ClimbTheWorld: Real-time stairstep counting to increase physical activity

Fabio Aiolli, Matteo Ciman, Michele Donini and Ombretta Gaggi
Department of Mathematics
Via Trieste, 63
University of Padua, Italy
{aiolli, mciman, mdonini, gaggi}@math.unipd.it

ABSTRACT

The increasing number of people that are overweight due to a sedentary life requires persuasive strategies to convince people to change their behaviors. In this paper, we present a machine learning based technique to recognize and count stairsteps when a person climbs or descends stairs. This technique has been used as part of *ClimbTheWorld*, a real-time smartphone application that aims at persuading people to use stairs instead of elevators or escalators, since an engaging activity has more chance to change people's life habits. We perform a fine-grained analysis by exploiting smartphone sensors to recognize single stairsteps. Data-dependent sliding windows are used facilitating the learning process and reducing the computational cost. Finally, energy consumption is widely investigated to optimize the trade-off between classification precision and battery usage, to avoid exhausting smartphone battery.

General Terms

activity recognition, energy consumption, mobile computing, ubiquitous applications.

1. INTRODUCTION

In the last few decades the technological progress has completely changed people's lifestyle, making everyday activities much easier. Unfortunately, this progress also moved people to a sedentary lifestyle, thus increasing the occurrence of some diseases, like obesity (both in adults and children), heart diseases, diabetes, cancer etc., and the medical costs for their treatment. According to the World Health Organization, there are at least 3.2 million people per year dying for these diseases [14]. On the other hand, the widespread presence of smartphone in people pockets offers the possibility to use these devices for health support and health-associated activity recognition.

In this paper we present *ClimbTheWorld*, a smartphone *serious game* aiming at incentivize people to use stairs instead of elevators or escalators. *ClimbTheWorld* implements a new method for stairsteps recognition and counting. In order to increase people engagement and, consequently, the real chance to change their behaviors, the game requires a fine-grained classification, that is a real-time counting of the number of stairsteps along with a real-time feedback.

The advantages of *ClimbTheWorld* over other solutions are:

- it provides an *online*, *fine-grained* classification, i. e., it is able to recognize a single stairstep vs. a simple step, during the activity itself,
- no restrictions on the smartphone orientation,
- it does not require people to buy expensive tools (e. g., the Nike+ FuelBand bracelet[1]) but it uses only the smartphone sensors, and
- energy consumption has been considered during the development of the serious game.

Several studies have shown that energy consumption is a critical aspect for mobile applications [8, 10] and can influence performance on some user experience metrics [4]. In particular, battery lifetime is one of the most important aspects considered by users when dealing with applications on mobile devices. For this reason, applications developers must avoid to waste energy since this can require a user a too frequent recharge of the smartphone. This means that, for example, data acquisition frequency must be carefully considered since it is an extremely energy consuming task. Similarly, our approach gathers strictly necessary information only from data, avoiding the use of a high number of features computations that could lead the final classification to become too expensive. To the best of our knowledge, this is the first tentative to consider energy consumption as a key aspect in a classification problem.

Our approach also proposes a new solution to deal with the change of the smartphone position. Since accelerometer data is influenced by the rotation and the position of the smartphone, a new method is introduced to translate data received from the accelerometer to a fixed coordinate system. We use the standard built-in rotation sensor providing information about the rotation of the device and a time fixed buffer to eliminate the influence of the gravity on data we gather.

[1] `http://www.nike.com/us/en_us/c/nikeplus-fuelband`

MOBIQUITOUS 2014, December 02-05, London, Great Britain

DOI 10.4108/icst.mobiquitous.2014.258013

Other works in literature propose solutions for activity recognition, e. g., running, walking or driving. These approaches collect data for an interval of time, and, after another interval of time for calculation, guess which activity is performed by the user in that period. Our approach is different since we aim at counting the number of stairsteps, which is a more difficult task than recognizing an activity during a longer time interval. This means that we need a more fine-grained classification. As we discuss in Section 4, to recognize stairsteps from steps is a very difficult task since data retrieved from accelerometer are similar. Moreover, stairsteps recognition and counting is performed *in real time*.

Since the needed frequency of data analysis is extremely high (each stairstep requires about 500ms to be completed), we implemented a data-dependent window, instead of the traditional sliding window with a fixed time duration, and we apply classification only if that window is suitable to represent a stairstep. In this way we reduce the number of windows to analyze and hence the computational cost.

2. RELATED WORKS

Before the usage of smartphone sensors to perform activity recognition (and in particular step counting), an analysis of performances reached by pedometer during stair climbing was made by Ayabe et al. [3]. The purpose of their experiments was to understand how well pedometers perform during stair climbing and descending. They evaluate three different commercial pedometers and different stepping rate (from 40 to 120 steps·min^{-1}). Although they do no distinguish between steps and stairsteps, results show that pedometer can assess stairstep counting within an error rate of ±5%, being a great tool to count number of stairsteps (and steps) made by each user.

Thanks to the increasing performances of smartphones and their ability to acquire lot of data from the surrounded environment, mobile applications that use data from sensors for activity recognition and to promote better lifestyle received considerable interest from the research community.

Anjum and Ilyas [2] develop an application for *online* activity recognition like walking, running, climbing stairs, descending stairs, cycling, driving and remaining inactive. They provide an analysis of 5 seconds length windows using several classification algorithms like KNN, Naive Bayes, Decision Trees and Support Vector Machines. They collect data from the accelerometer, the gyroscope and the GPS. The precision of the results ranges between 79% (using Support Vector Machine) and 94% (Decision Tree). Their analysis also shows that data retrieved from gyroscope do not provide any useful information, therefore they remove all the features coming from this sensor. Moreover, they stated that the recognition of stairs climbing and descending is a really difficult task, achieving a much more lower precision, and often stair climbing is confused with walking (precision of 84.6% for going upstairs, 90,5% for going downstairs).

Shoaib et al. [12] analyze activity recognition using accelerometer, gyroscope and magnetometer, alone or combined together at a frequency of 50Hz. They consider four different positions of the smartphone (arm, belt, pocket and wrist). They showed that the magnetometer, both alone or in combination with other sensors, performs poorly since it causes overfitting for the classifier. Moreover, they found that the accelerometer performs better than the gyroscope in recognizing the six different activities, but, in contrast with the results provided by Anjum and Ilvas, this work showed that the combination of accelerometer and gyroscope data performs better than the individual performances.

