UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Classification of physical activity from the embedded smartphone sensors: algorithm development and validation

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Jordi Saumell i Ortoneda

In partial fulfilment

of the requirements for the degree in

Ciències i Tecnologies de Telecomunicació **ENGINEERING**

Advisors: Miguel Ángel García González

and

Federico Guede Fernández

Barcelona, June 2015

Abstract

Physical activity classification has grown in importance lately, for reasons such as positioning or health issues. Given the ubiquity of smartphones and the plethora of sensors they contain, these devices have become an extremely useful tool for this task. In that direction, this project provides an algorithm to count steps using the accelerometer of an Android smartphone.

This algorithm searches for patterns in the accelerometer's signal based on the correlation between consecutive fragments of the signal after a pre-processing step that adapts the data, to count steps under relatively unconstrained ways of carrying the smartphone. The accuracy of the designed algorithm is 92.5% using a database of eleven subjects and four different tests for each subject.

As some limitations have been found, a plan for improving the algorithm has been introduced, based on the experience acquired.

Resum

Classificar l'activitat física és cada cop més important, ja sigui per posicionament o per problemes de salut. Donada l'omnipresència dels *smartphones* i el conjunt de sensors que contenen, aquests aparells s'han convertit en eines verdaderament útils per aquesta tasca. En aquesta línia, aquest projecte proporciona un algorisme per comptar passos a partir de l'acceleròmetre d'un mòbil Android.

Aquest algorisme busca patrons en el senyal d'accelerometria basant-se en la correlació entre fragments consecutius de senyal després d'un pre-processament per adaptar les dades; per comptar passos amb maneres de portar el mòbil poc restrictives. La precisió de l'algorisme dissenyat és de 92.5% fent servir una base de dades d'onze subjectes i quatre proves diferents per cada subjecte.

Com s'han trobat certes limitacions, s'han plantejat uns possibles passos per millorar l'algorisme basats en l'experiència adquirida.

Resumen

Clasificar actividad física es cada vez más importante, ya sea para posicionamiento o por problemas de salud. Dada la omnipresencia de los *smartphones* y el conjunto de sensores que contienen, estos aparatos se han convertido en herramientas verdaderamente útiles para ésta tarea. En esta línea, este proyecto proporciona un algoritmo para contar pasos a partir del acelerómetro de un móvil Android.

Este algoritmo busca patrones en el señal de acelerometría basándose en la correlación entre fragmentos consecutivos de señal tras un preprocesado para adaptar los datos; para contar pasos con formas de llevar el móvil poco restrictivas. La precision del algoritmo diseñado es de 92.5% usando una base de datos de once sujetos y cuatro pruebas distintas para cada sujeto.

Aunque los resultados no son tan buenos como se pretendía, se han planteado unos posibles pasos para mejorar el algoritmo basados en la experiencia adquirida.

Acknowledgements

I would like to express my gratitude to my advisors Miguel Ángel García González and Federico Guede Fernández for their guidance, for aiding me in the decision making, and also for helping me find solutions to the different issues encountered while developing and testing the algorithm.

I would also like to thank all the volunteers, without whom I would not have had the measurements necessary to develop this thesis.

Revision history and approval record

Revision	Date	Purpose
0	13/06/2015	Document creation
1	07/07/2015	Document revision
2	09/07/2015	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Jordi Saumell Ortoneda	jsaumell420@gmail.com
Miguel Ángel García González	miquel.angel.garcia@upc.edu
Federico Guede Fernández	federico.guede@upc.edu

Written by:		Reviewed and approved by:	
Date	13/06/2015	Date	09/07/2015
Name	Jordi Saumell Ortoneda	Name	Miguel Angel García González Federico Guede Fernández
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

1.1. Objectives

This project aims at studying different approaches for obtaining the number of steps a user takes while walking, wearing his smartphone; and then building an application for Android 4.0 or above, which will use an efficient and reliable algorithm.

The project has been developed at the Electronic Engineering Department and it has been supervised by Miguel Ángel García González and co-supervised by Federico Guede Fernández.

The goals of this project are:

1. To study algorithms used for counting steps performed by a user; in different activities, such as walking or climbing stairs.
2. To choose from among these algorithms the one that, to our judgment, is most reliable on counting steps while wearing the smartphone on different but realistic places, such as the hand or the pocket.
3. To develop a user-friendly application for Android 4 or above, which will use the chosen algorithm.
4. To compare the developed application with the best reviewed commercial application.

1.2. Requirements and specifications

The project requirements are developing an application for Android 4.0 or higher versions, which should be able to determine the number of steps done by the user who is wearing the smartphone. It should also be easy to use and properly work with independence of the smartphone location (pocket, hand, backpack, etc., the only constraint is that the smartphone is carried by the user)

The sensitivity and precision (positive predictive value) for the step detection using the application must be higher than 95%. These specifications will be tested in controlled measurements in several volunteers.

1.3. Methods and procedures

This work is based on reference [1], where an algorithm for counting steps is described. This algorithm has been implemented, optimized and modified in order to improve the results when measuring different subjects on different testing conditions.

1.4. Work plan

Work Packages:

Project: Research	WP ref: (WP1)	
Major constituent: Research	Sheet 1 of 2	
Short description: Searching in the bibliography ways of measuring physical activity and also getting familiar with the Android API's to work with the sensors.	Planned start date: 02/03 Planned end date: 16/03	
	Start event: 02/03 End event: 16/03	
Internal task T1: Bibliography about measuring physical activity Internal task T2: API's	Deliverables: Documentation	Dates: 16/03

Project: Samples	WP ref: (WP2)	
Major constituent: Obtaining samples	Sheet 1 of 2	
Short description: Deciding how are the samples going to be obtained, and obtaining them.	Planned start date: 16/03 Planned end date: 30/03	
	Start event: 16/03 End event: 06/04	
Internal task T1: Establishing a procedure for obtaining sensor samples. Internal task T2: Obtaining the samples.	Deliverables: Samples	Dates: 06/04

<table>
<tr><td>Project: Building the algorithm</td><td colspan="2">WP ref: (WP3)</td></tr>
<tr><td>Major constituent: Implement Algorithms in Matlab</td><td colspan="2">Sheet 1 of 2</td></tr>
<tr><td rowspan="2">Short description:
Implementing the algorithms to process the data from the sensors in Matlab, and then comparing each of the algorithms to see which may give better results</td><td colspan="2">Planned start date: 30/03
Planned end date: 04/05</td></tr>
<tr><td colspan="2">Start event: 16/03
End event: 18/05</td></tr>
<tr><td>Internal task T1: Developing the algorithms

Internal task T2: Testing them</td><td>Deliverables:
Algorithm results</td><td>Dates: 18/05</td></tr>
</table>

<table>
<tr><td>Project: Improving the algorithm</td><td colspan="2">WP ref: (WP4)</td></tr>
<tr><td>Major constituent: Improving algorithm</td><td colspan="2">Sheet 2 of 2</td></tr>
<tr><td rowspan="2">Short description:
Implementing some improvements to the algorithm in the paper, and testing the results with these improvements.</td><td colspan="2">Planned start date: 04/05
Planned end date: 01/06</td></tr>
<tr><td colspan="2">Start event: 18/05
End event: 03/07</td></tr>
<tr><td>Internal task T1: Developing improvements

Internal task T2: Testing improvements

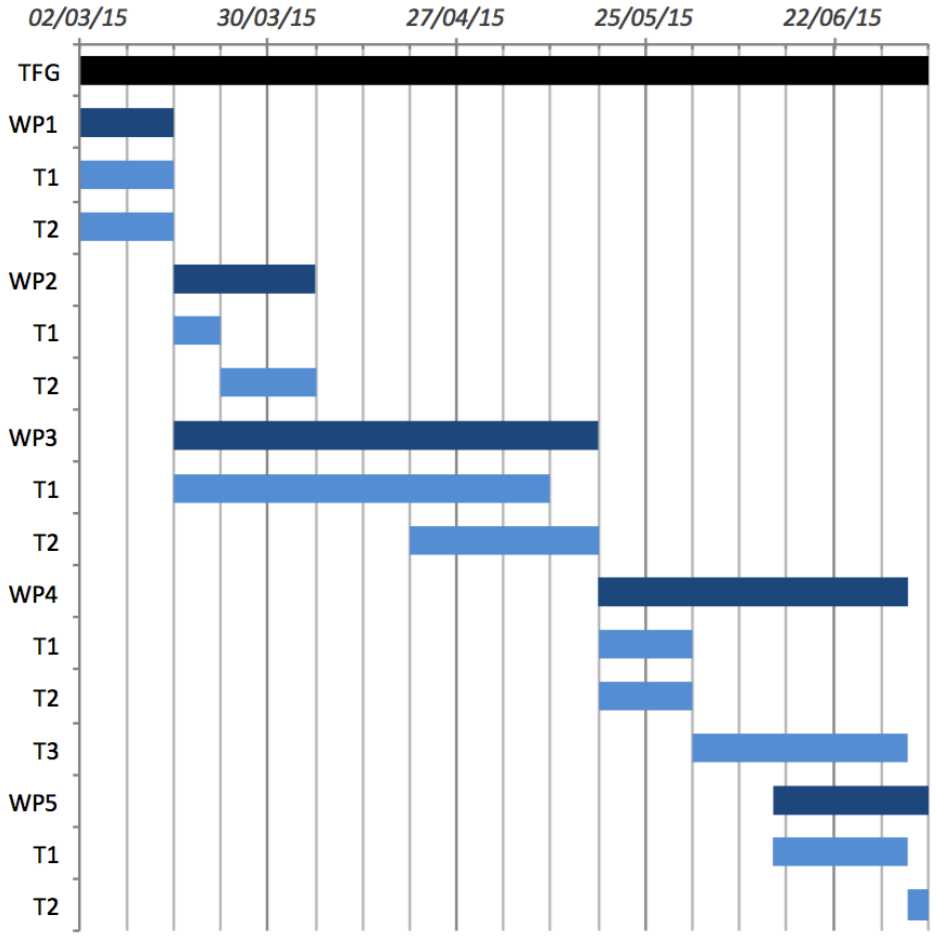
Internal task T3: Testing final algorithm</td><td>Deliverables:
Android software</td><td>Dates: 03/07</td></tr>
</table>

Project: Writing the Final Report	WP ref: (WP5)	
Major constituent: Writing	Sheet 2 of 2	
Short description: Writing the final report and revising it.	Planned start date: 01/06 Planned end date: 06/07	
	Start event: 13/06 End event: 09/07	
Internal task T1: Writing the final report Internal task T2: Revising the final report	Deliverables: Final Report	Dates: 09/07

Milestones:

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	1 2	Reading the bibliography API's	Information to decide what to do	1-2 1-2
2	1 2	Establishing procedure Obtaining samples	Samples needed for the Algorithms	3 4-5
3	1 2	Developing Algorithm Matlab Testing algorithm	Algorithm to be implemented	3-10 8-11
4	1 2 3	Improving algorithm Testing improvements Testing final results	Final results	12-13 12-13 14-18
5	1 2	Writing the Final Report Revising the Final Report	Final Report	15-17 18

Task	Start	End	Duration
TFG	02/03/15	09/07/15	18,4286
WP1: Research	**02/03/15**	**16/03/15**	**2**
T1: Bibliography	02/03/15	16/03/15	2
T2: API's	02/03/15	16/03/15	2
WP2: Samples	**16/03/15**	**06/04/15**	**3**
T1: Establishing the procedure	16/03/15	23/03/15	1
T2: Obtaining the samples	23/03/15	06/04/15	2
WP3: Building the algorithm	**16/03/15**	**18/05/15**	**9**
T1: Developing algorithm	16/03/15	11/05/15	8
T2: Testing it	20/04/15	18/05/15	4
WP4: Improving algorithm	**18/05/15**	**03/07/15**	**6,57143**
T1: Developing improvements	18/05/15	01/06/15	2
T2: Testing them	18/05/15	01/06/15	2
T3: Testing final algorithm	01/06/15	03/07/15	4,57143
WP5: Writing the Final Report	**13/06/15**	**09/07/15**	**3,71429**
T1: Writing the Final Report	13/06/15	03/07/15	2,85714
T2: Revising the Final Report	03/07/15	09/07/15	0,85714



1.5. Deviations from initial plan

The first deviations from the initial plan were those described in the critical review. Obtaining the samples took one week longer than expected because of the difficulty of scheduling the measurements with each volunteer.

As the obtaining of the samples did not occupy all the time for this scheduling issues, work package 3 started earlier and was done at the same time that samples were being obtained.

The development of the algorithm also took more time than initially planned, as it was more difficult than expected.

After the critical review, there have been several changes, too. The fourth work package initially consisted of porting the algorithm to Android and testing it, because this was one of the goals of this project. However, there have been several setbacks while developing the algorithm such as omissions in the selected journal paper or the algorithm being much more difficult to debug than expected. In addition, the results from the mentioned algorithm were much worse than the expected from the results published in the paper.

For these reasons, there was not enough time to both improve the algorithm and port it to Android. As porting an algorithm with poor results was completely useless, it was decided to substitute the work package of porting the algorithm to android for another that consisted of improving the algorithm –in Matlab.

The last work package, writing the Final Report, was initially given more time than strictly necessary, so that it could be shortened if the previous tasks had not been finished on time. As this is what has happened, the package has finally started a bit later than planned. And it also has been carried through simultaneously with the last part of work package four.

2. State of the art of the technology used or applied in this thesis:

2.1. Related research

Nowadays, we become more concerned about our health, and hence we pay more attention to physical activity as it has been proven to be very beneficial. Also, the interest on losing weight has lately spread enormously; therefore, measuring physical activity is helpful as it allows quantifying progress.

Some people specially need to do exercise, as they need to recover or improve their mobility. For instance, those who are recovering from an injury, or those who suffer a pathology that affects the movement. The elderly also need to do exercise frequently, or they could use a detector to call for help if they fell onto the ground. And in some of these cases it is necessary to verify that enough physical activity is being performed, as not everyone is willing to do it on their own, or to do it enough. Finally, some people are willing to do the required exercise, but could use some help in quantifying it.

At last, measuring and classifying physical activity is very helpful for indoor positioning. As the GPS rarely works indoors, and there is a lot of interest in indoor localization, a lot of work has been done in using classification of physical activity for aiding the positioning algorithms.

As it has been shown above, the interest on measuring activity is real. Now it is necessary find useful measuring tools.

For instance, if we place a pressure sensor in the shoe it will be relatively easy to track how many steps the user takes; nevertheless, it will be costly and uncomfortable, so it would be better to avoid this path if possible. The same can be applied to the hands, as we can attach sensors to the hands and arms to better detect what the user is doing. But this solution offers the same disadvantages in cost and discomfort.

Here is where the ubiquity of smartphones offers a solution without hassle or cost to the user. Smartphones are computationally powerful and have a plethora of sensors embedded. Nowadays, almost everyone has a smartphone and carries it wherever she/he goes. These smartphones carry accelerometers, gyroscopes, magnetometers, GPSs and many more sensors that may be used for a lot of monitoring purposes, with the only drawback of battery consumption.

For these reasons, there is a lot of research on measuring physical activity using smartphones, or accelerometers similar to those in smartphones. Many different approaches have been taken, some closer to real life whereas others are restricted to the laboratory. Nevertheless, there is a lot of work to be done yet, as the accuracy of these algorithms is still very low.

One of the fields with most literature about classification of physical activity is on activity recognition. Many have used machine learning techniques to distinguish between

different physical activities using the data obtained from different sensors, such as the accelerometer.

The relevant work from L. Bao and S. Intille [2], tested different classifiers to differentiate between twenty everyday activities from twenty subjects. The decision tree classifier offered the best results with 84% of accuracy, being fed with the mean, energy, frequency-domain entropy and correlation of the acceleration. Nevertheless, the data was obtained from 5 different accelerometers attached to different parts of the body, thus making the algorithm very impractical to be used in practical scenarios.

Because of this limitation, there is also research in algorithms that use only one accelerometer. In [3] they do the activity recognition with a predictive model that analyses the accelerometer's data from twenty-nine subjects to distinguish between six different activities.

There is also work done to estimate the gait length, such as [4], which combines a decision tree, a neural network and a support vector machine to estimate the user's step length in order to improve indoor location.

With the same purpose of indoor location, [5] compares three algorithms to detect steps and classify activities. Although the results are positive, the implemented algorithm has bad results when it is trained with one person and tested with another, or when the user walks fast. An interesting conclusion from [5] is that footwear significantly affects the shape of the acceleration signal.

Another paper that aims at counting steps for indoor location with a smartphone is [6], which uses the gyroscope, magnetometer and accelerometer for such purpose. The first two sensors are used to estimate the heading direction and to compensate the tilt whereas the accelerometer is used to compute steps. A comparison between ways of estimating position is done.

As the previous uses three sensors, there are many that even use foot plantar force sensors, or attach the accelerometer to the foot. Although these researches obtain good results, tying sensors to the foot is not practical for most everyday-use purposes. For instance, [7] compares two approaches to count steps in real time for an indoor navigation system. The first approach is computationally heavy and has a delay, and the second approach needs a foot plantar force sensor. The error rate is very low, but has the drawbacks of the sensors' position.

