Event Detection in Complex Environments: An Effective and Efficient Machine-Learning-Based Framework

Alfredo Cuzzocrea
DIA Dept., University of Trieste, Italy
alfredo.cuzzocrea@dia.units.it

Enzo Mumolo
DIA Dept., University of Trieste, Italy
mumolo@units.it

Abstract

In this paper we describe a falls detection and classification algorithm for discriminating falls from daily life activities using a MEMS accelerometer. The algorithm is based on a shallow Neural Network with three hidden layers, used as fall/non fally classifier, trained with daily life activities features and fall features. The novelty of this algorithm is that synthetic falls are generated as multivariate random Gaussian features, so only real daily life features must be collected during some day of normal living. Moreover, the features related to synthetic fall events are generated as complement of normal features. First of all, the features acquired during daily life are clustered by Principal Component Analysis and no Fall activities shall be recorded. The complement set of the normal features is found and used as a mask for Monte Carlo generation of synthetic fall. The two feature sets, namely the features recorded from daily life activities and those artificially generated are used to train the Neural Network. This approach is suitable for a practical utilization of a Neural Network based fall detection characterized by high Recall-Precision rate.

1 Introduction

The detection of falls of the elderly and people with deseases like epilepsy or Parkinson or simple people with motor difficulties, is today a problem of great public interest. This generated a wide range of research and led to the development of various falls detection and tele-monitoring systems to allow prompt intervention when a fall occours. Studies in the past years have shown that $1/3$ of Senior citizens over age 65 ([25]) are often victims of undetected falls and most of their injuries are due to a lack of intervention. Generally speaking the classification between daily life activities and falls is a difficult task because many daily life activities look like fall (for example running, sitting in a car or lying in bed) and many falls may look like daily life activities. All the types of errors of these systems are of great importance. In case of False Positive error, users are not motivated to use the falls detection system because in many normal life activities are wrongly detected as falls. In those cases, the operator soon gets tired of the false alarms. On the other hand, in case of False Negatives errors it happens that the system leads to lack of interventions in case of fall. Of course this situation is followed by problems of serious injury and also of mortality.

Generally speaking, while it is quite simple to gather accelerometer data during Normal Daily Living (Daily Life Activities or DLA), it is very difficult if not impossible to collect data during Falls, so the question is: how can the false positive rate be reduced if enough fall data is not available? The answer to this question is that we produce synthetic fall data starting from data collected during normal daily living.

Our approach reduces the amount of False Positives compared to threshold based systems by performing extensive training of a neural network if large quantities of features are gathered during normal life activities.

In the last decade, there has been a great deal of research that has examined the use of inertial sensors such as accelerometers and/or gyroscopes to realize systems for automatic detection of falls. The goal of these systems is to detect the falls of patients who have any difficulty in walking to quickly alert the operators who provide a suitable assistance. The characteristics these systems must have are on one side low cost, consumption and size, and on the other side high performance. While the first characteristics are always better met, the performance is still unsatisfactory. Typically, performance is measured in false positives and false negatives. False positives, if excessive, could demotivate the use of these systems because the recipient of a fall warning could get tired of it. Clearly, false negatives are the most dangerous for the patient's health because they correspond to missed falls. Normally these errors are measured by Sensitivity (ability to detect actual falls) and Specificity (ability to avoid false positives) or equivalently Recall and Precision. From the literature, the existing technology for falls detection systems can be roughly classified into three

DOI reference number: 10.18293/DMSVIVA2019-023

categories.

- With wearable sensors: These systems use a triaxial accelerometer or gyroscope or a combination of both to estimate the posture of the subject's body. The sensors are placed in different places, such as the waist, the thigh, the wrist, the shoes. Many systems use the smart-phones inertial sensors.

- With environmental sensors: The environmental sensors are nothing more than sensors positioned around the subject. Floor sensors such as pressure sensors on the mat, microphones, infrared sensors, microwave motion detectors are used to detect the fall. The classifiers of neural networks are used to classify daily activities.

- With image processing: In such methods, a camera is used to monitor body postures. Falls are detected using various image processing techniques such as pattern matching, posture recognition, skeleton extraction, background subtraction, optical flow processing, etc.

A fall can be described as the rapid change from standing or sitting towards an elongated position in earth or almost elongated [24]. This definition has been used in many studies. The paper is organized as follows. In Section 2 we describe the state of the art. In Section 3 we provide the main features of the accelerometer. In Section 4 we introduce preliminary definitions of our work. In Section 5 we describe our main approach. In Section 6 we provide the proposed metrics for performance evaluation. In Section 7 we provide the experimental evaluation and assessment of our proposed framework. Finally, in Section 8 we discuss concluding remarks and future work of our research. A preliminary version of this paper appears in the short paper [5].

2 Previous Work

The algorithms used in systems with wearable sensors can be divided into approaches that use thresholds-based heuristics and approaches that use machine learning tools [23]. The latter may be k-Nearest Neighbor (kNN), neural networks, hidden Markov models, or the two classes classification schemes based on the Support Vector Machines (SVM) classifiers. In all cases, however, it is important to have both falls and daily life activity features. In the case that the approaches are based on thresholds, the availability of both types of data is important to find optimal thresholds, while in the other case the data are important for correctly training the machine learning algorithms.