Authors	Use of cell phone	Real-time	Sensors	Orientation	Window	Activity	Stairstep counting
[3]	No	Yes	Acc., Gyr.	1	-	stairs	Yes
[2]	Yes	Yes	Acc., Gyr., GPS	4	5 s	stand, walk, run, cycle, stairs	No
[12]	Yes	Yes	Acc.	1	4 s	stand, walk, cycle, drive, run	No
[5]	Yes	No	Acc.	6	(*)	Walking	No
[15]	No	No	Acc., Gyr.	1	2 s	Stand, walk, run, stairs	No
Our solution	Yes	Yes	Acc., Rot.	Free (**)	(***)	Stairstep	Yes

(*) [5] allows different window sizes
(**) The only constraint is not to use the trouser pocket
(***) Our solution uses a data dependent window size

Table 1: Resume of the different approaches presented in Section 2 compared with our solution

Brajdic and Harle [5] discuss about walk detection and step counting algorithms with data acquired from a smartphone; the user can choose to carry the smartphone in six different positions. This work discusses some issues in common with our approach, but in our case we count stairsteps (so we need to discriminate between steps and stairsteps which is a more difficult task). The authors evaluate several algorithms using a dataset of 130 walks traces from 27 different users, getting that most of the algorithms are able to detect walking within a trace that contains only walking and idle periods, with a best median error of 1.3 using thresholding technique, but no one of the algorithms is 100% reliable.

Wu et al. [15] analyze activity recognition using an iPod Touch, acquiring data from the accelerometer and the gyroscope. They recognize activity like sitting, walking, jogging, going upstairs and downstairs. They extract both time, frequency and Fast Fourier Transform. They made data analysis offline, using 2-second window size with data acquisition at 30Hz. They achieved the best results using kNN. When sitting, walking and jogging, the accuracy was very high (90,1%-94,1%), while up and down stair walking was classified with lower precision (52.3%-79.4%).

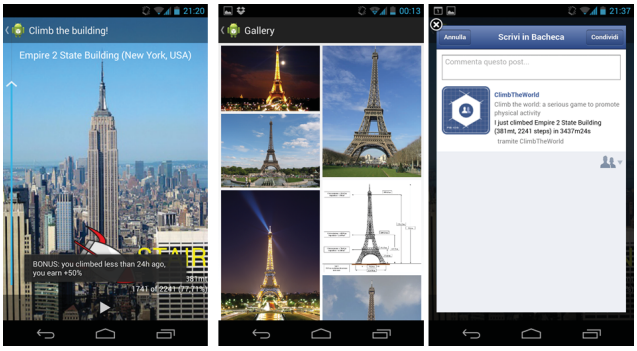


Figure 1: Three screenshots of *ClimbTheWorld*

HealthyLife [6] is a smartphone application that automatically recognizes users activities. It recognizes activities like walking, running, driving and staying-still. They use the accelerometer data and the GPS signal. Moreover, they apply *Ambiguity reasoning* to increase classification results and to disambiguate data. The idea is to apply a set of weak constraints that attaches to any possible answer of the classifier a violation cost that depends on the number and type of violated constraints. The right classification answer will be the one with violation cost equals to 0. The final precision of the system ranges from 100% (staying-still) to 73% (walking).

The analysis of the related works reveals several open issues that need further analysis. First of all, since we are pursuing a smartphone based real time recognition, energy consumption is a fundamental aspect, never considered before. For example, the usage of multiple sensors at the same time (accelerometer, gyroscope, magnetometer and GPS) will rapidly drain the battery, causing lot of stress to the user. For this reason, we will rely on the accelerometer and the rotation sensor only, avoiding the usage of much more expensive sensors. Moreover, differently from other solutions that recognize an activity over a large window of time, we aim at recognizing each single stairstep, and distinguish it from a simple step. As shown in Figure 3, this is not an easy task, since the signal obtained by a stairstep and a step has a similar behavior. Finally, all the previously proposed approaches impose a set of fixed positions for the smartphone and the training of a different classifier for each supported position. As we will see, we do not impose any position of the smartphone, we only ask to avoid the trousers pocket. Table 1 provides a comparison of our approach against the ones presented in this section.

3. GAME DESCRIPTION

ClimbTheWorld is a *serious game* developed as a smartphone application that aims to incentivize people to take stairs instead of elevators or escalators. The use of mobile technologies in real time persuasive solutions is a very good combination since smartphones are commonly available in the users pocket and can invisibly work and help people to change their unhealthy behavior.

The idea underlying the game is simple: the user has to climb real world buildings, e. g., the Empire State Building or the Eiffel Tower, making stairs during his/her everyday life. Once started, the game records and analyzes data from

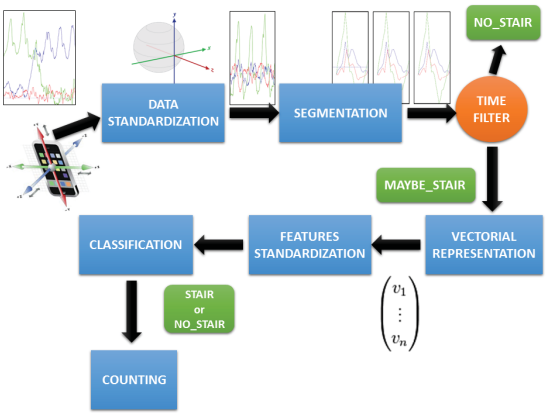


Figure 2: Pipeline overview of the system

the accelerometer and count the number of stairsteps made by the user, even when the application goes in background.

The game proposes different difficulty levels: easier levels correspond to a lower number of stairsteps necessary to reach the top of the building. Each stairstep in real life corresponds to one (or more) stairstep in the game. Once the user reaches the top of the building, a slideshow of pictures is displayed, showing the view from the top of the building. Different difficulty levels also bring different quality and number of provided photos.

To increase the engagement of the user, the game provides a set of bonuses, depending on the performance of the user. For example, if the user improves his/her performance with respect to the day before, he/she gets a 10% increase on the total number of stairsteps made. These bonuses are used to constantly encourage and help the user. For the same reason, we aims at designing a non-invasive application, so we do not fix the smartphone orientation and we consider energy consumption issues.

