Noname manuscript No.
(will be inserted by the editor)

Improving the Performance of Fall Detection Systems through Walk Recognition

Guglielmo Cola · Alessio Vecchio · Marco Avvenuti

the date of receipt and acceptance should be inserted later

Abstract Social problems associated with falls of elderly citizens are becoming increasingly important because of the continuous growth of aging population. Automatic fall detection systems represent a possible answer to some of these problems, as they are useful to obtain help in case of serious injuries and to reduce the long-lie problem. Nevertheless, widespread adoption of these systems is strongly influenced by their usability and trustworthiness, which are at the moment not excellent. In fact, the user is forced to wear the device according to placement and orientation restrictions that depend on the considered fall-recognition technique. Also, the number of false alarms generated is too high to be acceptable in real world scenarios. This paper presents a technique, based on walk recognition, that increases significantly both usability and trustworthiness of a smartphone-based fall detection system. In particular, the proposed technique automatically and dynamically determines the orientation of the device, thus relieving the user from the burden of wearing the device with predefined orientation. Orientation is then used to infer posture and eliminate a large fraction of false alarms ($\sim 98\%$).

Keywords Pervasive Healthcare · Activity Recognition · Wearable Sensors · Walk Recognition

1 Introduction

Falls are a major problem for elderly people and fall-related injuries are one of the most common causes for hospital admission or death. The long-lie problem is frequently associated with falls: elderly people may remain on the ground for a long period because they are shocked, injured, or too weak to get to their feet (Wild et al. 1981; Tinetti et al. 1993; Gurley et al. 1996). The problem of long-lie can be reduced through the use of a personal emergency response system, a small device equipped with a "help" button that can be carried or worn by the user. Unfortunately, in many circumstances, one may not be able to press the button, e.g. because of a loss of consciousness or as the result of severe injuries. A solution to this problem is represented by automatic fall detection systems: after a fall the system, without human intervention, sends an alarm message to the caregiver or to the patient's relatives. From the technical and research points of view, the most challenging part of the process is recognizing a fall, as it is an ill-defined process and it is difficult to characterize.

Some fall detection systems are based on the idea of instrumenting, with sensing devices, the environment where the patients live. Solutions include tracking of patients' movements with a camera (Anderson et al. 2006), infrared sensors placed in proximity of beds (Sixsmith and Johnson 2004), floor mats equipped with pressure sensors, or vibration and acoustic sensors (Zigel et al. 2009). Nevertheless, instrumenting the environment requires significant set up costs and poses some privacy concerns. Other techniques, on the contrary, are based on the idea of sensing the patients' movements through one or more sensors (accelerometers and/or gyroscopes) attached to the users' body. The number of required devices is a critical factor for obtaining a reasonable system usability, thus in the following we focus only on those solutions where monitoring is carried out by means of a single sensing device. In particular, we concentrate

G. Cola · A. Vecchio · M. Avvenuti
Dip. di Ingegneria dell'Informazione, University of Pisa, Italy
E-mail: firstname.lastname@iet.unipi.it

on methods based on accelerometric information, since it proved to be more useful with respect to angular velocity for detecting falls (Lindemann et al. 2005). Other critical factors that influence the usability and the acceptability of fall detection systems are their sensitivity and specificity: the former is the capacity of a system in detecting all falls, whereas the latter is its ability in detecting only real falls (filtering all fall-like impacts caused by activities of daily living, such as sitting on a chair).

In some previous work, information concerning the orientation of the sensing device is used to infer user's posture and therefore to reduce the number of false alarms (Karantonis et al. 2006; Kangas et al. 2008; Bourke et al. 2010). The basic assumption behind the techniques based on postural analysis is that the user is lying after a fall: a possible fall is confirmed only if the user's body is horizontal after an impact. In fact, posture recognition proved to be of paramount importance for the reduction of false alarms in fall detection systems. Unfortunately, the recognition of lying posture using a single accelerometer poses two requirements that significantly reduce system usability: *i*) one of the reference axes of the device must be aligned with the longitudinal axis of the user's body *ii*) the device must be integral with the user's body. Consider, for example, the use of a smartphone placed into a pocket: the alignment between the device and the longitudinal body axis can not be assured. Thus, in order to correctly apply postural recognition, the user would be forced to perform a calibration phase each time he/she changes the orientation of the device (for example, when extracting and reinserting the phone from/into the pocket). In summary, calibration, to the purpose of fall detection, consists in virtually aligning one of the device's axes with the longitudinal axis of the user's body (Avvenuti et al. 2013; Gietzelt et al. 2012).

In this paper we propose a technique that enables the detection of lying posture without affecting system usability. Users are allowed to wear the device without paying attention to its orientation and without a manual setup phase. This is achieved through dynamic and automatic calibration: the direction of the longitudinal axis of the user's body, in the coordinate system of the device, is automatically detected taking advantage of walk recognition. A specific walk recognition algorithm was designed and tested for this purpose. We then evaluated the benefits introduced by the use of posture detection: experimental results show that such information can reduce the number of generated false alarms by $\sim 98\%$, increasing significantly the trustworthiness of the fall detection process. A corollary contribution of this work is a comparative evaluation of the filtering effect of posture with respect to other information commonly used to distinguish real falls from false alarms. Results show that posture detection provides the greatest benefits, and highlight the importance of using such information in future research on fall detection.

The remaining of this paper is organized as follows. In Section 2 we describe the state of the art regarding user's posture in fall detection systems, walk recognition, and smartphone-based fall detection. Section 3 presents the principle of operation of our approach: posture information can be obtained through walk recognition and then incorporated in a fall detection system. The experimental settings and the data acquisition campaign are described in Section 4. In Section 5 we present the algorithm for the detection of walk segments. Then, in Section 6 we show how the posture information obtained from the walk segments increase the specificity of fall detection. Finally, we present our conclusions in Section 7.

2 State of the art

Here we recall the most significant work on: *i*) the use of posture information in fall detection systems; *ii*) recognition of walk segments in similar contexts; *iii*) smartphones as a platform for the detection of falls. Then, the major contributions of our work with respect to previous literature are highlighted.

2.1 Use of posture information in fall detection systems

One of the first papers describing the use of an unobtrusive and smart device for the classification of human movements and the detection of falls is Karantonis et al. (2006). The device, equipped with a tri-axial accelerometer, is firmly attached at the user's waist and aligned with the longitudinal axis of the human body. Thus, the system is able to determine the posture of the user by measuring the angle between the axis of the device aligned with the user's body and gravity. The tilt angle is then compared with fixed thresholds to discriminate between standing, sitting, and lying postures. A possible fall, detected by means of a threshold on the acceleration magnitude, is upgraded to a fall only if: no significant activity is recorded for at least $60s$; the user is in the lying posture.

