Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 15–25

EXPLORING SMARTPHONE SENSORS FOR FALL AND EMERGENCY DETECTION

I. N. FIGUEIREDO, C. LEAL, L. PINTO, J. BOLITO AND A. LEMOS

Abstract: Falling, and the fear of falling, is a serious health problem among the elderly. It often results in physical and mental injuries that have the potential to severely reduce their mobility, independence and overall quality of life. Nevertheless, the consequences of a fall can be largely diminished by providing fast assistance. These facts have lead to the development of several automatic fall detection systems. Recently, many researches have focused particularly on smartphone-based applications. In this paper, we study the capacity of smartphone built-in sensors to differentiate fall events from activities of daily living. We explore, in particular, the information provided by the accelerometer, magnetometer and gyroscope sensors. A collection of features is analyzed and the efficiency of different sensor output combinations is tested using experimental data. Based on these results, a new, simple, and reliable algorithm for fall detection is proposed. In addition, we also present the algorithm for an innovative emergency application for smartphones, designated by knock-to-panic (KTP). This ingenious system enables users to simply hit their devices in order to send an alarm signal to an emergency service. This application can not only be seen as a backup in case of failure of the fall detection algorithm, but also as a complete and autonomous emergency system. Moreover, the simple and fast activation of KTP makes it a viable and potentially superior alternative to traditional emergency calls. In fact, it is our belief that KTP can be useful to the general population and not only to elderly persons. The two proposed methods are threshold-based algorithms and are designed to require a low battery power consumption.

Keywords: Fall detection, Knock-to-panic, Smartphone, Tri-axial accelerometer, Threshold-based method.

Received June 2, 2015.
This work was partially supported by the research project PTDC/MATNAN/0593/2012 of FCT (the Portuguese national funding agency for science, research and technology) and by CMUC (the Centre for Mathematics of the University of Coimbra - UID/MAT/00324/2013), funded by the Portuguese Government through FCT/MEC and co-funded by the European Regional Development Fund through the Partnership Agreement PT2020. The authors would also like to thank Gonçalo Quadros (CEO at *Critical Software*) for having suggested the study of the interesting KTP topic, as well as Luís Neves (CEO at *OnCaring*) and João Diogo Ramos (CEO at *Retmarker*) for their important organizational and technical assistance. *OnCaring* collaborated with the academic investigators, provided the smartphones, collected the data, implemented in JAVA and installed in the Android smartphones the proposed algorithms.

1. Introduction

Statistics and facts related with falls in elderly people is somewhat worrying. For instance, approximately one in every three people over the age of sixty five experience a fall, at least once a year, and these are the leading cause of hospitalization for this age group [17]. Another very concerning aspect of falls, among the elderly, is their reluctance to seeking treatment after suffering an injury. A further common occurrence is the elderly's inability to stand up after a fall, even when the physical injuries are not severe [18]. This event, known as "long lie", has a strong psychological impact on those affected, and it has been revealed that this experience is associated with a very high risk of morbidity and mortality [8]. Moreover, the economic impact of falls was estimated in 2000 to be $US19 billion in the US only [16]. All of this is even more relevant when one considers that the number of old people (above 60 years old) in the world is expected to increase from 841 million in 2013 to more than 2 billion in 2050 [14].

The previous findings, stress the necessity for healthcare providers to focus on measures to reduce the risk and severity of falls-related injuries. Automatic fall detection systems are an important component in this effort and is a current major research topic. In a general sense, fall detection is a multidisciplinary and complex problem which involves two main components: the conception of suitable devices and the recording of the device signals that are subsequently processed, with appropriate algorithms, in real time, with the goal of correctly distinguishing real falls from activities of daily living (ADL). Traditionally, fall detection systems are based on body-attached accelerometers and gyroscopes. The features generally used for fall detection are the magnitude of the acceleration, posture monitoring, change in orientation, vertical velocity, angular velocity, and angular acceleration [3, 11, 4, 19, 6]. Automated image analysis systems based on video camera images have also been proposed [13]. Other approaches like the *GoSafe* system (http://www.lifelinesys.com/content/), known as *PERS* (personal emergency response system), a commercial wearable device from *Philips*, allow users to push a emergency button in the event of a fall. However, these solutions have some drawbacks. The wearable devices are not well tolerated by the elderly while camera-based solutions can not be applied without violating privacy.

In response to some of these issues, many researchers are focusing their efforts on smartphone-based applications. In fact, the increasing popularity of smartphones makes them an attractive platform for the development of new fall detection systems. Moreover, smartphones are well accepted, even among the elderly population, and the already built-in communication facilities, including, *e.g.*, SMS (short message service) and GPS (global position system), makes them a perfect candidate for an automatic fall detection system that covers the detection and communication stages. The increasing number of built-in sensors, such as accelerometer, gyroscope, and magnetometer, is also highly advantageous to researchers. On the other-hand, smartphone systems also offers many challenges. Issues like the huge variety of devices, and inherently the massive amount of software and hardware, make the task of developing new algorithms a formidable challenge. Moreover, the portability of smartphones makes it almost impossible to predict or simulate all the scenarios that an algorithm might encounter. The reliability and quality of the smartphones sensors is also a question of concern. Finally, we can not neglect the impact of the algorithms on the smartphone performance and battery life. Smartphones are multitasking systems, which have limited resources, and they were not intended for this purpose.

Smartphone-based fall detectors have already been presented in the literature [1, 7, 20] and some dedicated applications, like *e.g.* the iFall [15], are available in the Android Play Store. A common aspect in all of these studies is the use of threshold-based algorithms and accelerometer data. For instance, Dai *et al.* in [7] propose four thresholds for the difference between the maximum and minimum values of the magnitude of the acceleration vector and vertical acceleration in four defined time windows. In the algorithm of Abbate *et al.* [1], a fall is simple defined as a peak in the magnitude of the acceleration higher than a threshold followed by a time period where the magnitude of the acceleration is lower than another threshold. The reason behind threshold methods and acceleration-only features is identical, to reduce battery drain. Threshold methods have lower complexity and computational cost, while the accelerometer has low power consumption when compared *e.g.* with the gyroscope. Note that accelerometers are also cheaper than gyroscopes and, therefore, more common in smartphones. For further review on fall detection methods and challenges we refer to [9, 10].

Whereas fall detection have emerged and gained considerable attention in the literature, emergency detection in a more general setting is an area in which little research has been carried out. Solutions similar to the aforementioned *PERS* systems could potentially be used, however, the poor acceptance by elderly and general population may be a difficult problem to overcome. One rare example of a study involving the detection of emergency scenarios using smartphones is presented in [12]. In that work, the authors defined an emergency scenario as a shock followed by a period of low or no activity. To identify this pattern they suggest the use of the smartphone accelerometer sensor. Namely, they characterize this type of event as a large and rapid change in the acceleration followed by a period of lower acceleration. According to the authors, a car accident or a sudden fall are two examples of the type of emergency events that can be detected by the algorithm. The reliability and accuracy of such approach is questionable and no experimental results were provided. Therefore, we believe that a better and more effective algorithm needs to be developed.

The purpose of this paper is to describe a new fall detection algorithm and an innovative emergency detection algorithm, herein designated by knock to panic (KTP) algorithm, as well as the evaluation of their performance in collected data. The fall detection algorithm is devised to detect a fall (a sudden incontrollable descent) suffered by the smartphone user. The KTP algorithm is prepared to identify a knock to panic movement. This latter consists in the movement of hitting the device, executed by the smartphone user. Both algorithms are threshold-based and rely only on the data information provided by the smartphone built-in accelerometer sensor. These two characteristics are of utmost importance since they help reducing battery power consumption, a crucial issue for smartphone users. In addition, the tests conducted for evaluating the two algorithms (and reported herein), as well as the currently on going tests in real environments (in senior care or senior monitoring contexts, both residential or domiciliary) carried out by our collaborative partner (http://oncaring.com/) confirm the very good performance of these detection algorithms. Moreover modified algorithms (for fall and KTP detection) are also briefly described in order to increase the battery life-time.

The remainder of the paper is organized in two main parts. In the first one, Section 2, we discuss fall detection. More precisely, in subsection 2.1, we detail the material and experimental setup, in subsection 2.2 we present the features extracted for analysis, and in subsection 2.3 algorithms and numerical results are discussed. We finish this first part in subsection 2.4 with a brief presentation of a support vector machine binary classifier. The second part, Section 3, is dedicated to KTP. The description of the algorithm takes place from subsection 3.1 to subsection 3.5 and we evaluate the algorithm performance with experimental data in subsection 3.6. Finally, we draw some conclusions and discuss future research directions.

2. Fall Detection

In this section we essentially describe threshold-based algorithms for fall detection for smartphone users. A fall is a sudden incontrollable descent suffered by the smartphone user. It has a particular pattern that is encoded in the signals transmitted by the built-in sensors of the smartphone. These signals can be analysed and processed, in order to automatically recognize the target fall pattern movement. We start by describing the type of smartphones and sensors used in our tests as well as the data collected for the experiments, in subsection 2.1. Then we explain, in subsection 2.2, the relevant features for fall classification using the outputs of different sensors. In subsection 2.3 we define 4 different threshold-based algorithms and evaluate and compare their correspondent performance on the used data set. Finally, in subsection 2.4 we present the results obtained with a more sophisticated algorithm, a support vector machine (SVM) binary classifier, in our data set, and compare its performance with the most promising threshold-based algorithm defined in subsection 2.3, that relies only on the acceleration sensor. The latter algorithm has an important advantage over the SVM algorithm, because since it is simpler, the computational power is less, and consequently it does not increase too much the battery power consumption, which is a practical and crucial aspect for smartphone users.

2.1. Experimental Setup. In this subsection we describe the smartphones, the sensors, and the data that we gathered for the fall detection study. The smartphones used were a Samsung Galaxy Nexus and a Samsung Galaxy Nexus S, both equipped with the Android operating system (version 4.1.2). These devices have a wide range of sensors, including triaxial accelerometer, triaxial gyroscope, and magnetometer. In addition to these sensors, Android also provides, *e.g.*, the linear acceleration and the orientation of the device. These quantities are usually obtained through the fusion of sensor data, however, the exact method is not available, and the actual implementation may differ from device to device. For completeness, a very brief description of these sensors and quantities is given herein. We also refer to Figure 1, where the smartphone reference frame (x-, y- and z- axis) and orientation angles (roll, pitch and azimuth) are depicted.