Finally, the game gives the possibility to a user to share his/her performance with friends through *Facebook* to further increase the user engagement. Figure 1 shows some screenshots of the final application, i. e., the user interface during stairstep counting, the photo gallery, and the user interface to post an achievement in *Facebook*.

4. PIPELINE OVERVIEW

In this section, we give a broad overview of the our system for recognition and counting of stairsteps. The system modules will be discussed in detail in the following sections. The main application and its pipeline is shown in Figure 2.

The overall system continuously acquires data from the sensors of a smartphone. This information is elaborated by the system to recognize time segments when the user is climbing or descending stairs. Finally, the system will return the counting. Since we want to provide real-time feedback to the user, and reduce the delay of the response of the application, all data elaboration is made on the smartphone and not on a server. This avoids problems of network availability, network delay or server overload, making the system

more responsive and interactive.

The first step of the pipeline is data acquisition. This is performed by the smartphone that, through its sensors, acquires data about the movements of the device.

The following module of the pipeline takes raw data acquired by sensors and standardize it. We need of this step because of the requirement to not fix the smartphone orientation and, as we will see in Section 5, raw data acquired by sensors can change a lot depending on the smartphone orientation. Thus, to overcome this problem, raw data is translated into a predefined and fixed coordinate system. In this way, data recording the same type of activity more likely corresponds to the same behavior, independently from the orientation of the smartphone.

Then, it follows the segmentation step, where the standardized data is split into consecutive segments or windows. In this module, in order to reduce the computational cost and improve the accuracy, a window segmentation technique based on data analysis (without fixing lasting time) is introduced. The proposed method allows to consider the time as additional information, then allows to introduce a simple filter which is able to immediately discard those windows that clearly have a duration unsuitable for a stairstep.

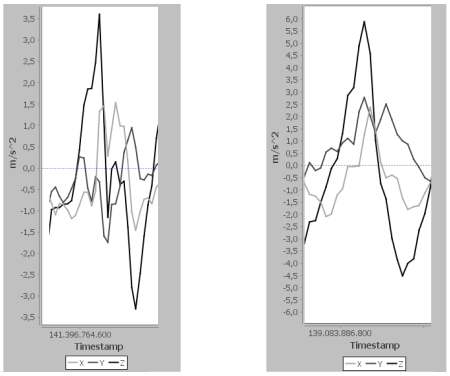
After segmentation, the extraction and standardization of the features for the classifier is performed.

We cast the stairstep recognition task in a classification setting where we have two possible labels ("STAIR" and "NO_STAIR") and the examples consist of vector representations of segmented windows. Note that, the training of the model is performed off-line only once (one single model shared by all different users), while the recognition phase have to be made *in real time*. This implies that the classification outputs must be promptly provided to the user without delays. We must note here that a simple high-pass filter or a peak detection algorithm would not be sufficient in our context. In fact, as it is shown in Figure 3, a stairstep and a step have almost the same shape, that is, a data peak on one axis. For this reason, we need a more complex method able to distinguish between this two different activities.

In the following sections, we provide a deep analysis of the main modules that compound the complete system. Finally, we present the results of the classification algorithms in a simulation of the system and an analysis of all the issues related to energy consumption.

5. SUPPORT TO THE ORIENTATION-INDEPENDENCE

The smartphone orientation is one of the first crucial issues to consider, since data recorded from the accelerometer during any activity is deeply affected by the device orientation. Requiring a user to keep the smartphone in a particular position is uncomfortable and thus not desirable, the application would not be pervasive anymore and this would likely lead the user to delete or misuse the application. For this reason, we propose a method that does not make any restrictive assumption about the smartphone position, in a way to enforce the pervasiveness of the application.



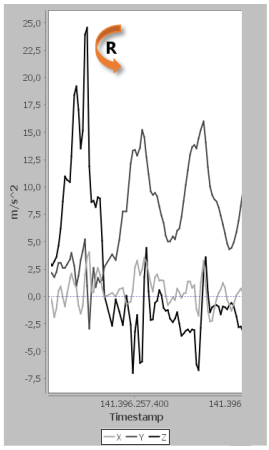
(a) Example of a step (b) Example of a stairstep

Figure 3: Comparison between a step and a stairstep: the signal behavior is almost the same

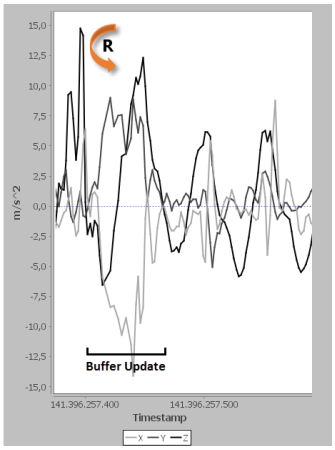
Figure 4a reports data recorded during the same walking activity while keeping the phone in a hand. The three lines represent signals recorded by the accelerometer for the X, Y and Z-axis, during a short interval of time (represented by the horizontal axis in the figure). The figure shows how data significantly varies due to a different orientation (rotation) of the device. In particular, we observe a clear switch of those signals that represent the acceleration on the axes involved in the rotation. This variation dramatically increases the difficulty of the following learning task since, in this case, the classifier is required to be trained on every possible orientations. Clearly, this approach is not reasonable, so we need to study how to standardize the signal such to enforce invariance with respect to orientation. Specifically, we would like the values read by the accelerometer to be referred to a fixed coordinate system, so that each activity will be represented by a similar signal behavior, independently from the position of the smartphone. In this way, how the user carries the smartphone becomes an insignificant problem, and the task to train an accurate classifier becomes easier.

One method to solve the problem above was previously proposed by Mizell [9]. The idea is the following: given a sampling interval, the gravity component $\mathbf{g} = (g_x, g_y, g_z) \in \mathbb{R}^3$ on each axis can be estimated by averaging over data read on each axis. When an accelerometer produces the original signal $\mathbf{a} = (a_x, a_y, a_z) \in \mathbb{R}^3$, it is possible to calculate the so called *dynamic component* of $\mathbf{a}$ as $\mathbf{d} = (a_x - g_x, a_y - g_y, a_z - g_z)$ where the influence of the gravity is eliminated. Finally, the vertical part $\mathbf{p}$ of the dynamic component $\mathbf{d}$ (parallel to the gravity) is computed as $\mathbf{p} = (\frac{\mathbf{d}\cdot\mathbf{m}}{\mathbf{m}\cdot\mathbf{m}})\mathbf{m}$, and the horizontal part (orthogonal to the gravity) as $\mathbf{h} = \mathbf{d} - \mathbf{p}$. As a positive aspect, we note that this method only requires the usage of the accelerometer, and this makes it less expensive in terms of energy consumption. On the other hand, it may loose relevant information since it translates a three-dimensional coordinate system (the one of the device) into a two-dimensional one (vertical and horizontal directions).