Most of the literature studies activity recognition or step counting under very restrictive conditions; such as wearing the sensor tied, or in a very specific position. Although these investigations are advances, they do not work in real life conditions. Some papers investigate less restrictive ways of carrying the device; as in [8], where they have developed an algorithm based on the Kalman filter and their own filter to count steps in iOS, allowing the user to have the smartphone in either the hand or the pocket –but in specific positions. This a priori knowledge on how the phone is carried allows to easily extract the acceleration component that is parallel to the gravity, but is not useful for real-life behavior.

The research described in [9] does not have the limitations seen in [8], as it compares different step counting algorithms with 27 people and in (almost) unconstrained smartphones -six different smartphone placements. As counting steps requires a lot of computation, it reduces battery life. In this paper, they implement a walking detector in

order to run the step count algorithm only when the user is actually walking. Their conclusion is that algorithms based on standard deviation are good at detecting walking, and the windowed peak detection is the best in step counting regardless of the smartphone placement.

The paper in [10] is aimed at encouraging a healthier lifestyle. To do that, it develops an algorithm to count steps and trains a feed-forward artificial neural network to prevent cheating. This cheating prevention system increases the robustness of the algorithm. Besides robustness, it is also useful in circumstances in which it is specifically important to detect cheating –for example, some kids who need to do exercise for their health try to trick the app by shaking the smartphone with their hand.

Once reviewed the related investigations, we are going to focus in those that rely on different approaches for counting steps to compare them. Among the different approaches that have been developed to count the number of steps using the data from the accelerometer, one of them that is widely used consists of setting thresholds and verifying whether the shape of the signal surpasses those thresholds. As the walking signal is similar to a sinusoid, each time a threshold is surpassed means there is a peak, and therefore, a step. In [11] they go further and take into consideration the fact that in different activities, different thresholds are required, and for this reason they have used a finite state machine. In this finite state machine it is necessary to first recognize the activity and then verify whether the threshold is crossed. Although systems based on thresholds provide good results, these thresholds vary with different subjects. Because of this variation, threshold-based algorithms are not useful for a system supposed to work with mostly everyone.

Also based on the sinusoid-like shape of the acceleration, others have developed algorithms that try to detect the peaks and valleys of the signal. They look for patterns, either by detecting hills, or by detecting local maxima. Particularly, in [12] they look for a local maximum between two local minima and verify that the value of consecutive minima and maxima is similar. These algorithms fail when the location of the device respect to the user changes.

Then, there are other systems based on the correlation between portions of the acceleration signal. Given a start point, such systems look for a reasonable end point –usually considering the minimum and maximum speed at which most human beings walk-, compute the correlation and verify whether a threshold is exceeded. If the threshold is exceeded –i.e., there is a step- the next end point will be at the same time distance than the first one plus or minus a random time interval. However, if the user changes the position of the smartphone this approach misses several steps. In [13] they use this method, but fail to consider when the user changes how the smartphone is being carried.

So, it seems that a possible good approach would be the one followed by [13] as long as the issue of changing the carrying way is addressed. With this in mind, in [1] they develop an algorithm to count steps that is tested in different activities, and permits changing how the device is carried in the middle of the activity. Despite this freedom in how the device is carried -which increases complexity- the results are very positive, with fewer than 2.5% of error.

2.2. The selected algorithm

After having caught up with the state of the art, it was necessary to choose what direction to follow. Developing the whole algorithm from scratch was more work than it could be done in the available time. With this in mind, it was necessary to decide for one of the described algorithms to start with, and then try to improve it. As it has been mentioned above, the paper [1] has an algorithm to count steps while carrying the phone in different ways and even if the way of carrying the device changes, all with a low error rate. Since this is the goal of this project and the algorithm is explained in the document, it seemed as a good option to start with. Also, the paper has an extended comparison of the algorithms in the literature and the explanation of how the algorithm works seemed reasonable. The only doubt on whether it was a good choice was that it does not explain how many subjects were used to obtain the results, which could mean that testing it with different people would lead to worse results.

3. Project development:

3.1. Data collection

As the algorithm has to be tested, it was necessary to obtain samples from different users and in different conditions. To do so, an itinerary was designed so that all ways of carrying the device and physical activities could be tested.

This itinerary –see Figure 1- consists of around twenty meters of walking with no slope, followed by twenty meters more with some slope. After that, around ten meters of walking without slope and twenty steps of climbing stairs. Then around thirty meters of walking, six steps of climbing stairs and around ten meters more of walking. Afterwards, the subject does the same route but in the opposite direction. With this circuit, the user walks up and down a slope, climbs stairs up and down, and walks without slope.

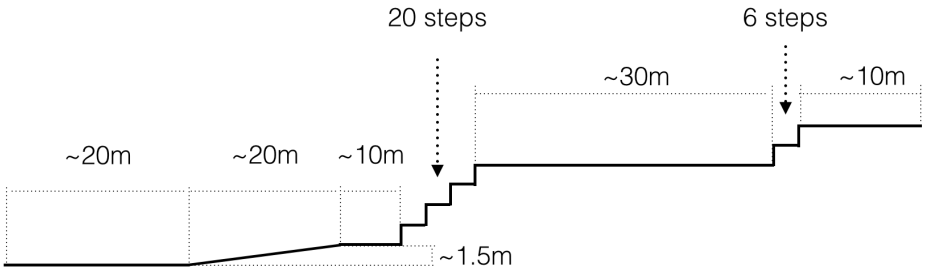


Figure 1 Itinerary

Every user did the itinerary four times, the first one with the smartphone in the trousers' side pocket. Then another time balancing the phone in the hand, but without using it. The third round was also with the phone in the hand, but this time using it –or simulating using it. Finally, in the last it was carried in a backpack.

These tests were taken in eleven subjects, 4 female and 7 male. The heights for the female subjects are in a range that goes from 1.63m to 1.75 and their weights from 52kg to 65; the heights for the male go from 1.70m to 1.92 and their weights from 70 to 85kg. Most of the subjects were in their twenties, except for one who was in his thirties and another in his forties.

As the algorithm has to be compared to the available software in the Google Play Store, "Noom" was picked as it seemed to be the best pedometer application considering ratings and number of user downloads. To compare Noom to our algorithm, Noom was running at the same time that samples were being taken and in the same device, so that it could be compared at the moment of analyzing the results.

To verify the accuracy when counting steps, both for our algorithm and for Noom, it is necessary to know the actual number of steps that had been walked. Therefore, steps were counted while the samples were being recorded. To know how many steps were walked in each different activity –walking or climbing stairs- I developed an Android application –see Figure 2- to manually count a step each time the top of the screen is touched, and change the activity each time the bottom of the screen is touched. This software also records the duration in seconds of each activity and then saved the results in the device. In the third image the results can be seen. Each row represents an activity,

the first column corresponds to the number of steps and the second column to the number of seconds spent in that activity.

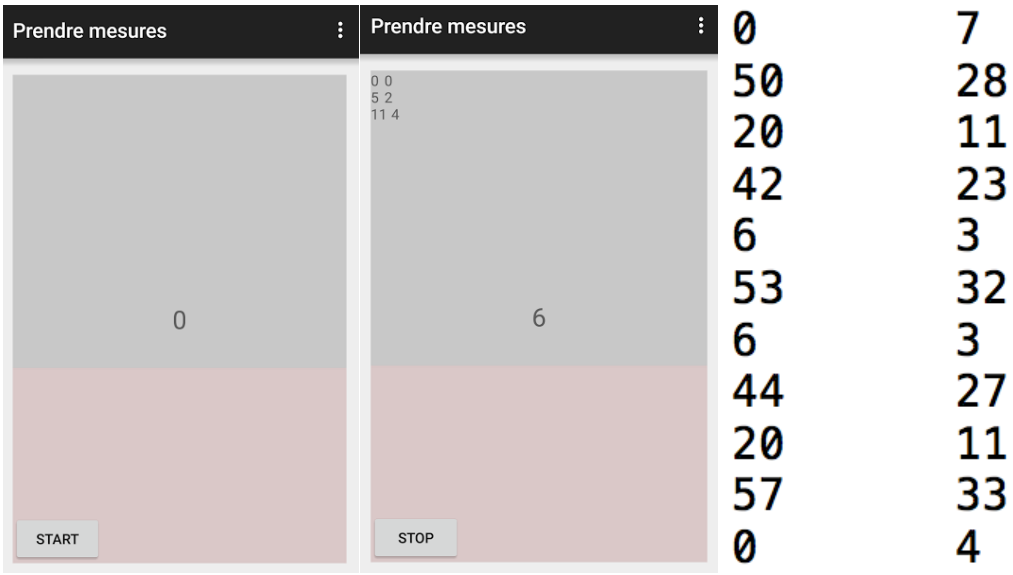


Figure 2 App to count steps

As the device registering the acceleration had to be held in particular ways, it could not be used for counting the number of steps. For that reason, a procedure to take samples was established, in which the subject walked while carrying the device with Noom and the software to take samples running. Another subject –external observer- walked next to the former with another smartphone counting the steps and verifying that the smartphone was being carried as it was required. At the end, the number of steps counted by Noom was written down next to the time at which the user had started walking.

Besides the itinerary, samples were also taken for several minutes while traveling by bus, both using the smartphone and having the smartphone in the pocket. The bus travel was used to verify that the algorithm did not count steps with the movement of the bus, as several systems currently do.

It was also planned taking samples from a user jogging. Nevertheless, because of poor results running the algorithm on the samples from the first described itinerary, more time and effort was spent on improving the algorithm; therefore, the running samples were not taken.

3.2. Theoretical background

The algorithm to count steps uses the smartphone's accelerometer to estimate those steps. Before describing the algorithm, it is necessary to discuss the behavior of the accelerometer while the user walks.

First of all, the data read from the sensor is discrete, so every time the software asks for the acceleration, it is given a value for each axis in return. As we have discrete samples, one parameter to take into account is the sensor's sampling frequency. The software provided by the advisors to read the accelerometer data changes this sampling frequency depending on the amount of activity, i.e., when the sensor detects a lot of movement it

increases the sampling frequency and when it does not detect movement it does the opposite. As the algorithm would be much more complex with a variable sampling frequency, it will be necessary to evenly resample the signal.

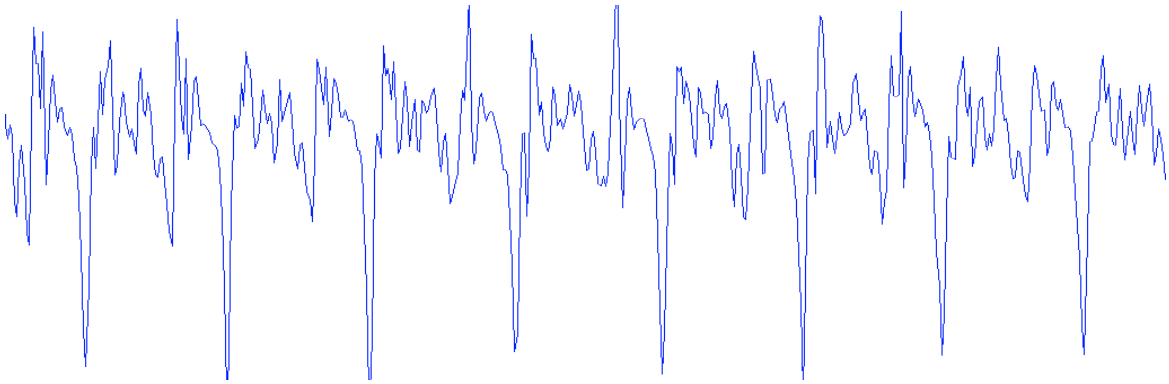


Figure 3 Periodicity example. Acceleration signal of a user walking, 'x' axis in m/s^2

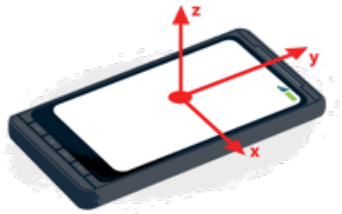


Figure 4 Smartphone with acceleration axes

The acceleration given by the sensor consists on three axes where none of these is necessarily perpendicular with the earth. When walking, if we observe the acceleration's signal vertical component –i.e., parallel to the gravity- and the horizontal component that is parallel to the walking direction, we will observe that it has a nearly periodical behavior –as it can be observed in Figure 3- and the fundamental frequency is related to the steps' speed. With this periodicity in mind, it appears to be convenient to use these two components for computing the number of steps; nevertheless, this would require the smartphone to be held in a specific position –see Figure 4. As this restriction is not acceptable, it is necessary to calculate the device's tilt and apply two rotations –as in Figure 5- to the acceleration to obtain the vertical and horizontal components described before. The (a) image shows the device in the original position. In (b) it can be seen the rotation represented by $ß_y$, and in (c) the rotation that $ß_x$ represents, after the correction on $ß_y$ has been applied. Finally, in (d) it can be seen the result of applying both transformations, which is as if the phone was parallel to the horizon.

For the first rotation, the gravity after the transformation will be $(g''_{x,ti}, g''_{y,ti}, g''_{z,ti}) = (0, 0, 9.8)$ m/s^2. Using this equality we will be able to compute $ß_x$ and $ß_y$ by solving the equations 1, 2 and 3. Nevertheless, these equations require the gravity, which is unknown. As we have the acceleration, which is composed of the addition of gravity, movement and noise; we can assume that the movement has null average and therefore, we can filter the signal with a low-pass filter to obtain an estimation of the gravity.

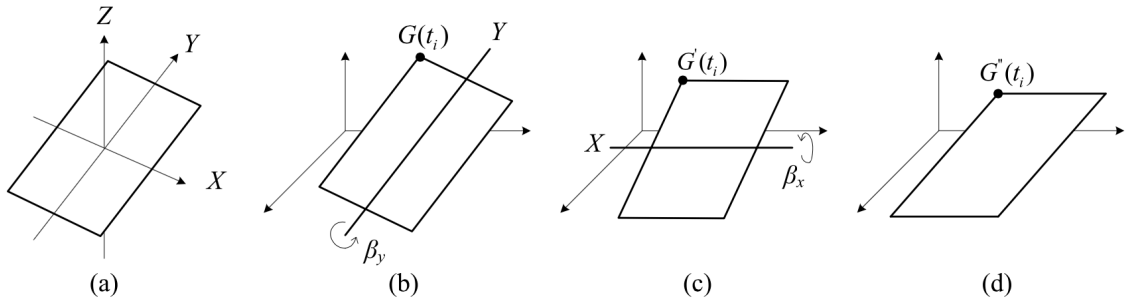


Figure 5 Rotation to be applied to the signal

$$g''_{x,ti} = g_{x,ti} \cdot \cos(ß_y) - g''_{z,ti} \cdot \sin(ß_y) \quad (1)$$

$$g''_{y,ti} = g_{y,ti} \cdot \cos(ß_x) + g_{x,ti} \cdot \sin(ß_y) \cdot \sin(ß_x) + g_{z,ti} \cdot \cos(ß_y) \cdot \sin(ß_x) \quad (2)$$

$$g''_{z,ti} = g_{y,ti} \cdot \sin(ß_x) + g_{x,ti} \cdot \sin(ß_y) \cdot \cos(ß_x) + g_{z,ti} \cdot \cos(ß_y) \cdot \cos(ß_x) \quad (3)$$

For the transformation, the Eq. 4 represents the rotation in the $ß_y$ angle, and the Eq. 5 the $ß_x$ angle. After applying both rotations to the sensor acceleration, we obtain the acceleration we would obtain if the smartphone were hypothetically parallel to the horizon.

$$\begin{cases} x'_p = x_p \cdot \cos(ß_y) + z_p \cdot \sin(ß_y) \\ y'_p = y_p \\ z'_p = -x_p \cdot \sin(ß_y) + z_p \cdot \cos(ß_y) \end{cases} \quad (4)$$

$$\begin{cases} x''_p = x'_p \\ y''_p = y'_p \cdot \cos(ß_x) - z'_p \cdot \sin(ß_x) \\ z''_p = y'_p \cdot \sin(ß_x) + z'_p \cdot \cos(ß_x) \end{cases} \quad (5)$$

The procedure described can be seen in the flowchart in Figure 6.

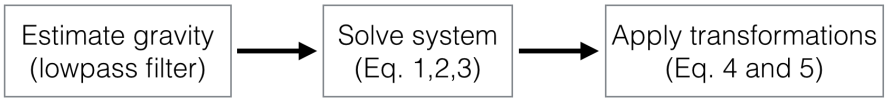


Figure 6 Preprocessing flowchart

To verify that the transformation works properly, two experiments have been conducted. The first was having the phone parallel to the ground, and rotating it so that $ß_x$ increased until 90º. The other was doing the same, but for $ß_y$.

In Figure 7 we have the experiment in which the smartphone was initially on the table, and was rotated so that it was in landscape mode. In the right figure we can see as the 'z' axis -green- is around 9.8 m/s^2, as it is expected. In the part with movement, there is more noise, and the signal is slightly above the gravity, due to the movement. And the same for the 'y' axis -red-, it is slightly above 0 at the part of the movement, because when rotating from being on the table, the device was also moving slightly upwards. The left figure represents the acceleration before the transformation and the 'z' acceleration goes from the value of the gravity to zero, whereas the 'y' acceleration does the opposite.