A comparison between low-complexity fall detection algorithms for wearable accelerometers is presented in Kangas et al. (2008). The three algorithms under evaluation are based on the following features: impact + posture; start of fall + impact + posture; and start of

fall + velocity + impact + posture. Posture information is calculated similarly to Karantonis et al. (2006). Besides the performance of the three algorithms, it is important to notice that posture information has been considered as fundamental and its analysis has been always included.

A similar study is described in Bourke et al. (2010), where a number of fall detection algorithms have been compared by measuring their performance against a rather large dataset. The experimental results showed that an algorithm that uses velocity, impact, and posture information can obtain a low false alarm rate (less than 1 per day) still having high sensitivity. Also in this case, it is required to firmly attach the device to the user's body (using a standard belt and a modified commercial mobile-phone carry case).

Two other works requiring a predefined orientation of the device are Estudillo-Valderrama et al. (2009) and Tolkiehn et al. (2011). In Estudillo-Valderrama et al. (2009) a distributed fall detection architecture is presented; the adopted algorithm is the one described in Estudillo-Valderrama et al. (2008), and lying posture is detected similarly to Karantonis et al. (2006). In Tolkiehn et al. (2011), tilt variations are used to detect falls and fall directions. In this system, a barometric pressure sensor is combined with the accelerometer to slightly improve detection accuracy.

In Gjoreski et al. (2011), further evidence about the importance of posture as a method for increasing accuracy of fall detection is provided: about 20% accuracy improvement can be obtained. Nevertheless, also in such work, detection of posture relies upon predefined placement of accelerometers to the user's body. Moreover, an individual calibration phase is required to compensate for the slightly different ways people wear the device.

All of the above described systems confirm the importance of posture information in fall detection systems. Nevertheless, the user is forced to wear the device according to a predefined orientation. The use of posture detection when the device orientation is unknown is addressed in Curone et al. (2010). This system aimed at the context of worker's surveillance and relied on a fundamental assumption: the user is upright while dressing the device. This assumption cannot be applied in our reference scenario.

2.2 Walk recognition

Recognition of human activities by using the accelerometer that is embedded in commonly available smartphones is described in Kwapisz et al. (2011). The authors evaluated different classification systems (J48, logistic regression, and neural network) in recognizing six different activities, including walking, on a set of 29 users carrying a smartphone in theirs pants front leg pocket. As far as walking is concerned, all the three classifiers obtained accuracy values of approximately 90%.

Other work showed that it is possible to obtain effective human activity recognition also when the position of the device is not known a priori (Xu et al. 2012). In particular such work showed that, through sparse signal representation, activities such as making a step can be reasonably recognized and, at the same time, the position of the device can be estimated (out of 14 possible activities and 7 possible locations). In this case, movement information (accelerometer and gyroscope signals) is collected using wireless sensor nodes (TelosB motes) and not commonly available smartphones.

Recognition of walking activity and its use for inferring some properties of the device has been discussed also in Kunze et al. (2005), where the authors describe a technique to automatically recognize the part of the body where the sensing device is located (wrist, head, trouser pocket, breast pocket). The technique operates in two stages: first it detects the time segments where the user is walking, then a classifier is used in such regions to select the most probable location of the device. The good classification results and the fact that walking is the most common human activity advocate the use of walking as a source of useful information for inferring device properties.

2.3 Smartphone-based fall-detection systems

Detection of falls by using the patient's mobile phone is obviously an attractive idea, as it would not force the user to carry an additional device. Moreover, smartphones already include all the communication functionality needed for sending alert messages to the caregivers, and are nowadays provided with enough computing power to support the use of non trivial signal analysis methods (Sposaro and Tyson 2009; Yavuz et al. 2010).

In Abbate et al. (2012) a smartphone-based fall detection system is presented. The system can acquire kinetic information using both the smartphone internal accelerometer or an external sensing unit. In both cases, the sensing device is attached to the user's belt. The implemented technique is able to recognize some fall-like activities, like sitting on a chair or lying on a bed, so that they are not confused with real falls and thus reducing the number of false alarms. The system shows excellent performance in terms of accuracy and

the user is not forced to wear the device according to a predefined orientation, since the detection algorithm uses only the magnitude of acceleration. Nevertheless, the device cannot be placed in the pockets of the user's trousers, as it would be subject to spurious movements.

PerFallD is another smartphone-based fall detection system (Dai et al. 2010). PerFallD can operate in two modes: using only the smartphone's accelerometer or using also an additional element that must be carried by the user attached on his thigh. This additional element is made of magnetic material and causes peculiar variations on the magnetic field that are detected by the smartphone's compass. However, while this element may increase the performance of the system, its use is also detrimental in terms of usability.

The problem of fall classification by machine learning using mobile phones is studied in Albert et al. (2012). The authors evaluate the performance, in terms of sensitivity and specificity, of five machine learning classifiers using data collected through a smartphone. The device was attached to the users' body through a belt and it was placed in a standard position so that the direction of the three axes was known. The evaluation has been carried out using a rather large set of acceleration features, avoiding a manual selection of the most relevant ones and relying on the machine learning classifiers.

All the previous systems are characterized by sensitivity and specificity values that range from good to excellent. Nevertheless, they all force the user to attach the device to the user's body in a rather unnatural way: it cannot be carried in one of the user's pockets, they all require to fix the device on his/her belt. Moreover, either the orientation of the device is fixed and known (placing an additional burden on the user) or the techniques cannot use the distinct acceleration values available on the three axes. It is clear that the performance of fall detection systems could only get better if the techniques proposed so far would make use of disaggregated acceleration information.

2.4 Contribution

With respect to the state-of-the-art techniques and systems, the major contributions of this work are summarized as follows.

- For the first time, the reduction of false alarms associated to the use of posture information in fall detection is evaluated and compared to other commonly used filtering criteria, such as vertical velocity or post impact activity. All these criteria can be considered as the "building blocks" of more complex fall detection techniques, thus the evaluation of their effect in increasing the specificity of fall detection is important for the design of future systems.
- The use of walk segments for passively collecting information about device orientation and user's posture in the context of fall detection is here presented and, as far as we know, it is completely novel with respect to previous work. The reduction of false alarms obtained in the real world (97 hours of monitoring) is approximately 98%.
- The adoption of these techniques in a smartphone-based fall detection system provides significant benefits for the user in terms of usability: *i*) the user is no longer forced to wear the device according to a fixed orientation; *ii*) the calibration phase is no longer needed.