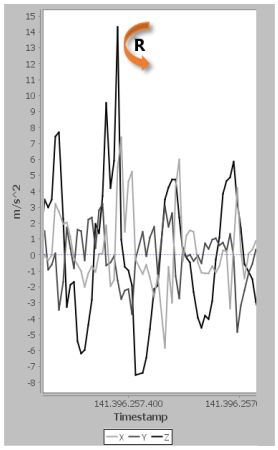
Here, we propose a new method to increase the precision of the transformation and preserving the orientation-invariance property. How this method works and how it changes the signal acquired by the accelerometer is shown in Figure 4b.



(a) Raw data acquired from the accelerometer: same activity but different axis values after rotation of the smartphone



(b) Our method applied to a walking activity while rotating the smartphone



(c) Native linear acceleration plus rotation method

Figure 4: The comparison among the raw signal (a) and the signal from the methods for the orientation-independence: our method (b), Linear (c). The instant in which the rotation takes place is denoted by the letter "R". Each line represents data acquired with reference to one axis X, Y and Y

The fixed target coordinate system is the following: the X axis is defined as the one tangential to the ground pointing approximately toward East. The Y axis is tangential to the ground pointing toward the geomagnetic North. Finally, the Z axis is orthogonal to the ground plane and points toward the sky[2]. To implement our method, we need the rotation vector sensor in addition to the accelerometer of the smartphone, and a buffer used to estimate the gravity component acting on each axis.

Firstly, we remove the gravity component from the accelerometer signal using a buffer which stores data acquired during the last 500ms, and averaging over the axes. The vector $\mathbf{g} = (g_x, g_y, g_z)$ is used to compute the dynamic component vector $\mathbf{d} = (a_x - g_x, a_y - g_y, a_z - g_z)$ where $\mathbf{a} = (a_x, a_y, a_z)$ is the original accelerometer reading. Once the vector $\mathbf{d}$ has been computed, it is rotated to the fixed coordinate system using information coming from the rotation vector sensor. This sensor provides information about the current orientation of the device with a three-component vector, where each component represents a rotation angle around an axis. Now, we can apply a three dimensional rotation to the vector $\mathbf{d} = (d_x, d_y, d_z)$ to obtain a new vector $\mathbf{d}' = (d'_x, d'_y, d'_z)$ representing the real movement of the user with respect to our target coordinate system. As we can see in Figure 4b, the signal remains affected from the rotation for a very short interval of time only which is bounded by the buffer size.

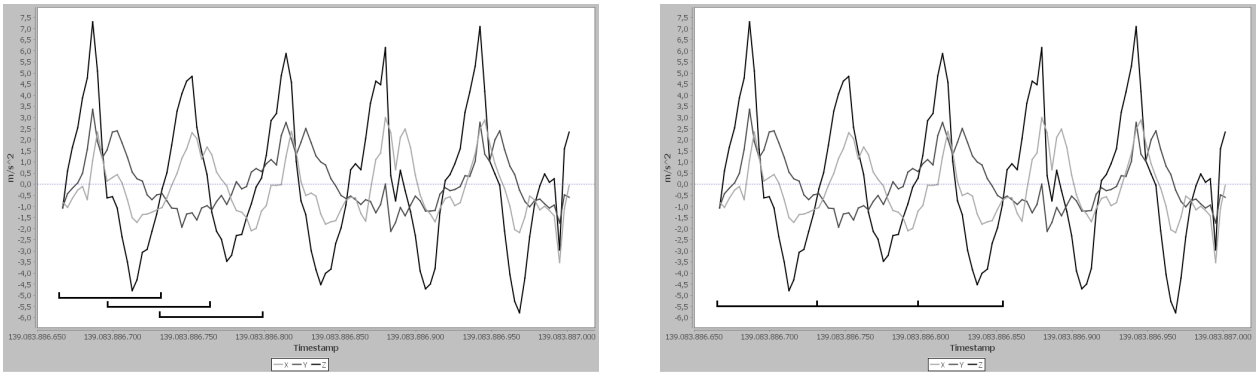
Android natively supports the automatic gravity removal using the linear acceleration sensor that directly retrieves the dynamic component vector $\mathbf{d}$. Starting from this vector and using the rotation data as described above, linear acceleration readings can be rotated to our fixed coordinate system. An illustration of the rotation-sensor based method applied on the Android native solution is provided in Figure 4c. As we will see in Section 8.1 and in Section 8.2, our method performs better that the native solution both in terms of precision and energy consumption.

A positive aspect of our approach is that, using a buffer to calculate and remove gravity, data from the accelerometer can be represented as a smoothed line which permits us to avoid the problem of sensitivity and calibration of different smartphone sensors, which varies on different models. With our approach this error is smoothed. We made several tests with a Samsung Galaxy i9250, acquiring data from the accelerometer and then applying our approach, and from the linear sensor. The smartphone was placed with the screen pointing to the sky. Data acquired at $125Hz$ for 5 minutes, and the readings averaged on each axis. Since the smartphone was not moving, the theoretical result is $A_t = (0.0, 0.0, 0.0)$. As mentioned before, this is only a theoretical result since every sensor has its own noise. The final average values obtained were $A_l = (-0.04869, 0.01913, 0.62103)$ using the linear sensor and $A_o = (0.03083, -0.01564, 0.42016)$ using our method. As we can see, even if we were not able to completely remove the noise from the accelerometer sensor, its influence on the final data is reduced.

6. SEGMENTATION WITH DATA-DEPENDENT WINDOW

The standard method to perform activity recognition is by classification of *sliding windows*. This approach is simple: since during data acquisition we do not know when a particular activity starts, a window is defined as the set of temporal consecutive data that lasts a fixed amount of time. When this maximum amount of time is reached, all data belonging to the window is analyzed and classified. With this approach, activities which start in the middle of a particular window can be missed. A natural extension is to consider *overlapping sliding windows*. In this case, windows are built of the same duration and overlap. In this way, it

[2]`http://developer.android.com/guide/topics/sensors/sensors_overview.html`



(a) Sliding windows using time

(b) Data-dependent sliding windows

Figure 5: The comparison between the segmentation in windows of possible stairsteps obtained from the time-driven sliding windows (a) and the data-dependent sliding windows (b)

is possible to reduce the number of missing activities due to unknown starting time. Figure 5a gives an example of a segmentation of sliding windows with overlapping of 50%.