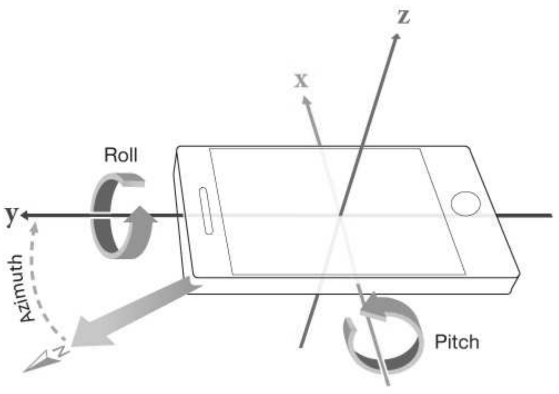

Figure 1. Smartphone reference frame and orientation angles.

The accelerometer sensor measures the acceleration force in meters per square second (m/s^2) applied to the device along each axis. We recall that this acceleration signal includes the effect of gravity. The gyroscope returns the angular velocity in radians per second (rad/s), *i.e.*, the rate of rotation around each one of the three axes. The magnetometer measures the magnetic field in microtesla (μ/T), and it is used to calculate the azimuth. The orientation sensor gives the rotation angles (roll, pitch, and azimuth) of the smartphone with respect to the correspondent axis, and their calculation involves the accelerometer, magnetometer and gyroscope sensors. Finally, the linear acceleration is the acceleration without gravity, also known as dynamic acceleration. This linear acceleration can be obtained by projecting the acceleration into a fixed coordinate system, using the orientation information, and removing the known gravity vector. Alternatively, a high-pass filter can be applied, for deriving the linear acceleration.

The data reported in this subsection were collected by the company *Oncaring* (http://oncaring.com/). The data-set consists of simulated falls and ADL. The simulated falls were performed by one young adult, and for safety reasons a mattress was used. The ADL collection involved a larger number of individuals. In total five young adults executed multiple ADL tasks. During these experiments, the smartphone was positioned in the trouser front pocket or in a belt worn around the hip. A total of 74 falls and 136 ADL

ADL	Total number
Walk	14
Walk downstairs and upstairs	9
Run	5
Sit down and stand up from a chair	15
Sit down and stand up from the floor	15
Sit down and stand up from a low stool	4
Sit down and stand up from a couch	4
Lie down and stand up from the bed	15
Lie down and stand up from the floor	6
Pick an object off the floor	8
Get in and out of the tub	5
Bump into someone	7
Sit down abruptly on a chair	8
Roll over on the floor	4
Kneel down to the floor	8
Get in and out of the car	5
Vertical jump	4

FALLS	Total number
Backward fall ending lying in lateral position	6
Backward fall ending lying	4
Forward fall with arm protection ending lying	7
Forward fall ending lying in lateral position	11
Lateral fall to the right ending lying	4
Lateral fall to the left ending lying	5
Lateral fall against wall ending lying	6
Fall from a chair ending lying	10
Run and fall ending lying	11
Walk and fall ending lying	10

Table 1. List of ADL movements and simulated falls performed by the volunteers.

were recorded overall. Among these, 6 falls and 37 ADL were obtained with the Samsung Galaxy Nexus S. The complete list of ADL activities is given in Table 1. This set of ADL was chosen in such a way that it would be representative of daily activities that can potentially cause false positives in fall detection threshold-based algorithms relying only on the acceleration sensor. Table 1 also lists a description of the type of falls that were collected. This data-set is relatively diverse with respect to both falls and ADL, as this Table 1 indicates.

For all the movements the signal of the gyroscope, accelerometer, liner acceleration, and orientation were recorded. Unfortunately, the gyroscope data obtained with the Samsung Galaxy Nexus S were lost because of technical problems. The accelerometer and gyroscope amplitude range was set at ± 20 m/s^2 and ± 50 rad/s, respectively. The sampling frequency was fixed at 100 Hertz (Hz) in the Samsung Galaxy Nexus and at 50 Hz in the Samsung Galaxy Nexus S. The acquired data is inherently affected by noise measurement. Here, we applied a first-order exponential low-pass filter (cut-off frequency of 5 Hz) for smoothing and noise reduction. The choice of this type of filter is motivated by their simplicity and suitability for real-time implementation.

2.2. Feature Extraction. The chosen features analyzed in this paper for fall detection are described in this section. We note that identical or similar parameters have been previously suggested by other authors, and without being exhaustive we refer for instance to [1, 7, 20, 15, 3, 11, 4, 19, 6] for more details.

We remark that these features, which correspond to appropriate quantifications of the sensors' signals, are then used for a binary classification of the different movements into fall or non-fall. This decision is based on a simple thresholding approach, that is afterwards explained in subsection 2.3, by checking for each movement the different features sequentially.

Firstly, we introduce some useful notations. At time t^n we denote the accelerometer data by $A^n = (A^n_x, A^n_y, A^n_z)$, for $n = 1, \dots, N$, with N the total number of samples, represent by $L^n = (L^n_x, L^n_y, L^n_z)$ the linear acceleration signal, and the output of the orientation sensor, roll, pitch, and azimuth angles, respectively, is represented by $O^n = (O^n_r, O^n_p, O^n_a)$. The measurements of the gyroscope at time t^n are denoted by $G^n = (G^n_x, G^n_y, G^n_z)$. Moreover, we also introduce the vectors

$$A = (A^n)_{n=1}^N, \quad L = (L^n)_{n=1}^N, \quad O = (O^n)_{n=1}^N, \quad \text{and} \quad G = (G^n)_{n=1}^N$$

as well as

$$A_x = (A^n_x)_{n=1}^N, \quad A_y = (A^n_y)_{n=1}^N, \quad A_z = (A^n_z)_{n=1}^N,$$

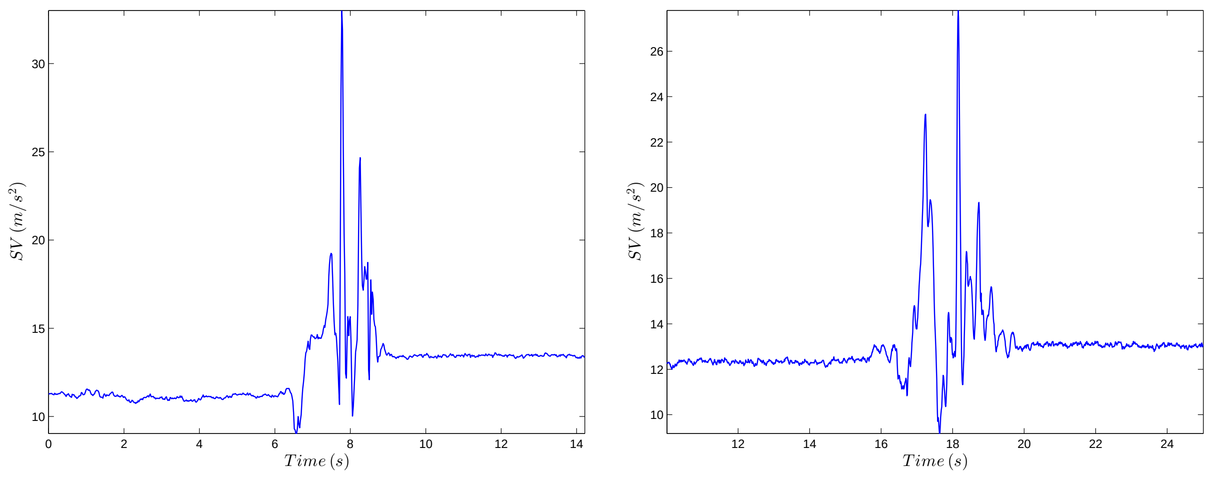

FIGURE 2. The SV pattern for a simulated "backward fall ending lying" (left) and the ADL "sit down and stand up from the floor" movement (right).

and similar definitions apply to L_x, L_y, L_z, to O_r, O_p, O_a and to G_x, G_y, G_z. In addition we remark that the number of samples can vary with the frequency of the signal.

Feature SV (sum of the components of the acceleration vector). The first feature used to distinguish between fall and ADL is the sum of the absolute value of the components of the acceleration vector (SV), which is defined as follows,

$$SV = (SV^n)_{n=1}^N, \quad SV^n = |A_x^n| + |A_y^n| + |A_z^n|, \tag{1}$$

with $|\cdot|$ representing the absolute value. A typical example of this quantity for a fall event is shown in Figure 2. The peaks in SV occur as a consequence of the impact of the human body with the ground. In the same figure, we also plot the measured SV for an ADL. Note the smaller maximum for SV in the latter.

Although SV seems to be an important feature for fall detection it might not be enough for distinguishing correctly a fall from a non-fall. Therefore extra features, as the following ones, are necessary for a more correct classification.