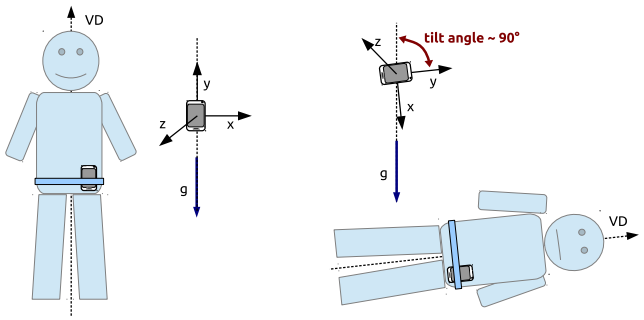


Fig. 1 Lying posture detection with aligned device. In this example, VD is aligned with the y axis of the device. The tilt angle between y and gravity is measured and compared against a threshold (e.g. 50°) in order to detect postural transitions from standing to lying

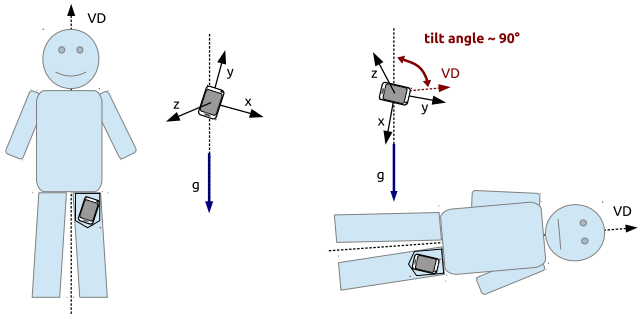


Fig. 2 Lying posture detection with misaligned device. The direction of gravity while the user is known to be upright is used to estimate the direction of VD. The tilt angle between VD and gravity is then monitored and used to detect postural transitions

3 Method

The acceleration measured by accelerometers always includes a component due to gravity, which can be extracted through low-pass filtering of raw acceleration

samples (Mizell 2003). The component due to gravity can be used to find the direction of gravity with respect to the current orientation of the device.

Let us call vertical direction (VD) the direction of the longitudinal axis of the human body. If one of the axes of the device is aligned with VD, posture can be detected as shown in Figure 1. In this example, the tilt angle between the y axis and gravity is used for posture detection.

A more realistic scenario is shown in Figure 2. The device is not aligned with the user's body and VD is unknown with respect to the coordinate system of the device. The estimation of VD is generally achieved with a calibration step, during which the user is required to stay upright for a few seconds. Indeed, gravity and VD are almost aligned when the user is upright and the direction of gravity can be used to estimate VD.

In order to automatically find VD and detect the posture without requiring a calibration step, we propose a technique based on the idea of measuring the direction of gravity while the user is walking. The reason for taking advantage of walk is threefold: while walking, the user is known to be upright; walk is a frequently occurring activity; walk can be recognized with high specificity by computer programs. Obtaining VD by means of walk recognition not only removes the necessity of wearing the device according to a predefined orientation, but also allows the user to freely reposition the device while in use (VD is automatically updated as soon as the user walks).

The posture detection method we propose can be used in a fall detection system as follows: *i*) whenever the user is walking, VD is estimated; *ii*) VD is used after an impact to understand whether the user is standing or not; *iii*) if after an impact the user is standing, then the event is discarded as a false alarm. A flowchart representation of the proposed method is shown in Figure 3. More detailed descriptions of the walk recognition and fall detection algorithm are given in Section 5 and 6, respectively. Such an approach to posture detection greatly increases the usability of the system, as it removes the need of placing and keeping the device according to a predefined alignment. Even with a binary meaning (i.e., upright/lying), posture can be used to classify a large number of impacts as non-falls, thus improving the trustworthiness of the fall detection system by reducing the number of false alarms.

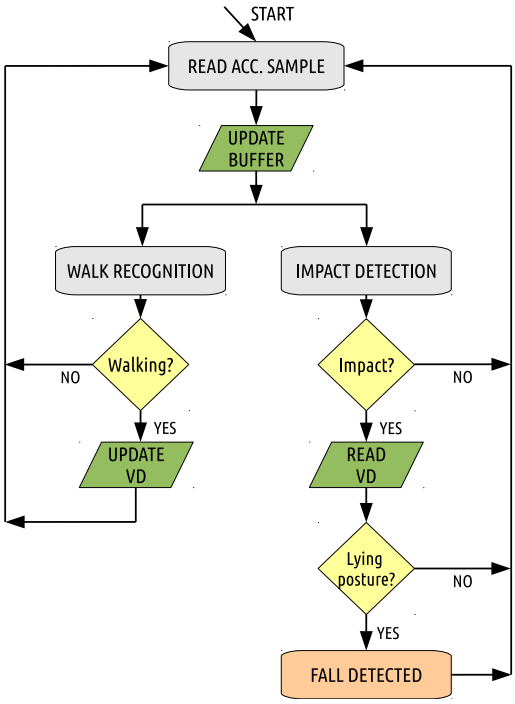


Fig. 3 Flowchart representation of the proposed method: walk recognition is executed in parallel with impact detection, in order to keep the estimation of VD updated. Whenever an impact is detected, the latest VD estimation is used to infer the user's posture and confirm a possible fall

Table 1 Volunteers' characteristics

User ID	Gender	Age	Height[cm]	Weight[kg]
1	F	26	160	55
2	F	26	166	50
3	F	34	170	60
4	F	57	166	66
5	F	61	166	77
6	M	22	168	58
7	M	26	192	80
8	M	28	175	62
9	M	40	177	81
10	M	68	175	95

4 Experimental setup and data acquisition

We carried out a data acquisition campaign to evaluate the performance of both the walk recognition algorithm and the fall detection system that we designed and implemented.

Movement traces have been acquired using a Shimmer 2 device (Realtime Technologies Inc. 2010), which is equipped with a tri-axial accelerometer. The Shimmer device has been encapsulated in a smartphone-like container to mimic the form factor of commonly available smartphones and thus to obtain acceleration traces that are consistent with those obtained in real-world settings. Besides the size, also the weight of the smartphone-like container has been calibrated to correspond to the weight of an ordinary smartphone (130 g, approximately the weight of an iPhone 4). We did not directly use a real smartphone for three main reasons: first, on smartphones, the scale of

accelerometers is often limited in the $\pm 2g$ range[1], and thus too small to capture the large variations that occur during falls; second, this enabled a fine-grained control of the sampling activities without the restrictions imposed by mobile operating systems; third, since the same device has been used to collect acceleration data also during some falls, the use of a real smartphone has been discouraged by its fragility.

Acceleration has been sampled at $51.2Hz$ and stored using the persistent memory of the device. Then, traces have been transferred onto a PC for off-line analysis and to ensure repeatable evaluation of the proposed techniques. During data acquisition, the device has always been worn in a front trouser pocket. Although this is not the only position where a smartphone can be placed, other possibilities include bags and jackets, trouser pockets are the most common placement. We preferred to defer the analysis of different locations until a later time. Moreover, it is important to notice that several walk-based techniques proved to be robust also when the device is placed in the user's jacket (Kunze et al. 2005). As far as orientation is concerned the device has been worn with no specific attention, as it is usually done with smartphones.

Ten volunteers have been involved in a collection campaign. Gender, age and physical characteristics of the volunteers are shown in Table 1. The campaign included both short walk sessions, aimed at evaluating the walk recognition and orientation procedures, and long monitoring sessions, to assess the final goal of the system, i.e. its ability to remove possible false alarms, in terms of falls, occurring during the normal activities of daily living.