This kind of approach has shown nice results in different contexts. However, it is not suitable for fine-grained classification problems as recognizing single stairsteps which last for very short intervals of time. The first problem is related to the presence of different user behaviors. Since every user is different from an other, how a person makes a particular activity, e.g., a stairstep, can have different duration and frequency. Within a fixed sliding window approach, it is not possible to manage this variability, i.e., a window could be too short, or too long, to give good results for a particular user. Moreover, an initial calibration step which asks the user to perform a particular activity, at his/her speed, to calibrate the size of the window, could decrease the predisposition of the user to use the application.

Another problem of the fixed overlapping sliding window is related to the training phase and involves the input data for the classifier: if we use examples of positive elements, i.e., stairsteps, which perfectly fits inside a single window, the training set is no longer adherent to reality. If we use examples which do not fit into a single window, it is more difficult to discriminate between positive and negative instances.

Finally, the overlapping sliding window approach brings a problem concerning energy consumption. In fact, since data is analyzed twice, due to the overlapping, then the total number of analyzed windows increases and this negatively affects the duration of the battery.

In order to reduce the input errors and the computational cost required by classification on a smartphone, we decided to change the way windows are segmented from a *time-based* segmentation to a *data-based* segmentation. In particular, we consider the typical behavior of the signal when a user climbs a stairstep. As shown in Figure 3b, the movement is characterized by a positive local maximum followed by a negative local minimum for the Z-axis, while the X-axis and the Y-axis get a much lower variation. Note that, with our method, this pattern is invariant to device orientation. Hence, our assumption is that a window which is suitable to be a stairstep always presents data with this particular pattern. An example illustrating how windows are built is presented in Figure 5b.

Using the approach described above, the duration of a window becomes an important additional feature that gives further information and can also be used during classification. A simple filter is defined discarding all the windows that do not respect a predefined time interval, i.e., the time necessary to perform a particular activity (in our case, to make a stairstep). In order to fix this time interval, we have roughly analyzed our training set and obtained that making a stairstep without running requires at least 300ms, and at most 2 seconds. Empirically, the adoption of this simple filter reduced significantly the number of windows to consider, thus saving energy.

Summing up, the proposed solution has several advantages with respect to the time-based sliding window. The first one is that we are sure that each stairstep will fit in a window and this makes the learning task much more easier. Moreover, it reduces the possible errors in the training set. The second advantage is that just the windows suitable for stairsteps are taken into account for the analysis, reducing the total computational cost for the smartphone. Finally, this approach allows to deal with user variability. In fact, even if different users require different amount of time to complete a stairstep, the window will be appropriately resized to contain it without the need of calibration.

7. REPRESENTATION AND FEATURES STANDARDIZATION

In the representation phase, the dynamic window resulting from the segmentation phase is transformed into a vector of real values representing the information obtained by the device sensors contained in that particular time window. This mapping will permit a natural application of standard machine learning methods to our task of recognizing whether the user is climbing stairs or not. The way data are represented is crucial for the effectiveness of a learning algorithm. Specifically, a good representation method will be able to maintain the information which is really relevant for the

task and reduce the noise in the data as much as possible.

In the following we formally describe our choices for the representation module. In particular, let the frequency of sampling be fixed. Then, each window obtained by the segmentation step will consist of a sequence of n vectors, $\mathbf{v}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i \in \{1, .., n\}$, each one consisting of the standardized values of acceleration with respect to the three axes.

We also consider two additional computed values that, in our opinion, can carry precious information. Firstly, we consider the norm of the vector $\mathbf{v}_i$ which well represents the global amount of activity being performed and, secondly, the average value of horizontal accelerations, namely $\frac{x_i+y_i}{2}$ which combines the horizontal activities in our fixed coordinate system in a single value.

More formally, for any single standardized vector $\mathbf{v}_i$ of a dynamic window, a new vector $\mathbf{s}_i \in \mathbb{R}^5$ can be built as follows:

$$\mathbf{v}_i = (x_i, y_i, z_i) \mapsto \mathbf{s}_i = (x_i, y_i, z_i, \|\mathbf{v}_i\|_2, \frac{x_i + y_i}{2})$$

Now, the entire sequence of observations in a window can be represented in the matrix $\mathbf{S} \in \mathbb{R}^{5\times n}$ where the vectors $\mathbf{s}_i$ are accommodated in columns. Finally, the representation of the sequence corresponding to the entire dynamic window is obtained by applying a simple transformation $\Phi : \mathbb{R}^{5\times n} \to \mathbb{R}^{74}$, by evaluation of different statistical estimates over the rows of $\mathbf{S}$.

Specifically, the 74 dimensions of $\Phi(\mathbf{S})$ are created evaluating standard statistics (i.e. average, standard deviation, variance and difference between minimal and maximal values) which are very common in time series analysis or evaluating the ratio among the statistical features above and the correlations from different sources [13, 11].

Table 2 presents a detailed description of the entire set of features we decided to compute in order to represent a sequence of acceleration measures in a dynamic window. The first set of 20 features considers standard statistics computed over the rows $\mathbf{S}_j$ of the sequence matrix $\mathbf{S}$. The next group of 40 features considers the ratio of the same statistics computed on different rows. An additional group of 4 features (61-64) takes in account the ratio between $\mathbf{S}_3$ and $\mathbf{S}_5$ statistics (the ratio between vertical and horizontal activities), and two other features, the Magnitude Area (MA) and the Signal Magnitude Area (SMA) of $\{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\}$. Finally, the last 10 features correspond to the value of the correlation between pairs of rows. For reason of computational complexity and energy consumption we didn't use the features in the Fast Fourier Transform (FFT) family.

In order to fairly compare our orientation-independence supporting method with other methods presented earlier in this paper, we used a similar representation mapping. In particular, exactly the same mapping is used for the Linear method. Concerning the Mizell method, that returns two measurements vectors instead of one, we have considered the natural extension of the function Φ applied to the two vectors independently and concatenating the resulting vectors, thus obtaining a new vector of dimension $74 \times 2 = 148$.