Feature OV (orientation variation). It is expected that during a fall event, the acceleration signal will exhibit rapid and significant variations along all the three directions x, y and z. In order to quantify this behavior we define

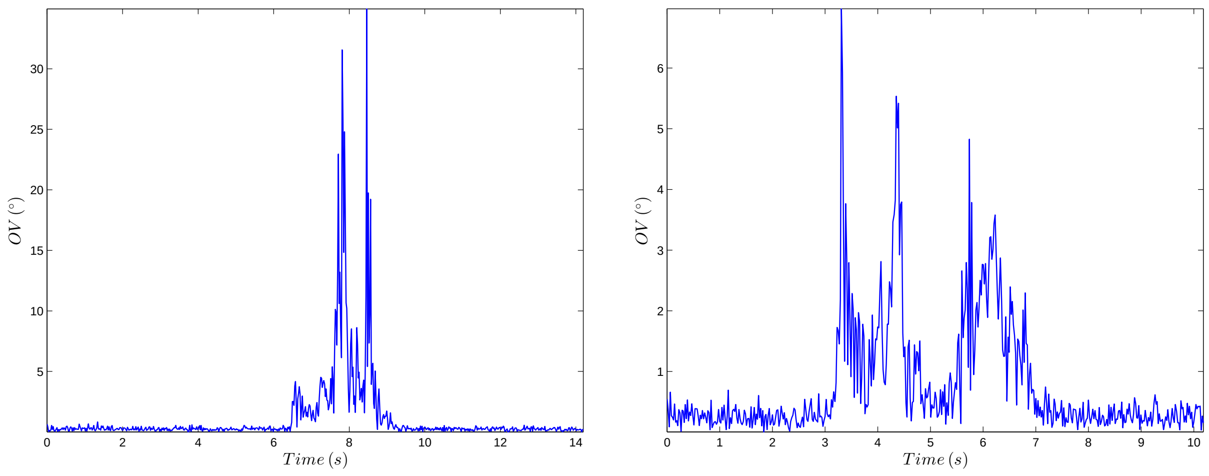

FIGURE 3. Plot of OV resulting from a simulated fall (left) and the ADL "sit down and stand up from a chair" movement (right).

a new fall feature based on the change of the angle between two consecutive acceleration readings. This quantity, herein called orientation variation (OV), is defined by

$$OV = \left[\cos^{-1}\left(\frac{A^n \cdot A^{n+1}}{\|A^n\|\|A^{n+1}\|}\right)\right]\frac{180}{\pi}, \quad \text{for } n = 1, \ldots, N-1, \tag{2}$$

where $\| \ \|$ denotes the Euclidian norm, the dot symbol in the numerator stands for the scalar product of two vectors and $\frac{180}{\pi}$ results from radians to degrees conversion. As an example, Figure 3 shows the feature OV corresponding to the acceleration signal of a simulated fall and a particular ADL, for comparison. The analysis of both figures reveals that during the fall event, OV reaches higher values than during the ADL task. For the fall the maximum value is bigger than 30 degrees (°) and smaller than 7° for the ADL.

Feature CO (change in orientation). Another feature that we have studied is the change in orientation (CO) of the device before and after a fall. We implement this feature by estimating the angle between two acceleration vectors. We proceed as follows. First, we define a time window of size $4s$ centered at the predicted fall time, that corresponds to the maximum of feature SV. Then, we create two vectors, $\bar{A}_b$ and $\bar{A}_e$, by averaging the acceleration data over the first second and last second of the window of size

$4s$, respectively. Finally, as in (2), we define the angle

$$CO = \left[\cos^{-1}\left(\frac{\bar{A}_b \cdot \bar{A}_e}{\|\bar{A}_b\|\|\bar{A}_e\|}\right)\right]\frac{180}{\pi}. \tag{3}$$

We expect a significant difference between the initial and final position of the device, and this variation must be reflected in the value of CO. Moreover, we take a $4s$ window because we estimate that a fall lasts approximately $2s$. Feature VA (vertical acceleration). The vertical component of acceleration (VA), is a feature usually considered in fall detection algorithms [4, 7]. It is defined by

$$VA = \frac{L \cdot g}{\|g\|}, \tag{4}$$

where g is the gravity vector, which can be calculated by subtracting the linear acceleration from the acceleration, that is $g = A - L$. VA provides a measure for the magnitude of the linear acceleration L in the direction of the earth gravity vector. Thus, it is plausible to expect that a fall will produce large values of VA, which may be useful to separate fall from non-fall. This prediction is illustrated in Figure 4 that shows the profile of VA during a fall and during an ADL movement.

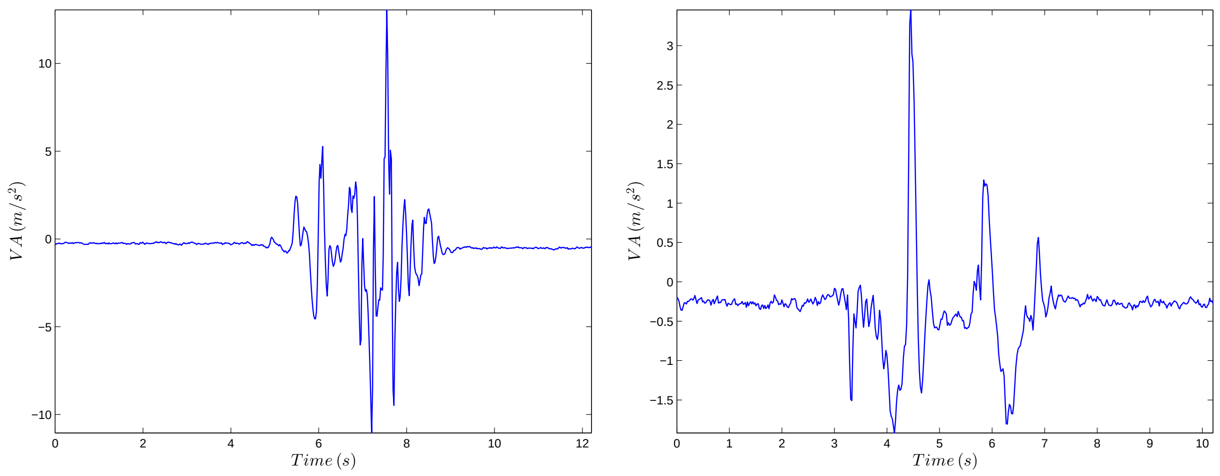

FIGURE 4. The vector VA resulting from a simulated "forward fall ending lying in lateral position" (left) and the ADL "sit down and stand up from a chair" movement (right).

Feature OD (orientation angles). Now we explore the orientation sensor, by defining a feature that is based on the difference of consecutive values of the sensor output, the roll, pitch and azimuth angles. More precisely, we measure the quantity

$$OD = |O_r^{n+1} - O_r^n| + |O_p^{n+1} - O_p^n| + |O_a^{n+1} - O_a^n|, \quad \text{for } n = 1, \dots, N-1, \tag{5}$$

where $|.|$ stands for the absolute value. With this approach we attempt to capture the rapid change in orientation that occurs during a fall. The pitch (O_p) and azimuth (O_a) angles belong to the interval $[-180°, 180°]$ and $[0°, 360°]$, respectively. Note that when a measured value oversteps these limits, a discontinuity or jump of 360° occurs. For instance, the azimuth angle jumps from 0° to 360° (or 360° to 0°) when the measured value is approaching 0° (or 360°). Therefore before calculating (5), it is necessary to correct the pitch and azimuth angles. Regarding the roll angle it does not show this behavior, and its values change continuously in the interval $[-90°, 90°]$.

Figure 5 displays the OD feature for a fall and an ADL movement. In this particular example, the difference in magnitude between the two cases is evident.

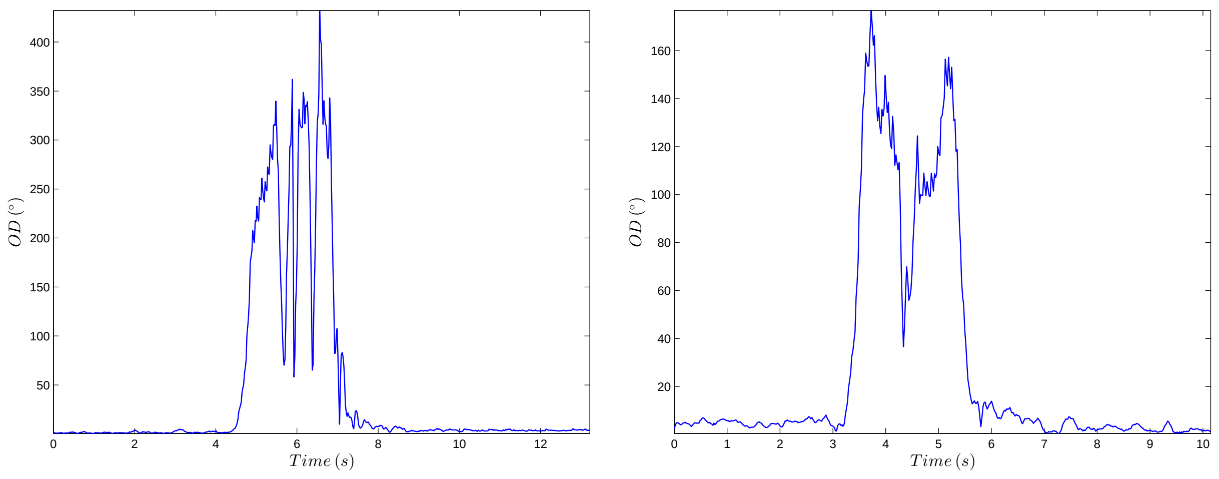

FIGURE 5. The vector OD for a simulated "backward fall ending lying in lateral position" (left) and the ADL "pick an object off the floor" movement (right).

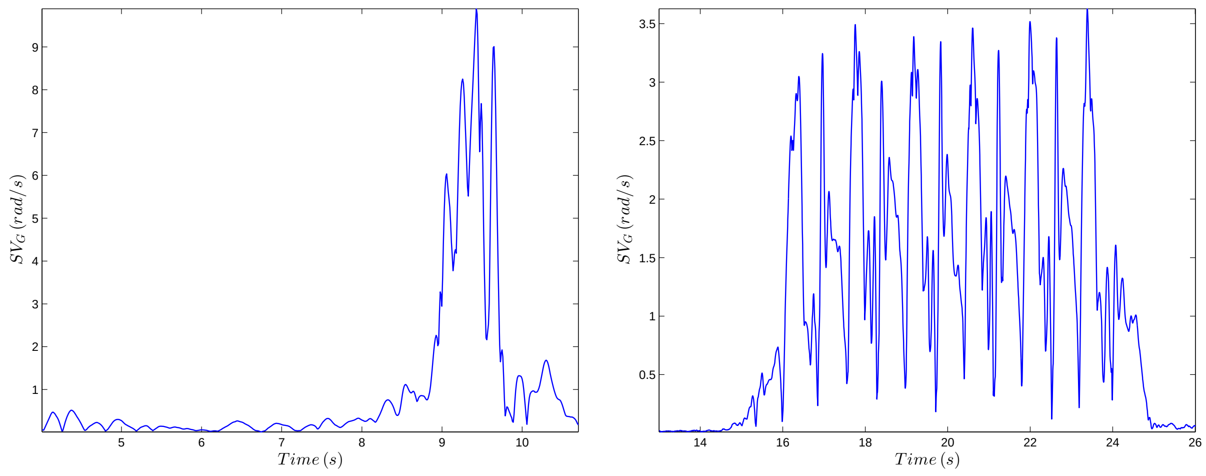

FIGURE 6. The calculated SV_G for a simulated "lateral fall against wall ending lying" (left) and the ADL "walk" movement (right).

