcoming them.

2 Procedure

As per the information provided from *van hees*, the overall procedure for the calibration analysis was split in the following steps.

1. The threshold for finding the Non-movement windows or calibration epochs were found by visual inspection of the Acceleration Vs Time plot. The region were a horizontal line was observed in the plot was used to estimate the threshold. Threshold value was defined as that value for which the standard deviation in each of the three axis in the time window was lower than it. The main reason behind this lies in the extended period of constant acceleration from the observation which was assumed to be an unrealistic scenario. Thus, it was assigned as the region for calculation of threshold.

2. The Non-movement windows were then used calculate a window specific average and standard deviation values. These values were reserved for later comparison with the optimised values. The length non-movement windows was based of the frequency of observed data and a provided time period of *10s*. Thus each window consisted of 300 data points.

3. The average calibration error, defined as the average absolute value of the difference between *1g* and euclidean norm of the acceleration vector components was then calculated. The euclidean norm was calculated as :

$$E_{norm} = \sqrt{(a_x)^2 + (a_y)^2 + (a_z)^2}$$

   Thus the average calibration error becomes :

$$\left\langle |1 - \sqrt{(a_x)^2 + (a_y)^2 + (a_z)^2}| \right\rangle$$

4. The calibration error is minimised by linearly transforming the vector components of the acceleration, i.e. ,

   $$a_i' = d_i + g_i \times a_i \qquad , where i = x, y, z$$

   Here , d_i is known as offset and g_i is the gain factor and are specific to each axis. Thus, in total we will have 6 unknown parameters after the linear translation.

5. Thus, the new quantity to minimize becomes

   $$\left\langle |1 - \sqrt{\sum_i (d_i)^2 + \sum_{i,j} (d_i \times g_j \times a_j) + \sum_i (g_i \times a_i)^2}| \right\rangle$$

   where as usual i, j = x,y,z.

6. Thereafter, the offset and gain for each of the axis was obtained from minimizing the above equation.

7. The linearly transformed components of the acceleratin are then obtained.

8. As a final confirmation, standard deviation centered around *1g* was calculated for the initial and final acceleration and compared.

3 Workflow

The first hurdle in the procedure came at the step (step 5 from previous section) where the error is minimized and optimum values for the six parameters are obtained. This was challenging because personally I don't have much recent experience with parameter estimation or curve fitting techniques. I have taken some linear regression classes and studied least square method for obtaining slope and intercept for linear models. So, minimizing the multi-variable non-linear equation seemed a bit difficult initially. I say 'initially' because after working out some formulaes by hand alongwith partial derivatives, I was able to come up with some ideas to achieve the objective.

3.1 Brute Force

The first method which is relatively straight forward and simple but inefficient and time consuming to an extent that it becomes not viable is iterative incremental method. An initial guess of the six parameters are provided and then there are slowly incremented by user-defined Δ quantities until an optimum value is obtained. After figuring out the method mentioned in next section, this option wasn't focused much.

3.2 Curve-fitting

The *curve_fit* method from `optimize` class **scipy** package in python was used to fit the linearly transformed components to a 3D sphere of radius 1. For the input data of size this method takes about *s*. Some thought process that went while trying to minimize the average calibration error were :

$$\text{norm} = \sqrt{\sum_i (d_i)^2 + \sum_{i,j} (d_i \times g_j \times a_j) + \sum_i (g_i \times a_i)^2}$$

$$avg_e rror = \langle |1 - \text{norm}| \rangle$$

$$\Downarrow$$

$$avg_e rror \to 0$$

as

$$norm \to 1$$

$$\Downarrow$$

(squaring on both sides)

$$norm^2 \to 1$$

4 Results

The following results were obtained after the analysis :

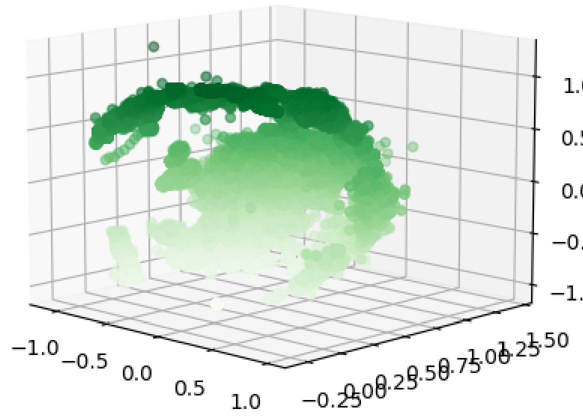
- `Offsets`
  - $d_x = 0.533$
  - $d_y = 0.618$
  - $d_z = 0.577$
- `Gain`
  - $g_x = -2.059e^{-10}$
  - $g_y = -2.802e^{-10}$
  - $g_z = -2.309e^{-10}$

Apart from these some other useful information are :

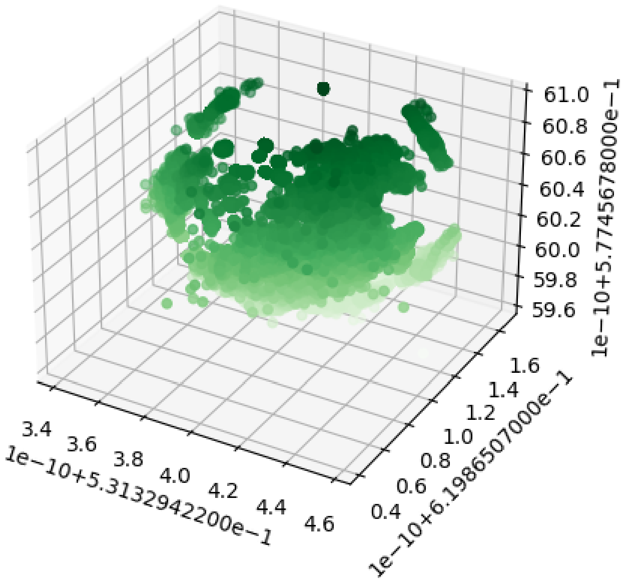
1. Total execution time for the analysis script is : 154s
2. The standard deviation centered around zero for the initial components of acceleration and the one after linear transformation are respectively, : *0.168* and *9.132* e^{-10}.

3. The average values of acceleration (euclidean norm) before and after transformation are respectively : *1.008* and *1.00*

The data points are plotted across a surface and compared to that of a 3D sphere with radius zero as shown below.



(a) Initial Raw



(b) Transformed

Figure 1: Plotting the acceleration vector in 3D space for comparisons with a sphere of radius *1g*. (a) Acceleration before linear transformation (raw input data) (b) Acceleration after finding minimising constants and linear transforming

References

[1] Vincent T. van Hees *et al 'Autocalibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents'*, Journal of Applied Physiology.