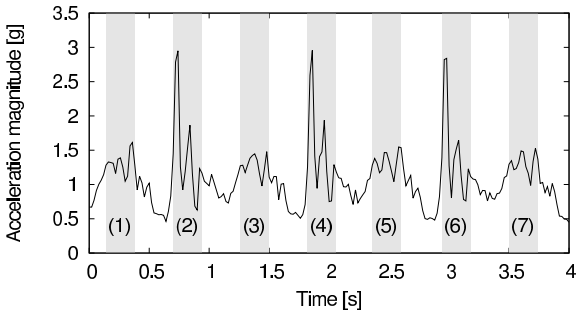


Fig. 4 Example of acceleration pattern during a short walk: the groups of peaks produced by each step have been highlighted and numbered

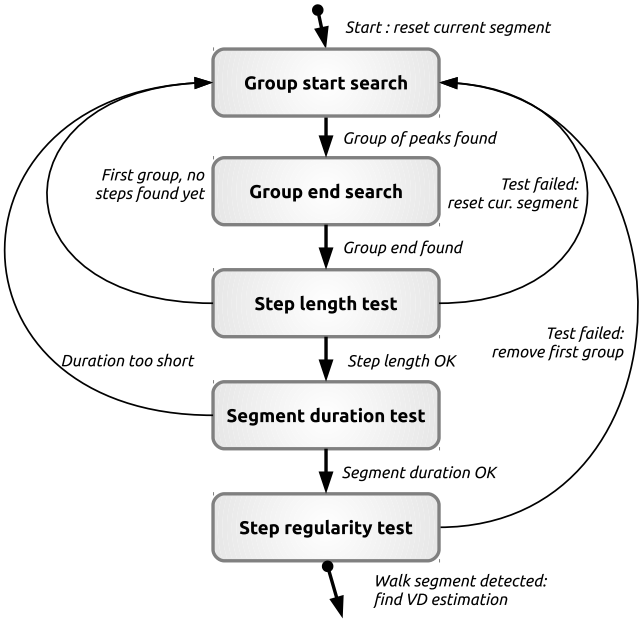


Fig. 5 Walk detection as a finite state machine

5 Real time walk recognition and estimation of device orientation

The algorithm for the recognition of walk segments has the following specific requirements: *i*) low computational load; *ii*) high specificity; *iii*) reasonable level of sensitivity.

Having a low computational load is fundamental for an application that is going to be executed on a smartphone. To reduce the computational load, our walk recognition algorithm does not operate in the frequency domain, but it is based only on temporal analysis of the samples of the acceleration magnitude (Euclidean norm). High specificity in detecting walk segments is strictly connected, in our system, with fall detection accuracy. A misdetected walk segment would lead to a wrongly estimated VD and, thus, to errors in lying posture detection. The effect of these errors on fall detection accuracy may be detrimental, since posture is used for both identifying false alarms and confirming real falls. Finally, a reasonable level of sensitivity in detecting walk segments is required in order to quickly update VD when the user changes the orientation of the device.

5.1 Description of the walk recognition algorithm

During a walk each leg goes through two fundamental states: the *stance phase*, when the foot is in contact with the ground; and the *swing phase*, when the leg swings forward and all the body weight is placed on the other leg (Lai et al. 2009).

[1] In general, the range supported by the HW is wider and this limit is imposed by OSes. Thus, we may expect to have smartphones with a fall-detection capable range in the next future, as the API and the OSes evolve.

The cyclic repetition of these states produces a typical acceleration magnitude pattern, as the one shown in Figure 4. It can be observed the presence of a group of peaks for each step made. These groups are generated at the end of every *swing* state, when the foot hits the ground. Conversely, relatively lower accelerations are produced while a leg is swinging.

Another interesting consideration concerns the different characteristics of the odd and even groups of peaks, clearly visible in Figure 4. This difference is due to the fact that the device is carried in a trouser pocket, thus the peaks produced by the leg corresponding to the side of the body where the sensor is placed are generally higher (even numbers in Figure 4).

The algorithm for the recognition of walk segments can be represented as a finite state machine (shown in Figure 5). When the machine is started, no groups of peaks have been found yet and the current possible walk segment is empty. The machine is in the *Group start search* state, where it analyzes the acceleration magnitude of samples waiting for a new group of peaks. We define an acceleration magnitude peak as a sample greater than the previous and the next samples. A new group starts when a peak greater than the `peak_intensity` threshold is detected. After that, the group start has been found and the machine moves to the *Group end search* state.

In this state, possible new peaks are searched and added to the current group of peaks. This process ends as soon as one of the following conditions occurs: *i*) no new peaks are found for a time longer than the `group_int_max` interval; *ii*) a time longer than `group_dur_max` has passed since the start of the group. When the group ends, the following information is saved and added to the current walk segment: `group_start`, corresponding to the time the first peak in the group occurred; `group_end`, corresponding to the time the last peak in the group occurred; `group_time`, calculated as the middle time between the start and the end times of the group. At this point, the machine moves to the *Step length test* state.

In our algorithm, the duration of each step is estimated using the difference between the `group_time` values of consecutive groups of peaks. In the *Step length test* state, the machine tests whether the duration of the last step lies between two thresholds: `step_dur_min` and `step_dur_max`. If the last step meets the duration requirements, the machine moves to the *Segment duration test*. Otherwise, the current walk segment is reset and the machine returns to the *Group start search* state.

In *Segment duration test*, the duration of the current walk segment is checked. This duration is calculated as the difference between the end time of the last group of peaks and the start time of the first group in the segment. If the duration of the segment is shorter than a `seg_dur_min` interval, then the machine returns to the *Group start search* state. Conversely, if the segment is long enough, the machine moves to the *Step regularity test* state.

In *Step regularity test*, two standard deviation values are calculated: `OSD` (Odd Step Durations) and `ESD` (Even Step Durations). Such values are calculated using the durations of the odd and the even steps respectively. The test is passed only if both `OSD` and `ESD` are smaller than a `step_dev_max` threshold. If the test is not passed, the first group of peaks belonging to the current possible walk segment is discarded and the machine moves back to the *Group start search*. Instead, if the regularity test is passed, the current possible walk segment is actually identified as a walk segment. Thus, it can be used to estimate VD in the coordinate system of the device. In our implementation, this estimation is done averaging the values of the acceleration samples belonging to the walk segment, considering the x, y, and z components separately. After this estimation has been calculated, all other information about the walk segment is discarded and the machine returns to the *Group start search* state.

5.2 Selection of thresholds

The minimum duration of a walk segment `seg_dur_min` has been chosen on the base of the following considerations. If the minimum duration of a segment is too short, then the estimation of VD may be highly inaccurate for at least two reasons: first, because the estimation is made on a relatively small set of samples; second, because it is more difficult to ensure the specificity of walk recognition by testing the regularity of a small number of consecutive steps. On the other hand, we expect short walks to be very frequent. This is true especially indoors where, due to the limited space, long sequences of steps are rare. According to our experimental data, a minimum duration of $6s$ represents a satisfactory trade-off between walk recognition sensitivity and the accuracy in estimating VD.