Feature SV_G and SV_{GA} (gyroscope). Finally, we consider the information of the gyroscope sensor. What we predict is that a fall leads to significant values in the angular velocity and angular acceleration, when compared to an ADL movement. Moreover, in comparison with ADL tasks, those values should be sufficiently large to provide an accurate distinction between fall and ADL. As mentioned in subsection 2.1, the angular velocity is available as the output of the gyroscope sensor. The components of the angular acceleration, denoted herein by $GA^n = (GA^n_x, GA^n_y, GA^n_z)$ at time n, can be approximated by forward finite differences, that is

$$GA^n_x = \frac{G^{n+1}_x - G^n_x}{\Delta t}, \quad \text{for } n = 1, \dots, N-1, \tag{6}$$

where Δt is the inverse of the signal frequency (similarly definitions apply to GA^n_y and GA^n_z). Based on this reasoning, we study the values of the following two features, the sum of the absolute value of the components of the angular velocity

$$SV_G = (SV^n_G)^N_{n=1}, \quad SV^n_G = |G^n_x| + |G^n_y| + |G^n_z|, \tag{7}$$

and the sum of the absolute value of the components of the angular acceleration

$$SV_{GA} = (SV^n_{GA})^N_{n=1}, \quad SV^n_{GA} = |GA^n_x| + |GA^n_y| + |GA^n_z|. \tag{8}$$

The SV_G curves displayed in Figure 6 reveal that, at least in this case, our expectations were confirmed. In the simulated fall, the maximum value of SV_G is bigger than the double of the value of SV_G obtained in the ADL. The analysis of SV_{GA}, not shown here, allows to draw an identical conclusion.

2.3. Fall Detection Algorithms and Results. In this subsection we describe and analyse four fall detection threshold-based algorithms, relying on the features previously defined. Besides the evaluation of the performance of these algorithms, a main goal is also to understand if the use of the linear acceleration, orientation or gyroscope data can significantly improve the performance of the fall detection algorithm that relies only in the acceleration sensor, but somehow incorporates the information about the orientation of the device. Therefore, we consider the following algorithms.

- Alg_1 that only uses acceleration data by combining sequentially the features SV, OV and CO.
- Alg_2 that consists of Alg_1 plus feature VA (the vertical acceleration).
- Alg_3 that adds the orientation feature OD to Alg_1.
- Alg_4 that adds the gyroscope information features SV_G and SG_{GA} to Alg_1.

Next, a detailed description of the threshold-based algorithm Alg_1 is presented. We omit the descriptions of the other three algorithms, they are only briefly outlined, because of the their similarity with the structure of Alg_1.

Algorithm Alg_1

1. Let C_{sv}, C_{ov} and C_{co} be 3 pre-defined and fixed thresholds.
2. Alg_1 starts by applying a low-pass filter to the original acceleration signal in a sliding window of size $2s$. Then, with the filtered data, the feature SV is computed, as defined in (1) and let t_{sv} be the time corresponding to the maximum value of SV in this window.
3. If this maximum value is above the critical threshold C_{sv}, the movement might be a fall and Alg_1 proceeds to the next point. Otherwise the movement is not a fall and Alg_1 stops.

4. Then equation (2) is used to compute OV within the time window $[t_{sv}-1, t_{sv}+1]$. If the maximum value of OV in $[t_{sv}-1, t_{sv}+1]$ exceeds a threshold value C_{ov}, the movement might be a fall and Alg_1 proceeds to the next point. Otherwise the movement is not a fall and Alg_1 stops.

5. The feature CO is computed, as explained in subsection 2.2, that is by averaging the acceleration data over $[t_{sv}-2, t_{sv}-1]$ and over $[t_{sv}+1, t_{sv}+2]$, and by using equation (3).

   If the obtained value of CO is above a threshold C_{co} the algorithm considers that a fall has occurred and an emergency signal is sent to the monitoring system. Otherwise Alg_1 ends and the movement is not a fall.

Algorithms Alg_2, Alg_3, and Alg_4 correspond to variants of Alg_1 obtained by including extra different features. In all cases, these extra features are calculated in the time window $[t_{sv}-1, t_{sv}+1]$ as in the computation of OV. For Alg_2, we monitor the maximum value and the l_1-norm of VA (defined by $\sum_{n=1}^{N}|VA^n|$ and normalized by the total number of points N, whenever necessary, to account for frequency differences). For Alg_3 we measure the maximum value of OD, and for Alg_4 the maximum values of the features SV_G and SV_{GA} are considered. Moreover, as described above, three thresholds are used in Alg_1, C_{sv}, C_{ov}, and C_{co}. Besides these three thresholds, we also define the thresholds C_{va} and $C_{va}^{l_1}$ for Alg_2, C_{od} for Alg_3 and C_{svg} and C_{svga} for Alg_4, with obvious meaning.

All these algorithms were tested on the data-set described in Table 1 of subsection 2.1. The performance of the algorithms was quantified by using sensitivity and specificity measures. In this case, sensitivity measures the capacity of the algorithm to detect a fall. It is defined by

$$\text{Sensitivity} = \frac{\text{number of TP}}{\text{number of TP} + \text{number of FN}},$$

where TP represents a true positive, that is a fall correctly detected by the algorithm and FN represents a false negative, that is a fall not detected by the algorithm. On the other hand, specificity measures the reliability of the algorithm, *i.e.*, the capacity of the algorithm to reject non-fall events, and is defined by

$$\text{Specificity} = \frac{\text{number of TN}}{\text{number of TN} + \text{number of FP}}.$$

Here, TN stands for a true negative, that is an ADL not identified as a fall by the algorithm, and FP stands for a false positive, that is an ADL erroneously classified as fall, by the algorithm.

We explain now, how we have chosen the thresholds C_{sv}, C_{ov}, C_{co}, C_{va}, $C_{va}^{l_1}$, C_{od}, C_{svg} and C_{svga}, for the features SV, OV, CO, VA, OD, SV_G and SV_{GA}, respectively. We first choose a training subset of the whole data set, and in this training subset we compute the values of all the features (SV, OV, CO, VA, OD, SV_G and SV_{GA}) for all the movements (fall and ADL). Among all the values we choose, for each feature, the maximal value (which is the threshold) such that the 6 thresholds provide approximately 100% of sensitivity for Alg_1, Alg_2, Alg_3 and Alg_4. This high level of sensitivity is essential in fall detection systems. Afterwards, with these set of thresholds we compute the specificity and the sensitivity for the whole dataset. The used threshold values as well as the maximum value for all the falls recorded are shown in Table 2.

Feature	Fall Value	Fall description	Threshold
SV	23.45	Fall from a chair ending lying	$C_{sv} = 23$
OV	18.53	Backward fall ending lying	$C_{ov} = 18$
CO	66.33	Forward fall with arm protection ending lying	$C_{co} = 65.5$
VA	4.93	Fall from a chair ending lying	$C_{va} = 4.72$
VA	1.49	Backward fall ending lying	$C_{va}^{l_1} = 1.48$
OD	253.88	Forward fall with arm protection ending lying	$C_{od} = 250$
SV_G	6.02	Backward fall ending lying	$C_{svg} = 5.9$
SV_{GA}	111.21	Backward fall ending lying	$C_{svga} = 110$

Table 2. Features and thresholds adopted in the fall detection algorithms.

The results obtained for each algorithm are summarized in Table 3 and reveal that Alg_1 performs well with 92.65% for the specificity. This specificity value corresponds only to 10 FP. Comparing these results with the performance of algorithms Alg_2, Alg_3 and Alg_4 it is possible to conclude that apparently the features used in Alg_1 are good enough and no significant advantage is gained by increasing the number of features with extra sensor data.

This conclusion is very important when we take into consideration the computational and power costs. We cannot neglect the limited computational and battery power of smartphone devices. In fact, the power required by the gyroscope, used in Alg_2, Alg_3, and Alg_4, is much higher than the power demand of the accelerometer sensor used in Alg_1.

Table 3 shows that Alg_2 eliminates only two more FP, leading to 8 FP, whereas with Alg_3 the number of FP is 9, *i.e.*, only one more FP is correctly classified. A direct comparison between Alg_1 and Alg_4 (last two rows in Table 3 that correspond to the results in a subset of the data set that also has the gyroscope information) shows almost identical results. In this case the specificity of Alg_1 is 91.92% and that of Alg_4 is 93.94%. These values correspond to 8 FP and 6 FP, respectively.

	TP	FP	TN	FN	Sensitivity	Specificity
Alg_1	74	10	126	0	100%	92.65%
Alg_2	74	8	128	0	100%	94.12%
Alg_3	74	9	127	0	100%	93.38%
Alg_1	68	8	91	0	100%	91.92%
Alg_4	68	6	93	0	100%	93.94%

TABLE 3. Sensitivity and specificity results for each algorithm. The last two rows correspond to the results in a subset of the data set that has the gyroscope information (some data do not have this information).

To gain more insight about Alg_1 and the influence of the 3 different features SV, OV and CO, we present in Table 4 the sensitivity and specificity results for different features combinations. In particular, we observe that none of these 3 features can be removed without substantially reducing the sensitivity. We also note that feature OV has a strong positive impact in the algorithm performance. To the best of authors knowledge this is the first time that the feature OV is used in a fall detection algorithm.