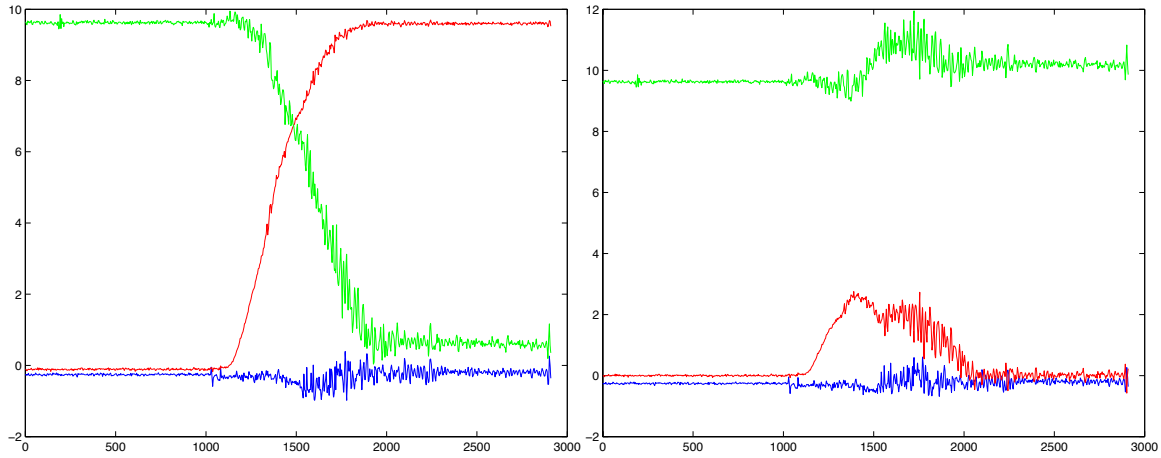


Figure 7 Accelerations in m/s^2 before and after rotation, plotted with the number of samples. Rotating to landscape position

And for the second experiment the result is also positive, but it has not been added for redundancy reasons. The 'z' axis is around the value of the gravity, the 'y' is at zero, and the 'x' -blue- slightly below in the part of the movement.

Although in theory these transformations should work properly, in some cases they did not. Some samples that were supposed to be at 9.5 m/s^2 were shifting from 9.5 m/s^2 to -9.5 m/s^2. The reason for this was that for values near to zero in the z-axis gravity, the added noise would make these values alternate between being above and below zero. The solution was shifting those small values with the difference between the standard deviation and the mean, and adding or subtracting depending on whether the mean was above or below zero.

3.3. Data preprocessing

Before actually running the algorithm on the signal recorded with the accelerometer, it is necessary to preprocess such signal as it has been explained in "theoretical background". With this procedure, the results will be better and the algorithm simpler.

The first step in this stage is the interpolation, choosing as resampling frequency a value of 200Hz, which is the maximum frequency used when the accelerometer detects activity. Using a lower frequency would mean losing data in the critical instants –when the user is moving- and choosing a higher frequency implies increasing the amount of data with no benefit –as all of it has to be estimated.

But for the interpolation to work, it is necessary to do a previous step. When the smartphone is overloaded with other tasks it may change the order at which values are sent –i.e. send the second value before the first- or send two values with the same timestamp. As the interpolation does not work with a non-monotonic time variable, it is necessary to sort the signal and delete the repeated time values (with the corresponding acceleration) prior to interpolating.

Now that the signal has been interpolated, the gravity is needed to continue. As in [1] the gravity is used with no explanation of how it is obtained, we have created a way to

estimate the gravity. To do the estimation, the interpolated signal is filtered with a moving average filter of 1 second of duration. If the length of the filter is too short, the supposition of zero-mean movement will not be valid, and thus the estimation will be erratic. On the other hand, a long filter will return values from the past, and the consequence will be a time delay between the tilt and the signal. With these considerations in mind, the filter was chosen by testing different lengths and picking the shortest that was not erratic at eyesight.

Because of the convolution used when filtering, the first resulting samples are not usable. Therefore, the first 'n' samples will be given the value of the sample 'n+1', where 'n' is the length of the filter.

Once the gravity has been estimated, all the necessary steps to compute the tilt of the phone have been done. The tilt is calculated with the formulae described before, but first the gravity values are modified to ensure that there are no issues with values near to zero. Then the transformation is applied and the vertical (*z*) and horizontal (*y*) accelerations are obtained.

With the *z* and *y* signals, we should be ready to start counting steps. But, if it was possible discarding part of the signal it would be much better in terms of computation. Some of the irrelevant time points would be those in which there is no activity. In an ideal world, the values for the acceleration would be 9.8m/s^2 for the *z* and 0 for the *y*. But in reality there is noise, so it has to be taken into consideration.

Leaving the smartphone steady, on a table, and recording the values from the acceleration, it would vary between -0.3 m/s^2 and +0.3 m/s^2 for the *y*; and the same for the *z* but adding 9.5. So, with the device used –a Nexus 5- the gravity is 9.5 m/s^2 instead of 9.8. Knowing the values that the noise could take, a threshold was set on the positive value –*th_usp*-, and another on the negative –*th_lsp*. Each time one of the noise thresholds is crossed, the time point corresponding to that crossing is recorded in a list. This process is done for both the *y* and *z* signals, and two separated lists store the relevant points –*section points*.

Now that the data has been properly preprocessed it can be analyzed to obtain the number of steps.

3.4. Step counting procedure

This algorithm analyses the signal of acceleration to count the number of steps, using the Y and Z axis, as explained before. The same procedure is applied to each of these axes separately and the results are combined to obtain the estimation of the number of steps.

As it has been explained before, when walking the acceleration signal is almost periodical –seen in Figure 3-, having one period of fluctuation for each step. Because of this periodicity, to analyze one axis we look for segments, which are portions of signal with two regular fluctuations. Therefore, one segment will correspond to two consecutive steps.

As the algorithm is complex, there is a flowchart in Figure 8 to help with the explanation. Some steps done before exiting have been omitted for readability, but are described in the text. The increments of samples or signal are not in the flowchart either, for the same

reasons. “th” corresponds to “th_ia_g”, “th2” corresponds to “th_low_g” and “th3” corresponds to “th_ir_g”.

The correlation is used to verify that there are two consecutive regular fluctuations. Given one segment of signal, it is split into two halves and the correlation between each half is computed. The nearer such correlation is to 1, the stronger the correlation will be. If it exceeds a certain threshold, it will be considered that there is a segment –so, two steps. Also, when the correlation between consecutive segments is needed, if one segment is shorter than the other it will be zero padded.

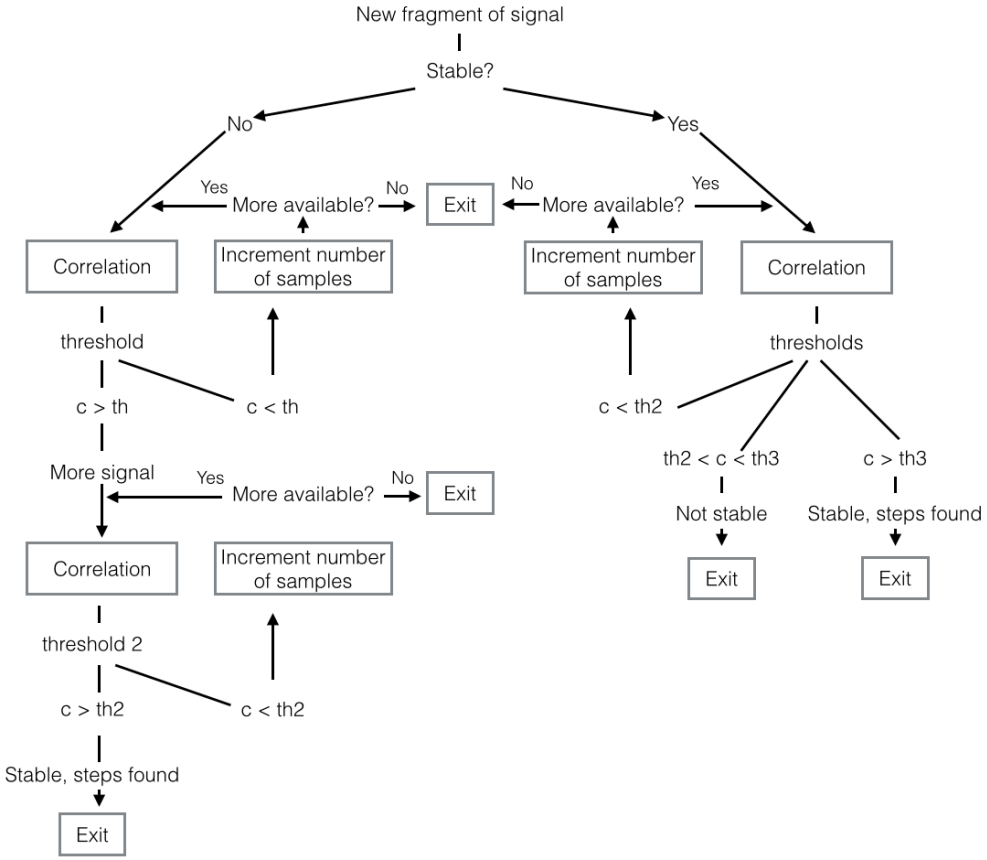


Figure 8 Algorithm flowchart

As we need segments of signal, it becomes necessary to define a way of obtaining them. As the start of the segment, we choose the first section point available. The end will be a point at a distance between the minimum reasonable gait length and the maximum reasonable gait length, which will be determined later on.

The procedure of identifying segments consists of starting with a portion of signal at the minimum length, and increasing the length until the correlation exceeds the threshold. If the maximum length has been reached without locating any segment, the starting point will be changed to the next section point and the same procedure will be taken. On the other hand, if a segment is found, we will look for another segment consecutive to the first one. Nevertheless, the next segment is supposed to have the same length plus or minus a delta. This delta corresponds to the possible variation of walking speed.

When looking for the second segment there are two possibilities. The first is that we do not find a second segment correlated to the first, and in this case we will look for a first

segment with the first initial time point that is posterior to the end of the segment we had recently found. However, if we do find a second segment correlated to the first one, we will consider the signal to be stable.

If the signal is stable, we consider three different situations when looking for another stable segment while using two different correlation thresholds. The first case would be if the correlation was below both thresholds after examining all the possible lengths, and we would consider that the signal is no longer stable and we would start looking for a new segment with the next section point. In the second case, however, the correlation is above the lowest threshold and below the highest threshold. This means that the user was walking but he is moving differently, either because he is stopping or because he is changing how he holds his phone. In this case, the portion of signal will be considered a segment, but the signal will no longer be stable. Finally, if both thresholds are surpassed we have a valid segment and we keep looking for the next one.

Every time a segment is found its last time point is registered so that it can be compared to the other axis.

Every time the stability is lost –i.e. the signal was stable and it is considered that it no longer is- compensation is applied. This compensation consists on supposing that during a certain period of time the user has been walking at the same speed that he was previously. The proportion of steps corresponding to that period is computed and the decimals are truncated.

As until now each axis has been treated separately, it is necessary to decide a way of combining them. Considering that this system never considers that the user walks faster than he is actually walking, and steps can be lost if the signal is not stable enough, it is reasonable to use the axis that counts most steps. Therefore, both axes are combined so that in a short period of time, the one that counts the most steps is taken as valid. To do this combination, the signal is split in periods of five seconds and after adding the steps from the segments and the compensated steps from each axis, the maximum value is taken.

3.5. Matlab code

Although this algorithm is intended to be run in an Android device, it has been developed in Matlab. There are a couple reasons for developing it in Matlab; first, the debugging is much easier in Matlab, especially because it allows plotting any signal to understand what is happening any time. The second reason was that it was necessary to optimize certain variables, and as the optimization has taken many hours in a computer, in a smartphone it would have been impossible. The code can be found in the annex.

The first lines of code initialize the variables and do the data preprocessing. After these phases, the actual data analysis is run in a while loop. On each loop, the next time point is given to the function that does the data processing. This algorithm is split into two sections, one that is called if the signal is stable and another for the opposite case. Each section starts by adding the possible end time points –between the minimum and maximum lengths- to a queue, and in a loop it computes the correlation until the queue is

empty or the correlation is higher than the threshold. The rest of the code is simply following what was explained in the previous section

It is important to note that having so many loops, the algorithm may be a bit slow if there are not steps in the signal. But when the signal is stable, it processes it very fast as it only computes a few correlations for each segment, and a segment consists of several hundreds of samples.

3.6. Constants

As it may be expected from the description of the algorithm in the step counting procedure, it relies on several constants. Unfortunately, the value of each of these constants was unknown and had to be either optimized or determined by intuition.

As optimizations require a lot of computational time, the least number of constants that had to be optimized the better. This means that the algorithm could perform slightly better if different values for those constants had been used, but computing all of them was unfeasible.

The first two constants are “th_lsp” and “th_usp”, which are the noise thresholds used to obtain the section points. These have been obtained by observing the values of the acceleration in the accelerometer of a static smartphone.

Then, “delta_y” and “delta_z” correspond to the variation of length that can be found from one segment to the following one. As the variations of gait speed are usually around 2% and at most 5%, a value near to 2% for the slowest gaits was taken. This is the same as 15 samples at a sampling frequency of 200.

The minimum duration of a gait was considered to be of 0.3 seconds, and the maximum of 2.5 seconds. As a segment consists of two steps, “d_min_y” and “d_min_z” have been assigned the value of 130 samples –approximately 0.3*200*2, where 200 is the sampling frequency and 2 to the number of steps. “d_max_y” and “d_max_z” have been assigned to 1000 –2.5*200*2.

The reason for having some constants doubled for the ‘y’ and ‘z’ axes is that in [1] a ‘y’ axis’ segment is considered to be either 2 or 4 steps, without further explanation. As in our observations only 2 steps occurred, the constants were doubled in case they might be useful if the reason was discovered in the future, but left with the same value.

Finally, there are three remaining constants, the correlation thresholds. But as the result highly depends on these variables, even for small changes, these constants could not be guessed intuitively. Instead, they were computed by optimization.

To do this optimization, the code was run 400 times on each of the 44 files with samples, and on each time a random value for each variable was chosen. The first, “th_ia_g”, had a random value between 0.3 and 0.7. “th_ir_g” was allowed to take values between 0.3 and 0.65. Finally “th_low_g” was set to a value that depended on “th_ir_g”, as it has to be smaller. The value was between 0 and 0.3 less than the value that had been assigned to “th_ir_g”. But, as it had been observed that having the same value for “th_ir_g” and “th_low_g” also provided good results, one every four iterations both variables were assigned the same value.

The results of the data analysis and the random values generated on each iteration were stored in variables to be analyzed. The values considered to be optimum were those that provided a minimum error in all the 44 different tests. The criteria for choosing the error was adding the absolute value of the difference between the number of steps estimated and the real number of steps for the 44 results, and selecting the iteration with the lowest value.

3.7. Algorithm pitfalls and improvements

Rotation matrix pitfall

In the first implementation of the algorithm using the procedure described in [1] each part was tested individually to verify that it functioned as expected. In order to test the rotations, two exercises were done, which consisted of recording samples changing the $ß_x$ and $ß_y$ of the smartphone. In both exercises the phone started flat on the table, and was tilted progressively so that the angle that started in 0º ended at 90º; one exercise was for $ß_x$ and the other for $ß_y$.

The result of one exercise was correct, whereas the other did not work properly. After verifying several times the code without success, and doing a lot of testing with different samples, it was noticed that the modulus of the transformed signal sometimes changed. Rotations are unitary, and it was verified that the transformation from the paper was actually non-unitary.

By looking at the determinant of the transformation matrix it was verified that there was a problem with a sign, and it was finally discovered that there was a minus missing. But even after correcting this minus, the transformations did not work as expected, although they were unitary. After more testing, the solution was found to be changing the sign of $ß_x$ and $ß_y$.

So, these were two important errors in the paper that supposed a lot of time wasted. These were not the only issues with the transformation, though. The other issue that is not explained in the paper is described in the methodology section. Because of the noise of the signal, values near to zero changed of sign leading to a transformed signal that was sometimes at $9.8 m/s^2$ and sometimes at $-9.8 m/s^2$, which made the data analysis not work.

Number of steps per segment

In the paper it is said that one segment in the Y axis may be either two steps or four steps. It is not explained why is this possible, or why does this occur to Y and not to Z.

In the implementation of the algorithm both axes had the same distance between consecutive peaks, and hence it was ignored that a segment in Y might have four steps instead of two.

As it will be explained later, the algorithm tends to count less steps than there actually are, because it looses stability. And comparing both axes, Y is usually less stable than Z. One possibility would be that multiplying by two the number of steps in each segment of Y provided better results, but if it was for this reason, it would be cheating.

Correlations

The correlation of portions of the signal is the essence of how the data is analyzed to count steps, but how it is supposed to be applied is not described. As the values are in the range of -1 and 1, the correlation used is the normalized correlation that Matlab has implemented. The problem is that this correlation requires both signals to be of the same length, and usually it is necessary to apply the correlation to signals of different length.

The solution that has been implemented adds zeroes to the shorter signal of the two. It is important to notice that if the shorter signal is the first one, the zeroes are added at the start, whereas if the shortest signal is the second one, the zeroes are added in the end. This is because to compare two signals to verify if they are two periods of a periodical wave, these signals have to be consecutive, without zeroes between them.

Another approach could be using interpolation to have both signals of the same length, but it was not implemented.

Maximum of the correlation

When analyzing the samples to detect steps, one of the most relevant parts of the process is when the correlation of portions of signal is computed and compared to a threshold. This comparison is done to verify if the acceleration in that instant is near to periodical. If for example the threshold is not the appropriate one, a non periodical signal may be considered to be periodical or a periodical signal may be considered non periodical, leading to completely unreliable results.