Features No.	Description
$1-5$	$Ave(\mathbf{S}_j)$, $j \in \{1, \ldots, 5\}$
$6-10$	$Std(\mathbf{S}_j)$, $j \in \{1, \ldots, 5\}$
$11-15$	$Var(\mathbf{S}_j)$, $j \in \{1, \ldots, 5\}$
$16-20$	$Max(\mathbf{S}_j) - Min(\mathbf{S}_j)$, $j \in \{1, \ldots, 5\}$
$21-30$	$\frac{Ave(\mathbf{S}_j)}{Ave(\mathbf{S}_h)}$, $j, h \in \{1, \ldots, 5\}, j > h$
$31-40$	$\frac{Std(\mathbf{S}_j)}{Std(\mathbf{S}_h)}$, $j, h \in \{1, \ldots, 5\}, j > h$
$41-50$	$\frac{Var(\mathbf{S}_j)}{Var(\mathbf{S}_h)}$, $j, h \in \{1, \ldots, 5\}, j > h$
$51-60$	$\frac{Max(\mathbf{S}_j)-Min(\mathbf{S}_j)}{Max(\mathbf{S}_h)-Min(\mathbf{S}_h)}$, $j, h \in \{1, \ldots, 5\}, j > h$
61	$\frac{Max(\mathbf{S}_3)}{Max(\mathbf{S}_5)}$
62	$\lvert\frac{Min(\mathbf{S}_3)}{Min(\mathbf{S}_5)}\rvert$
63	$\sqrt{Ave(\mathbf{S}_1)^2 + Ave(\mathbf{S}_2)^2 + Ave(\mathbf{S}_3)^2}$
64	$Ave(\lvert\mathbf{S}_1\rvert + \lvert\mathbf{S}_2\rvert + \lvert\mathbf{S}_3\rvert)$
$65-74$	$Corr(\mathbf{S}_j, \mathbf{S}_h)$, $j, h \in \{1, \ldots, 5\}, j > h$

Table 2: 74 features extracted from dynamic windows

Finally, it's well-known that the classification algorithms are badly influenced by features with different orders of magnitude. For this reason, we have rescaled all the features from the dynamic windows used to train the classifiers between $[-1, 1]$ with a linear transformation, in order to improve the classification results from the classifiers. The linear transformations (that are fixed and different for each feature) are applied to all the features before the classification takes place in order to improve the classification performance.

8. EXPERIMENTS AND RESULTS

In this section, we present experimental results we have obtained with a simulation of our system, with respect both to classification and energy consumption.

8.1 Classification setting and results

First, we have created a dataset with raw information captured from device sensors for any of the three methods selected: Mizell, Linear and our method. The dataset is composed by a total of 8000 windows with 1500 stairsteps. This information was acquired through direct data collection from seven different users (five adults and two children) using different smartphones. We recorded this data keeping the devices consistently with the body movements, e. g., hand held, in a backpack or in a handbag. The users hadn't other constraints with respect to where or how to keep the device when the data was collected. From this raw data, the methods presented in Section 7 have been used to compute a features vector for each dynamic window contained in the datasets. Data has been then manually labeled in order to use it with the supervised learning algorithms.

Several different algorithms are available in the machine learning literature to tackle this task. For the experiments presented in this paper we use algorithms from three different families: decision trees (DT), K-nearest neighbors (KNN) and a kernel-based SVM-like algorithm, namely KOMD [1] (Kernel Optimization of the Margin Distribution). We used our implementation of the KOMD algorithm, while for the decision trees and the k-nearest neighbors we used the Weka [7] implementation.

Frequency 20Hz									
	Mizell			Linear			Our method		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
DT	$0.741_{\pm 0.050}$	$0.728_{\pm 0.027}$	0.735	$0.720_{\pm 0.033}$	$0.737_{\pm 0.054}$	0.729	$0.750_{\pm 0.040}$	$0.753_{\pm 0.038}$	0.751
KNN	$0.758_{\pm 0.021}$	$0.725_{\pm 0.052}$	0.741	$0.740_{\pm 0.036}$	$0.722_{\pm 0.035}$	0.731	$0.746_{\pm 0.036}$	$0.719_{\pm 0.049}$	0.733
KOMD	$0.797_{\pm 0.017}$	$0.794_{\pm 0.028}$	0.796	$0.808_{\pm 0.016}$	$0.796_{\pm 0.018}$	0.802	$\mathbf{0.811_{\pm 0.029}}$	$\mathbf{0.800_{\pm 0.013}}$	**0.806**

Frequency 30Hz									
	Mizell			Linear			Our method		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
DT	$0.767_{\pm 0.022}$	$0.765_{\pm 0.035}$	0.766	$0.740_{\pm 0.035}$	$0.744_{\pm 0.032}$	0.742	$0.759_{\pm 0.043}$	$0.762_{\pm 0.030}$	0.760
KNN	$0.766_{\pm 0.055}$	$0.783_{\pm 0.032}$	0.774	$0.769_{\pm 0.038}$	$0.741_{\pm 0.031}$	0.755	$0.789_{\pm 0.019}$	$0.737_{\pm 0.053}$	0.762
KOMD	$0.798_{\pm 0.044}$	$0.801_{\pm 0.031}$	0.800	$0.832_{\pm 0.033}$	$0.832_{\pm 0.033}$	0.832	$\mathbf{0.859_{\pm 0.034}}$	$\mathbf{0.860_{\pm 0.023}}$	**0.860**

Frequency 50Hz									
	Mizell			Linear			Our method		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
DT	$0.753_{\pm 0.050}$	$0.736_{\pm 0.042}$	0.744	$0.718_{\pm 0.055}$	$0.742_{\pm 0.030}$	0.730	$0.761_{\pm 0.071}$	$0.747_{\pm 0.029}$	0.754
KNN	$0.794_{\pm 0.052}$	$0.772_{\pm 0.023}$	0.783	$0.802_{\pm 0.051}$	$0.768_{\pm 0.033}$	0.784	$0.812_{\pm 0.025}$	$0.772_{\pm 0.026}$	0.792
KOMD	$0.806_{\pm 0.025}$	$0.803_{\pm 0.034}$	0.804	$0.850_{\pm 0.024}$	$0.854_{\pm 0.033}$	0.852	$\mathbf{0.864_{\pm 0.033}}$	$\mathbf{0.872_{\pm 0.039}}$	**0.868**

Table 3: Experimental results of Precision and Recall with standard deviation and F_1 score for the algorithms and methods used with frequency of sampling of **20**Hz, **30**Hz and **50**Hz. The result highlighted has the largest value of $\boldsymbol{F_1}$

Raw data was originally sampled with a frequency of 50Hz but since we are interested in finding the best trade-off between performance and energy consumption, we have also created two different datasets with sub-sampled frequencies of 20Hz and 30Hz. A higher frequency corresponds to a more accurate information given to the system, but obviously, it also corresponds to a higher energy consumption for the devices.