The thresholds-based heuristics approaches are methods that use thresholds and appropriate functions derived from inertial parameters. The simplest approach to detecting a fall could be to detect the ground position of the person, by means of a horizontal inclination detection sensor. This method is suitable for monitoring "isolated subjects" but less suitable for the detection of falls of an elderly person in his home environment as the hours of sleep are not regular. Therefore this method provides many "false positives". A complementary solution is to detect the person lying on the floor, using floor tiles equipped with sensors. But when the falls do not end on earth, or if the floor does not have these sensors, obviously they are not detectable. When it falls, the person often hits the ground or an obstacle. The "shock impact" causes an intense inversion of the polarity of the acceleration vector in the direction of the trajectory, which can be detected with an accelerometer or a shock detector, which is actually a threshold accelerometer. Although most of the falls occur in the "front" plane (forward or backward), the direction of the trajectory of fall and is obviously variable from one fall to another. Also the position of the sensor on the body relative to the point of impact modifies the signal recorded at the moment of shock. The lack of movement can be used to detect the fall as, after the "serious" fall, in which the person can be seriously injured, they often remain immobilized in one position. A motion / vibration sensor, positioned on the body (e.g., wrist or ankle), can be used or, again more simply, infrared sensors of presence disseminated in the home. The choice of latency time (the delay before the decision) that should be long enough to reduce "false positives", which translates into a longer delay before intervention, represents a critical problem for these approaches. As discussed earlier, during a fall there is a temporary "fall free" period, during which the vertical speed increases linearly over time due to gravitational acceleration. If you measure the vertical speed of the normal movements of the person (getting up, lowering, sitting down), you can discriminate these speeds from what you do during the fall, which would exceed an appropriate threshold. The intrinsic of analytical methods lies in the choice of this threshold, which if too low causes "false positives" and if too high causes "false negatives". Also this threshold differs from subject to subject. Image processing of video signals can also be used to detect one fall identifying the lying posture using analysis of the visual scene or detecting brusque movements using the revelation of movements with respect to the background. The latter method usually consists in subtracting successive images to keep only the variations, which are then sorted according to their direction and / or their width. While these techniques are well established in controlled environments (e.g., laboratory), they must be modified in environments uncontrolled where parameters such as lighting or framing can be arbitrary. Furthermore, if the subject moves in a 3-dimensional space, it may need more complex techniques, namely the

use of 2 cameras ("VisionStereo"). These image techniques are absolutely feasible at present, both technically and economically, thanks to the presence on the market of low cost cameras (web cam), with the possibility to transmit images in wireless mode over short distances and availability of appropriate processing algorithms. However the acceptance of these technologies for images poses big problems of privacy, as it requires the positioning of video cameras in the living space of the person, and in particular in the bedroom and the bathroom.

An alternative to heuristics approaches, machine learning methods can be used to detect falls. These methods are based on the data acquired on the real working system, used in a preliminary training phase. "Supervised" or "unsupervised" classification algorithms can be used. In case of classification algorithms with "supervised" learning, the person wearing the device performs a series of voluntary actions in order to identify the parameters in the normal cases. In the case of "unsupervised" learning, it is possible to record the movements of the person, within a few hours or several days, and then perform a statistical analysis of the measured speed. These approaches to developing fall detection algorithms are based on observing of the data (the training period) and then on the classification. The choice of classification algorithms is very broad. If you use a supervised method, a simpler choice is to train a neural network, which will then be used to automatically classify the signal. Only the situations encountered during training can be recognized, all others can be shuffled into a class called "others" if the algorithm " is " unsupervised ", falls can be isolated if the training period is much longer than the fall event. Furthermore, it is likely that the first event fall is not detected since its class is still unknown before its premiere appearance.

Regarding the thresholds-based heuristics approaches, Bourke *et al.* [2] uses signals from triaxial accelerometers mounted on the trunk and the thigh to distinguish falls from the *Activities of Daily Living* (ADLs). They propose a higher fall threshold (UFT) and a lower fall threshold (LFT) in an attempt to optimize the balance of false positives and false negatives. Likewise, Kangas *et al.* [13] attached a triaxial accelerometer to the waist, wrist and head of volunteers who performed simulated drops and ADLs in laboratory. Their algorithms considered the phases of pre-impact, impact and post-impact of the fall, separately and in combination, and achieved up to 100% of specificity and sensitivity of 95%, using a single sensor mounted at the waist. However, this algorithm has not been tested in real environments. The only study that examined its accuracy in the real world was conducted by Bagala *et al.* [6], which evaluated fall detection methods (including the Bourke and Kangas algorithms described above) using data from real falls, achieving much better results. In laboratory settings, the development of improved algorithms for automatic fall detection in the elderly requires an understanding of real-life fallout scenarios in older adults and the integration of such information into the design of laboratory experiments. The common fall scenarios are often absent in the majority, if not in all, of the previous laboratory experiments of fall, and the consequent discrepancy in the sensor data is, perhaps, the main cause of the lack of accuracy of the fall detection algorithms, when tested on real scenarios. In [9] a system is described that uses a tri-axial accelerometer and gyroscope. The detection algorithm uses three thresholds: one to recognize pre-fall situations, one to detect the maximum of the acceleration vector module and one to detect the maximum angular velocity. The algorithm described in [10] also uses three thresholds, one for the local minima of the acceleration module and two for the local maxima of the acceleration module and the angular velocity module. Systems based on machine learning algorithms use classifiers that are trained with both ADL and falls data. However, in a realistic context, due to the lack of sufficient availability of falls data and the lack of knowledge and understanding of what could be the falls, approaches based on the detection of anomalies and classification of a single class can be used. These techniques can not identify falls directly because fall data is not available for classifier training. However, they can identify falls indirectly by classifying them as abnormal activities. In these approaches, therefore, abnormal activities are classified as deviations from normal behavior. Naturally, the concept of normal activities must be clearly defined to identify abnormal activities. Moreover, even if the data of normal activities are not sufficient, then these techniques can produce excessive false positives. Recent research projects [4], [22], [16] show that falls can be identified without actually acquiring them. As evidenced by Klenk et al. [11], simulated falls differ significantly from real-world falls. Thus, having simulated falls in the training data set could lead to achieve classifiers that show different behaviors with real-world falls However, other authors such as Zhou and others [21] have presented a method to detect falls using transitions between activities to model falls. Zhou and others trained supervised classifiers using the normal activities collected by a mobile device, then used transitions between these activities to train a One-class Support Vector Machine (OSVM) and showed that it performs better than an OSVM trained only with activities normal. Micucci et al. [22] evaluates methods of detecting falls that do not require dropping data during training on different data sets collected using the smart-phone accelerometer. Their results show that in most cases, the One Class k-Nearest approach Neighbor classifier OCNN behaves better or equivalently to supervised SVM and KNN classifiers that require both types of data, i.e. data for normal and abnormal activities. In other words, Micucci et al.