With respect to the number of FP of Alg_1 displayed in Table 3, most of them are due to the ADL "sit down abruptly on a chair" movement (4 FP), followed by "get in and out of the car" movement (3 FP). This means, that in more than 50% of the cases, these two ADL originate FP. These are very

Features	TP	FP	TN	FN	Sensitivity	Specificity
$SV + OV$	74	38	98	0	100%	72.06%
$SV + CO$	74	56	80	0	100%	58.82%
$OV + CO$	74	19	117	0	100%	86.03%

Table 4. Sensitivity and specificity results for Alg_1 by considering only two different combinations among the 3 features SV, OV and CO.

similar movements, that can generally be classified as "sitting abruptly". The pattern for this type of movement resembles a fall, and therefore it poses a real challenge to fall detection systems, especially the ones based solely on acceleration signals. The remaining FP were caused by the ADL "kneel down to the floor" movement (1 FP), and the ADL "lie down and stand up from the floor" movement (2 FP).

We also have tested two algorithms that only use the features SV and VA, and SV and OD, and obtained specificity values of 46.10% and 40.28%, respectively, which reveals that these features are not reliable. These values can also be compared with those given in Table 4.

The results obtained in this study seem to be in disagreement with some results reported in the literature, but this discrepancy is apparent, and the reason is due to the fact that the other authors use different location for the sensors and also the quality and type of the chosen sensors is different from ours. For instance in [3], 100% sensitivity and specificity were obtained with an algorithm that is based only on gyroscope data. In our data-set, an algorithm that relies only on SV, SV_G, and SV_{GA} results in 100% sensitivity and 71.84% specificity. However, in [3] the authors use an improved gyroscope sensor fixed to the trunk, which is a better place for the sensor in fall detection. We also tried to extract vertical velocity features from the vertical acceleration in a similar way to the one presented in [4]. However, our results were not satisfactory, while in that study the authors also obtained 100% sensitivity and specificity. But, again, we must stress that in this case better sensors than ours were used, and in addition, the sensors were fixed to the trunk.

Remark 2.1. *Finally we remark that we have conducted some experiments using a modified version of the fall detection algorithm Alg_1, with the goal* *of decreasing the smartphone battery consumption. The changes are listed below:*

1. *No low-pass filter is applied to the original acceleration signal.*
2. *In step 4 of algorithm Alg_1 the maximum value of OV is computed in the time interval centered in t_{sv} with size $0.7s$, instead of $2s$ as indicated before.*
3. *For the computation of feature CO the time interval $[t_{sv} - 2, t_{sv} + 2]$ of size $4s$, is replaced by a shorter time interval also centered in t_{sv}, but with size $3s$, and the averages $\bar{A}_b$ and $\bar{A}_e$ are created by averaging the acceleration data over, respectively, the first half second and last half second of this new window of size $3s$.*

The experimental results show the performance is a little bit worse as shown in Table 5 (compare with Table 3). We observe that if we keep the low-pass filter the results are very similar to those obtained with Alg_1 in Table 3. By removing the low-pass filter the values for feature SV increase, therefore the value C_{sv} was changed to 39.

Modified Alg_1	*Sensitivity*	*Specificity*	*TP*	*FP*	*TN*	*FN*
Changes 1, 2 and 3	*98.65%*	*82.35%*	*73*	*24*	*112*	*1*
Changes 2 and 3	*98.65%*	*91.91%*	*73*	*11*	*125*	*1*

TABLE 5. Sensitivity and specificity results for the modified fall detection algorithm Alg_1.

2.4. Support Vector Machine Classifier. In this subsection, we briefly discuss the results obtained with a support vector machine (SVM) binary classifier on our dataset and compare SVM and Alg_1 performances. For this SVM classifier, we use exactly the same features of Alg_1, that is SV (sum of the absolute value of the components of the acceleration vector), OV (orientation variation) and CO (change in orientation).

Before proceeding, we observe that in our study we did not consider the possibility of implementing this type of SVM algorithm for fall detection in smartphones, because as it requires more computational power than the threshold-based algorithms, it causes some technical difficulties associated with the lack of energy and computing power in smartphones devices. One possible solution to this problem would be to transmit the data (or at least some of the data), in real time, to a remote computer to perform further analysis. This procedure would allow the use of more sophisticated classification methods like, for example, SVM, whose results are herein discussed.

For this experiment, we used the SVM Toolbox LIBSVM developed for Matlab software [5], and the RBF (Radial Basis Function) kernel was adopted. In order to avoid over-fitting, two-fold cross validation was employed. Therefore, our database was partitioned into two sets. Each set contained equal number of falls and ADL, 37 and 68, respectively, and an attempt was made to evenly distribute all the falls and ADLs across both sets.

The classification performance was evaluated by the accuracy measure, which is defined as follows

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of positives + Number of negatives}}.$$

In particular, we are more interested in the accuracy of the testing set, since it is considered a better indicator of the algorithm effectiveness than in the accuracy of the training set. Two parameters need to be optimized for SVM. We have followed the recommended "grid-search" procedure. Thus several pairs of these two parameters were tried, and the one with the best accuracy for the cross-validation of the testing sets was selected.

The best accuracy results for the set of 3 features $SV + OV + CO$, and also for other combinations, with only 2 out of these 3 features, are given in Table 6. For the 3 features $SV + OV + CO$, SVM gives an accuracy of

Features	Accuracy	Sensitivity	Specificity
$SV + OV + CO$	96.19%	95.95%	96.32%
$SV + OV$	85.24%	82.43%	86.77%
$SV + CO$	88.57%	85.14%	90.44%
$OV + CO$	95.71%	95.95%	95.59%

Table 6. Performance of the SVM classifier.

96.19%, a sensitivity of 95.95%, and a specificity of 96.32%. We also point out the performance of the combination $OV + CO$ is very similar to the one obtained with $SV + OV + CO$. In fact, the same sensitivity was obtained and the specificity value is only slightly lower. These results confirm the quality of the three features $SV + OV + CO$ and of the features selected, especially OV and CO, to accurately classify fall and ADL events.

Finally, in Table 7, we reveal the number of actions that were misclassified by SVM using the three features $SV + OV + CO$. It is interesting to note, that among the 5 FP, 4 are "sit down abruptly on a chair" movements, and 1 is the ADL "kneel down to the floor" movement. These 5 actions, with the exception of 1 ADL "sit down abruptly on a chair" movement, were already misclassified by the threshold-based approach Alg_1. The 3 FN correspond to "a lateral fall to the right ending lying" , "a forward fall with arm protection", and "a walk and fall ending lying".

Features	TP	FP	TN	FN
$SV + OV + CO$	71	5	131	3

Table 7. Classification results for $SV + OV + CO$ using SVM.

By comparing these results with those obtained for Alg_1 in Section 2.3, we conclude that they are very similar, and consequently the threshold-based algorithm Alg_1, although simpler has a performance as good as the more elaborated and complex SVM algorithm.

3. KTP Detection

In this section we focus our attention on the knock to panic (KTP) detection. After defining the KTP protocol, we describe an automatic threshold-based algorithm. It relies only on the acceleration data and the KTP pattern (see Figure 7). The structure of the KTP algorithm involves the following sequential three steps: Step 1- knocks' detection, Step 2 - verification of the stability conditions and Step 3 - verification of the oscillatory conditions. In Step 1 we look for the primary feature in the KTP pattern and Steps 2 and 3 correspond to secondary features of the KPT pattern that are imposed in order enforce the KTP detection and to avoid misclassification with other movements. At the end of this section we describe the results of the tests conducted for evaluating the KTP detection algorithm.

3.1. Protocol Definition. By definition, KTP consists in the movement of hitting the device three or more times along the face parallel to the screen in the front or back of the device - this procedure is refer herein as "knocks". During this movement the smartphone must be completely stationary (and located in the trouser front pocket or in a belt worn around the hip). If these conditions are fullfiled, a specific acceleration pattern is observed in the smartphone accelerometer response signals. An example of such pattern is depicted in Figure 7. As can be observed the three knocks, made by the user smartphone, are clearly visible in the A_z component of the acceleration (blue curve) along the z-axis, that is perpendicular to the screen plane, while the acceleration signals A_x, A_y along the x- and y-axis (green and red curves) are almost constant. We refer the reader to Figure 1 for an illustration of the smartphone reference frame. We describe the techniques used to correctly identify this KTP pattern in the next 3 subsections.

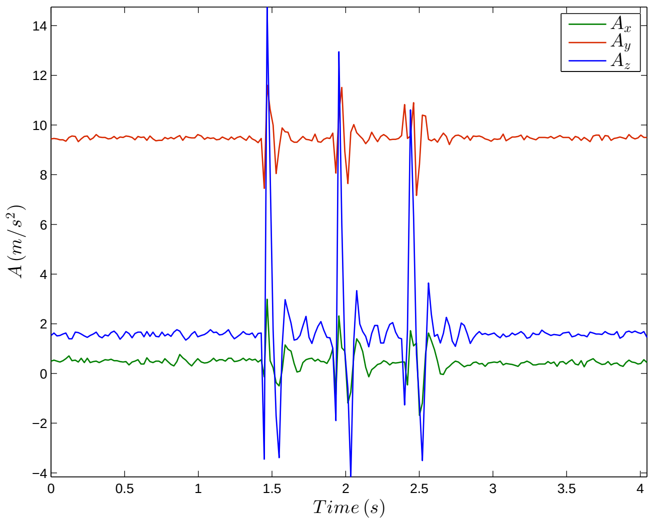

FIGURE 7. KTP pattern represented by the accelerometer data.

3.2. Step 1 - Knocks' Detection. In the following, the algorithm for KTP detection is described in detail. We start with some useful notation. We denote by $|S|$ the number of elements of a set S. As before, we define the acceleration vector $A^n = (A_x^n, A_y^n, A_z^n)$, for $n = 1, \dots, N$, for time t^n, where $t^1 = 0$ is the initial time and $t^N = T$ the final time. Let $\bar{v}_I$ denote the mean of a vector v in the time interval $I = [t^1, t^2]$, with $0 \leq t^1 < t^2 \leq T$. We define the sets I_t, I_m and I_k as

$$I_t = [t^m,\, t^m + 0.5[\, \cup\, [t^m + 0.5,\, t^m + 2.5] = I_m \cup I_k, \quad \text{for } 0 \leq t^m \leq T - 2.5. \tag{9}$$

That is, I_t denotes a time interval of size $2.5s$, which is written as the union of the time interval I_m of size $0.5s$ with the time interval I_k of size $2.0s$.