In our binary classification task, the distribution of the labels is greatly imbalanced since we have far more negative examples than positive ones. For this, we evaluated *Precision* and *Recall* (instead of accuracy) to obtain a more proper performance estimation for the different experimental settings. Then, we used the F_β-score (with $\beta = 1.0$) to combine recall and precision in a single effectiveness score. The precision (or positive predictive value) is calculated as $TP/(TP+FP)$, the recall (or sensitivity) as $TP/(TP + FN)$ while the F_1-score as $2(Precision \cdot Recall)/(Precision + Recall)$, where TP stand for True Positives, FP as False Positives and FN as False Negatives.

Starting from the original datasets with manually assigned labels for each dynamic window, each obtained dataset has been split randomly in three independent subsets: Training set (80% of the examples, used to train the supervised algorithm), Validation set (10% of the examples of the original dataset, used to find the best parameters for the algorithms) and the Test set (10% of the examples, used to evaluate the algorithms performance).

We repeated several times the same type of simulation with different randomly splits (training, validation and test sets) and averaged the results. The obtained results over the test set are summarized in Table 3.

These results clearly show that the proposed method for the support to the orientation-independence combined with the KOMD classification algorithm obtains the best performance against all the other combination of methods and algorithms. The results are satisfactory considering the high difficulty of the classification task, with the few samples contained in each single dynamic window. Also, we were interested in finding the best trade-off between energy consumption and performance. For this, we can see that the best compromise is obtained at 30Hz, as we can notice that the difference in performance between 50Hz and 30Hz is not statistically significant considering the standard deviation. Moreover, our method with KOMD shows an higher F_1 score at 30Hz than the other classifiers at 50Hz.

8.2 Energy consumption results

When working with mobile devices, and in particular with smartphones, energy consumption is an issue to consider very carefully. In fact, if an application wastes a lot of energy, requiring a lot of computational resources, the user may need to recharge the smartphone very often, in the worst case, more than once a day. As already discussed, waste of energy can be one of the reasons for users to remove an application from the smartphone.

Therefore, a deep analysis of how an application uses resources, and which are the computationally expensive tasks, is a fundamental requirement during development. In particular, the accelerometer and the rotation sensor can retrieve data at different frequencies, with different costs in terms of energy consumption. Different frequencies lead to different classification precision, so it is necessary to find the right trade-off between classification precision and energy consumption.

Several applications and tools to measure energy consumption on smartphone were proposed in literature [8, 10]. Some of them must be installed on the smartphone and work, in background, analyzing the power consumption of running applications [16]. Even if this information could provide an idea of how much expensive is a particular application, the data provided cannot be considered too precise, since the

inevitable overhead introduced by this applications is unknown, and not fixed.

For this reason, to measure the energy consumption we decided to use a tool, that differently from the other software solutions, do not require to be installed on the smartphone but measures the energy that is "requested" by the smartphone to the battery. In this way, it is possible to measure the real requested energy without overhead. The tool is the Monsoon Power Monitor[3], i. e., an hardware device with a software for remote data reading. This tool allows to analyze data of energy consumption of any device that uses a single lithium battery. The information retrieved are "Energy Consumption", "Average power" and "Average current", "Expected battery life" etc. This information are extremely important, since it is possible to evaluate how energy is used, for how much time, which are the tasks that are more expensive etc. Therefore, we can compare the three different methods for data acquisition and standardization, i. e., the Mizell's method, the linear sensor and our method to eliminate gravity information and translate each data to a fixed coordinates system, to understand which is the best method, in terms of classification precision and energy consumption.

We use a Samsung Galaxy i9250, with a theoretical battery capacity of 1750mAh, as test smartphone. Even if this smartphone is not up to date, we do not want to consider absolute values of energy consumption, but we want to determine which is the less expensive method, and this information is independent from the chosen smartphone.

Our results are presented in Table 4. The considered metric is "Consumed Energy", which is the energy necessary to acquire data from the smartphone and apply one of the three methods for two minutes. We tested the three methods with the three target frequencies of data acquisition (20Hz, 30Hz and 50Hz), and we repeated the experiments several times to get a mean final value.

	Energy consumption (μAh)		
Frequency	**Mizell**	**Linear**	**Our method**
20Hz	$7852_{\pm 34}$	$8726_{\pm 3}$	$8695_{\pm 8}$
30Hz	$7930_{\pm 16}$	$8802_{\pm 28}$	$8713_{\pm 27}$
50Hz	$8094_{\pm 30}$	$8884_{\pm 12}$	$8871_{\pm 25}$

Table 4: Energy consumption analysis for the three orientation methods

As it is easy to see, the Mizell method is the less consuming one because, differently from the other two methods, it requires only data acquired from the accelerometer, thus reducing the total amount of consumed energy. Unfortunately, this method has low performance, since it performs worse than the other two methods for every data acquisition frequency, with the higher precision equal to 80%, while the other two methods can reach 86%.

Comparing together the other two methods, we can see that our method is slightly less expensive compared to the use of the linear acceleration sensor. This is an extremely good result, since we are able to reach higher precision at every acquisition frequency, consuming less energy. For the final application, we decided to use the KOMD algorithm and our method at a frequency of 30Hz, since with the higher value (50Hz) the reached precision is not significantly better, but the energy consumption is higher.

Another key aspect introduced by our approach is the data-driven window (segmentation) instead of time-based windows. Beyond the simplification of the learning and the classification task, we want to compare these two kind of window definition in terms of energy consumption. We tested both approaches in our application acquiring data at 30Hz, and using KOMD as classification algorithm. For the time-based windows, we tested the overlapping sliding windows approach with a duration of 500ms and an overlapping of 50%. The final results are reported in Table 5.