How this comparison is done, therefore, is crucial for the final results. The original procedure, as described in the methodology section, compares correlations until the result is above the threshold, and then it stops. This is not the best option, since being above the threshold does not imply having the maximum value. If this maximum is achieved when the period of the signal is considered to be larger –i.e. more samples are included in the correlation- the period that will be found will be shorter than the actual period. This is especially important if we remember that the speed at which the user walks may vary, as it becomes more likely to find a period that is too small.

This problem with the periods can be better understood looking at Figure 9 where we have two periods of a sinus and the correlation is computed in two different manners. For the image on the left, the first operand of the correlation goes between the start and the middle of the signal –where you can see a red point- and corresponds to one period of the sinusoid. The second operand of the correlation goes from the middle to the second red point, which less than one period of the sinusoid. This is what would happen if the algorithm had found one segment and was programmed to stop searching as soon as the result of the correlation exceeded the threshold. In a situation like the left image, the correlation is vey high, but we have not reached the second period of he signal –the second step.

The image on the right represents the situation of the signal not being stable, and the algorithm looking for a fragment of signal that can be split into two halves that are correlated –one segment- and thus represent two steps. When the window reaches the size of the second red point, it splits the wave into two halves and the resulting correlation

has a high value. However, it can be seen that it would be better to stop at sample number 100 instead of 94, because it is where the second period of the signal ends.

In both examples it can be seen that when the threshold is exceeded it is better to continue computing correlations until a maximum is found, as it will be seen next.

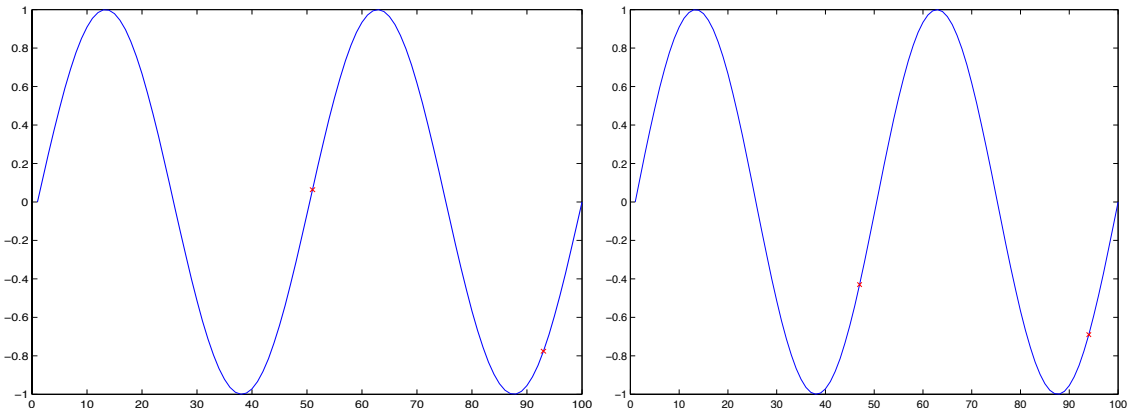


Figure 9 Two examples of incorrect periods with sinusoidal signals. Correlations of 0.9689 and 0.9539 respectively, with 100 samples

To verify how did the correlation evolve in a real situation, the algorithm was stopped while running at a random point and the correlation was evaluated moving the window 50 samples to the left and right of the point at which the threshold had been reached. This can be seen in Figure 10, where we have two examples. The algorithm would stop at the position 50 in both examples. In the first one we can see that the correlation does not improve a lot from the position 50 to the maximum at 65; however, in the second example the difference is significant. Although there are some cases like the one in the left, where the difference is minor, there are several situations similar to the second example, so it seems reasonable to introduce this improvement.

To verify the behavior of the algorithm, with and without the modification, the proportion of the stable signal was compared and with the modification there was much more signal considered stable. This was a positive result, because in the samples tested the user was walking so most of the signal was supposed to be stable.

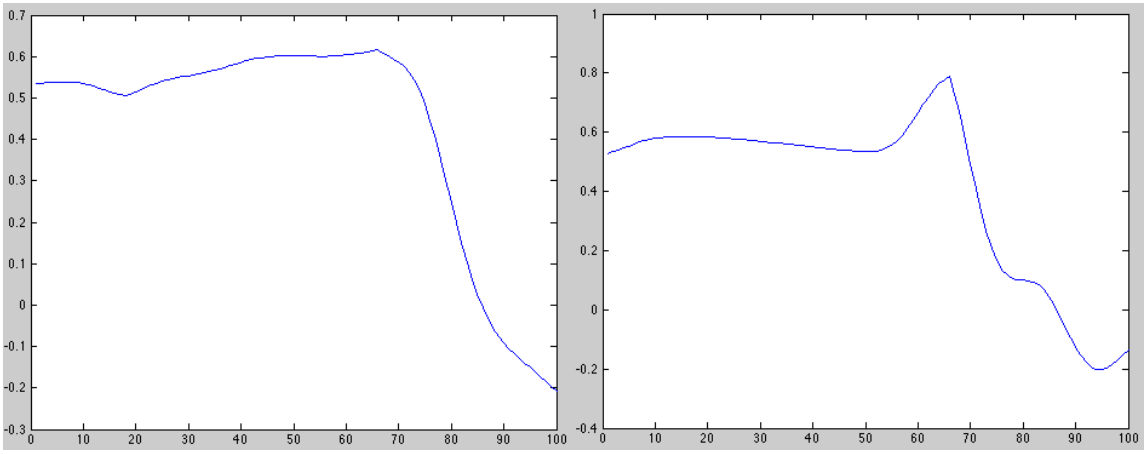


Figure 10 Values of correlation when changing window size

Variance

In some tests in which the user was holding the smartphone in the hand and using it, but without walking, the algorithm counted some steps –although not many. Also, the same occurred while seating in the bus.

To reduce these errors, a simple addition was made. It was noticed that the variance of the signal was much smaller when the user was standing or sitting with the phone in the hand, compared to the user walking.

With this difference in mind, a threshold for the variance may be established to avoid counting steps when the variance was too small for the user to be walking. This has not been added to the algorithm because it was noticed and added in the end of the project and there was not enough time.

A median filter was applied to the variance because it had an impulsive behavior in cases where the user was not walking, and after the filtering the signal in the bus or while standing had a very low variance –almost always below 0.5 m^2/s^4-, whereas the signal walking had higher values –as it can be seen in Figure 11.

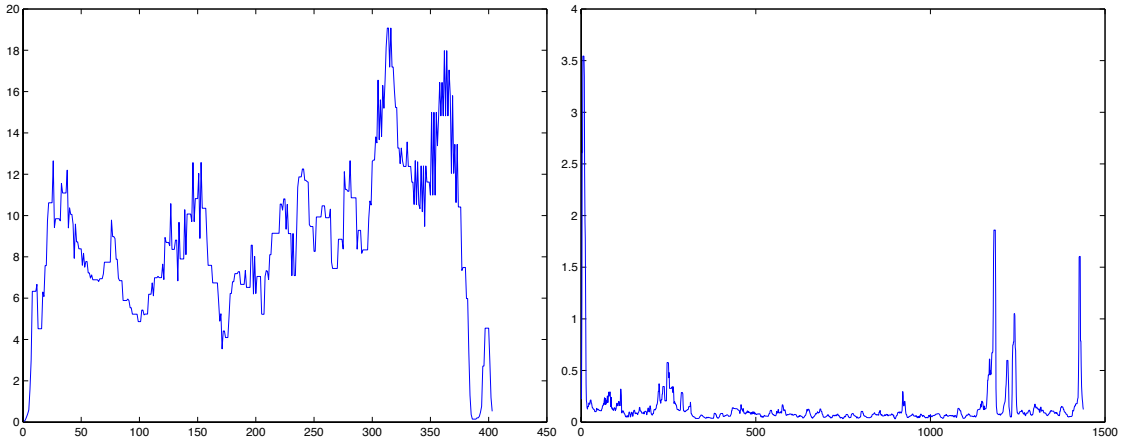


Figure 11 Comparison of variances. Left is walking; right is seated in the bus. m^2/s^4 and samples

Compensation of the steps lost

As it has been explained before, when the stability of the signal is lost the steps are estimated to reduce the number of steps that are missed. Nevertheless, the approach described is not the same than how it is supposed to be done according to the paper.

In the paper, the compensation is done from the instant that the stability is lost to the next instant in which the signal is stable again. This should provide excellent results when tested with the user walking all the time, but if the user stops walking for some time and resumes afterwards, it will count a lot of steps that are not there. Although the results will be better in most cases, this compensation has been rejected because it implies counting steps that do not exist.

4. Results

In order to measure the accuracy of the step counting, the database was analyzed. As previously commented, this database consists on eleven users and four tests done by each one of the users. The database was split in train and test, using subjects 1, 6 and 8 as the test partition, and the rest for training.

As explained in methodology, all the data was analyzed 400 times trying different random values for the three variables to be optimized. The procedure of training consisted of selecting the values where there was the minimum error after adding the absolute value of the errors of each test done by each user -32 tests in total.

This value was found in the iteration number 332, and the optimal values found are *th_ia=0.3088* and *th_low=th_ir=0.3231*. In this iteration, the error –defined as the absolute difference between the actual steps walked and the estimation divided by the actual steps walked- for the train database was 5.62%, and in the test recordings was 7.65%. The total error of Noom was 27.93%.

Subject	**Pocket**		**Using the phone**		**Phone in hand**		**Backpack**	
	Noom	**Real**	**Noom**	**Real**	**Noom**	**Real**	**Noom**	**Real**
1	-119	299	8	308	-91	298	209	311
2	-118	304	-207	305	-72	301	-118	325
3	-28	299	-134	293	-228	294	-72	305
4	88	281	48	289	-85	300	22	300
5	-81	272	-22	259	-77	264	-29	268
6	-153	276	-187	268	-89	278	-20	272
7	-31	274	-17	270	-163	276	-38	270
8	158	274	46	273	-59	268	13	278
9	-4	280	-18	273	-106	277	-25	279
10	-97	293	-31	280	-56	288	55	303
11	-92	293	-48	299	-150	295	-26	284
Mean	-43.36		-51.09		-106.91		-2.63	

Table 1 Relative error of Noom and total steps

Subject	Pocket		Using the phone		Phone in hand		Backpack	
	Noom	Our	Noom	Our	Noom	Our	Noom	Our
1	-39.8%	2.3%	2.6%	-7.4%	-30.5%	13%	67.2%	11.6%
2	-38.8%	-5.7%	-67.9%	-6.6%	-23.9%	-9.5%	-36.3%	-9.5%
3	-9.36%	-1.7%	-45.7%	-17%	-77.6%	2.4%	-23.6%	-6.6%
4	30%	3.1%	17.1%	4.6%	-29.5%	-6.6%	7.3%	5.6%
5	-29.8%	-0.4%	-8.5%	0.4%	-29.1%	7%	-10.8%	7.8%
6	-54.5%	0.7%	-64.7%	13.2%	-29.7%	8.3%	-6.67%	6%
7	-11.2%	-2.5	-6.34%	4%	58.6%	4.9%	-14%	7%
8	56.4%	-5.7%	16.9%	8.1%	-21.3%	-4%	4.66%	9.3%
9	-1.56%	7.3%	-6.6%	9.5%	-39.6%	4%	-9%	8.6%
10	-25.4%	4.4%	-11.5%	-3.7%	-20.3%	2.2%	-20.4%	0%
11	-31.4%	3.1%	-16.1%	6.4%	-50.9%	6.1%	-9.2%	12%

Table 2 Error of Noom and our algorithm

Train database		Pocket	Using the phone	Phone in hand	Phone in backpack
Noom	mean	23.44%	22.46%	41.18%	16.31%
	std	0.2306	0.2644	0.1978	0.1737
Our algorithm	mean	3.53%	5.81%	6.02%	7.14%
	std	0.0444	0.0609	0.0793	0.0771

Table 3 Results for each way of carrying the smartphone tested, for the train database

Test database		Pocket	Using the phone	Phone in hand	Phone in backpack
Noom	mean	50.23%	28.05%	27.17%	26.18%
	std	0.6023	0.4356	0.0510	0.3978
Our algorithm	mean	7.07%	9.78%	4.01%	8.96%
	std	0.0947	0.0278	0.0404	0.0280

Table 4 Results for each way of carrying the smartphone tested, for the test database

Whole database		Pocket	Using the phone	Phone in hand	Phone in backpack
Noom	mean	30.71%	23.99%	37.36%	19.00%
	std	0.3317	0.2951	0.1794	0.2719
Our algorithm	mean	4.5%	6.89%	5.47%	7.64%
	std	0.0586	0.0616	0.0696	0.0711

Table 5 Results for each way of carrying the smartphone tested, including the 11 users

The Table 1 shows the results from the itinerary described in methodology. The columns represent the data of users walking with the smartphone in the pocket, of the user using the phone, of the user balancing the smartphone in his hand, and of the user carrying the smartphone in a backpack, respectively. For each way of carrying, the real number of steps and the relative error of the Noom's estimation are written. The error and the variation of the error from test to test are large.

The Table 2 presents the error of Noom next to the error in our algorithm, both as a percentage, also differentiating between users and tests. The error is defined as the division between the estimated number of steps and the real number of steps. A comparison at first sight shows much smaller errors in our algorithm and les variability, but this will be better analyzed in the following tables.

Table 5 shows the mean and standard deviation of the error for Noom and for our algorithm, with each activity in a column and using the whole database. Here the error is defined as in the previous paragraph. Here it can be seen as the mean error rate for Noom is much larger than for our algorithm, and the standard deviation is also much larger for Noom.

For Noom it is easier to compute the number of steps with the phone in the backpack, and the standard deviations are similar except for the phone in the hand –but this case has the worst error rate. Our algorithm behaves slightly better with the phone in the pocket, either considering the mean error or the standard deviation, but in general the results are similar for all activities.

The Tables 3 and 4 are the same than the 5th, but splitting the database into test and train, respectively. The results are similar, with the particularity of a lower error rate in the case of “phone in hand” for the test database. Ignoring this particularity, the test database has worse results than the train database as it might be expected, but the difference is relatively small.

The positive conclusions from these results are the low error rate, the low standard deviation, and the fact that the results are nearly independent of the activity being performed. Also, the error rates are similar between different users, although there are some small differences.

When tested on the bus, or while standing with the device in the hand, it did count several steps that should not have been counted. Nevertheless, this should not happen with the addition of the variance check. Noom also counted some steps while in the bus.

Another drawback of this algorithm is that it generally counts fewer steps than it should, because it looses stability. However, in these results this does not occur because the correlation thresholds have been set to lower values to avoid this. This leads to more false positives, which may be corrected with the variance check described before.

Also, it has been observed that the results may differ a lot if the values for the optimized variables are changed. If the values for the variables are incorrectly chosen, the results are completely unreliable. The error rate may become higher than 50%. This behavior was not expected because the values given in the paper were approximate; instead, it was discovered when observing the results. Such a high dependence on the values of these variables is a drawback, as it might mean that the algorithm will perform worse when tested in a larger database.

It has not been possible to achieve an error rate as low as it was set in the goals of this project -5%. The main difficulties for achieving a high accuracy are the noise of the signal and the fact that only one sensor has been used –the accelerometer.

The paper in which this project is based had an error rate below 2.5%, usually around 0.1%, which is very far from our results.

The first possible reason for this difference in success is that in such paper, the number of users used is not mentioned, so it is likely to be one single user. With one user it is easier to achieve good results, but these results are less reliable.

The other possible reason is the one explained in “compensation of the steps lost”, in the previous section. In the original algorithm, when stability is lost all the steps are estimated from the previous speed, until another stable segment is found. However, this has been considered to be undesirable. It may provide better results in our tests, as the user is mostly walking, but if the user stops and resumes walking many inexistent steps will be counted.

5. Budget

It has been estimated that 370 hours have been dedicated to this thesis, and considering the cost of a junior engineer to be of 8€ per hour, this part of the cost is a total of 2960€.

Considering the total cost of a computer to be of 800€, a life expectancy of 5 years, and six months of use, the part of the cost of this thesis related to the computer is of 80€.

Finally, the Matlab license for students has a price of 69€ [14] and the parallel computing toolbox of 7€. Adding the IVA the total cost of the software is of 92€.

Therefore, the total cost of this project is estimated to be of 3132€.

Description	Cost	Quantity	Total
Hours of work	8€	370	2960€
Computer	800€	1/10	80€
Matlab license	92€	1	92€
Total	-	-	**3132€**

Table 6 Budget

6. Conclusions and future development:

As seen previously, this thesis had as goals developing and Android application to count steps with a smartphone, without restrictions on how the device is carried and a very low error rate. The algorithm implementation was more difficult than expected, which meant that there was no time to port it from Matlab to Android, and the results were a bit worse than the objective.

The first conclusion is that the algorithm that has been developed outperforms Noom significantly, taking into account both the mean error and the standard deviation of the error.

Also, three parameters of the algorithm have been optimized and others have been deducted, to achieve the best results given the database used. There were not clear values at [1] for these parameters, so it was necessary to determine them.

Several improvements have been applied to the original algorithm. Some of these improvements have benefits in the reliability of the algorithm and others in the accuracy.

A database of 11 subjects has been collected. This database reinforces the value of the results extracted –as several databases used in similar projects are smaller. It is also useful for other related projects, as it will not be necessary to generate the data again, or it will be possible to achieve larger databases with part of the work done.