The KTP algorithm is devised to continuously analyze the KT patterns in a rolling window of $2.5s$. Moreover, since the KTP movement is by definition a quick movement (in practice the user is in panic), we then stipulate that it has to be executed in a time window of $2s$. The set I_k is the time interval where the knocks are expected to occur and I_m is the preceding short interval.

This first step of the algorithm involves the detection of the knocks, at least 3. As illustrated in Figure 7, when KTP is executed at least three large peaks appear in the A_z component of the acceleration. The highest peaks are in the positive part of the z- axis if the knocks are executed in the front of the device, otherwise if the knocks are applied in the back of the device the highest peaks appear in the negative part of the z- axis. In order to detect these peaks we proceed as follows.

- Let C_z be a critical pre-defined threshold. Firstly, we compute $\bar{A}_{z,I_m}$ (the mean value of A_z in I_m), and secondly, we look for values of A_z bigger than $\bar{A}_{z,I_m}+C_z$, or smaller than $\bar{A}_{z,I_m}-C_z$, in the time interval I_k. Thus, we define the set of time knocks candidates, T^+_{knocks} when the user hits the front of the smartphone, or T^-_{knocks} when the user hits the back of the smartphone, by

$$\begin{aligned} T^+_{knocks} &= \{t^n \in I_k : \; A^n_z \geq \bar{A}_{z,I_m} + C_z\} \\ T^-_{knocks} &= \{t^n \in I_k : \; A^n_z \leq \bar{A}_{z,I_m} - C_z\}. \end{aligned} \tag{10}$$

  These sets identify the times at which the component of the acceleration along the z- axis is above the line $\bar{A}_{z,I_m}+C_z$ for T^+_{knocks}, or below the line $\bar{A}_{z,I_m}-C_z$ for T^-_{knocks}.

- Note that a minimum time span between knocks must be observed. Thus let C_t be a pre-defined small time range. In order to eliminate the elements of T^+_{knocks} (or T^-_{knocks}) that are separated apart by less than C_t, we update them according to (assuming that the elements of these sets are already sorted in ascending order)

$$\begin{aligned} T^+_{knocks} &:= T^+_{knocks} \setminus \{t^m \in T^+_{knocks} : \; t^{m+1} - t^m \leq C_t\} \\ T^-_{knocks} &:= T^-_{knocks} \setminus \{t^m \in T^-_{knocks} : \; t^{m+1} - t^m \leq C_t\}. \end{aligned} \tag{11}$$

After this, if the number of elements in T^+_{knocks} or in T^+_{knocks} is still bigger or equal than 3, that is if the condition

$$|T^+_{knocks}| \geq 3 \quad \text{or} \quad |T^-_{knocks}| \geq 3 \tag{12}$$

is still verified, then 3 possible knocks are detected and the algorithm proceeds to the next step for checking other conditions.

The values used for the thresholds C_z and C_t, in the tests, are indicated in Table 8. Throughout this paper, whenever we refer to T^+_{knocks} we can also refer to T^-_{knocks} (depending if the knocks are executed on the front or back of the smartphone) with convenient adjustments.

3.3. Step 2 - Stability Conditions. The KTP protocol demands that the device must be completely stopped. As is shown in Figure 7, this requirement results in an almost constant acceleration, especially for the A_x and A_y components. In this second step we model this property in the interval $I_t = I_m \cup I_k$, see (9).

Let C_m be a pre-defined threshold, and T^+_k the maximum time interval spanned by T^+_{knocks} (respectively T^-_k if T^-_{knocks} is used). Then consider the following sets,

$$\begin{aligned}
S_{1,x} &= \{A^n_x : \; t^n \in I_t \;\text{ and }\; |\bar{A}_{x,I_m} - A^n_x| \leq C_m\} \\
S_{2,x} &= \{A^n_x : \; t^n \in I_t\} \\
S_{3,x} &= \{A^n_x : \; t^n \in T^+_k \;\text{ and }\; |\bar{A}_{x,I_m} - A^n_x| \leq C_m\}, \\[1em]
S_{1,y} &= \{A^n_y : \; t^n \in I_t \;\text{ and }\; |\bar{A}_{y,I_m} - A^n_y| \leq C_m\} \\
S_{2,y} &= \{A^n_y : \; t^n \in I_t\} \\
S_{3,y} &= \{A^n_y : \; t^n \in T^+_k \;\text{ and }\; |\bar{A}_{y,I_m} - A^n_y| \leq C_m\}, \\[1em]
S_{1,z} &= \{A^n_z : \; t^n \in I_t \;\text{ and }\; |\bar{A}_{z,I_m} - A^n_z| \leq C_m\} \\
S_{2,z} &= \{A^n_z : \; t^n \in I_t\} \\
S_{3,z} &= \{A^n_z : \; t^n \in T^+_k \;\text{ and }\; |\bar{A}_{z,I_m} - A^n_z| \leq C_m\},
\end{aligned} \tag{13}$$

where $\bar{A}_{x,I_m}$, $\bar{A}_{y,I_m}$ and $\bar{A}_{z,I_m}$ are the mean values of A_x , A_y and A_z in I_m (the "normal" time interval that preceds the interval I_k where knocks might exist), respectively. We note that $S_{1,\cdot}$ represents the values of the acceleration component that are contained in a strip of thickness $2C_m$ and centered at the mean value of that acceleration component in I_m. $S_{2,\cdot}$ is the set values of the acceleration component in all the interval I_t. Finally $S_{3,\cdot}$ represents the values of the acceleration component, for the time interval I_k where the possible knocks were detected, that are contained in a strip of thickness $2C_m$ and centered at the mean value of that acceleration component in I_m.

Let $C_{1,x}$, $C_{1,y}$, $C_{1,z}$, $C_{3,x}$, $C_{3,y}$, $C_{3,z}$ be positive pre-defined thresholds smaller or equal to one, herein understood as percentages. Then, we define the two following stability conditions

$$|S_{1,x}| \geq C_{1,x}|S_{2,x}|, \quad |S_{1,y}| \geq C_{1,y}|S_{2,y}|, \quad |S_{1,z}| \geq C_{1,z}|S_{2,z}|, \tag{14}$$

$$|S_{3,x}| \geq C_{3,x}|S_{2,x}|, \quad |S_{3,y}| \geq C_{3,y}|S_{2,y}|, \quad |S_{3,z}| \geq C_{3,z}|S_{2,z}|. \tag{15}$$

The condition (14) states that a certain percentage of values of A_x, A_y, and A_z in the interval I_t must be located around the respective mean calculated in the interval I_m. Condition (15) has the same meaning, but it is applied only in the interval T_k^+, i.e., in the interval where the possible knocks were detected.

The values used, in the tests, for all these thresholds are indicated in Table 8. We remark that the percentages $C_{1,z}$ and $C_{3,z}$ should be always much smaller than $C_{1,x}$ and $C_{3,x}$ or $C_{1,y}$ and $C_{3,y}$, because the stability conditions are mainly satisfied by the A_x and A_y components of the acceleration (as can be seen in Figure 7). In addition, $C_{3,z}$ should be smaller than $C_{1,z}$, because $C_{3,z}$ is associated with the interval I_k where the knocks might occur while $C_{1,z}$ is linked to the entire interval I_t.

3.4. Step 3 - Oscillatory Condition. Note that while the KTP peaks are clearly more pronounced in the A_z component, they can also sometimes be observed in the A_x and A_y components (see Figure 8). This can happen, for instance, if some movement occurs in those directions during the KTP action. Nevertheless, the peaks, which are visible in Figures 7 and 8, are characterized by a quick rise and an equally rapid decay. In order to capture this behavior, let C_o be another predefined threshold and we introduce the following sets

$$\begin{aligned} S_{4,x} &= \{A_x^n : \ t^n \in I_t \ \text{ and } \ |A_x^n - \bar{A}_{x,I_m}| \geq C_o\} \\ S_{4,y} &= \{A_y^n : \ t^n \in I_t \ \text{ and } \ |A_y^n - \bar{A}_{y,I_m}| \geq C_o)\} \\ S_{4,z} &= \{A_z^n : \ t^n \in I_t \ \text{ and } \ |A_z^n - \bar{A}_{z,I_m}| \geq C_o\}, \end{aligned} \tag{16}$$

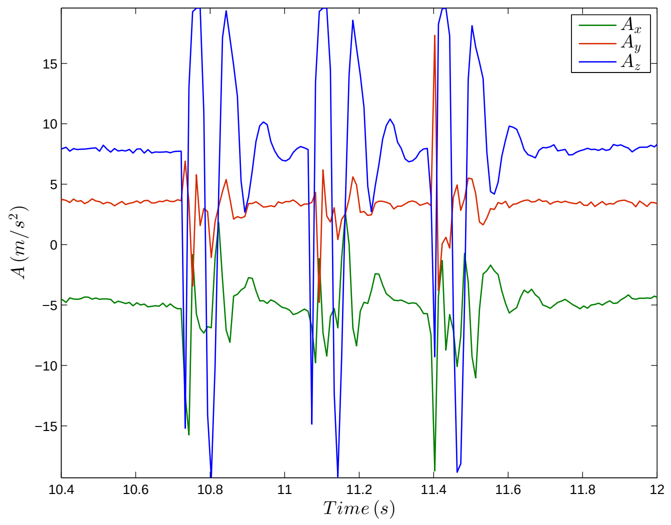

FIGURE 8. KTP pattern showing also some oscillations in the A_x and A_y components.

representing the components A_x, A_y and A_z of the acceleration that are outside the strip, with thickness $2C_o$, and centered respectively at $\bar{A}_{x,I_m}$, $\bar{A}_{y,I_m}$ and $\bar{A}_{z,I_m}$ (those are the acceleration averages in the "normal" time interval I_m, that preceds the interval I_k where knocks might exist). Then, we denote by $seq_{4,x}$, $seq_{4,y}$ and $seq_{4,z}$ the number of elements, of the longest sequences of consecutive elements in $S_{4,x}$, $S_{4,y}$ and $S_{4,z}$, respectively. The oscillatory condition associated with (16) is then defined as follows

$$seq_{4,x} \leq C_{4,x}|S_{2,x}|, \quad seq_{4,y} \leq C_{4,y}|S_{2,y}| \quad \text{and} \quad seq_{4,z} \leq C_{4,z}|S_{2,z}|, \qquad (17)$$

where $C_{4,x}$, $C_{4,y}$ and $C_{4,z}$ are 3 pre-defined thresholds, that represent percentages. This condition (17) specifies bounds for the number of consecutive acceleration points that are outside the aforementioned strips. Cleary we should allow a large percentage for the acceleration component along the z-axis. The values for the thresholds C_o, $C_{4,x}$, $C_{4,y}$ and $C_{4,z}$ used in the tests are displayed in Table 8.