All the other thresholds have been tuned according to the following procedure. First, the `peak_intensity`, `group_dur_max`, and `group_int_max` thresholds have been found by means of exhaustive search: all the possible triplets in a reasonable search space have been used to evaluate walk recognition results on the training set (maximizing the number of detected walk segments). The triplet that provided the best results has been used to determine the remaining thresholds.

Table 2 First detection times

User	Average [s]	Worst [s]
1	6.42	6.52
2	6.44	6.95
3	6.28	6.46
4	6.24	6.39
5	6.55	7.21
6	6.41	6.62
7	6.46	6.56
8	6.33	6.41
9	6.43	7.17
10	6.31	6.56
Global	6.39	7.21

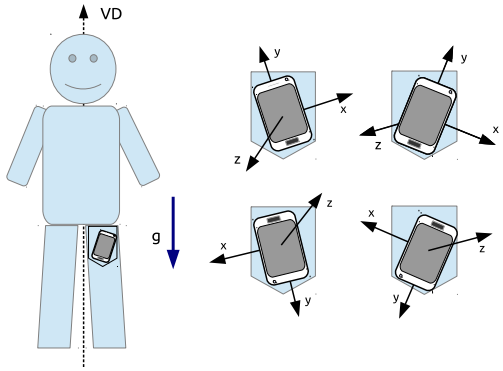


Fig. 6 Device placement examples during long monitoring experiments. While the user is standing, the movements of the device inside a pocket are expected to affect only the x and y components of VD in the coordinate system of the device

In particular, the tuning algorithm selects the longest `step_dur_min`, the shortest `step_dur_max`, and the lowest `step_dev_max` which do not lead to a reduction in the total number of walk segments detected.

5.3 Walk recognition results and discussion

A first evaluation has been carried out using our dataset of short walk tracks. To reduce the dependency of results from the training set, we used the leave-one-out cross-validation technique: let N be the number of users, the walk recognition algorithm has been tuned using the tracks of $N-1$ users and evaluated on the tracks of the remaining user; the procedure has been repeated N times.

A performance index that is particularly interesting in our case of study is represented by the *first detection time*, defined as the end time of the first walk segment found in a walk track. This index is relevant because it corresponds to the delay introduced by the system to compute the first VD estimation since the user started walking. Table 2 shows the *first detection time* obtained in the worst and average case for every user in our dataset. In the global worst case, the walk recognition algorithm was able to find a walk segment after $7.21s$. This confirms the effectiveness of the walk recognition method (note that each segment has a duration of $6s$, thus the initial transient phase that is not included in the walk segment is slightly above $1s$).

A second evaluation has been carried out using the long monitoring tracks. The walk recognition algorithm has been evaluated on each user's track, using the parameters obtained from the walk tracks of the other users. Table 3 shows the results. For each user, the duration of the long monitoring track, the number of walk segments found, the average and the maximum interval between consecutive walk segments are shown. The last row of the table shows the global results.

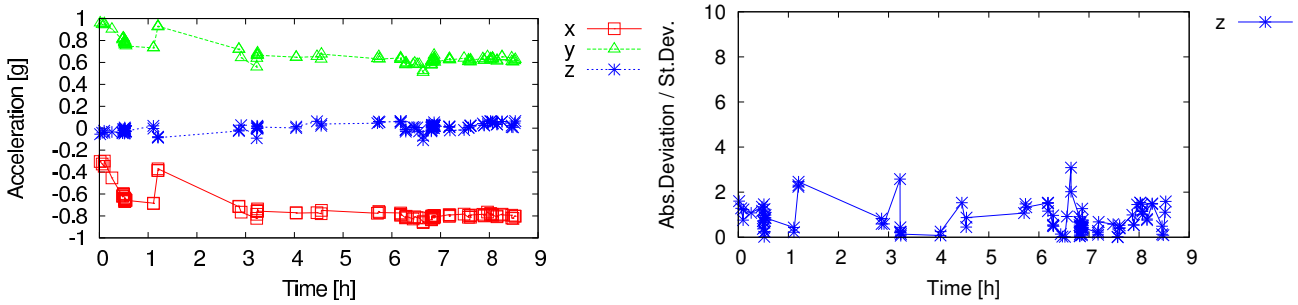
To the purpose of fall detection, the average interval between consecutive walk segments is particularly significant, as it determines the time needed for obtaining a new VD estimation. This interval is influenced by two main factors: first, by how frequently the user actually walks; second, by how much the algorithm is able to detect walking when it happens. The first factor is not under our control, while the second depends on the sensitivity of the walk recognition algorithm. Analyzing the long monitoring tracks more in detail, we were able to find out that the longest intervals without walk segments were registered when users had been sitting for a long time (e.g. driving, working at the office, watching TV). Conversely, when users were performing less sedentary activities (e.g. visiting shops), the walk recognition algorithm proved to be able to find a new segment with adequate frequency (within few minutes).

The long monitoring tracks were labeled with the actions performed by users. However, to reduce the burden, users were asked to annotate the activities/locations at a rough level, e.g. at home, at the office, shopping, driving, specifying the start and end time with minute precision. This approximate annotation made it impossible to calculate the specificity of the walk recognition algorithm, as this would have required second-level precision (each segment is $6s$ long) and a detailed specification of activities (e.g. by recording a video).

Nevertheless, some assumptions can be made if we consider the placement of the sensor during the long monitoring experiments. As illustrated in Figure 6, the device was placed in a front trouser pocket, with the z axis of the device almost orthogonal to VD. Hence, we expect the VD component along the z axis to remain almost constant and close to $0g$ across different estimations. Also, we expect only minor variations regarding the x and the y components of consecutive VD estimations, if correctly produced by

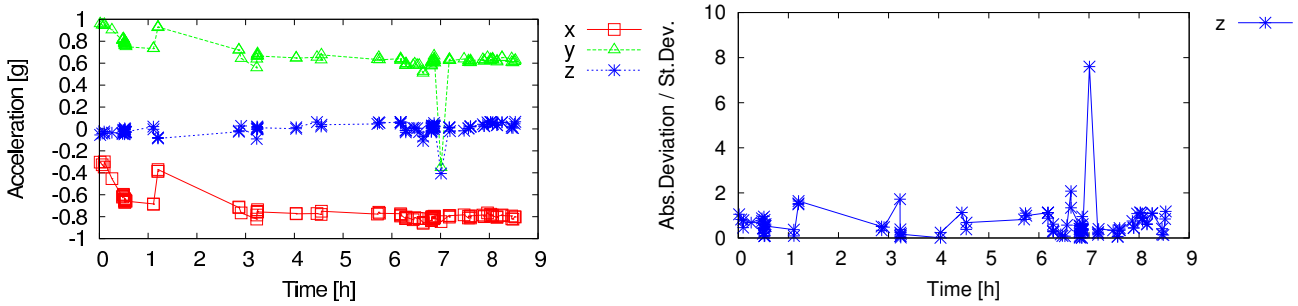
Table 3 Walk recognition results on long monitoring tracks