use the one-class k-Nearest Neighbor (kNN) classifier and the one-class SVM classifier. These classifiers have been trained only with ADL and FALL instances. If the anomaly score is higher than a given threshold, the new instance is classified as an anomaly / fall, otherwise is classified as an ADL. Micucci et al compares the anomaly detectors with a two classes kNN and a two-classes radial basis SVM. These classifiers have been trained and tested with both instances of ADL and FALL. This is the case that we are looking for in a real scenario. The main contribution of [22] is the discovery that to design an effective method of detection of falls, it is not necessary to acquire data of falls but it is sufficient to classify the test data as anomalous. As for the HMM models, the traditional way to detect unseen abnormal activities appears as a model of normal activity using an HMM, appears the likelihood of a test sequence with the trained models and if it is below a pre-defined threshold then identify it as an anomalous activity [20]. Another common method to detect anomalous activities is to model the normal activities by a common HMM instead of modeling them separately. In [17] two HMM algorithms are presented that are normal HMM, in which the system noise covariance of the normal dynamics is used to determine the region with highest likelihood which are far from normality based on which events can be classified as 'not normal'. Their results show high detection rates for falls on two activity recognition data sets, albeit with an increase in the number of false alarms. In [16], Khan *et al.* experimentally show that this approach can give better results than supervised classification with limited fall data. When the number of fall data increases, the performance of supervised classifiers improves, but falling data collection can take a long time.

3 Accelerometer Features

The output of the MEMs accelerometer are the three component of the acceleration vector according to the three axis x, y, z, namely a_x, a_y, a_z, each of them related to the current time instant. From this signal, many features are extracted, see for example [14, 27]. We first compute the modulus of the acceleration vector, namely $Acc = \sqrt{a_x^2 + a_y^2 + a_z^2}$. Let us give a look to Figure 1 which is the time evolution of Acc for a typical fall. In this case, it is a Fall backward while trying to sit on a chair, taken from Mobifall v.2. It is worth noting that the overall time frame is the typical fall time. Here we choose the following measures: the maximum value of the modulus, labeled as ***Peak*** in Figure 1, the length between the two arrows, labeled as ***Base***, and the modulus of the slopes of the three components within the signal frame. The slope is computed as follows. Calling t_1^i, t_{1+N}^i the first and last time instant of

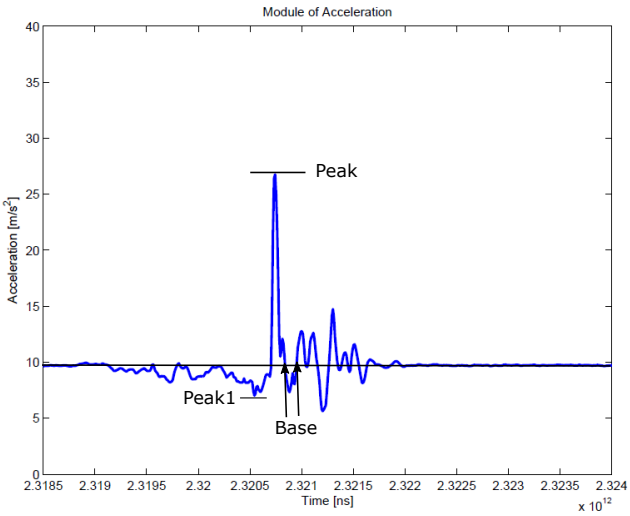

Figure 1. Features Extracted from the Graph of the Module

the N samples i-th frame, let us consider the values:

$$max_{a_x} = max\{a_x(t_1^i), \ldots, a_x(t_{1+N}^i\},$$
$$min_{a_x} = min\{a_x(t_1^i), \ldots, a_x(t_{1+N}^i)\},$$
$$max_{a_y} = max\{a_y(t_1^i), \ldots, a_y(t_{1+N}^i\},$$
$$min_{a_y} = min\{a_y(t_1^i), \ldots, a_y(t_{1+N}^i)\},$$
$$max_{a_z} = max\{a_z(t_1^i), \ldots, a_z(t_{1+N}^i)\}, \quad min_{a_z} = min\{a_z(t_1^i), \ldots, a_z(t_{1+N}^i)\}.$$

Then, in the interval t_1^i, t_{1+N}^i, the slopes of the three components are: $slope_x = max_{ax} - min_{ax}$, $slope_y = max_{ay} - min_{ay}$, $slope_z = max_{az} - min_{az}$. The modulus of the slope component is

$$Slope = \sqrt{slope_x^2 + slope_y^2 + slope_z^2} \qquad (1)$$

Another feature we use is the Ratio of the Peak over Base, as described in (2)

$$Ratio = \frac{Peak}{Base} \qquad (2)$$

These features have been chosen because they require very low computation, and so they can be used also on embedded processors with very little computational power.

4 Preliminary Definitions

We now make some preliminary definition useful in Section 5. Assume we use N features describing Falls and ADL. Then, consider the following definitions.