	Segmentation approach	
	Time-based window	**Data-driven window**
Energy consumption (μAh)	$14535_{\pm 17}$	$12554_{\pm 14}$

Table 5: Energy consumption comparison between time-based and data-driven windows

As we can see from this result, the data-driven approach lets save more energy (more than 15%) with respect to the time-based sliding windows. This comes from the fact that the number of windows built using the data-driven approach that are classified using the KOMD classifier are much lower, since only those with a suitable pattern and duration are further analyzed. With time-based windows, this first filter is not applied, and so the classifier has to classify all the possible windows, that means more energy consumption.

8.3 Counting task results

In this section, the best trade-off between classification and energy consumption has been selected to simulate the real task that our algorithm has to tackle, i.e. counting stairsteps. As quality measure for this task we have evaluated the average of the stairsteps counted by the algorithm with respect to the real number of stairsteps climbed by the user. The combination of KOMD with our orientation-independence method has been chosen as classifier. We have selected the frequency of 30Hz for sampling the raw signals from the sensors as a trade-off between performance and energy consumption. We have evaluated the average of the performance using different random splits of the training set (80%) and the test set (20%) in order to obtain significant results from the dataset used in Section 8.1. The counting average that we have obtained from this experiments is $99.4\%_{\pm 4.1}$. This result is evaluated over approximately 1600 windows and 300 stairsteps (that are not used in the training part). So, our algorithm counted in average about 298 stairsteps (over 300) and 1302 other windows (over 1300). In order to complete the analysis of the performance of our algorithm, in the same experiments we have obtained an *Area Under Curve* (AUC) of $93.2\%_{\pm 1.3}$ and an *Accuracy* of $86.3\%_{\pm 1.7}$.

9. CONCLUSIONS AND FUTURE WORKS

In this paper, we have presented a new method for *real time* stairsteps recognition and counting. Our proposal is independent from the orientation of the smartphone and from

[3]http://www.msoon.com/LabEquipment/PowerMonitor/

the walking speed of the users, since it is based on a self-adjusting size of the window for the analysis of data.

We reported experiments showing that using our orientation-invariant method to translate data from sensors into a fixed coordinates system and a data-driven window segmentation, we are able to obtain a very accurate recognition, comparable or even better than the one obtained with more time-demanding methods. In particular, using a state-of-the-art kernel method (KOMD) we obtain the best performance against all the other combination of methods and algorithms, with a precision of 86% and a recall of 87%, with a sampling rate of 50Hz.

The goal of this study was the use of the classification algorithm inside *ClimbTheWorld*, a real time smartphone application that aims at persuading people to use stairs instead of elevators or escalators. For this reason, key aspects as battery lifetime and low resources of the device have been taken into account. Our experiments have shown that the proposed method performs better than the native Android solution in terms of consumed energy. The best trade-off between energy saving and precision is reached with KOMD classification algorithm combined with our method for data acquisition, using a sampling rate of 30Hz (in this case, the obtained precision is 85%, recall is equal to 86%). To the best of our knowledge, this is the first work that deals with energy consumption issues in a classification problem. Finally, we have obtained a counting performance of 99.4% using this configuration.

Although precision and recall (and consequently the counting performance) obtained by our solution are not affected by the smartphone orientation, we noticed a performance degradation when the user is carrying the smartphone on his/her trouser pocket.

Future works will be dedicated to solve this problem. A possible approach could be to recognize this case using information from proximity or light sensors. Another solution could be to analyze the user movement to find out when he/she puts the smartphone on his/her trouser pocket.

Another issue we would like to investigate in the future is the refinement of the output from the classification algorithm using data history. For instance, if the classifier recognizes a stairstep, the following window has a good chance to contain a stairstep too.

10. REFERENCES

[1] F. Aiolli, G. D. S. Martino, and A. Sperduti. A kernel method for the optimization of the margin distribution. In *ICANN*, pages 305–314, 2008.

[2] A. Anjum and M. Ilyas. Activity recognition using smartphone sensors. In *IEEE Consumer Communications and Networking Conference (CCNC 2013)*, pages 914–919, Jan 2013.

[3] M. Ayabe, J. Aoki, K. Ishii, K. Takayama, and H. Tanaka. Pedometer accuracy during stair climbing and bench stepping exercises. *Journal of sports science & medicine*, 7(2):249, 2008.

[4] L. Bloom, R. Eardley, E. Geelhoed, M. Manahan, and P. Ranganathan. Investigating the relationship between battery life and user acceptance of dynamic, energy-aware interfaces on handhelds. In *in Proc. Int. Conf. Human Computer Interaction with Mobile Devices & Services*, pages 13–24, 2004.

[5] A. Brajdic and R. Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*, pages 225–234.

[6] T. M. Do, S. W. Loke, and F. Liu. Healthylife: An activity recognition system with smartphone using logic-based stream reasoning. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 188–199. Springer, 2013.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[8] R. Mittal, A. Kansal, and R. Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual International Conference on Mobile Computing and Networking (MobiCom '12)*, pages 317–328.

[9] D. Mizell. Using gravity to estimate accelerometer orientation. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers (ISWC'03)*, page 252.

[10] A. Pathak, Y. C. Hu, and M. Zhang. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys '12)*, pages 29–42.

[11] S. Pirttikangas, K. Fujinami, and T. Nakajima. Feature selection and activity recognition from wearable sensors. In H. Youn, M. Kim, and H. Morikawa, editors, *Ubiquitous Computing Systems*, volume 4239 of *Lecture Notes in Computer Science*, pages 516–527. Springer Berlin Heidelberg, 2006.

[12] M. Shoaib, H. Scholten, and P. Havinga. Towards physical activity recognition using smartphone sensors. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 80–87, Dec 2013.

[13] P. Siirtola and J. Röning. Recognizing human activities user-independently on smartphones based on accelerometer data. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(5):38–45, 06/2012 2012.

[14] World Health Organization. Physical Inactivity: A Global Public Health Problem http://www.who.int/dietphysicalactivity/factsheet_inactivity/en/.

[15] W. Wu, S. Dasgupta, E. E. Ramirez, C. Peterson, and G. J. Norman. Classification accuracies of physical activities using smartphone motion sensors. *Journal of medical Internet research*, 14(5):e130, 2012.

[16] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha. Appscope: Application energy metering framework for android smartphones using kernel activity monitoring. In *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, USENIX ATC'12.