User	Duration [h]	Walk segments	Avg Interval [min]	Max Interval [min]
1	8.20	372	1.31	134.11
2	11.87	474	1.50	132.64
3	9.41	117	4.75	99.38
4	9.24	211	2.59	34.84
5	8.79	77	6.68	192.99
6	10.62	443	1.43	102.89
7	9.08	50	10.69	140.04
8	12.25	425	1.73	43.85
9	9.16	138	3.90	97.78
10	8.58	396	1.29	78.07
Global	97.21	2703	2.14	192.99



(a) (b)

Fig. 7 (a) VD estimations against time; (b) absolute deviation over standard deviation ratio on the z axis



(a) (b)

Fig. 8 (c) VD estimations against time with a wrong estimation performed while driving; (d) absolute deviation over standard deviation ratio on the z axis

the walk recognition algorithm. Significant differences are possible only if the device is extracted from the pocket and repositioned in a different way. In the latter case, we expect an abrupt change along the x and the y, followed by estimations that confirm the new orientation of the device with respect to the user's body. In any case, the z component of the estimated VD should remain close to $0g$ during our experiments.

To verify this hypothesis, we plotted the VD estimations obtained for each user against the time when they were found. Figure 7a shows one of these plots where the acceleration on the z axis is almost constant and very close to $0g$, while the acceleration along the x and y axes is characterized by some fluctuations. Figure 7b highlights the dispersion on the z axis: for each value it is shown the ratio between the absolute deviation from the average and the standard deviation. The maximum absolute deviation from the average is 3.1 times the standard deviation in the example.

Figures 8a and 8b, instead, have been produced by artificially adding the recognition of a walk segment during an activity that does not correspond to walking (driving, in this particular case). Such a wrong

estimation can be immediately identified simply from the observation of the plots. In particular, in Figure 8b it can be noticed that the absolute deviation over standard deviation ratio on the z axis presents an abnormal value corresponding to the fake walk segment. The fake walk segment produces a value on the z axis with an absolute deviation from the average equal to 7.6 times the standard deviation.

We inspected all the traces and verified that the pattern corresponds to the expected one. This information cannot be used to state that all walk segments were collected when the user was actually walking. However, we can state with reasonable confidence that no segment was collected while the user was in a non-upright position, such as lying or sitting.

6 Use of posture in a fall detection system

In general, accelerometer-based fall detection systems detect impacts by means of a fixed threshold on the acceleration magnitude (Karantonis et al. 2006; Kangas et al. 2008; Bourke et al. 2010; Abbate et al. 2011, 2012). Unfortunately, these impacts include real falls as well as fall-like impacts due to activities of daily living (ADLs), such as sitting or walking, that lead to false alarms. Approaches for reducing false alarms try to discriminate between ADLs and real falls using vertical velocity estimation (Degen et al. 2003; Bourke and Lyons 2008), post-fall activity detection (Karantonis et al. 2006; Abbate et al. 2012), and posture information (Karantonis et al. 2006; Kangas et al. 2008; Bourke et al. 2010).

The major drawback of existing approaches based on posture is the need for a fixed alignment of the device with respect to user's body, or for a manual calibration. Additionally, in case of manual calibration, the procedure must be repeated each time the orientation of the device changes. The walk recognition technique described in Section 5 can be used to address this limitation, since it allows the fall detection system to automatically and dynamically estimate VD each time the user walks.

In the following, we describe a fall detection system that uses our technique for the automatic estimation of device orientation to infer posture information. The performance of the fall detection system is then evaluated on the long monitoring tracks of our dataset, to measure both the overall results achieved by the system and the relative contribution to filter false alarms provided by posture information with respect to other filtering techniques.

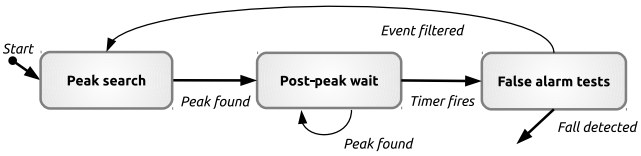


Fig. 9 Fall detection as a finite state machine

6.1 Fall detection algorithm

The fall detection algorithm can be described through the finite state machine shown in Figure 9. The initial activity of the fall detection algorithm (*Peak search* state) is finding a *fall-like impact*, called *impact* thereafter, and defined as follows. An impact is found when the magnitude of the acceleration signal exceeds a predefined `impact_peak` threshold. Values ranging from $2.5g$ to $3.5g$ have been used in the literature for this threshold (Bourke et al. 2007; Li et al. 2009). In our implementation, we set `impact_peak` to $3g$: the $3g$ value is small enough to avoid false negatives, as real falls are likely to produce a peak that exceeds such threshold, but not too small to generate numerous false alarms.

After an impact has been detected, the machine moves to the *Post-peak wait* state and starts a `bouncing_timer`. This timer is used to wait for the end of the impact phase. During the interval specified by the timer, the acceleration samples are still analyzed: if another magnitude sample above $3g$ is detected, the timer is restarted. When the timer finally fires, the machine moves to the *False alarm tests* state.

In *False alarm tests*, the algorithm performs a set of tests in order to confirm that the impact is a real fall. The set of tests that we have implemented includes: *post-impact activity test*, *vertical velocity estimation test*, *AAMV index test*, *activity ratio test*, and *lying detection test*. Only if all of these tests are passed the impact is definitely classified as a real fall.

The *Post-impact Activity Test (PAT)* is based on the assumption that, immediately after a fall, the user generally lies on the ground and produces little or no variations in the acceleration signal. The PAT has been implemented as described in Abbate et al. (2012). In particular, only an interval of $2.5s$ after the impact was considered, in order to allow fast detection of falls. If enough movement is detected, the impact is discarded as a false alarm.

The *Vertical Velocity estimation Test (VVT)* has been used to reduce the incidence of false alarms (Degen et al. 2003; Bourke and Lyons 2008). The estimation is based on the numerical integration of the acceleration magnitude after gravity has been subtracted. In order to increase the estimation accuracy, the acceleration signal related to the impact is low-pass filtered with

Table 5 Fall detection algorithm: impacts above $3g$

User	Impacts	Impacts/h
1	16	1.95
2	55	4.63
3	14	1.49
4	54	5.84
5	100	11.38
6	287	27.02
7	23	2.53
8	99	8.08
9	72	7.86
10	763	88.93
Global	1483	15.26

Table 6 Fall detection specificity results (%)

User	PAT	VVT	AAMVT	ART	LDT	All
1	81.3	43.8	37.5	43.8	100	100
2	94.5	3.6	50.9	16.4	100	100
3	85.7	7.1	78.6	35.7	92.9	100
4	25.9	14.8	64.8	35.2	100	100
5	74.0	18.0	72.0	77.0	99.0	100
6	98.6	1.0	57.8	30.0	99.0	100
7	30.4	56.5	69.6	82.6	91.3	100
8	94.9	3.0	29.3	14.1	98.0	100
9	98.6	2.8	90.3	68.1	97.2	100
10	78.8	16.5	78.9	62.9	98.3	100
Global	82.3	12.3	69.5	51.6	98.4	100

$15Hz$ cut-off frequency, as described in Bourke and Lyons (2008). The vertical velocity threshold has been selected analyzing the database of falls presented in Abbate et al. (2012): setting this value to $0.7m/s$ seems to be a reasonable choice to minimize the risk of false negatives. If no velocity estimations above the threshold are found, the impact is discarded as a false alarm.