Also, an observation made while dealing with this project is that when scheduling it is important to take into consideration that there may be unexpected setbacks or that some things may be more difficult than it seems. These unexpected difficulties have required changing one work package for another, in this project.

Now that the conclusions have been explained, it is relevant to describe the future development points, so that there is a guide for improving the algorithm.

The first obvious improvement to apply would be using more sensors. The signal is very noisy and estimating the gravity to obtain the tilt has some drawbacks, as reliability and delay. If instead, the gyroscope and magnetometer had been used, the tilt could be extracted easily and reliably. The problem of using more sensors is that they drain more battery.

Another possible improvement could be using interpolation when computing the correlation between signals of different lengths, instead of adding zeroes, but it is not clear whether the improvement will be worthwhile.

For debugging the code, all parts of it were thoroughly tested –an example is how the issue with the rotation was found. For testing the behavior of the stability part, the signal was being plotted with colors for the stability or lack of stability. This has allowed to better understanding the behavior of the algorithm, as many samples have been observed.

This observation has allowed seeing beforehand that this approach for counting steps tends to count fewer steps, and rarely counts more, with high correlation thresholds. The reason is that when it is capable of finding the steps, it finds the exact amount of steps and no more; but when it looses the stability, it does not count steps at all. Therefore, as

it can be said the upper bound of the step count is the actual number of steps, which is worse than how most algorithms actually behave –being both above and below. After hours of using an app with some error rate but unbiased, the instants in which it counts more steps than it should may compensate those instants when it counts less than it should.

For this reason, this approach might be a bad path. Nevertheless, there are some possible future improvements that may change this.

As discussed in the last part of "algorithm issues and improvements", the compensation in the paper was not correct, but it would offer better results in most cases. The most reasonable approach to improve the step counting with an approach similar to the one followed in this thesis could consist of actually using the compensation in the paper – when the signal looses stability, steps are estimated to be at the same speed than before, until the signal is stable again- but with some modifications.

The first and less complex modification would be checking the amount of time in the interval between the loss of stability, and the recovery of stability. If both events are distant in time, it is more likely that the user will have stopped walking, and it will be reasonable not to compensate. On the other hand, if stability is recovered soon, it most likely means that the user has moved differently, thus loosing stability, but he has actually been walking.

This approach is compliant with the behavior observed in many samples. As the plots of the signal's stability were done several times, what happened to stability was observed repeatedly. In these observations it could be seen that stability usually lasted for a few seconds, and was lost and recovered afterwards. This pattern occurs continuously and is the reason for most missed steps.

In Figure 12 the stability after running the algorithm in one file of samples can be seen. The green part of the signal is stable, the yellow one corresponds to the periods of time when stability is lost –there is no yellow because th_low_g=th_ir_g, and blue means there is no stability at all. This behavior is seen in all the tests done to all the subjects. Also, when the thresholds are higher, it behaves much worse with stability being lost more commonly.

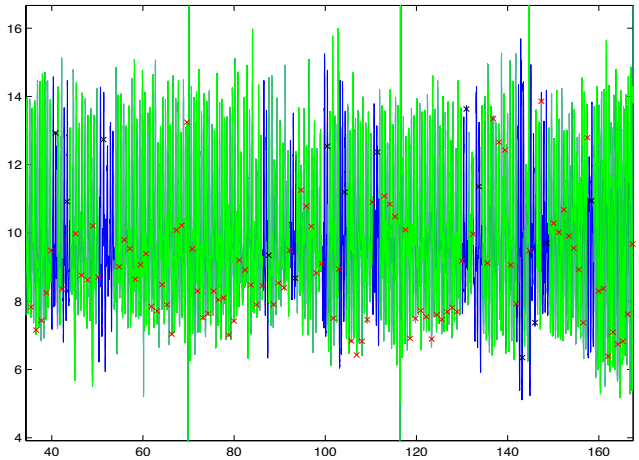


Figure 12 Stability of the signal

The other possible modification would be adding a layer for detecting whether the user is actually walking, separate from the algorithm to count steps. This addition would probably

be made with machine learning –as it has been seen in the [9] or especially in [10], for example. This would prevent the step counting from being done when the user was not walking, thus saving a lot of computing.

The issue with the bus, and the simple addition of the variance check also supports the use of machine learning to aid the step count algorithm. Without any check, the algorithm might count several steps in situations where there were none.

The machine learning would also allow estimating the steps when there is no stability, because it would be known that the user is really walking. Nevertheless, if a machine learning layer is added it may be reasonable to opt for a simpler algorithm for the step counting, as the approach in this thesis does the walking detection and step counting at the same time.

To sum up, to reliably count steps it is necessary to either try another algorithm with a better performance, or apply some improvements to this one, such as using more sensors and adding a machine learning layer to modify the behavior of the step counting layer.

Bibliography:

[1] Meng-Shiuan Pan, Hsueh-Wei Lin. "A Step Counting Algorithm for Smartphone Users: Design and Implementation". *IEEE Sensors Journal*, vol. 15, no. 4, April 2015. DOI: 10.1109/JSEN.2014.2377193.

[2] Bao, L. and Intille, S. 2004. Activity Recognition from User- Annotated Acceleration Data. *Lecture Notes Computer Science 3001*, 1-17.

[3] Kwapisz, Jennifer R., Gary M. Weiss, and Samuel A. Moore. "Activity Recognition Using Cell Phone Accelerometers." *ACM SIGKDD Explorations Newsletter SIGKDD Explor. Newsl.* 12.2 (2011): 74. Web.

[4] Lee, Jung Ho, Shin, Beomju, Lee, Seok, Park, Jinwoo, Kim, Jaehun, Kim, Chulki, Lee, Taikjin, "A Step Length Estimation Based on Motion Recognition and Adaptive Gait Cognition Using a Smartphone", *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)*, Tampa, Florida, September 2014, pp. 243-249.

[5] Corina Kim Schindhelm. "Activity Recognition and Step Detection with Smartphones: Towards Terminal Based Indoor Positioning System". *2012 IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. Sydney, NSW. pp 2454-2459. DOI: 10.1109/PIMRC.2012.6362769.

[6] Ando, B., Baglio, S, Lombardo, C.O., V. Marletta. "An advanced tracking solution fully based on native sensing features of smartphone". *2014 IEEE Sensors Applications Symposium (SAS)*. Pp 141-144.

[7] Rui Terra, Lino Figueiredo, Ramiro Barbosa, and Ricardo Anacleto. 2013. Step count algorithm adapted to indoor localization. In *Proceedings of the International C* Conference on Computer Science and Software Engineering* (C3S2E '13), Ana Maria Almeida, Jorge Bernardino, and Sudhir Mudur (Eds.). ACM, New York, NY, USA, 128-129. DOI=10.1145/2494444.2494457

[8] Kinh Tran; Tu Le; Tien Dinh, "A high-accuracy step counting algorithm for iPhones using accelerometer," *Signal Processing and Information Technology (ISSPIT), 2012 IEEE International Symposium on*, vol., no., pp.000213,000217, 12-15 Dec. 2012. DOI: 10.1109/ISSPIT.2012.6621289

[9] Agata Brajdic and Robert Harle. 2013. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing* (UbiComp '13). ACM, New York, NY, USA, 225-234. DOI=10.1145/2493432.2493449

[10] Tomlein, M., Bielik, P., Krátky, P., Mitrík, S., Barla, M., Bieliková, M. "Advanced pedometer for smartphone-based activity tracking". *International Conference on Health Informatics (HEALTHINF 2012)*

[11] M. Alzantot and M. Youssef, "UPTIME: Ubiquitous pedestrian track- ing using mobile phones," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2012, pp. 3204–3209.

[12] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2012, pp. 197–210.

[13] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2012, pp. 293–304.

[14] http://es.mathworks.com/academia/student_version/

Glossary

Noom: app with which our algorithm is compared. It was the app that seemed to be the best one in the Play Store according to downloads and user reviews.

Stability: when the algorithm considers the user is walking, the signal is considered to be stable. On the other hand, if it does not know if the user is walking, the signal is considered to be non-stable. When at least two consecutive segments are found, the signal is considered stable.

Segment: portion of signal in which there are two steps. The first half of a segment is correlated to the second half.

Section point: point considered to be relevant for counting steps, and candidate to being starting point of the first segment of a series of segments. A point is a section point if the previous point was between the noise thresholds, and the current point has crossed on of these thresholds.

Starting point: first time point of a portion of signal to be analyzed.

Compensate: to estimate the number of steps when stability is lost but the user is expected to be walking.

Appendix

In the optimization, the order of the results matrix does not correspond to the correct order of activities –it corresponds to the chronological order in which samples were taken. The next line is the proper order.

```
order                                                                    =
[1,3,2,4,6,7,5,8,9,11,10,12,13,14,15,16,17,19,18,20,22,23,21,24,...
    25,27,26,28,29,30,31,32,33,34,35,36,38,39,37,40,41,42,43,44];
```

Main function to count steps

```
function [segments,c_segments, segments_time, compensate_time, ...
    compensate_start_time,  variances]  =  step_count(ax,  ay,  az,  t,
th_ia_g, th_low_g...
    ,th_ir_g, is_z_axis, modify_to_find_max_g)
%   [segments,c_segments, segments_time, compensate_time, ...
%       compensate_start_time]  =  step_count(ax,  ay,  az,  t,  th_ia_g,
th_low_g...
%    ,th_ir_g, is_z_axis, modify_to_find_max_g)
%
%       -- OR --
%
%   [segments,c_segments, segments_time, compensate_time, ...
%    compensate_start_time] = step_count()
%
%   segments: number of stable segments (2*segments is a reference of
%       the number of steps)
%   c_segments: number of non-stable segments
%   segments_time: time at which stable segments are found
%   compensate_time: time at which semi-stable segments end
%   compensate_start_time: time at which semi-stable segments start
%   variances: variances of the signal
%
%   ax, ay, az, t: acceleration for each axis and time
%   th_ia_g, th_low_g, th_ir_g: correlation thresholds being optimised
%   is_z_axis: whether 'z' or 'y' axis should be analysed
%    modify_to_find_max_g: whether the algorithm modification to find
better
%       maxima (of correlation) should be applied
%
%
```

```
%% General variables

% Variables to choose what to do
plot_stability = 1; % To plot data, for debugging

if (nargin == 0)
    optimizing = 0;
else
    optimizing = 1;
end
if (optimizing)
    z_axis = is_z_axis;
    modify_to_find_max = modify_to_find_max_g;
else
    z_axis = 1;
    modify_to_find_max = 1;
end



init_stable_t = [];
end_stable_t = [];
init_semistable_t = [];
end_semistable_t = [];
segments_time = []; %Time when a segment is recorded, steps increased
compensate_time = []; %Time when a segment is compensated
compensate_start_time = []; %Start of a segment that is compensated


segments = 0;
c_segments = 0;

filter_length = 200;
%% Values
step_c= 0;
```

```
stable = 0; % whether the current portion of signal is stable
tp_cur = 1; % current time point
tp_prev = 0; % previous segment's initial time
window = 0; % length of a segment, used to find the next one
th_lsp = -0.3; % noise negative threshold
th_usp = 0.3; % noise positive threshold
delta_y = 15; % y window variation
delta_z = 15; % z window variation
d_min_y = 2*65; % min length of y segment
d_min_z = 2*65; % min length of z segment
d_max_y = 2*500; % max length of y segment
d_max_z = 2*500; % max length of z segment
%correlation thresholds
th_ia = 0.3088;
th_ir = 0.3231;
th_low = 0.3231;

%Reading thresholds if they are provided
if (optimizing)
    th_ia = th_ia_g;
    th_low = th_low_g;
    th_ir = th_ir_g;
end

%% Adapting Data

% Reading samples
if (~optimizing)
    [ax, ay, az, t]=accelData('AcelerometerData_15_04_16_11_18_36.dat','/cris/',0);
end
%Sometimes samples have the same or previous timestamp. Reordering
[t,i]=sort(t);
ax=ax(i);
ay=ay(i);
az=az(i);
k=find(diff(t));
t=t(k);
ax=ax(k);
```

```
ay=ay(k);
az=az(k);
% Interpolating so that sampling is at 200Hz
axi=interp1(t,ax,t(1):1/filter_length:t(length(t)));
ayi=interp1(t,ay,t(1):1/filter_length:t(length(t)));
azi=interp1(t,az,t(1):1/filter_length:t(length(t)));
%Filtering the signal to estimate the gravity
axif=filter(ones(size(1:filter_length))/filter_length,1,axi);
ayif=filter(ones(size(1:filter_length))/filter_length,1,ayi);
azif=filter(ones(size(1:filter_length))/filter_length,1,azi);
gx = axif;
gy = ayif;
gz = azif;
    %The fist samples are not usable, because of the filtering
gx(1:filter_length) = gx(filter_length+1);
gy(1:filter_length) = gy(filter_length+1);
gz(1:filter_length) = gz(filter_length+1);

%Prevent changing from +0 to -0 if near to zero
k=find(abs(gz)<0.1);
k2=find(abs(gy)<0.1);
gzm=gz;gzm(k)=0.1*ones(size(k)).*sign(gz(k));
gym=gy;gym(k2)=0.1*ones(size(k2)).*sign(gy(k2));
if mean(gzm)>0 && mean(gzm)-std(gzm)<0
    dev=mean(gzm)-std(gzm);
    gzm = gzm-dev;
elseif mean(gzm)<0 && mean(gzm)+std(gzm)>0
    dev=mean(gzm)+std(gzm);
    gzm = gzm-dev;
end
% Obtaining the device's tilt
beta_y = (unwrap(atan(gx./gzm)));
beta_xd = (gx.*sin(beta_y)+gzm.*cos(beta_y));
beta_x = (atan( -gym./(beta_xd) ));

% Applying the rotation to the acceleration
x = axi .* cos(-beta_y) + azi .* sin(-beta_y);
y = ayi .* cos(-beta_x) + (axi.*sin(-beta_y) - azi.*cos(-beta_y) ) .*
sin(-beta_x);
```

```
z = ayi .* sin(-beta_x) - (axi.*sin(-beta_y) - azi.*cos(-beta_y) ) .*
cos(-beta_x);


%temporal variable after resampling
t2 = linspace(0,t(length(t)),length(z));



%% Relevant points


%Variables for debugging
no_1_segment = 0;
no_2_segment = 0;
no_3_segment = 0;
corr_1 = 0;
corr_2 = 0;


tp_cur = 1; %current time point


% Selecting data from z axis or from y axis
if (z_axis)
    gravity = mean(z); %in this axis noise is not around 0, but around
the gravity value
    %finding section points (th_usp or th_lsp is surpassed)
    section_point = find([false(1) (z(1:length(z)-1)<th_usp+gravity & ...
        z(2:length(z))>=th_usp+gravity)         |         (z(1:length(y)-
1)>th_lsp+gravity...
        & z(2:length(z))<=th_lsp+gravity)]);
    signal = z;
    delta = delta_z;
    d_min = d_min_z;
    d_max = d_max_z;
else
    %finding section points (th_usp or th_lsp is surpassed)
    section_point = find([false(1) (y(1:length(y)-1)<th_usp &...
        y(2:length(y))>=th_usp) | (y(1:length(y)-1)>th_lsp & ...
        y(2:length(y))<=th_lsp)]);
    signal = y;
    delta = delta_y;
    d_min = d_min_y;
    d_max = d_max_y;
```

```
end

%% Data analysis
%obtaining first time point
[not_empty, tp_cur] = nextTpcur(section_point, tp_cur, signal);
while (not_empty)

    % Analizing segment of data
    [segments, c_segments, stable, window, tp_prev, tp_cur,...
    no_1_segment,no_2_segment,no_3_segment,corr_1,corr_2,...
    init_stable_t, end_stable_t, init_semistable_t, end_semistable_t,...
    compensate_time, compensate_start_time, segments_time]...
    = segment_finding(...
        segments, c_segments, signal, stable, t2, tp_cur,...
        tp_prev, window, th_ia, th_ir, th_low, delta, d_min, d_max,...
        no_1_segment,no_2_segment,no_3_segment,corr_1,corr_2,...
        init_stable_t,          end_stable_t,          init_semistable_t,
end_semistable_t,...
        segments_time, compensate_time, compensate_start_time,...
        modify_to_find_max);

    % Obtaining next point to analyze. If none, while will be exited
    if (~stable)
        [not_empty, tp_cur] = nextTpcur(section_point, tp_cur, signal);
    end

end

lengthVariances = 100;
index = 1:(length(signal)/lengthVariances-1);
variances = 0*index;
for i=index
    variances(i) = var( signal((i*lengthVariances-lengthVariances/2):...
        (i*lengthVariances+lengthVariances/2)) );
end

% Plot the signal, with the stable and semistable segments (debugging)
if (plot_stability)
    figure('name', 'Signal with stability')
```

```
    plot(t2,signal)
    hold on
    for (i=1:length(init_stable_t))

plot(t2(init_stable_t(i):end_stable_t(i)),signal(init_stable_t(i)...
            :end_stable_t(i)),'g');
    end
    for (i=1:length(init_semistable_t))
        plot(t2(init_semistable_t(i):end_semistable_t(i)),signal(...
            init_semistable_t(i):end_semistable_t(i)),'y');
    end
    plot(t2(segments_time), signal(segments_time),'xr')
    plot(t2(compensate_time), signal(compensate_time),'xk')
    hold off
    segments
    c_segments

end

end



function [segments, c_segments, stable, window, tp_prev, tp_cur,...
    no_1_segment,no_2_segment,no_3_segment,corr_1,corr_2,...
    init_stable_t, end_stable_t, init_semistable_t, end_semistable_t,...
    compensate_time, compensate_start_time, segments_time] ...
    = segment_finding(...
    segments, c_segments, signal, stable,t2,...
    tp_cur,  tp_prev,  window,  th_ia,  th_ir,  th_low,  delta,  d_min,
d_max,...
    no_1_segment,no_2_segment,no_3_segment,corr_1,corr_2,...
    init_stable_t, end_stable_t, init_semistable_t, end_semistable_t,...
    segments_time, compensate_time, compensate_start_time,...
    modify_to_find_max)

if (~stable) % Not stable, searching for two segments
    % Searching for first segment
    q_tp = linspace(tp_cur+d_min, tp_cur+d_max, d_max-d_min+1);
```

```
    keep_searching = 1;
    while (keep_searching)
        if (isempty(q_tp))
            no_1_segment = no_1_segment +1;
            return;
            %Empty queue, segments not found
        else
            tp_j = q_tp(1); %remove 2 first from the queue
            q_tp  =  q_tp(3:length(q_tp));  %removing  2  to  have  an  even
length

            correlation = correlate(signal, tp_cur, tp_j);
            if (correlation < th_ia)
                %not correlated, next value
            else
                %  (modification)  looking  for  the  first  maximum  of
correlation
                if (modify_to_find_max)
                    corrVal = correlation;
                    for a = (1:25)
                        newCorr = correlate(signal, tp_cur, tp_j+2*a);
                        if (newCorr < corrVal)
                            tp_j = tp_j+2*a-2;
                            break;
                        else
                            corrVal = newCorr;
                        end
                    end
                end
                corr_1 = corr_1+1;
                window = tp_j-tp_cur;
                keep_searching = 0;
            end
        end
    end

    %Searching for second segment
    q_tp = linspace(tp_j+window-delta, tp_j+window+delta, 2*delta+1);
    keep_searching = 1;
    while (keep_searching)
```

```
        if (isempty(q_tp))
            no_2_segment  = no_2_segment +1;
            return;
            %Empty queue, segment not found
        else
            tp_k = q_tp(1); %remove first
            q_tp = q_tp(2:length(q_tp));
            correlation = correlate(signal, tp_cur, tp_j, tp_k);
            if (correlation < th_low)
                %not correlated, next value
            else
                % Correlated, we have 2 consecutive segments

                %  (modification)  looking  for  the  first  maximum  of
correlation
                if (modify_to_find_max)
                    corrVal = correlation;
                    for a = (1:50)
                        newCorr   =   correlate(signal,   tp_cur,   tp_j,
tp_k+a);
                        if (newCorr < corrVal)
                            tp_k = tp_k+a-1;
                            break;
                        else
                            corrVal = newCorr;
                        end
                    end
                end
                stable = true;
                window = tp_k-tp_j;
                tp_prev = tp_j;
                tp_cur = tp_k;
                [segments,segments_time] = record_segment(segments, ...
                    segments_time, tp_k);
                keep_searching = 0;
                corr_2 = corr_2+1;
                init_stable_t = [init_stable_t tp_prev];
                end_stable_t = [end_stable_t tp_cur];
            end
        end
```

```
    end
else % stable
    if (window-delta < d_min) %verifying that segment is not too small
        q_tp = linspace(tp_cur+d_min, tp_cur+d_min+delta, delta+1);
    else
        q_tp    =    linspace(tp_cur+window-delta,    tp_cur+window+delta,
2*delta+1);
    end
    keep_searching = 1;
    while (keep_searching)
        if (isempty(q_tp))
            no_3_segment = no_3_segment+1;
            %Empty queue, not found
            time = tp_cur+window;
            if (time>length(t2))
                time  =  length(t2);  %it  might  be  later  than  the  last
sample
            end
            [c_segments, compensate_time, compensate_start_time] = ...
                compensate_SC(c_segments, compensate_time, ...
                compensate_start_time,                                  time,
segments_time(length(segments_time)));
            tp_prev = 0;
            window = 0;
            stable = false;
            return;
        else
            tp_j = q_tp(1); %remove first
            q_tp = q_tp(2:length(q_tp));

            correlation = correlate(signal, tp_prev, tp_cur, tp_j);
            if (correlation < th_low)
                %not correlated, next value
            elseif (correlation >= th_low) && (correlation < th_ir) %was
<th_ia
                %The user has moved it a bit but is still walking
                init_semistable_t = [init_semistable_t tp_cur];
                end_semistable_t = [end_semistable_t tp_j];
                time = tp_cur+window;
                if (time>length(t2))
```

```
                    time = length(t2); %it might be later than the last
sample
                end
                [c_segments,    compensate_time,    compensate_start_time]
= ...
                    compensate_SC(c_segments, compensate_time, ...
                    compensate_start_time,                          time,
segments_time(length(segments_time)));
                tp_prev = 0;
                window = 0;
                stable = false;
                tp_cur  =  tp_j;  %  Not  in  the  paper,  but  considered
reasonable
                return;
            else
                %Segment correlated. Update and look for the next

                % looking for the first maximum of correlation
                if (modify_to_find_max)
                    corrVal = correlation;
                    for a = (1:50)
                        newCorr  =  correlate(signal,  tp_prev,  tp_cur,
tp_j+a);
                        if (newCorr < corrVal)
                            tp_j = tp_j+a-1;
                            break;
                        else
                            corrVal = newCorr;
                        end
                    end
                end
                window = tp_j-tp_cur;
                keep_searching = 0;
                tp_prev = tp_cur;
                tp_cur = tp_j;
                [segments,segments_time]    =    record_segment(segments,
segments_time, tp_j);
                end_stable_t(length(end_stable_t)) = tp_cur;
            end
        end
    end
```

```
end


end


function [not_empty, next_tp_cur_t] = nextTpcur(section_point, tp_cur_t,
signal)

not_empty = 1;
tp_cur = find(section_point>tp_cur_t,1);
if (tp_cur+1 > length(section_point))
    not_empty = 0;
    next_tp_cur_t = 0;
else
    tp_cur = tp_cur+1;
    next_tp_cur_t = section_point(tp_cur);
    if (isempty(tp_cur))
        not_empty = 0;
    end
end


end

function    [segments,    segments_time]    =    record_segment(segments,
segments_time, time)
segments_time = [segments_time time];
segments = segments+1;

end

function [c_segments, compensate_time, compensate_start_time] = ...
    compensate_SC(c_segments, compensate_time, compensate_start_time, ...
    time, start_time)
compensate_time = [compensate_time time];
compensate_start_time = [compensate_start_time start_time];
c_segments = c_segments+1;
```

```
end


function [result] = correlate(signal, first, intermediate, last)
if (nargin==3)
    if (intermediate > length(signal))
            result = 0;
            return;
    end
    if (mod(intermediate-first,2)==0)
        result = corr(signal(first:(first+floor((intermediate-first-1)...
            /2)))',signal((first+floor((intermediate-first-
1)/2)+1):intermediate-1)');
    else
        result      =      corr(signal(first:(first+floor((intermediate-
first)/2)))'...
            ,signal((first+floor((intermediate-
first)/2)+1):intermediate)');
    end


else
    if (last > length(signal))
            result = 0;
            return;
    end
    if (last-intermediate-1 > intermediate-first)
        a            =            [zeros(1,last-2*intermediate-1+first),
signal(first:intermediate)];
        b = signal(intermediate+1:last);
    else
        a = signal(first:intermediate);
        b = [signal(intermediate+1:last), zeros(1,2*intermediate-first-
last+1)];
    end
    result = corr(a',b');
end


end
```