Definition 1 *The Features Space (FsS) is an hyper-cuboid with 2^N vertices and $2 \cdot N$ sides where all the original features points lie.*

Calling $max(feature_i)$, and $min(feature_i)$ respectively the maximum and minimum values the $i-th$ feature can reach in the current case, the length of the first side of the hypercuboid is $max(feature_1) - min(feature_1)$, of the second side is $max(feature_2) - min(feature_2)$ and so forth. The two vertices V_1 and V_2 with respectively the minimum and maximum Euclidean distance from the origin have coordinates $V_1 = (min(feature_1), min(feature_2), \ldots, min(feature_N))$ and $V_2 = (max(feature_1), max(feature_2), ldots, max(featu$ respectively. Of course, if we have only two features, FsS is a rectangle and if we have three features, FsS is a 3D cuboid.

Definition 2 *$DLAS \in FsF$ is a set whose elements are vectors of features collected in one or more days of Daily Living Activities (DLA).*

Definition 3 *$FAS \in FsF$ is a set whose elements are vectors of features collected during Fall Events.*

In Figure (2) we report an example of FsF, $DLAS$, FAS and vertices V_1, V_2 for two and three features respectively.

Definition 4 *$SynFS \in FsF$ is a set whose elements are features representing Synthetic Falls, and C_{SFS} is its Cardinality, i.e. the number of its elements. The contours of such sets will be used as $Masks$ in Monte Carlo synthetic generation of falls.*

As stated before, our assumption is that $SynFS$ can be viewed approximately as a complement to $ADLS$, provided that certain conditions are verified. $SynFS$ is represented by an N-dimensional sphere with center in V_2, which is the vertices of FsS most distant from the origin. The radius of SFS, initially equal to zero, is found with an iterative approach which increase its value until the desired number of elements of $DALS$ are included in it. In other words, $SynFS$ is defined as follows:

$$SynFS = \{z | z \in D\bar{L}AS \cup \{\textit{some elements of DLAS}\} \wedge C_{SFS} \leq \gamma\} \quad (3)$$

Let us look at the Figure 2 and Figure 3. Here we assume that only two features are used, so the sets can be drawn as a 2D plane. The points represented with squares are ADL elements. The circles' boundaries are related to some SFS with different Cardinalities and define two $Masks$ which will be noted as $Mask_{Cardinality}$.

We recall now the definition of a useful operator from Binary Morphology [15, 19].

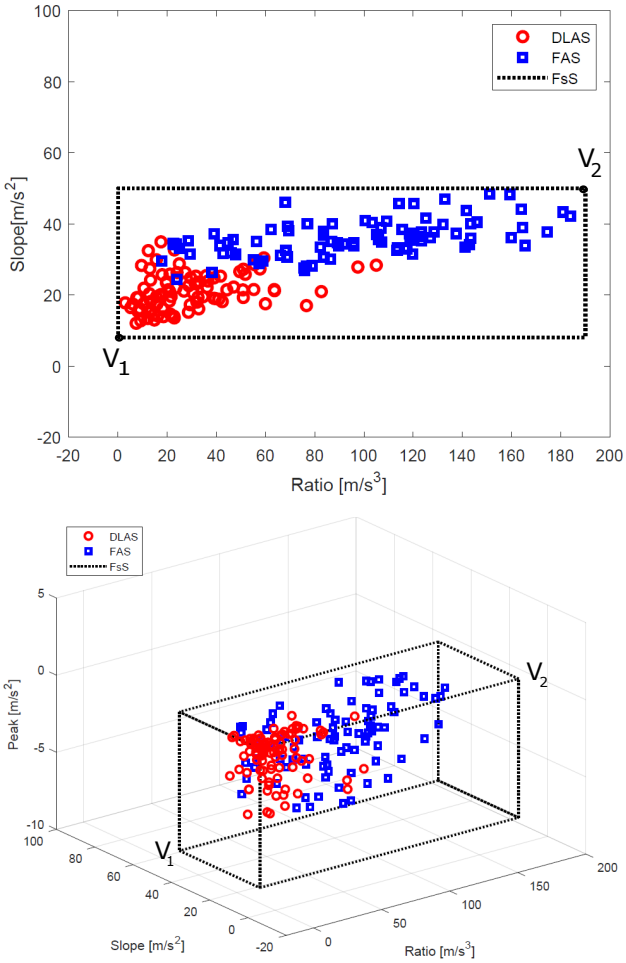

Figure 2. Example of 2D and 3D Features Obtained in Daily Living Activities and Fall Events

Definition 5 *The Delation of a given matrix A by a structuring element B, represented with the $\oplus$ symbol is defined as follows: $A \oplus B = \{x | B \cap A \neq \emptyset\}$.*

5 Description of the Approach

Our goal is to generate synthetic falls by using a Monte Carlo algorithm. In other words N-dimensional random vectors from Gaussian Distribution are generated and filtered by the Masks described above so that only the vectors falling within the Mask are retained. In order to reduce the computational complexity of the Monte Carlo algorithm, binary operations are performed.

First of all, Principal Component Analysis [1, 12] of DLAS data is performed to cluster DLA data. To clarify the

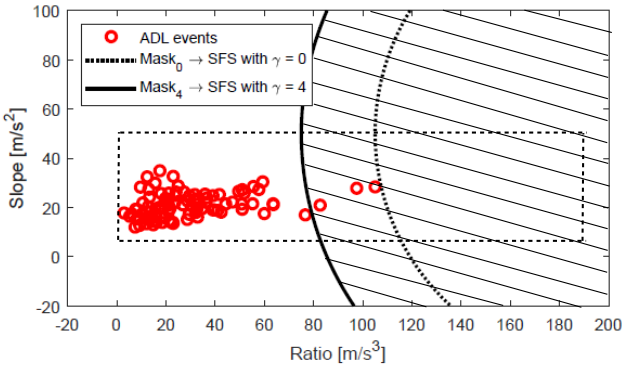

Figure 3. Example of Two Synthetic Fall Sets with Different Cardinality

algorithm description let us assume that only two features are used. Using the data of Figure 2 by PCA we approximate the shape of the data with the ellipse depicted in Figure 4. It is worth observing that the major and minor axis of the