The use of the *Average absolute Acceleration Magnitude Variation (AAMV) index* has been discussed in Abbate et al. (2011, 2012). According to the experimental results, falls are expected to produce faster variations in the acceleration magnitude with respect to sitting or lying. We implemented the AAMV index test (AAMVT) as described in Abbate et al. (2012), using a threshold equal to $0.26g$. If the value of the index is below the threshold, the fall-like impact is ignored.

The *Activity Ratio Test (ART)* is based on the *Activity Ratio Index (ARI)* described in Abbate et al. (2012). ARI measures the level of activity in an interval of $700ms$ properly centered at the fall-like impact. It is calculated as the ratio between the number of samples not in $[0.85g, 1.3g]$ and the total number of samples in the $700ms$ interval. Fall-like impacts are discarded as false alarms if the ratio is below 0.45.

Finally, posture information is considered through the *Lying Detection Test (LDT)*, which has been implemented by measuring the angle between the automatically estimated VD and gravity. If this tilt difference is below 50° the impact is discarded. We decided to set a 50° threshold to be conservative and reduce the risk of filtering out real falls. At the beginning of the track it may happen that a VD estimation has not been found yet: in this case, LDT is not executed.

6.2 Fall detection results and discussion

The fall detection algorithm was tested on the long monitoring tracks of our dataset. As no real falls occurred during the recording of these tracks, this test can be only used to measure the specificity of the fall detection algorithm. For the sensitivity, we tuned the filtering thresholds and parameters in order to ensure that all the simulated falls of the dataset presented in Abbate et al. (2012) were properly detected (100% sensitivity).

Table 7 Walk segments statistics

User	WS/h	AHP[g]
1	45.4	2.0
2	39.9	2.5
3	12.4	2.3
4	22.8	2.1
5	8.8	2.8
6	41.7	3.0
7	5.5	2.6
8	34.7	2.4
9	15.1	2.8
10	46.2	3.6
Global	27.8	2.6

As mentioned in Section 5.3, users were asked to annotate their current activity or location occurring during the long monitoring tracks. These were categorized at a rough level using the following labels: *home*, *office*, *transport*, *city*, and *outdoor*. *Home* includes activities such as resting, watching TV and housekeeping; *office* mainly includes short walks and long periods sitting at the desk; *city* refers to visiting shops or bars and walking outdoors; *transport* is used for the periods spent in a car or public transportation; finally, *outdoor* (performed only by user 10) includes activities mainly performed in the countryside, such as walking, jumping, kneeling, and bending. The percentages of different activities performed by each user are shown in Table 4.

The first step of the fall detection algorithm consists in the detection of impacts, and it is based on the $3g$ acceleration magnitude threshold. The results of impact detection applied to the long monitoring tracks are shown in Table 5, in terms of total number of impacts and impact rate (impacts per hour). These

Table 4 Long monitoring tracks, duration of activities

User	Duration [h]	Home %	Office %	Transport %	City %	Outdoor %
1	8.2	0.0	76.6	4.1	19.3	0.0
2	11.9	0.0	82.9	0.0	17.1	0.0
3	9.4	7.1	68.7	11.4	12.8	0.0
4	9.3	94.7	0.0	2.2	3.2	0.0
5	8.8	100	0.0	0.0	0.0	0.0
6	10.6	82.4	0.0	0.0	17.6	0.0
7	9.1	4.4	82.2	8.3	5.1	0.0
8	12.3	54.3	36.7	9.0	0.0	0.0
9	9.2	5.9	82.5	8.8	2.8	0.0
10	8.6	18.9	0.0	43.6	9.5	28.1
Global	97.2	28.2	52.3	8.2	8.8	2.5

figures highlight the need for techniques able to reduce the incidence of false alarms. Also, it can be observed a great variation in the impact rate of different users. This variation can not be solely explained by the peculiar activities a user performed. For example, user 9 showed an impact rate about 5 times greater than user 2, despite having performed similar activities. It is also worth noting that no impacts were produced while the users were traveling in a car.

The second step of the fall detection algorithm consists in filtering out false alarms by means of the five tests described in Section 6.1. Table 6 shows the results obtained by the different tests in terms of specificity (%), where the specificity reached by each filter is calculated as the ratio between the number of impacts recognized as false alarms and the total number of impacts. These results highlight the importance of the use of posture for the filtering of false alarms: in fact, the LDT was able to filter an average 98.4% of the impacts, bringing a leading advantage over all the other techniques. Post-impact activity detection techniques, i.e. PAT, AAMVT and ART, also bring a significant improvement (82.3%, 69.5% and 51.6%, respectively), while vertical velocity, i.e. VVT, seems to be the least relevant test (12.3%).

The high specificity achieved when filtering on the base of posture information suggests that most of the false alarms were produced while users were upright or walking. In order to evaluate if a relationship holds between walking and the impact rate of each user, we calculated the statistics shown in Table 7. For each user, the number of *Walk Segments per hour (WS/h)* and the *Average Highest Peak (AHP)* index are reported. The former can be used as a measure of the user's activity level, while the latter indicates the user's tendency to produce high acceleration peaks, and thus impacts, while walking. AHP has been calculated by averaging the highest acceleration magnitude peaks of walk segments. The combination of these two indexes together with the activities performed seems to offer a reasonable explanation of the impact rate experienced by each user. For example, the relatively low AHP value of user 1 determined a low number of impacts, while the relative high walking rate and high AHP value of user 10 determined the highest impact rate of the dataset.

Finally, it is worth highlighting that the use of posture on these long monitoring tracks was made possible by our technique based on automatic estimation of VD. The traditional approaches would have been inadequate, since they require manual calibration and/or predefined orientation of the device.

7 Conclusions

We presented a novel technique for increasing both usability and trustworthiness of fall detection systems. By finding segments of acceleration corresponding to walk periods, the orientation of the sensing device with respect to the user's body (and vice-versa) can be automatically determined. The advantage is twofold: manual calibration and alignment constraints are no longer necessary (as, instead, are with fall detection systems presented so far); and the sensing device can be worn in the user's trouser pockets. Such passively collected information is then used to understand whether, after an impact, the user's body is horizontal or not, thus reducing significantly the number of false alarms. Both the walk recognition algorithm, specifically designed for being included in a fall detection system, and the filtering effect due to posture information have been evaluated using a set of long monitoring tracks. Experimental results confirmed that the proposed method is able to improve the accuracy of a fall detection system, in terms of specificity.