Function to combine steps from both axes:

```
function [total_steps] = combine_steps(segments_timez, ...
    segments_timey, compensate_start_timez, compensate_timez, ...
    compensate_start_timey, compensate_timey)
%   [total_steps] = combine_steps(segments_timez, ...
%    segments_timey, compensate_start_timez, compensate_timec1z, ...
%    compensate_start_timey, compensate_timey)
%
%   For 200 iterations
%
%   total_steps: the final number of steps
%   segments_timez: time when a stable segment is found, in z axis
%   segments_timey: time when a stable segment is found, in y axis
%    compensate_start_timez: start time of a semi-stable segment, in z
axis
%    compensate_start_timey: start time of a semi-stable segment, in y
axis
%   compensate_timez: end time of a semi-stable segment, in z axis
%   compensate_timey: end time of a semi-stable segment, in y axis
%



temporal_block = 1000; % 5seconds. Length of each block of time
iterations = 400; % Number of iterations
iterations = 1;


total_c_steps = 0;
total_steps = 0*(1:iterations);
for j=1:iterations
    if          (segments_timez{j}(length(segments_timez{j}))          >
segments_timey{j}(length(segments_timey{j})))
        max_length = segments_timez{j}(length(segments_timez{j}));
    else
        max_length = segments_timey{j}(length(segments_timey{j}));
    end
    last_iteration   =   ceil(max_length/temporal_block);   %Number   of
timeblocks


    contribution = 0*1:last_iteration; % Variable to check if y and z
are
```

```
    %both contributing to the step count. 1 means z, 2 means y


    for i = 1:last_iteration


        steps_z   =   2*   length(find(segments_timez{j}>(i-1)*1000+1   &
segments_timez{j}<i*1000));
        steps_y   =   2*   length(find(segments_timey{j}>(i-1)*1000+1   &
segments_timey{j}<i*1000));


        %compensating the non-stable
        if (i~=last_iteration)
        index_c_z   =   (find(compensate_start_timez{j}>(i-1)*1000+1   &
compensate_start_timez{j}<i*1000));
        times_z         =         compensate_timez{j}(index_c_z)         -
compensate_start_timez{j}(index_c_z);
        c_steps_z = floor(sum(times_z)*steps_z/1000);
        index_c_y   =   (find(compensate_start_timey{j}>(i-1)*1000+1   &
compensate_start_timey{j}<i*1000));
        times_y         =         compensate_timey{j}(index_c_y)         -
compensate_start_timey{j}(index_c_y);
        c_steps_y = floor(sum(times_y)*steps_y/1000);
        else
            %in the last block, there is no compensation because it
would
            %be compensated when more data entered (in real time)
            c_steps_z=0;
            c_steps_y=0;
        end
        % Choosing the axis that has found more steps (y or z)
        if (steps_z+c_steps_z >steps_y+c_steps_y)
            contribution(i) = 1;
            total_steps(j) = total_steps(j) + steps_z + c_steps_z;
            total_c_steps = total_c_steps + c_steps_z;
        else
            contribution(i) = 2;
            total_steps(j) = total_steps(j) + steps_y + c_steps_y;
            total_c_steps = total_c_steps + c_steps_y;
        end
    end
end
end
```