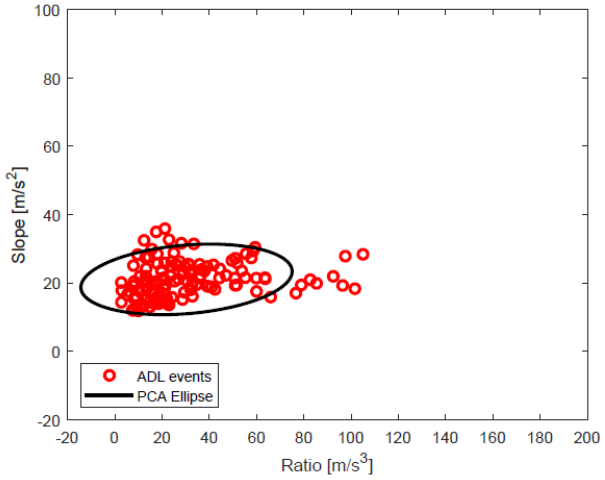

Figure 4. Shape of the Data Captured by PCA

ellipse are the first and second components of the data.

The binary operation starts by a binary version of the PCA ellipse, which is simply a projection of the PCA ellipse on a 100×100 binary matrix. It is worth noting that here we make use of binary data for complexity reduction. The ellipse is filled with ones. This is shown in Figure 5, where the binary matrix corresponding to Figure 4 is depicted. The set whose elements are the pixels inside the binary ellipse is called BE for binary ellipse. Then, a mask is generated

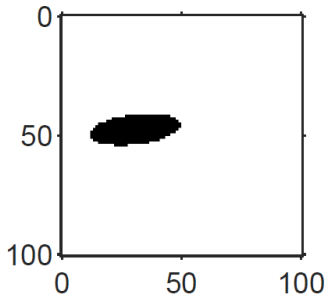

Figure 5. PCA Shape as a Binary Matrix

with Morphological Delation using n-dimensional polytope structuring elements. It is worth noting that we use delation instead cicles because the shapes are easier to binarize. In the 2D example we use an octagon. Starting from an octagon in V_2, a rough approximation of the spherical boundaries shown in Figure 2 are obtained using the following iteration: $mask = mask \oplus octagon$. Let M be the set of pixels set to one in the mask. Then we estimate the number of pixels of the intersection between the mask and the binary ellipse by performing the set operation $Len(M \cap BE)$. Now we introduce the threshold γ such that the number of pixels in the intersection between mask and binary ellipse be $\leq \gamma$. A loop is performed such as:

$$\textbf{while} Len(M \cap BE) \leq \gamma \textbf{then} M = M \oplus B \qquad (4)$$

where B is an octagon structuring element. In the upper panel of Figure 6, a sequence of Masks (the black surface) for different values if γ is reported. The panel at the bottom

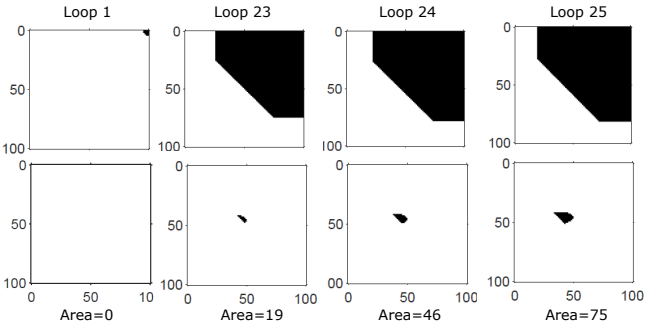

Figure 6. Portion of the Sequence of Masks for Increasing Values of γ

of Figure 6 shows the corresponding result of the set operation $M \cap E$. The number of pixels of $M \cap E$ increases until the area is greater than the threshold γ.

Synthetic falls, finally, are generated as random vectors $X = [X_1 \dots X_n]^T$, where $X_1 = \text{feature}_1, X_2 = \text{feature}_2, \dots, X_N = \text{feature}_N$, according to the Gaussian multidimensional probability distribution $N - Gauss$ reported in (5).

$$p(x; \mu, \Sigma^2) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right) \quad (5)$$

In 5 the mean is estimated as

$$\mu = [\frac{max(\text{feature}_1) + min(\text{feature}_1)}{2}, \dots$$
$$\dots, \frac{max(\text{feature}_N) + min(\text{feature}_N)}{2}] \quad (6)$$

so $\mu \in \mathbf{R}^n$. Morover, in (5) the covariance matrix is an $N \times N$ symmetric matrix. The diagonal elements are the variances of each feature. It is estimated as follows:

$$\Sigma(i,i) = \frac{(max(\text{feature}_i) - min(\text{feature}_i)^2)}{4}. \quad (7)$$

All the other elements are equal to zero. In Figure 7 we report an example of random generation of features according to the bi-variate Gaussian distribution.

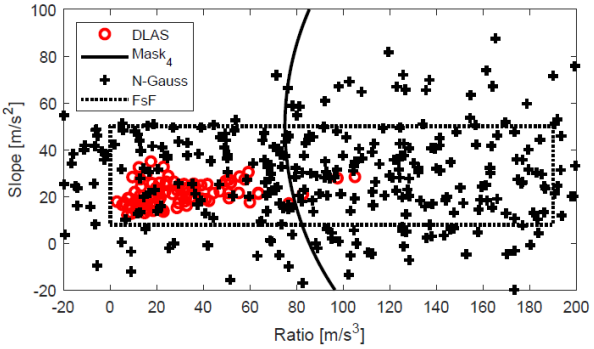

Figure 7. Example of Random Bi-Variate Gaussian Generation

Finally we evaluate the synthetic fall features as intersection of the random bi-variate features with the feature Space and the binary Mask. The synthetic falls features are reported in the example shown in Figure 8 where also the real fall features are reported for a first comparison.