Future work will focus on understanding how the different ways of carrying a smartphone (in a bag, in a jacket pocket) impact the proposed technique, as well as on the analysis of the energy consumption introduced by the proposed walk recognition algorithm.

References

Abbate S, Avvenuti M, Cola G, Corsini P, Light JV, Vecchio A (2011) Recognition of false alarms in fall detection systems. In: Proc IEEE Int Workshop Consum eHealth Platf Serv Appl, Las Vegas, NV, USA, pp 538–543

Abbate S, Avvenuti M, Bonatesta F, Cola G, Corsini P, Vecchio A (2012) A smartphone-based fall detection system. Pervasive Mob Comput 8(6):883 – 899

Albert MV, Kording K, Herrmann M, Jayaraman A (2012) Fall classification by machine learning using mobile phones. PLoS ONE 7(5):e36,556

Anderson D, Keller J, Skubic M, Chen X, He Z (2006) Recognizing falls from silhouettes. In: Conf Proc IEEE Eng Med Biol Soc, IEEE, pp 6388–6391

Avvenuti M, Casella A, Cesarini D (2013) Using gait symmetry to virtually align a triaxial accelerometer during running and walking. Electron Lett 49(2):120–121

Bourke AK, Lyons GM (2008) A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. Med Eng Phys 30(1):84–90

Bourke AK, O'Brien JV, Lyons GM (2007) Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. Gait Posture 26(2):194–199

Bourke AK, van de Ven P, Gamble M, O'Connor R, Murphy K, Bogan E, McQuade E, Finucane P, O'Laighin G, Nelson J (2010) Assessment of waist-worn tri-axial accelerometer based fall-detection algorithms using continuous unsupervised activities. In: Conf Proc IEEE Eng Med Biol Soc, pp 2782 –2785

Curone D, Bertolotti G, Cristiani A, Secco E, Magenes G (2010) A real-time and self-calibrating algorithm based on triaxial accelerometer signals for the detection of human posture and activity. IEEE Trans Inf Technol Biomed 14(4):1098–1105

Dai J, Bai X, Yang Z, Shen Z, Xuan D (2010) Mobile phone-based pervasive fall detection. Pers Ubiquitous Comput 14(7):633–643

Degen T, Jaeckel H, Rufer M, Wyss S (2003) Speedy: a fall detector in a wrist watch. In: Proc IEEE Int Symp Wearable Comput, pp 184–187

Estudillo-Valderrama M, Roa L, Reina-Tosina J, Naranjo-Hernandez D (2008) A proposal of a fall detection algorithm for a multidevice personal intelligent platform. In: IEEE Int Conf BioInforma BioEng, pp 1–4

Estudillo-Valderrama M, Roa L, Reina-Tosina J, Naranjo-Hernandez D (2009) Design and implementation of a distributed fall detection system - personal server. IEEE Trans Inf Technol Biomed 13(6):874–881

Gietzelt M, Schnabel S, Wolf KH, Bsching F, Song B, Rust S, Marschollek M (2012) A method to align the coordinate system of accelerometers to the axes of a human body: The depitch algorithm. Comput Meth Programs Biomed 106(2):97 – 103

Gjoreski H, Lustrek M, Gams M (2011) Accelerometer placement for posture recognition and fall detection. In: Conf Proc IEEE Intell Env, IEEE Computer Society, Washington, DC, USA, pp 47–54

Gurley RJ, Lum N, Sande M, Lo B, Katz MH (1996) Persons found in their homes helpless or dead. N Engl J Med 334(26):1710–1716

Kangas M, Konttila A, Lindgren P, Winblad I, Jms T (2008) Comparison of low-complexity fall detection algorithms for body attached accelerometers. Gait Posture 28(2):285 – 291

Karantonis DM, Narayanan MR, Mathie M, Lovell NH, Celler BG (2006) Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. IEEE Trans Inf Technol Biomed 10(1):156–67

Kunze K, Lukowicz P, Junker H, Tröster G (2005) Where am I: recognizing on-body positions of wearable sensors. In: Conf Proc Locat Context-Aware, Springer-Verlag, Berlin, Heidelberg, LoCA'05, pp 264–275

Kwapisz JR, Weiss GM, Moore SA (2011) Activity recognition using cell phone accelerometers. SIGKDD Explor Newsl 12(2):74–82

Lai D, Begg R, Palaniswami M (2009) Computational intelligence in gait research: a perspective on current applications and future challenges. IEEE Trans Inf Technol Biomed 13(5):687–702

Li Q, Stankovic JA, Hanson MA, Barth AT, Lach J, Zhou G (2009) Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In: Proc IEEE Int Workshop Wearable Comput Implant Body Sens, IEEE Computer Society, Berkeley, CA, USA, pp 138–143

Lindemann U, Hock A, Stuber M, Keck W, Becker C (2005) Evaluation of a fall detector based on accelerometers: a pilot study. Med Biol Eng Comput 43(5):548–551

Mizell D (2003) Using gravity to estimate accelerometer orientation. In: Proc IEEE Int Symp Wearable Comput, pp 252–253

Realtime Technologies Inc (2010) http://www.shimmer-research.com

Sixsmith A, Johnson N (2004) A smart sensor to detect the falls of the elderly. IEEE Pervasive Comput 3(2):42–47

Sposaro F, Tyson G (2009) iFall: An android application for fall monitoring and response. In: Conf Proc IEEE Eng Med Biol Soc, pp 6119–6122

Tinetti ME, Liu W, Claus EB (1993) Predictors and prognosis of inability to get up after falls among elderly persons. J Am Med Assoc 269(1):65–70

Tolkiehn M, Atallah L, Lo B, Yang GZ (2011) Direction sensitive fall detection using a triaxial accelerometer and a barometric pressure sensor. In: Int Conf Eng Med Biol Soc, pp 369–372

Wild D, Nayak U, Isaacs B (1981) How dangerous are falls in old people at home? Br Med J 282(6260):266

Xu W, Zhang M, Sawchuk AA, Sarrafzadeh M (2012) Robust human activity and sensor location corecognition via sparse signal representation. IEEE Trans Biomed Eng 59(11-2):3169–3176

Yavuz G, Kocak M, Ergun G, Alemdar HO, Yalcin H, Incel OD, Ersoy C (2010) A Smartphone Based Fall Detector with Online Location Support. In: Conf Proc PhoneSense, pp 31–35

Zigel Y, Litvak D, Gannot I (2009) A method for automatic fall detection of elderly people using floor vibrations and soundproof of concept on human mimicking doll falls. IEEE Trans Biomed Eng 56(12):2858–2867