Function to do the optimization of variables:

```
function [] = optimizing_constants(n)
if nargin == 0
    n = 200;
    n = 1;
end

dispNames = 0; %whether names of the analysed subjects should be
displayed

%mkdir('results');
%addpath('results');



%% Reading data
%path = './dades';
path = '/Users/lotdrops/Documents/Telecos/TFG/dades_mobils';


%jordi
[axy1,                                  ayy1,                                  azy1,
ty1]=accelData('AcelerometerData_15_03_27_13_27_36.dat',[path
'/jordi/'],0); %  passos
[axy2,                                  ayy2,                                  azy2,
ty2]=accelData('AcelerometerData_15_03_27_13_33_24.dat',[path
'/jordi/'],0); %  passos
[axy3,                                  ayy3,                                  azy3,
ty3]=accelData('AcelerometerData_15_03_27_13_38_50.dat',[path
'/jordi/'],0); %  passos
[axy4,                                  ayy4,                                  azy4,
ty4]=accelData('AcelerometerData_15_03_27_16_47_48.dat',[path
'/jordi/'],0); %  passos


%victor
[axv1,                                  ayv1,                                  azv1,
tv1]=accelData('AcelerometerData_15_04_08_12_38_44.dat',[path
'/victor/'],0); %  passos
[axv2,                                  ayv2,                                  azv2,
tv2]=accelData('AcelerometerData_15_04_08_12_43_40.dat',[path
'/victor/'],0); %  passos
[axv3,                                  ayv3,                                  azv3,
tv3]=accelData('AcelerometerData_15_04_08_12_48_21.dat',[path
'/victor/'],0); %  passos
```

```
[axv4,                                  ayv4,                                  azv4,
tv4]=accelData('AcelerometerData_15_04_08_12_53_13.dat',[path
'/victor/'],0); %  passos


%angel
[axa1,                                  aya1,                                  aza1,
ta1]=accelData('AcelerometerData_15_04_10_12_22_26.dat',[path
'/angel/'],0); %  passos
[axa2,                                  aya2,                                  aza2,
ta2]=accelData('AcelerometerData_15_04_10_12_26_58.dat',[path
'/angel/'],0); %  passos
[axa3,                                  aya3,                                  aza3,
ta3]=accelData('AcelerometerData_15_04_10_12_30_56.dat',[path
'/angel/'],0); %  passos
[axa4,                                  aya4,                                  aza4,
ta4]=accelData('AcelerometerData_15_04_10_12_34_52.dat',[path
'/angel/'],0); %  passos


%cris
[axc1,                                  ayc1,                                  azc1,
tc1]=accelData('AcelerometerData_15_04_16_11_06_50.dat',[path
'/cris/'],0); %293 passos
[axc2,                                  ayc2,                                  azc2,
tc2]=accelData('AcelerometerData_15_04_16_11_11_00.dat',[path
'/cris/'],0); %280 passos
[axc3,                                  ayc3,                                  azc3,
tc3]=accelData('AcelerometerData_15_04_16_11_15_02.dat',[path
'/cris/'],0); %288 passos
[axc4,                                  ayc4,                                  azc4,
tc4]=accelData('AcelerometerData_15_04_16_11_18_36.dat',[path
'/cris/'],0); %303 passos


%fede
[axf1,                                  ayf1,                                  azf1,
tf1]=accelData('AcelerometerData_15_04_10_14_50_25.dat',[path
'/fede/'],0); %272 passos
[axf2,                                  ayf2,                                  azf2,
tf2]=accelData('AcelerometerData_15_04_10_14_54_04.dat',[path
'/fede/'],0); %264 passos
[axf3,                                  ayf3,                                  azf3,
tf3]=accelData('AcelerometerData_15_04_10_14_57_15.dat',[path
'/fede/'],0); %259 passos
[axf4,                                  ayf4,                                  azf4,
tf4]=accelData('AcelerometerData_15_04_10_15_01_27.dat',[path
'/fede/'],0); %268 passos


%guillem
```

```
[axg1,                              ayg1,                              azg1,
tg1]=accelData('AcelerometerData_15_04_10_12_49_53.dat',[path
'/guillem/'],0); %  passos

[axg2,                              ayg2,                              azg2,
tg2]=accelData('AcelerometerData_15_04_10_12_54_20.dat',[path
'/guillem/'],0); %  passos

[axg3,                              ayg3,                              azg3,
tg3]=accelData('AcelerometerData_15_04_10_12_57_50.dat',[path
'/guillem/'],0); %  passos

[axg4,                              ayg4,                              azg4,
tg4]=accelData('AcelerometerData_15_04_10_13_02_46.dat',[path
'/guillem/'],0); %  passos


%irene

[axi1,                              ayi1,                              azi1,
ti1]=accelData('AcelerometerData_15_04_10_17_47_40.dat',[path
'/irene/'],0); %  passos

[axi2,                              ayi2,                              azi2,
ti2]=accelData('AcelerometerData_15_04_10_17_52_25.dat',[path
'/irene/'],0); %  passos

[axi3,                              ayi3,                              azi3,
ti3]=accelData('AcelerometerData_15_04_10_17_56_31.dat',[path
'/irene/'],0); %  passos

[axi4,                              ayi4,                              azi4,
ti4]=accelData('AcelerometerData_15_04_10_18_00_39.dat',[path
'/irene/'],0); %  passos


%joana

[axj1,                              ayj1,                              azj1,
tj1]=accelData('AcelerometerData_15_04_14_10_41_22.dat',[path
'/joana/'],0); %  passos

[axj2,                              ayj2,                              azj2,
tj2]=accelData('AcelerometerData_15_04_14_10_45_22.dat',[path
'/joana/'],0); %  passos

[axj3,                              ayj3,                              azj3,
tj3]=accelData('AcelerometerData_15_04_14_10_50_06.dat',[path
'/joana/'],0); %  passos

[axj4,                              ayj4,                              azj4,
tj4]=accelData('AcelerometerData_15_04_14_10_53_59.dat',[path
'/joana/'],0); %  passos


%magi

[axma1,                             ayma1,                             azma1,
tma1]=accelData('AcelerometerData_15_04_14_10_18_08.dat',[path
'/magi/'],0); %  passos

[axma2,                             ayma2,                             azma2,
tma2]=accelData('AcelerometerData_15_04_14_10_22_20.dat',[path
'/magi/'],0); %  passos
```

```
[axma3,                                ayma3,                                azma3,
tma3]=accelData('AcelerometerData_15_04_14_10_26_26.dat',[path
'/magi/'],0); %  passos
[axma4,                                ayma4,                                azma4,
tma4]=accelData('AcelerometerData_15_04_14_10_30_21.dat',[path
'/magi/'],0); %  passos


%miriam
[axmi1,                                aymi1,                                azmi1,
tmi1]=accelData('AcelerometerData_15_04_13_13_17_21.dat',[path
'/miriam/'],0); %  passos
[axmi2,                                aymi2,                                azmi2,
tmi2]=accelData('AcelerometerData_15_04_13_13_21_24.dat',[path
'/miriam/'],0); %  passos
[axmi3,                                aymi3,                                azmi3,
tmi3]=accelData('AcelerometerData_15_04_13_13_25_22.dat',[path
'/miriam/'],0); %  passos
[axmi4,                                aymi4,                                azmi4,
tmi4]=accelData('AcelerometerData_15_04_13_13_28_53.dat',[path
'/miriam/'],0); %  passos


%miguel angel
[axg1,                                 ayg1,                                 azg1,
tg1]=accelData('AcelerometerData_15_04_30_15_13_25.dat',[path
'/garcia/'],0); %  passos
[axg2,                                 ayg2,                                 azg2,
tg2]=accelData('AcelerometerData_15_04_30_15_17_38.dat',[path
'/garcia/'],0); %  passos
[axg3,                                 ayg3,                                 azg3,
tg3]=accelData('AcelerometerData_15_04_30_15_21_04.dat',[path
'/garcia/'],0); %  passos
[axg4,                                 ayg4,                                 azg4,
tg4]=accelData('AcelerometerData_15_04_30_15_25_09.dat',[path
'/garcia/'],0); %  passos




%% Setting errors

parfor i=1:n

    disp(['iteracio: ', num2str(i), ' de ' num2str(n)]);

    th_ia_g = rand(1)/20*8 +0.3;   %0.3-0.7
    th_ir_g = rand(1)/20*7 +0.3;   %0.3-0.65
```

```
    if (mod(i,4)==0) % one out of 4 times, the difference is 0
        th_low_g = th_ir_g;
    else
        th_low_g = th_ir_g - rand(1)/10*3; %0-0.3 of difference respect
ir
    end



    th_ia{i} = th_ia_g;
    th_low{i} = th_low_g;
    th_ir{i} = th_ir_g;



    %Modified algorithm
    %jordi
    [segmentsy1z2{i},c_segmentsy1z2{i},segments_timey1z2{i},
compensate_timey1z2{i},  compensate_start_timey1z2{i},  variancesy1z2{i}]
= step_count(axy1, ayy1, azy1, ty1,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsy1y2{i},c_segmentsy1y2{i},segments_timey1y2{i},
compensate_timey1y2{i},  compensate_start_timey1y2{i},  variancesy1y2{i}]
= step_count(axy1, ayy1, azy1, ty1,th_ia_g,th_low_g,th_ir_g, 0,1);
    [segmentsy2z2{i},c_segmentsy2z2{i},segments_timey2z2{i},
compensate_timey2z2{i},  compensate_start_timey2z2{i},  variancesy2z2{i}]
= step_count(axy2, ayy2, azy2, ty2,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsy2y2{i},c_segmentsy2y2{i},segments_timey2y2{i},
compensate_timey2y2{i},  compensate_start_timey2y2{i},  variancesy2y2{i}]
= step_count(axy2, ayy2, azy2, ty2,th_ia_g,th_low_g,th_ir_g, 0,1);
    [segmentsy3z2{i},c_segmentsy3z2{i},segments_timey3z2{i},
compensate_timey3z2{i},  compensate_start_timey3z2{i},  variancesy3z2{i}]
= step_count(axy3, ayy3, azy3, ty3,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsy3y2{i},c_segmentsy3y2{i},segments_timey3y2{i},
compensate_timey3y2{i},  compensate_start_timey3y2{i},  variancesy3y2{i}]
= step_count(axy3, ayy3, azy3, ty3,th_ia_g,th_low_g,th_ir_g, 0,1);
    [segmentsy4z2{i},c_segmentsy4z2{i},segments_timey4z2{i},
compensate_timey4z2{i},  compensate_start_timey4z2{i},  variancesy4z2{i}]
= step_count(axy4, ayy4, azy4, ty4,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsy4y2{i},c_segmentsy4y2{i},segments_timey4y2{i},
compensate_timey4y2{i},  compensate_start_timey4y2{i},  variancesy4y2{i}]
= step_count(axy4, ayy4, azy4, ty4,th_ia_g,th_low_g,th_ir_g, 0,1);
    if (dispNames)
        disp('Jordi done')
    end
    %victor
```

```
    [segmentsv1z2{i},c_segmentsv1z2{i},segments_timev1z2{i},
compensate_timev1z2{i},  compensate_start_timev1z2{i},  variancesv1z2{i}]
= step_count(axv1, ayv1, azv1, tv1,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsv1y2{i},c_segmentsv1y2{i},segments_timev1y2{i},
compensate_timev1y2{i},  compensate_start_timev1y2{i},  variancesv1y2{i}]
= step_count(axv1, ayv1, azv1, tv1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsv2z2{i},c_segmentsv2z2{i},segments_timev2z2{i},
compensate_timev2z2{i},  compensate_start_timev2z2{i},  variancesv2z2{i}]
= step_count(axv2, ayv2, azv2, tv2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsv2y2{i},c_segmentsv2y2{i},segments_timev2y2{i},
compensate_timev2y2{i},  compensate_start_timev2y2{i},  variancesv2y2{i}]
= step_count(axv2, ayv2, azv2, tv2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsv3z2{i},c_segmentsv3z2{i},segments_timev3z2{i},
compensate_timev3z2{i},  compensate_start_timev3z2{i},  variancesv3z2{i}]
= step_count(axv3, ayv3, azv3, tv3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsv3y2{i},c_segmentsv3y2{i},segments_timev3y2{i},
compensate_timev3y2{i},  compensate_start_timev3y2{i},  variancesv3y2{i}]
= step_count(axv3, ayv3, azv3, tv3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsv4z2{i},c_segmentsv4z2{i},segments_timev4z2{i},
compensate_timev4z2{i},  compensate_start_timev4z2{i},  variancesv4z2{i}]
= step_count(axv4, ayv4, azv4, tv4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsv4y2{i},c_segmentsv4y2{i},segments_timev4y2{i},
compensate_timev4y2{i},  compensate_start_timev4y2{i},  variancesv4y2{i}]
= step_count(axv4, ayv4, azv4, tv4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('victor done')

end

    %angel

    [segmentsa1z2{i},c_segmentsa1z2{i},segments_timea1z2{i},
compensate_timea1z2{i},  compensate_start_timea1z2{i},  variancesa1z2{i}]
= step_count(axa1, aya1, aza1, ta1,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsa1y2{i},c_segmentsa1y2{i},segments_timea1y2{i},
compensate_timea1y2{i},  compensate_start_timea1y2{i},  variancesa1y2{i}]
= step_count(axa1, aya1, aza1, ta1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsa2z2{i},c_segmentsa2z2{i},segments_timea2z2{i},
compensate_timea2z2{i},  compensate_start_timea2z2{i},  variancesa2z2{i}]
= step_count(axa2, aya2, aza2, ta2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsa2y2{i},c_segmentsa2y2{i},segments_timea2y2{i},
compensate_timea2y2{i},  compensate_start_timea2y2{i},  variancesa2y2{i}]
= step_count(axa2, aya2, aza2, ta2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsa3z2{i},c_segmentsa3z2{i},segments_timea3z2{i},
compensate_timea3z2{i},  compensate_start_timea3z2{i},  variancesa3z2{i}]
= step_count(axa3, aya3, aza3, ta3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsa3y2{i},c_segmentsa3y2{i},segments_timea3y2{i},
compensate_timea3y2{i},  compensate_start_timea3y2{i},  variancesa3y2{i}]
= step_count(axa3, aya3, aza3, ta3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsa4z2{i},c_segmentsa4z2{i},segments_timea4z2{i},
compensate_timea4z2{i},  compensate_start_timea4z2{i},  variancesa4z2{i}]
= step_count(axa4, aya4, aza4, ta4,th_ia_g,th_low_g,th_ir_g, 1,1);
```

```
    [segmentsa4y2{i},c_segmentsa4y2{i},segments_timea4y2{i},
compensate_timea4y2{i},  compensate_start_timea4y2{i},  variancesa4y2{i}]
= step_count(axa4, aya4, aza4, ta4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('angel done')

    end

    %cris

    [segmentsc1z2{i},c_segmentsc1z2{i},segments_timec1z2{i},
compensate_timec1z2{i},  compensate_start_timec1z2{i},  variancesc1z2{i}]
= step_count(axc1, ayc1, azc1, tc1,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsc1y2{i},c_segmentsc1y2{i},segments_timec1y2{i},
compensate_timec1y2{i},  compensate_start_timec1y2{i},  variancesc1y2{i}]
= step_count(axc1, ayc1, azc1, tc1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsc2z2{i},c_segmentsc2z2{i},segments_timec2z2{i},
compensate_timec2z2{i},  compensate_start_timec2z2{i},  variancesc2z2{i}]
= step_count(axc2, ayc2, azc2, tc2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsc2y2{i},c_segmentsc2y2{i},segments_timec2y2{i},
compensate_timec2y2{i},  compensate_start_timec2y2{i},  variancesc2y2{i}]
= step_count(axc2, ayc2, azc2, tc2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsc3z2{i},c_segmentsc3z2{i},segments_timec3z2{i},
compensate_timec3z2{i},  compensate_start_timec3z2{i},  variancesc3z2{i}]
= step_count(axc3, ayc3, azc3, tc3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsc3y2{i},c_segmentsc3y2{i},segments_timec3y2{i},
compensate_timec3y2{i},  compensate_start_timec3y2{i},  variancesc3y2{i}]
= step_count(axc3, ayc3, azc3, tc3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsc4z2{i},c_segmentsc4z2{i},segments_timec4z2{i},
compensate_timec4z2{i},  compensate_start_timec4z2{i},  variancesc4z2{i}]
= step_count(axc4, ayc4, azc4, tc4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsc4y2{i},c_segmentsc4y2{i},segments_timec4y2{i},
compensate_timec4y2{i},  compensate_start_timec4y2{i},  variancesc4y2{i}]
= step_count(axc4, ayc4, azc4, tc4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('cris done')

    end

    %fede

    [segmentsf1z2{i},c_segmentsf1z2{i},segments_timef1z2{i},
compensate_timef1z2{i},  compensate_start_timef1z2{i},  variancesf1z2{i}]
= step_count(axf1, ayf1, azf1, tf1,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsf1y2{i},c_segmentsf1y2{i},segments_timef1y2{i},
compensate_timef1y2{i},  compensate_start_timef1y2{i},  variancesf1y2{i}]
= step_count(axf1, ayf1, azf1, tf1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsf2z2{i},c_segmentsf2z2{i},segments_timef2z2{i},
compensate_timef2z2{i},  compensate_start_timef2z2{i},  variancesf2z2{i}]
= step_count(axf2, ayf2, azf2, tf2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsf2y2{i},c_segmentsf2y2{i},segments_timef2y2{i},
compensate_timef2y2{i},  compensate_start_timef2y2{i},  variancesf2y2{i}]
= step_count(axf2, ayf2, azf2, tf2,th_ia_g,th_low_g,th_ir_g, 0,1);
```

```
    [segmentsf3z2{i},c_segmentsf3z2{i},segments_timef3z2{i},
compensate_timef3z2{i},  compensate_start_timef3z2{i},  variancesf3z2{i}]
= step_count(axf3, ayf3, azf3, tf3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsf3y2{i},c_segmentsf3y2{i},segments_timef3y2{i},
compensate_timef3y2{i},  compensate_start_timef3y2{i},  variancesf3y2{i}]
= step_count(axf3, ayf3, azf3, tf3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsf4z2{i},c_segmentsf4z2{i},segments_timef4z2{i},
compensate_timef4z2{i},  compensate_start_timef4z2{i},  variancesf4z2{i}]
= step_count(axf4, ayf4, azf4, tf4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsf4y2{i},c_segmentsf4y2{i},segments_timef4y2{i},
compensate_timef4y2{i},  compensate_start_timef4y2{i},  variancesf4y2{i}]
= step_count(axf4, ayf4, azf4, tf4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('fede done')

    end

    %guillem

    [segmentsg1z2{i},c_segmentsg1z2{i},segments_timeg1z2{i},
compensate_timeg1z2{i},  compensate_start_timeg1z2{i},  variancesg1z2{i}]
= step_count(axg1, ayg1, azg1, tg1,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg1y2{i},c_segmentsg1y2{i},segments_timeg1y2{i},
compensate_timeg1y2{i},  compensate_start_timeg1y2{i},  variancesg1y2{i}]
= step_count(axg1, ayg1, azg1, tg1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsg2z2{i},c_segmentsg2z2{i},segments_timeg2z2{i},
compensate_timeg2z2{i},  compensate_start_timeg2z2{i},  variancesg2z2{i}]
= step_count(axg2, ayg2, azg2, tg2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg2y2{i},c_segmentsg2y2{i},segments_timeg2y2{i},
compensate_timeg2y2{i},  compensate_start_timeg2y2{i},  variancesg2y2{i}]
= step_count(axg2, ayg2, azg2, tg2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsg3z2{i},c_segmentsg3z2{i},segments_timeg3z2{i},
compensate_timeg3z2{i},  compensate_start_timeg3z2{i},  variancesg3z2{i}]
= step_count(axg3, ayg3, azg3, tg3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg3y2{i},c_segmentsg3y2{i},segments_timeg3y2{i},
compensate_timeg3y2{i},  compensate_start_timeg3y2{i},  variancesg3y2{i}]
= step_count(axg3, ayg3, azg3, tg3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsg4z2{i},c_segmentsg4z2{i},segments_timeg4z2{i},
compensate_timeg4z2{i},  compensate_start_timeg4z2{i},  variancesg4z2{i}]
= step_count(axg4, ayg4, azg4, tg4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg4y2{i},c_segmentsg4y2{i},segments_timeg4y2{i},
compensate_timeg4y2{i},  compensate_start_timeg4y2{i},  variancesg4y2{i}]
= step_count(axg4, ayg4, azg4, tg4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('guillem done')

    end

    %irene

    [segmentsi1z2{i},c_segmentsi1z2{i},segments_timei1z2{i},
compensate_timei1z2{i},  compensate_start_timei1z2{i},  variancesi1z2{i}]
= step_count(axi1, ayi1, azi1, ti1,th_ia_g,th_low_g,th_ir_g, 1,1);
```

```
    [segmentsi1y2{i},c_segmentsi1y2{i},segments_timei1y2{i},
compensate_timei1y2{i},  compensate_start_timei1y2{i},  variancesi1y2{i}]
= step_count(axi1, ayi1, azi1, ti1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsi2z2{i},c_segmentsi2z2{i},segments_timei2z2{i},
compensate_timei2z2{i},  compensate_start_timei2z2{i},  variancesi2z2{i}]
= step_count(axi2, ayi2, azi2, ti2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsi2y2{i},c_segmentsi2y2{i},segments_timei2y2{i},
compensate_timei2y2{i},  compensate_start_timei2y2{i},  variancesi2y2{i}]
= step_count(axi2, ayi2, azi2, ti2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsi3z2{i},c_segmentsi3z2{i},segments_timei3z2{i},
compensate_timei3z2{i},  compensate_start_timei3z2{i},  variancesi3z2{i}]
= step_count(axi3, ayi3, azi3, ti3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsi3y2{i},c_segmentsi3y2{i},segments_timei3y2{i},
compensate_timei3y2{i},  compensate_start_timei3y2{i},  variancesi3y2{i}]
= step_count(axi3, ayi3, azi3, ti3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsi4z2{i},c_segmentsi4z2{i},segments_timei4z2{i},
compensate_timei4z2{i},  compensate_start_timei4z2{i},  variancesi4z2{i}]
= step_count(axi4, ayi4, azi4, ti4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsi4y2{i},c_segmentsi4y2{i},segments_timei4y2{i},
compensate_timei4y2{i},  compensate_start_timei4y2{i},  variancesi4y2{i}]
= step_count(axi4, ayi4, azi4, ti4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('irene done')

    end

    %joana

    [segmentsj1z2{i},c_segmentsj1z2{i},segments_timej1z2{i},
compensate_timej1z2{i},  compensate_start_timej1z2{i},  variancesj1z2{i}]
= step_count(axj1, ayj1, azj1, tj1,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsj1y2{i},c_segmentsj1y2{i},segments_timej1y2{i},
compensate_timej1y2{i},  compensate_start_timej1y2{i},  variancesj1y2{i}]
= step_count(axj1, ayj1, azj1, tj1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsj2z2{i},c_segmentsj2z2{i},segments_timej2z2{i},
compensate_timej2z2{i},  compensate_start_timej2z2{i},  variancesj2z2{i}]
= step_count(axj2, ayj2, azj2, tj2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsj2y2{i},c_segmentsj2y2{i},segments_timej2y2{i},
compensate_timej2y2{i},  compensate_start_timej2y2{i},  variancesj2y2{i}]
= step_count(axj2, ayj2, azj2, tj2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsj3z2{i},c_segmentsj3z2{i},segments_timej3z2{i},
compensate_timej3z2{i},  compensate_start_timej3z2{i},  variancesj3z2{i}]
= step_count(axj3, ayj3, azj3, tj3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsj3y2{i},c_segmentsj3y2{i},segments_timej3y2{i},
compensate_timej3y2{i},  compensate_start_timej3y2{i},  variancesj3y2{i}]
= step_count(axj3, ayj3, azj3, tj3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsj4z2{i},c_segmentsj4z2{i},segments_timej4z2{i},
compensate_timej4z2{i},  compensate_start_timej4z2{i},  variancesj4z2{i}]
= step_count(axj4, ayj4, azj4, tj4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsj4y2{i},c_segmentsj4y2{i},segments_timej4y2{i},
compensate_timej4y2{i},  compensate_start_timej4y2{i},  variancesj4y2{i}]
= step_count(axj4, ayj4, azj4, tj4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)
```

```
    disp('joana done')
    end
    %magi
    [segmentsma1z2{i},c_segmentsma1z2{i},segments_timema1z2{i},
compensate_timema1z2{i},                    compensate_start_timema1z2{i},
variancesma1z2{i}]      =      step_count(axma1,      ayma1,      azma1,
tma1,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsma1y2{i},c_segmentsma1y2{i},segments_timema1y2{i},
compensate_timema1y2{i},                    compensate_start_timema1y2{i},
variancesma1y2{i}]      =      step_count(axma1,      ayma1,      azma1,
tma1,th_ia_g,th_low_g,th_ir_g, 0,1);
    [segmentsma2z2{i},c_segmentsma2z2{i},segments_timema2z2{i},
compensate_timema2z2{i},                    compensate_start_timema2z2{i},
variancesma2z2{i}]      =      step_count(axma2,      ayma2,      azma2,
tma2,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsma2y2{i},c_segmentsma2y2{i},segments_timema2y2{i},
compensate_timema2y2{i},                    compensate_start_timema2y2{i},
variancesma2y2{i}]      =      step_count(axma2,      ayma2,      azma2,
tma2,th_ia_g,th_low_g,th_ir_g, 0,1);
    [segmentsma3z2{i},c_segmentsma3z2{i},segments_timema3z2{i},
compensate_timema3z2{i},                    compensate_start_timema3z2{i},
variancesma3z2{i}]      =      step_count(axma3,      ayma3,      azma3,
tma3,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsma3y2{i},c_segmentsma3y2{i},segments_timema3y2{i},
compensate_timema3y2{i},                    compensate_start_timema3y2{i},
variancesma3y2{i}]      =      step_count(axma3,      ayma3,      azma3,
tma3,th_ia_g,th_low_g,th_ir_g, 0,1);
    [segmentsma4z2{i},c_segmentsma4z2{i},segments_timema4z2{i},
compensate_timema4z2{i},                    compensate_start_timema4z2{i},
variancesma4z2{i}]      =      step_count(axma4,      ayma4,      azma4,
tma4,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsma4y2{i},c_segmentsma4y2{i},segments_timema4y2{i},
compensate_timema4y2{i},                    compensate_start_timema4y2{i},
variancesma4y2{i}]      =      step_count(axma4,      ayma4,      azma4,
tma4,th_ia_g,th_low_g,th_ir_g, 0,1);
    if (dispNames)
    disp('magi done')
    end
    %miriam
    [segmentsmi1z2{i},c_segmentsmi1z2{i},segments_timemi1z2{i},
compensate_timemi1z2{i},                    compensate_start_timemi1z2{i},
variancesmi1z2{i}]      =      step_count(axmi1,      aymi1,      azmi1,
tmi1,th_ia_g,th_low_g,th_ir_g, 1,1);
    [segmentsmi1y2{i},c_segmentsmi1y2{i},segments_timemi1y2{i},
compensate_timemi1y2{i},                    compensate_start_timemi1y2{i},
variancesmi1y2{i}]      =      step_count(axmi1,      aymi1,      azmi1,
tmi1,th_ia_g,th_low_g,th_ir_g, 0,1);
    [segmentsmi2z2{i},c_segmentsmi2z2{i},segments_timemi2z2{i},
compensate_timemi2z2{i},                    compensate_start_timemi2z2{i},
variancesmi2z2{i}]      =      step_count(axmi2,      aymi2,      azmi2,
tmi2,th_ia_g,th_low_g,th_ir_g, 1,1);
```

```
    [segmentsmi2y2{i},c_segmentsmi2y2{i},segments_timemi2y2{i},
compensate_timemi2y2{i},                    compensate_start_timemi2y2{i},
variancesmi2y2{i}]      =      step_count(axmi2,      aymi2,      azmi2,
tmi2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsmi3z2{i},c_segmentsmi3z2{i},segments_timemi3z2{i},
compensate_timemi3z2{i},                    compensate_start_timemi3z2{i},
variancesmi3z2{i}]      =      step_count(axmi3,      aymi3,      azmi3,
tmi3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsmi3y2{i},c_segmentsmi3y2{i},segments_timemi3y2{i},
compensate_timemi3y2{i},                    compensate_start_timemi3y2{i},
variancesmi3y2{i}]      =      step_count(axmi3,      aymi3,      azmi3,
tmi3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsmi4z2{i},c_segmentsmi4z2{i},segments_timemi4z2{i},
compensate_timemi4z2{i},                    compensate_start_timemi4z2{i},
variancesmi4z2{i}]      =      step_count(axmi4,      aymi4,      azmi4,
tmi4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsmi4y2{i},c_segmentsmi4y2{i},segments_timemi4y2{i},
compensate_timemi4y2{i},                    compensate_start_timemi4y2{i},
variancesmi4y2{i}]      =      step_count(axmi4,      aymi4,      azmi4,
tmi4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('miriam done')

    end

    %miguel angel

    [segmentsg1z2{i},c_segmentsg1z2{i},segments_timeg1z2{i},
compensate_timeg1z2{i},  compensate_start_timeg1z2{i},  variancesg1z2{i}]
= step_count(axg1, ayg1, azg1, tg1,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg1y2{i},c_segmentsg1y2{i},segments_timeg1y2{i},
compensate_timeg1y2{i},  compensate_start_timeg1y2{i},  variancesg1y2{i}]
= step_count(axg1, ayg1, azg1, tg1,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsg2z2{i},c_segmentsg2z2{i},segments_timeg2z2{i},
compensate_timeg2z2{i},  compensate_start_timeg2z2{i},  variancesg2z2{i}]
= step_count(axg2, ayg2, azg2, tg2,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg2y2{i},c_segmentsg2y2{i},segments_timeg2y2{i},
compensate_timeg2y2{i},  compensate_start_timeg2y2{i},  variancesg2y2{i}]
= step_count(axg2, ayg2, azg2, tg2,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsg3z2{i},c_segmentsg3z2{i},segments_timeg3z2{i},
compensate_timeg3z2{i},  compensate_start_timeg3z2{i},  variancesg3z2{i}]
= step_count(axg3, ayg3, azg3, tg3,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg3y2{i},c_segmentsg3y2{i},segments_timeg3y2{i},
compensate_timeg3y2{i},  compensate_start_timeg3y2{i},  variancesg3y2{i}]
= step_count(axg3, ayg3, azg3, tg3,th_ia_g,th_low_g,th_ir_g, 0,1);

    [segmentsg4z2{i},c_segmentsg4z2{i},segments_timeg4z2{i},
compensate_timeg4z2{i},  compensate_start_timeg4z2{i},  variancesg4z2{i}]
= step_count(axg4, ayg4, azg4, tg4,th_ia_g,th_low_g,th_ir_g, 1,1);

    [segmentsg4y2{i},c_segmentsg4y2{i},segments_timeg4y2{i},
compensate_timeg4y2{i},  compensate_start_timeg4y2{i},  variancesg4y2{i}]
= step_count(axg4, ayg4, azg4, tg4,th_ia_g,th_low_g,th_ir_g, 0,1);

    if (dispNames)

    disp('miguel angel done')
```

```
    end

    datestr(now)

end

save('optimizing_results_uni.mat');
end
```