3.5. KTP Algorithm. We summarize now the KTP threshold-based algorithm.

1. Set the thresholds C_z, C_t for koncks' detection, C_m, $C_{1,x}$, $C_{1,y}$, $C_{1,z}$, $C_{3,x}$, $C_{3,y}$, $C_{3,z}$ for the stability conditions and C_o, $C_{4,x}$, $C_{4,y}$, $C_{4,z}$ for the oscillatory conditions.

2. Read the data given by the acceleration sensor in a rolling window of size $2.5s$.

3. Detect the possible knocks by checking condition (12). If this condition is verified proceed to the next step, otherwise no KTP is detected.

4. Check the stability conditions (14) and (15). If they are verified proceed to the next step, otherwise no KTP is detected.

5. Check the oscillatory conditions (17). If it is verified a KTP is detected and an emergency signal is sent to the monitoring system, otherwise the algorithm ends without any KTP detection.

3.6. Numerical Experiments for KTP. To test the efficiency of the proposed algorithm, a data set with 326 movements was collected by the company *Oncaring* (http://oncaring.com/) by a group of 5 different users. In total 117 KTP and 209 ADL (like walking, running, jumping, and sitting), were recorded. Specific movements that could generate FP, such as "using the phone", were also monitored. The smartphones used were the same described in subsection 3.1. From the total data samples, 19 KTP and 73 ADL were obtained with the Samsung Galaxy Nexus S. For all the experiments the smartphone was placed in the trouser front pocket or in a belt worn around the hip. Moreover, KTP movements were recorded with the user in different positions, *e.g.*, lying on the floor, seated on a chair or standing-up.

For defining the 13 thresholds C_z, C_t, C_m, $C_{1,x}$, $C_{1,y}$, $C_{1,z}$, $C_{3,x}$, $C_{3,y}$, $C_{3,z}$, C_o, $C_{4,x}$, $C_{4,y}$, $C_{4,z}$, we have applied the same strategy described in Section 2.3 for the fall detection, by defining a training subset of the whole dataset and setting these thresholds such that a desired level of specificity (about 99%) of specificity is reached. We note that, in contrast to the fall detection algorithm where we set the thresholds in the training data set to reach 100% of sensitivity, for the KTP algorithm the goal is to reach about 100% of specificity. In fact, for the KTP detection algorithm it is essential to have no false alarms, since the user can repeat the KTP movement several times if the alarm is not activated the first time the user hits the device for doing a KTP movement. A list of the thresholds used is given in Table 8.

In Table 9 we present the results of our algorithm. As this table shows, the algorithm has a high sensitivity (91.45%) and even higher specificity

Threshold	Value	Step
C_z	7	Knocks' detection
C_t	0.15	
C_m	2	Stability Conditions
$C_{1,x}$	0.7	
$C_{1,y}$	0.7	
$C_{1,z}$	0.4	
$C_{3,x}$	0.4	
$C_{3,y}$	0.4	
$C_{3,z}$	0.1	
C_o	4	Oscillatory Condition
$C_{4,x}$	0.03	
$C_{4,y}$	0.03	
$C_{4,z}$	0.04	

Table 8. Thresholds adopted for KTP detection.

(96.19%). We also point out that all the FP were due to the action of using the phone, *e.g.*, writing SMS. This type of FP can be very difficult to identify. One possible solution is to activate the algorithm only when the smartphone screen is locked. In the FN group, 2 were consequence of malfunction of the smartphone. In the other 8 cases the algorithm stopped in Step 3, since the condition (12) ($|T^+_{knocks}| \geq 3$ or $|T^-_{knocks}| \geq 3$) was not satisfied. This was due to the fact that the threshold C_z was to high. Unfortunately, we were unable to eliminate more FN without significantly increasing the number of FP. For instance, taking $C_z = 5$ we only increase the sensitivity to 92.31% while the specificity reduces to 83.73%.

Sensitivity	Specificity	TP	FP	TN	FN
91.45%	96.19%	107	8	202	10

Table 9. Sensitivity and specificity results for the algorithm KTP.

Remark 3.1. *Finally, we briefly explain some simplifications we have carried out in the KTP algorithm, described in Section 3.5, in order to reduce* *the computational execution, and thus for decreasing the smartphone battery consumption:*

- *In step 2 we have included the following initial extra condition, in the rolling window of size* 2.5*s*

$$|\max_n A_z^n - \min_n A_z^n| \geq C_z$$

  This extra condition acts as a prediction of condition (10)*. If it is verified, then it is likely that condition* (12) *will be also verified, so the KTP algorithm proceeds with steps 3, 4 and 5. Otherwise it is considered that there is no KTP movement and no computations are performed in this window.*

- *We have also simplified conditions (14-15), by imposing them only for the A_z component, so only the conditions involving $S_{1,z}$, $S_{2,z}$ and $S_{3,z}$ are used.*

- *We have simplified condition (17), by imposing it again only for the A_z component.*

Clearly these simplifications have improved the battery life-time, but the experimental results show the performance is a little bit worse with respect to the specificity as shown in the Table 10 (to compare with Table 9)

Sensitivity	*Specificity*	*TP*	*FP*	*TN*	*FN*
91.45%	*93.81%*	*107*	*13*	*197*	*10*

Table 10. Sensitivity and specificity results for the simplified KTP algorithm.

3.6.1. *KTP (smartphone placed over a hard surface).* Now we analyze a very specific scenario, namely when the device is placed on a hard surface such as a table, and the user hits the device at least 3 times. This particular situation presents some challenges since the accelerometer signal can be very different from the KTP pattern that is produced when the device is located in the trouser front pocket or in a belt worn around the hip. Moreover, we also observed that the shape of the signal can be very different from one device to another. These issues are well illustrated in Figures 9 and 10.

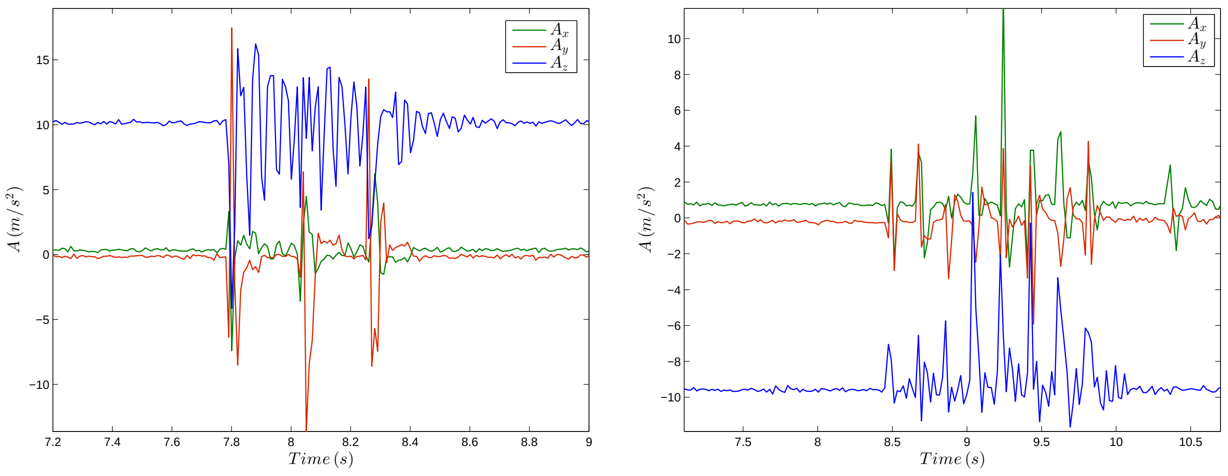

FIGURE 9. Examples of table KTP patterns represented by the accelerometer response for the smartphone Samsung Galaxy Nexus.

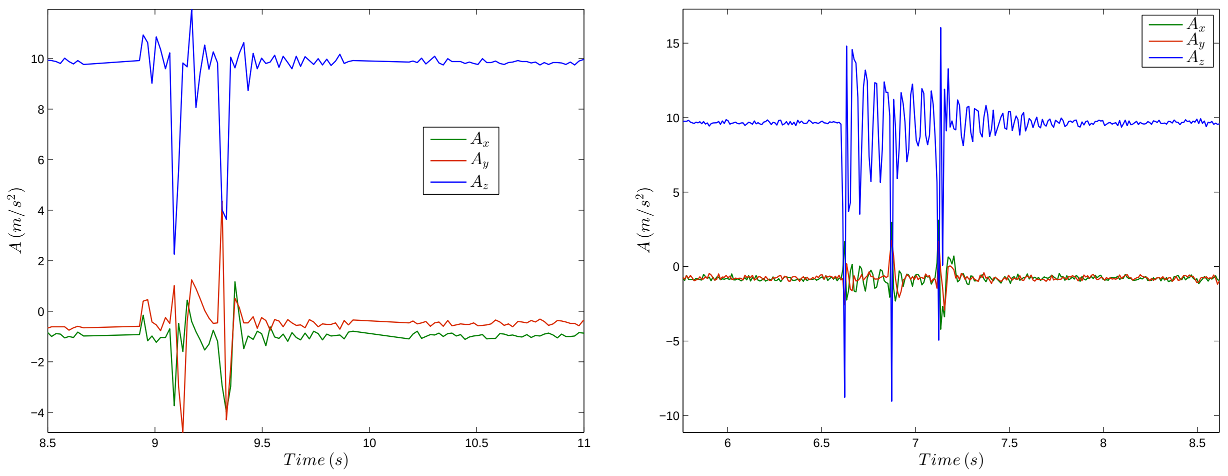

FIGURE 10. Examples of table KTP patterns represented by the accelerometer response for the smartphone Samsung Galaxy Nexus S.