In Figure 9 we recall that the neural network has input data derived from the sensors. Then we have three hidden layers and one exit layer. The dimension of this network allows to perform the network training using usual back-propagation.

The key point of our approach is the following. The neural network is trained with three features, namely Ratio,

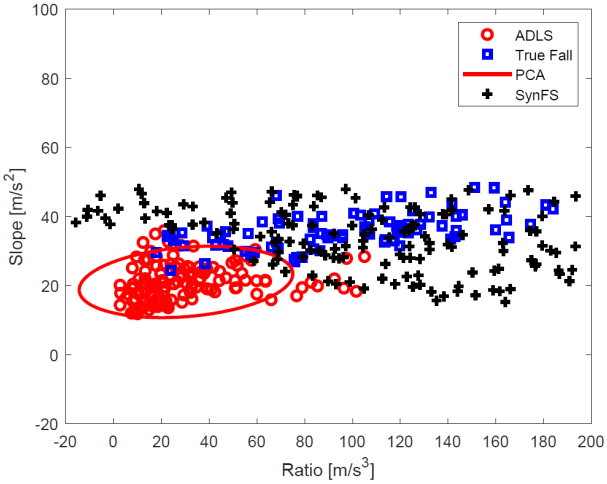

Figure 8. Example of Synthetic Fall Features

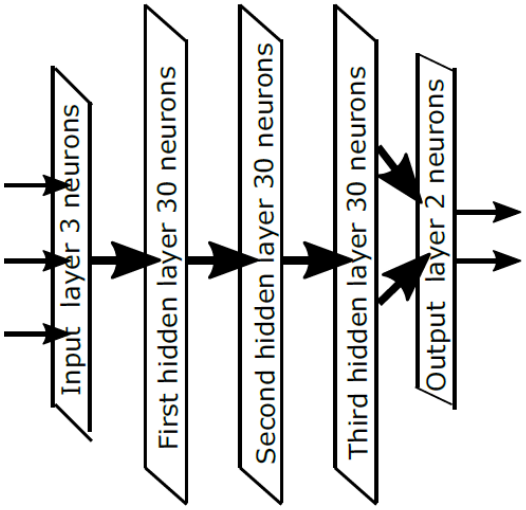

Figure 9. Neural Network Structure Trained with Real DLA and Synthetic Falls

Base and Peak1 extracted from ADL and falls. The DAL output of the network is set to zero during daily living features and the FALL output is set to one for synthetic falls features. When a real fall happens, the network should be able to detect it by looking which output is greater tahn the other.

It is worth recalling that the problem with training such

neural network classifier is, in practice, we have only real ADL features and it is not easy to have real falls. To solve this difficulty, we generate features which are classified as Falls. In other words, we generate features which are points in the set that is complement to the set of the DAL set.

6 Metrics for Performance Evaluation

The starting point for measuring the quality of a classifier is to obtain the rate of false positive (fp), false negative (fn), true positive (tp) and true negative (tn) from the classifier. In our case of a falls detector, let us suppose that there is a fall. If the detector detects it, a **tp** is measured. If it does not detect it, we have a **fn**. For example, on 100 actual fall events, the detector could have 80 **tp** and 20**fn**. If there has not been a fall, the detector could say that there was a fall (**fp**) or that there was no fall (**tn**). So on 100 non-fall events, we could have 80 **tn** and 20 **fp**.

In other words:

- TP (true positive): This is a situation in which a fall occurs and the system correctly detects it.
- FN (false negative): In this situation we have the fall happens, but the device does not announce it.
- TN (true negative): This is the situation in which a fall does not occur and the system correctly detects that there has not been a fall.
- FP (false positive): In this situation the fall does not happen but the device incorrectly announces that it has detected a fall.

These measures are also called:
tp $\rightarrow$ **hit** **fn** $\rightarrow$ **correct rejection**
fp $\rightarrow$ **false alarm**
tn $\rightarrow$ **miss**

It is sometimes convenient to measure errors in a more concise way. The most used measures are:

$$Precision = \frac{tp}{tp + fp}$$

This parameter measures the following quantity: the proportion of positive responses that have really fallen.

$$Recall = \frac{tp}{tp + fn}$$

This parameter represents the system's ability to detect a fall every time it occurs. The algorithm is good if the recall approaches 1 because in this case there are no false negatives. In other words, this parameter measures the proportion of falls that have been correctly identified.

7 Experimental Results

First we give a look to the data sets used for experiments. The first is the MobiAct v2.0 data set [7]. MobiAct contains data of four different types of falls and nine different daily living activities from a total of 57 subjects with more than 2500 trials. As well as being used to obtain experimental results, in this paper MobiAct is used as representative data set in all the Figures.

The second is UMAFall, which contains data from 17 subjects performing 8 different types of ADL and 3 different types of falls. Sensing point of MobiAct and UMAFall is the right trouser pocket.

The third data set used for experimental result is the SisFall [8, 26], selecting data related to the waist sensor point. It was generated with 38 participants performing repetitions of 19 ADL and 15 fall types.

We first obtain False Positive, and Negative as well as True positive and True Negative. All the results reported in the following are averaged over these data sets. Of course as usual we try an input signal from the test section of the data sets and we look if the output is correct or not. These results are reported in Figure 10 and in Figure 11.