Script to generate the matrix of results:

```
%jordi
error2(1,:) = -299 +combine_steps(segments_timey1z2, ...
segments_timey1y2, compensate_start_timey1z2, compensate_timey1z2, ...
compensate_start_timey1y2, compensate_timey1y2);
error2(2,:) = -298 +combine_steps(segments_timey2z2, ...
segments_timey2y2, compensate_start_timey2z2, compensate_timey2z2, ...
compensate_start_timey2y2, compensate_timey2y2);
error2(3,:) = -308 +combine_steps(segments_timey3z2, ...
segments_timey3y2, compensate_start_timey3z2, compensate_timey3z2, ...
compensate_start_timey3y2, compensate_timey3y2);
error2(4,:) = -311 +combine_steps(segments_timey4z2, ...
segments_timey4y2, compensate_start_timey4z2, compensate_timey4z2, ...
compensate_start_timey4y2, compensate_timey4y2);

%victor
error2(5,:) = -301 +combine_steps(segments_timev1z2, ...
segments_timev1y2, compensate_start_timev1z2, compensate_timev1z2, ...
compensate_start_timev1y2, compensate_timev1y2);
error2(6,:) = -304 +combine_steps(segments_timev2z2, ...
segments_timev2y2, compensate_start_timev2z2, compensate_timev2z2, ...
compensate_start_timev2y2, compensate_timev2y2);
error2(7,:) = -305 +combine_steps(segments_timev3z2, ...
segments_timev3y2, compensate_start_timev3z2, compensate_timev3z2, ...
compensate_start_timev3y2, compensate_timev3y2);
error2(8,:) = -325 +combine_steps(segments_timev4z2, ...
segments_timev4y2, compensate_start_timev4z2, compensate_timev4z2, ...
compensate_start_timev4y2, compensate_timev4y2);

%angel
error2(9,:) = -299 +combine_steps(segments_timea1z2, ...
segments_timea1y2, compensate_start_timea1z2, compensate_timea1z2, ...
compensate_start_timea1y2, compensate_timea1y2);
error2(10,:) = -294 +combine_steps(segments_timea2z2, ...
segments_timea2y2, compensate_start_timea2z2, compensate_timea2z2, ...
compensate_start_timea2y2, compensate_timea2y2);
error2(11,:) = -293 +combine_steps(segments_timea3z2, ...
segments_timea3y2, compensate_start_timea3z2, compensate_timea3z2, ...
compensate_start_timea3y2, compensate_timea3y2);
```

```
error2(12,:) = -305 +combine_steps(segments_timea4z2, ...
segments_timea4y2, compensate_start_timea4z2, compensate_timea4z2, ...
compensate_start_timea4y2, compensate_timea4y2);

%cris
error2(13,:) = -293 +combine_steps(segments_timec1z2, ...
segments_timec1y2, compensate_start_timec1z2, compensate_timec1z2, ...
compensate_start_timec1y2, compensate_timec1y2);
error2(14,:) = -280 +combine_steps(segments_timec2z2, ...
segments_timec2y2, compensate_start_timec2z2, compensate_timec2z2, ...
compensate_start_timec2y2, compensate_timec2y2);
error2(15,:) = -288 +combine_steps(segments_timec3z2, ...
segments_timec3y2, compensate_start_timec3z2, compensate_timec3z2, ...
compensate_start_timec3y2, compensate_timec3y2);
error2(16,:) = -303 +combine_steps(segments_timec4z2, ...
segments_timec4y2, compensate_start_timec4z2, compensate_timec4z2, ...
compensate_start_timec4y2, compensate_timec4y2);

%fede
error2(17,:) = -272 +combine_steps(segments_timef1z2, ...
segments_timef1y2, compensate_start_timef1z2, compensate_timef1z2, ...
compensate_start_timef1y2, compensate_timef1y2);
error2(18,:) = -264 +combine_steps(segments_timef2z2, ...
segments_timef2y2, compensate_start_timef2z2, compensate_timef2z2, ...
compensate_start_timef2y2, compensate_timef2y2);
error2(19,:) = -259 +combine_steps(segments_timef3z2, ...
segments_timef3y2, compensate_start_timef3z2, compensate_timef3z2, ...
compensate_start_timef3y2, compensate_timef3y2);
error2(20,:) = -268 +combine_steps(segments_timef4z2, ...
segments_timef4y2, compensate_start_timef4z2, compensate_timef4z2, ...
compensate_start_timef4y2, compensate_timef4y2);

%guillem
error2(21,:) = -300 +combine_steps(segments_timeg1z2, ...
segments_timeg1y2, compensate_start_timeg1z2, compensate_timeg1z2, ...
compensate_start_timeg1y2, compensate_timeg1y2);
error2(22,:) = -281 +combine_steps(segments_timeg2z2, ...
segments_timeg2y2, compensate_start_timeg2z2, compensate_timeg2z2, ...
compensate_start_timeg2y2, compensate_timeg2y2);
```

```
error2(23,:) = -289 +combine_steps(segments_timeg3z2, ...
segments_timeg3y2, compensate_start_timeg3z2, compensate_timeg3z2, ...
compensate_start_timeg3y2, compensate_timeg3y2);
error2(24,:) = -300 +combine_steps(segments_timeg4z2, ...
segments_timeg4y2, compensate_start_timeg4z2, compensate_timeg4z2, ...
compensate_start_timeg4y2, compensate_timeg4y2);

%irene
error2(25,:) = -276 +combine_steps(segments_timei1z2, ...
segments_timei1y2, compensate_start_timei1z2, compensate_timei1z2, ...
compensate_start_timei1y2, compensate_timei1y2);
error2(26,:) = -278 +combine_steps(segments_timei2z2, ...
segments_timei2y2, compensate_start_timei2z2, compensate_timei2z2, ...
compensate_start_timei2y2, compensate_timei2y2);
error2(27,:) = -268 +combine_steps(segments_timei3z2, ...
segments_timei3y2, compensate_start_timei3z2, compensate_timei3z2, ...
compensate_start_timei3y2, compensate_timei3y2);
error2(28,:) = -272 +combine_steps(segments_timei4z2, ...
segments_timei4y2, compensate_start_timei4z2, compensate_timei4z2, ...
compensate_start_timei4y2, compensate_timei4y2);

%joana
error2(29,:) = -280 +combine_steps(segments_timej1z2, ...
segments_timej1y2, compensate_start_timej1z2, compensate_timej1z2, ...
compensate_start_timej1y2, compensate_timej1y2);
error2(30,:) = -273 +combine_steps(segments_timej2z2, ...
segments_timej2y2, compensate_start_timej2z2, compensate_timej2z2, ...
compensate_start_timej2y2, compensate_timej2y2);
error2(31,:) = -277 +combine_steps(segments_timej3z2, ...
segments_timej3y2, compensate_start_timej3z2, compensate_timej3z2, ...
compensate_start_timej3y2, compensate_timej3y2);
error2(32,:) = -279 +combine_steps(segments_timej4z2, ...
segments_timej4y2, compensate_start_timej4z2, compensate_timej4z2, ...
compensate_start_timej4y2, compensate_timej4y2);

%magi
error2(33,:) = -274 +combine_steps(segments_timema1z2, ...
segments_timema1y2, compensate_start_timema1z2, compensate_timema1z2, ...
compensate_start_timema1y2, compensate_timema1y2);
```

```
error2(34,:) = -273 +combine_steps(segments_timema2z2, ...
segments_timema2y2, compensate_start_timema2z2, compensate_timema2z2, ...
compensate_start_timema2y2, compensate_timema2y2);
error2(35,:) = -268 +combine_steps(segments_timema3z2, ...
segments_timema3y2, compensate_start_timema3z2, compensate_timema3z2, ...
compensate_start_timema3y2, compensate_timema3y2);
error2(36,:) = -278 +combine_steps(segments_timema4z2, ...
segments_timema4y2, compensate_start_timema4z2, compensate_timema4z2, ...
compensate_start_timema4y2, compensate_timema4y2);

%miriam
error2(37,:) = -276 +combine_steps(segments_timemi1z2, ...
segments_timemi1y2, compensate_start_timemi1z2, compensate_timemi1z2, ...
compensate_start_timemi1y2, compensate_timemi1y2);
error2(38,:) = -274 +combine_steps(segments_timemi2z2, ...
segments_timemi2y2, compensate_start_timemi2z2, compensate_timemi2z2, ...
compensate_start_timemi2y2, compensate_timemi2y2);
error2(39,:) = -270 +combine_steps(segments_timemi3z2, ...
segments_timemi3y2, compensate_start_timemi3z2, compensate_timemi3z2, ...
compensate_start_timemi3y2, compensate_timemi3y2);
error2(40,:) = -270 +combine_steps(segments_timemi4z2, ...
segments_timemi4y2, compensate_start_timemi4z2, compensate_timemi4z2, ...
compensate_start_timemi4y2, compensate_timemi4y2);

%miguel angel
error2(41,:) = -293 +combine_steps(segments_timeg1z2, ...
segments_timeg1y2, compensate_start_timeg1z2, compensate_timeg1z2, ...
compensate_start_timeg1y2, compensate_timeg1y2);
error2(42,:) = -299 +combine_steps(segments_timeg2z2, ...
segments_timeg2y2, compensate_start_timeg2z2, compensate_timeg2z2, ...
compensate_start_timeg2y2, compensate_timeg2y2);
error2(43,:) = -295 +combine_steps(segments_timeg3z2, ...
segments_timeg3y2, compensate_start_timeg3z2, compensate_timeg3z2, ...
compensate_start_timeg3y2, compensate_timeg3y2);
error2(44,:) = -284 +combine_steps(segments_timeg4z2, ...
segments_timeg4y2, compensate_start_timeg4z2, compensate_timeg4z2, ...
compensate_start_timeg4y2, compensate_timeg4y2);
```