Nevertheless, we have applied the KTP algorithm to a data-set of 43 KTP movements, when the device is placed over a table, herein called "table KTP" movements. The results are given in Table 11. As can be seen, approximately 65% of table KTP movements were detected by the KTP algorithm described in Section 3.5. To improve these results, we have developed an extra module in the KTP algorithm, the “table module”, that models the particular table KTP pattern.

	TP	FN
KTP Algorithm	28	15
KTP Algorithm + Table module	41	2

Table 11. Results for KTP and extended KTP algorithms in the dataset with 43 table KTP.

The structure of this additional module includes the KTP algorithm, as described in Section 3.5, but the following changes:

- There is an additional stability condition, defined by

$$\begin{aligned} \max S_x^* \leq C_{t,x}, &\quad \max S_y^* \leq C_{t,y}, \\ \max S_z^* \leq C_{t,z}, &\quad \min S_z^* \leq C_{t,z}^1. \end{aligned} \tag{18}$$

  where

$$S_x^* = \{A_x^n : \; t^n \in I_m\}, \quad S_y^* = \{A_y^n : \; t^n \in I_m\}, \quad S_z^* = \{A_z^n : \; t^n \in I_m\},$$

  and max and min denote the maximum and minimum absolute values, respectively, of the acceleration components in the correspondent sets.

- The threshold values are shown in Table 12, and they are different from those used in the KTP algorithm (displayed in Table 8)

Note that condition (18) allows to identify when the device is stationary on a flat surface, in the time interval I_m that precedes the time interval I_k where the table knocks might occur. In fact, since the smartphone is placed over the table, the first two conditions state that the acceleration components A_x and A_y along the x- and y- axes are close to zero, whereas the second conditions force A_z, the acceleration component along the z- axis to be very close to the standard value of the gravitational acceleration (this is the convention for Android smartphones).

The results with this new table module are also presented in Table 11, and they reveal that 95% of the table KTP movements are now detected with the extended KTP algorithm.

We have also tested the KTP algorithm with the table module, in the full data-set containing 160 KTP (117+43), referred in Section 3.6 obtained after

Threshold	Value	Step
C_z	2	Knocks' detection
C_t	0.04	
C_m	2	Stability Conditions
$C_{1,x}$	0.9	
$C_{1,y}$	0.9	
$C_{1,z}$	0.7	
$C_{3,x}$	0.8	
$C_{3,y}$	0.8	
$C_{3,z}$	0.2	
$C_{t,x}$	1.5	
$C_{t,y}$	1.5	
$C_{t,z}$	11	
$C^1_{t,z}$	9	
C_o	4	Oscillatory Condition
$C_{4,x}$	0.02	
$C_{4,y}$	0.02	
$C_{4,z}$	0.03	

TABLE 12. Thresholds adopted for the new module table KTP.

adding the 43 table KTP. In Table 13 we present the results. In comparison with Table 9 there are no false positives generated by the extra module, so the specificity is kept.

Sensitivity	Specificity	TP	FP	TN	FN
93.75%	96.19%	150	8	202	10

TABLE 13. Sensitivity and specificity results for the KTP algorithm + Table module in the full data-set with a total of 160 KTP (43 table KTP) and 210 ADL.

We conclude this section with an important observation. The KTP protocol may not be respected by the table module, as the algorithm may detect KTP movements with less than the three predefined knocks. This problem arises because of the acceleration signal itself (which is related to the hardware) and because the thresholds C_t and C_z indicated in Table 12 are too small. So further studies on these issues should be done, namely the dependence of threshold values on the characteristics of the device.

Moreover we emphasize that the table module could be an independent application to be used in different contexts.

4. Conclusion

In this paper, we have analised smartphone sensors to determined their reliability to discriminate between falls and ADL. Based on our results, the accelerometer appears to be the most reliable sensor. Using the information provided by this sensor a novel algorithm was proposed and tested. The algorithm Alg_1 is simple and can easily be implemented in smartphones platforms. In our data-set, its performance leads to 100% sensitivity and 92.65% specificity. The SVM analysis confirms the good performance of the proposed methodology. These results are promising, but more experiments must be done and other issues should be explored, *e.g.*, the influence of individual physical factors and smartphone location in the body should be further investigated. Furthermore, we also need to address the false positive results generated by the ADL "sitting abruptly" movement. Another, aspect that needs further study is the difference in acceleration signals from simulated and real falls. The study presented in [2] suggests that there are important differences, and the performance of fall detection algorithms with real data is seriously affected. In cooperation with our industry partner we are now testing the algorithm in real world environment. This will allow us to collect valuable data to explore these issues in future research.

We have also described in detail a KTP detection algorithm. Some numerical results for evaluating the performance of the algorithm were also given. Currently, this algorithm is also being tested in realistic situations (in senior care and senior monitor services), and so far the results confirm the reported experimental findings. Both sensitivity and specificity are very high, reaching values of 90%. Moreover, a very good level of acceptance and approval has been observed among both elderly and non-elderly users (*e.g.* caregivers). We believe that KTP has a huge potential, and if these preliminary results are confirmed, this inventive application may become a widespread emergency detection tool in the future. In addition we remark that the intensity of the knocks depend on the strength applied by the user. Young and elderly people clearly might generate knocks with different intensities. Thus adaptive thresholds should be considered and implemented in the future.

To finish this conclusion, we notice that no correlation was observed between the two smartphones and any specific type of fall, ADL or KTP. This fact suggests that the results are unbiased with respect to differences in frequency and hardware. However, the number of samples obtained with the lower frequency Samsung Galaxy Nexus S is considerably smaller, and therefore the robustness of the algorithms relative to different types of devices and frequencies needs additional research. This topic is currently under study.

Conflicts of interest statement: Isabel N. Figueiredo, Carlos Leal and Luís Pinto work at the University of Coimbra, Portugal and have no conflict of interest to declare in relation to the presented work. Jason Bolito and André Lemos work at *OnCaring* (http://oncaring.com/), a company for *Senior Care solutions from the Critical Group*. *OnCaring* has no commercial relationship with the remaining authors.

References

[1] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini, and A. Vecchio. A smartphone-based fall detection system. *Pervasive and Mobile Computing*, 8:883–899, 2012.

[2] F. Bagala, C. Becker, A. Cappello, L. Chiari, K. Aminian, J.M. Hausdorff, W. Zijlstra, and J. Klenk. Evaluation of accelerometer-based fall detection algorithms on real-world falls. *PLoS ONE*, 7:e37062, 2012.

[3] A.K. Bourke and G.M. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical Engineering and Physics*, 30:84–90, 2008.

[4] A.K. Bourke, K.J. O'Donovan, and G. ÓLaighin. The identification of vertical velocity profiles using an inertial sensor to investigate pre-impact detection of falls. *Medical Engineering and Physics*, 30:937–946, 2008.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011.

[6] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. In *Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005.

[7] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan. Mobile phone-based pervasive fall detection. *Personal and Ubiquitous Computing*, 14:633–643, 2010.

[8] R.J. Gurley, N. Lum, M. Sande, B. Lo, and M.H. Katz. Persons found in their homes helpless or dead. *The New England Journal of Medicine*, 334:1710–1716, 1996.

[9] M.A. Habib, M.S. Mohktar, S.B. Kamaruzzaman, K.S. Lim, T.M. Pin, and F. Ibrahim. Smartphone-based solutions for fall detection and prevention: Challenges and open issues. *Sensors*, 14:7187–7208, 2014.

[10] R. Igual, C. Medrano, and I. Plaza. Challenges, issues and trends in fall detection systems. *Biomedical Engineering Online*, 12, 2013.

[11] M. Kangas, A. Konttila P. Lindgren, I. Winblad, and T. Jämsä. Comparison of low-complexity fall detection algorithms for body attached accelerometers. *Gait and Posture*, 28:285–291, 2008.
[12] H. Ketbdar and T. Polzehl. Fall and emergency detection with mobile phones. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, 2009.
[13] A. Leone, G. Diraco, and P. Siciliano. Detecting falls with 3d range camera in ambient assisted living applications: A preliminary study. *Medical Engineering and Physics*, 33:770–781, 2011.
[14] Department of Economic and Social Affairs Population Division. *World Population Ageing 2013*. United Nations, New York, 2013.
[15] F. Sposaro and G. Tyson. ifall: An android application for fall monitoring and response. In *31st Annual International Conference of the IEEE EMBS*, 2009.
[16] J.A. Stevens, P.S. Corso, E.A. Finkelstein, and T.R. Miller. The costs of fatal and non-fatal falls among older adults. *Injury Prevention*, 12:290–295, 2006.
[17] A.M. Tromp, S.M.F. Pluijm, J.H. Smit, D.J.H. Deeg, L.M. Bouter, and P. Lips. Fall-risk screening test: A prospective study on predictors for falls in community-dwelling elderly. *Journal of Clinical Epidemiology*, 54:837–844, 2001.
[18] D. Wild, U.S. Nayak, and B. Isaacs. How dangerous are falls in old people at home? *British Medical Journal*, 282:266–268, 1981.
[19] G. Wu. Distinguishing fall activities from normal activities by velocity characteristics. *Journal of Biomechanics*, 33:1497–1500, 2000.
[20] Z. Zhao, Y. Chen, S. Wang, and Z. Chen. Fallalarm: Smart phone based fall detecting and positioning system. *Procedia Computer Science*, 10:617–624, 2012.

I. N. Figueiredo
CMUC, Department of Mathematics, University of Coimbra, Portugal.
E-mail address: isabelf@mat.uc.pt
URL: http://www.mat.uc.pt/∼isabelf/

C. Leal
CMUC, Department of Mathematics, University of Coimbra, Portugal.
E-mail address: carlosl@mat.uc.pt

L. Pinto
CMUC, Department of Mathematics, University of Coimbra, Portugal.
E-mail address: luisp@mat.uc.pt

J. Bolito
Oncaring, Coimbra, Portugal.
E-mail address: alemos@oncaring.com
URL: http://oncaring.com/

A. Lemos
Oncaring, Coimbra, Portugal.
E-mail address: alemos@oncaring.com
URL: http://oncaring.com/