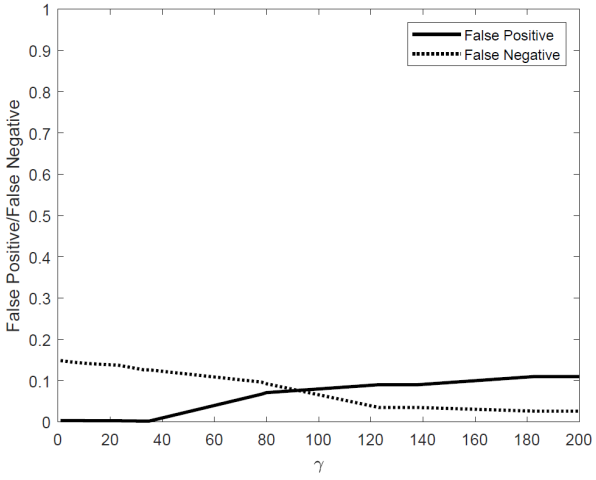

Figure 10. False Positive and False Negative Obtained with Real ADL and Falls Data

Then we obtain the values of Recall and Precision, reported in Figure 12.

These performances have been compared with the Bourke algorithm [2], which is three threshold based. To find the optimum values of the thresholds, we noticed that the most important threshold is the third one, so we found its performances at various values of the third threshold. The results are reported in Figure 13 which shows that the val-

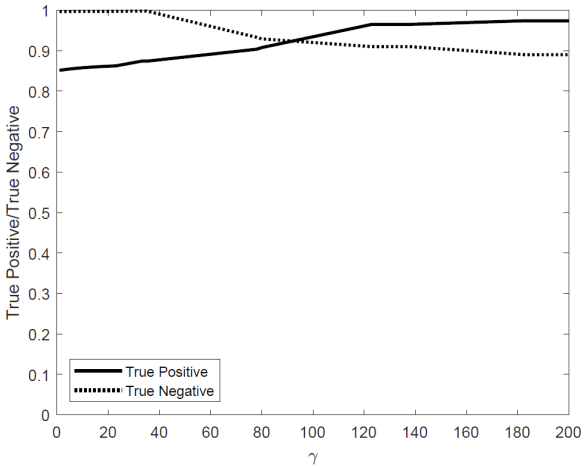

Figure 11. True Positive and True Negative Obtained with Real ADL and Falls Data

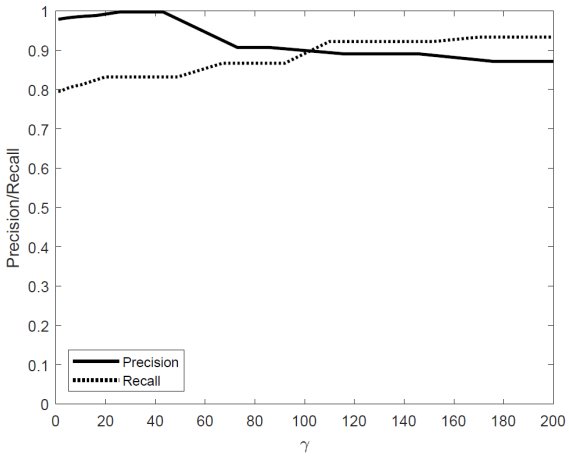

Figure 12. Recall and Precision Obtained with Real ADL and falls Data

ues of Recall/Precision are quite lower than our algorithm.

8 Final Remarks and Future work

Many fall detection systems are based on thresholds applied on features derived from inertial sensors. In this paper we report a novel algorithm which is able to achieve high performance, namely an equal Recall and Precision value more of 90% and a false error rate in this point less than

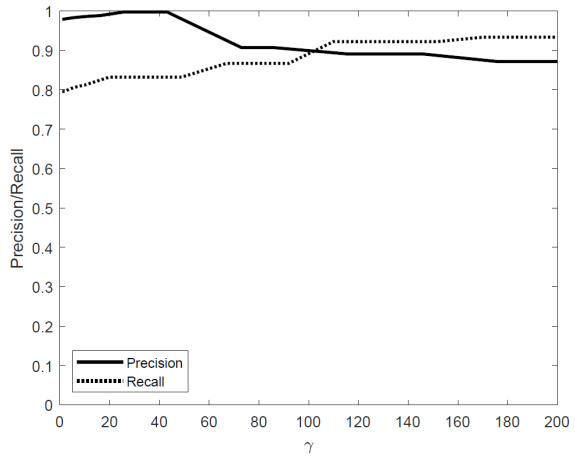

Figure 13. Recall and Precision Performances of the Bourke Algorithm for Various Values of the Third Threshold

8%. The main feature of this algorithm are that it only requires collecting features in periods of normal daily living and then, from these features, it estimates features of artificial falls. The availability of many features which describe normal and fall events allow to use Machine Learning approaches which are very powerful classifiers provided that sufficient amount of data is given. An important aspects of the described approach is that we use only one type of inertial sensor, namely the accelerometer. Future work will be focused on the fusion of the described results obtained with only an accelerometer with other types of inertial sensors, for example a gyroscope or a magnetometer. In this way many other data could be given to the neural network. The computation complexity is very low because it needs only to compute a trained neural network, so it can be performed in real time. Moreover, we used only three features. It would be interesting to see how much the performance increase if other features are used, hence leading to a feature hyperspace as indicated in Section 3. Another possible direction of research consists in making our algorithm compliant with emerging features of novel big data systems (e.g., [18, 28, 3, 29]).

References

[1] H. Abdi and L. J. Williams. Principal component analysis. *WIREs Comput. Stat.*, 2(4):433–459, July 2010.

[2] A. Bourke and G. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Med Eng Phys.*, 30(1):84–90, 2008.

[3] P. Braun, J. J. Cameron, A. Cuzzocrea, F. Jiang, and C. K. Leung. Effectively and efficiently mining frequent patterns from dense graph streams on disk. In *18th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2014, Gdynia, Poland, 15-17 September 2014*, pages 338–347, 2014.

[4] M. C, I. R, P. I, and C. M. Detecting falls as novelties in acceleration patterns acquired with smartphones. *PLoS ONE*, 2014.

[5] A. Cuzzocrea, E. Mumolo, and M. Tessarotto. Towards an effective and efficient machine-learning-based framework for supporting event detection in complex environments. In *Proceedings of the 43rd IEEE Computer Society Signature Conference on Computers, Software and Applications, COMPSAC 2019, Milwaukee, WI, USA, July 15-19, 2019*, 2019.

[6] B. F, B. C, C. A, C. L, A. K, H. JM, Z. W, and K. J. Evaluation of accelerometer-based fall detection algorithms on real-world falls. *PLoS One*, 7(5):1–9, 2012.

[7] V. G., C. C., M. T., P. M., and M. Tsiknakis. The mobiact dataset: Recognition of activities of daily living using smartphones. In *International Conference on Information and Communication Technologies for Ageing Well and e-Health*, pages 143–151, 2016.

[8] S. Gasparrini, E. Cippitelli, S. Spinsante, and E. Gambi. A depth-based fall detection system using a kinect® sensor. *Sensors*, 14(2):2756–2775, 2014.

[9] H. W. Guo, Y. T. Hsieh, Y. S. Huang, J. C. Chien, K. Haraikawa, and J. S. Shieh. A threshold-based algorithm of fall detection using a wearable device with tri-axial accelerometer and gyroscope. In *Intelligent Informatics and Biomedical Sciences (ICIIBMS), 2015 International Conference on*, pages 54–57. IEEE, 2015.

[10] Q. T. Huynh, U. D. Nguyen, L. B. Irazabal, N. Ghassemian, and B. Q. Tran. Optimization of an accelerometer and gyroscope-based fall detection algorithm. *J. Sensors*, 2015.

[11] K. J, B. C, L. F, N. S, M. W, A. W, Z. W, H. JM, van Lummel RC, C. L, and L. U. Comparison of acceleration signals of simulated and real-world backward falls. *Medical Engineering & Physics*, pages 368–373, 2011.

[12] T. T. Jolliffe and J. Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, pages 1–16, 2016.

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

[14] A. O. KANSIZ, M. A. GUVENSAN, and H. I. TURKMEN. Selection of time-domain features for fall detection based on supervised learning. In *Proceedings of the World Congress on Engineering and Computer Science 2013 Vol II*, pages 1–6, 2013.

[15] L. R. Kennell, R. W. Ives, and R. M. Gaunt. Binary morphology and local statistics applied to iris segmentation for recognition. In *ICIP*, pages 293–296. IEEE, 2006.

[16] S. S. Khan, M. E. Karg, D. Kulic, and J. Hoey. Towards the detection of unusual temporal events during activities using hmms. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1075–1084, 2012.

[17] S. S. Khan, M. E. Karg, D. Kulic, and J. Hoey. X-factor hmms for detecting falls in the absence of fall-specific training data. In *Ambient Assisted Living and Daily Activities - 6th International Work-Conference, IWAAL 2014, Belfast, UK, December 2-5, 2014. Proceedings*, pages 1–9, 2014.

[18] K. Li, H. Jiang, L. T. Yang, and A. Cuzzocrea, editors. *Big Data - Algorithms, Analytics, and Applications*. Chapman and Hall/CRC, 2015.

[19] S. Z. Li and A. K. Jain, editors. *Encyclopedia of Biometrics*. Springer US, 2009.

[20] S. Lühr, S. Venkatesh, G. A. W. West, and H. H. Bui. Explicit state duration HMM for abnormality detection in sequences of human activity. In *PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, August 9-13, 2004, Proceedings*, pages 983–984, 2004.

[21] Z. M., W. S., C. Y., C. Z., and Z. Z. An activity transition based fall detection model on mobile devices. *In: Park J., Jin Q., Sang-soo Yeo M., Hu B. (eds) Human Centric Technology and Service in Smart Space. Lecture Notes in Electrical Engineering, vol 182. Springer, Dordrecht*, 2012.

[22] D. Micucci, M. Mobilio, P. Napoletano, and F. Tisato. Falls as anomalies? an experimental evaluation using smartphone accelerometer data. *Journal of Ambient Intelligence and Humanized Computing*, pages 87–99, 2017.

[23] V. Mirchevska, M. Lustrek, and M. Gams. Combining domain knowledge and machine learning for robust fall detection. *Expert Systems*, 31(2):163–175, 2014.

[24] N. Noury, P. Rumeau, A. Bourke, G. Laighin, and J. Lundy. A proposal for the classification and evaluation of fall detectors. *IRBM*, 29(6):340 – 349, 2008.

[25] T. Shi, X. Sun, Z. Xia, L. Chen, and J. Liu. Fall detection algorithm based on triaxial accelerometer and magnetometer, vol. 24, no.2, pp157-163, 201. *Engineering Letters*, 24(2):157–163, 2016.

[26] A. Sucerquia and J. D. Lpez. Sisfall: A fall and movement dataset. *Sensors*, pages 1–14, 2016.

[27] S. Wu and Z. Wang. Applying online feature selection for fall detection. In *IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 1–5, 2016.

[28] Z. Wu, W. Yin, J. Cao, G. Xu, and A. Cuzzocrea. Community detection in multi-relational social networks. In *Web Information Systems Engineering - WISE 2013 - 14th International Conference, Nanjing, China, October 13-15, 2013, Proceedings, Part II*, pages 43–56, 2013.

[29] C. Yang, J. Liu, C. Hsu, and W. Chou. On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism. *The Journal of Supercomputing*, 69(3):1103–1122, 2014.