Hips Do Lie! A Position-Aware Mobile Fall Detection System

Christian Krupitzer*, Timo Sztyler†, Janick Edinger*, Martin Breitbach*,
Heiner Stuckenschmidt†, and Christian Becker*

*Chair of Information Systems II,
University of Mannheim, Germany
Email: {christian.krupitzer, janick.edinger,
martin.breitbach, christian.becker}@uni-mannheim.de

†Data and Web Science Group,
University of Mannheim, Germany
Email: {timo, heiner}@informatik.uni-mannheim.de

***Abstract*—Ambient Assisted Living using mobile device sensors is an active area of research in pervasive computing. Multiple approaches have shown that wearable sensors perform very well and distinguish falls reliably from *Activities of Daily Living*. However, these systems are tested in a controlled environment and are optimized for a given set of sensor types, sensor positions, and subjects. In this work, we propose a self-adaptive pervasive fall detection approach that is robust to the heterogeneity of real life situations. Therefore, we combine sensor data of four publicly available datasets, covering about 100 subjects, 5 devices, and 3 sensor placements. In a comprehensive evaluation, we show that our system is not only robust regarding the different dimensions of heterogeneity, but also adapts autonomously to spontaneous changes in the sensor's position at runtime.**

I. Introduction

Pervasive fall detection in the domain of Ambient Assisted Living is an active field of research in pervasive computing [1]–[5]. Indeed, falls are a major reason for serious injuries of elderly people and also have critical consequences in terms of an independent life [6]. Especially the absence of help and the remaining on the ground lead to difficult-to-treat long-term effects. The loss of self-confidence and the change in behavior to prevent falls can cause a physical as well as a psychological decline in health which in turn results in a premature death [7]. Inertial sensors which are embedded in smart devices allow to recognize critical falls in an unobtrusive way. Existing work already provides evidence concerning the feasibility and reliability of such approaches (e.g., [4], [8], [9]). However, a problem of many existing approaches is that they rely on simplifying assumptions, e.g., that the device is always attached to the same sensor position or that a single algorithm works smoothly in all situations [10]. These assumptions limit their applicability in real world scenarios. In particular, several researchers investigated and presented fall detection approaches but most of them are optimized to a specific dataset or restricted to a specific setup [11]–[13], e.g., at a specific position such as wrist-worn or hip-attached.

The detection of falls with inertial sensors can use a threshold-based algorithm or a machine learning-based approach [1]. Researchers state that the choice depends on the sensor position of the device but also on the subject's physique as a varying height, weight, or age results in different movement patterns [14]–[16]. For this reason, acceleration and gyration patterns may differ significantly for the same activity which entails a different interpretation of these sensor signals.

In this paper, we present a framework that automatically applies the most suitable fall detection algorithm. In this context, we use the current sensor position but also physical characteristics of the subject for selecting the best performing setup. As a subject-specific model is unfeasible in our scenario, we follow a leave-one-subject-out approach [16]. Subsequently, the model is personalized at runtime. For this purpose, we use the FESAS framework [17] that enables to build the required self-adaptive system, i.e., it exchanges the required algorithms and modules on demand to ensure a high reliability. In this context, we also rely on the self-improving layer [18], an architectural extension which supports online learning, i.e., learning of new system configurations. We aim to clarify the difference in performance but also the feasibility compared to non-adaptive systems. The contributions of our work are the following:

- We present a self-adaptive fall detection system that is able to select the most suitable configuration in respect of the environment.
- We show that the presented approach surpasses non-adaptive systems in a real world setting.
- We perform comprehensive experiments regarding different fall detection algorithms in respect of the device's sensor position.

This paper is structured as follows: In Section II, we discuss existing fall detection systems. Section III describes our system architecture, i.e., the self-adaptive system. In Section IV, we explain the corresponding implementation in detail. Subsequently, Section V outlines our experimental results and Section VI clarifies the benefits of autonomously adapting of the fall detection algorithm. Finally, Section VII covers the conclusion and future work.

II. Existing Fall Detection Systems

Mubashir *et al.* [2] proposed a tripartite classification of fall detection systems in (i) wearables, (ii) ambient-based, and (iii) vision-based. Wearables include different types of devices with sensors that are attached to a human's body. Ambient-based approaches integrate different sensors into the environment for detecting falls using vibration, video, and audio signals. Vision-based fall detection systems combine various algorithms to detect falls in video streams. Compared to vision-based and ambient-based approaches, wearables for fall detection have several advantages [2]. First, the components for wearable fall detection systems are relatively cheap. Second, they are mobile and can be used indoor and outdoor as they are attached to the user. Third, they are less intrusive and, hence, easier to set up. Fourth, they better protect the user's privacy compared to systems of the other two categories that monitor the user's environment. Due to this flexibility, we focus on the class of wearable sensors. In the following, we present an overview over the positions of wearables as well as algorithms for fall detection that are present in literature.

A. Positions of Wearables for Fall Detection

A wearable fall detection system depends on sensor measurements to distinguish falls and Activities of Daily Living (ADL) [19]. The most common sensors used in various fall detection systems are accelerometer, gyroscope, magnetometer, and inclinometer [3], [20]. According to [3], the majority of wearables for fall detection are attached to the chest, waist, or thigh. Alternatively, other wearable fall detection systems use placements at the forehead, ear, neck, shoulder, back, wrist, ankle, or foot. Extremities such as arms are involved in nearly every movement and a device worn at the wrist is therefore very active. In contrast, a waist-worn sensor stays steady most of the time. Consequently, most wearable fall detectors are attached to the torso [3], [8]. Figure 1 shows the placements of the related approaches.

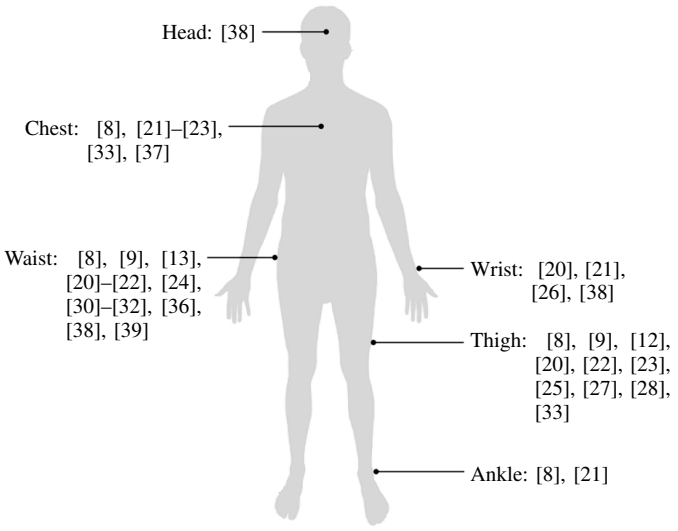

Figure 1. Overview of the positions of wearable-based fall detection approaches in the literature.

B. Fall Detection Algorithms

The fall detection process uses two approaches to distinguish between falls and ADLs [1]: (i) threshold-based and (ii) machine learning-based. The threshold-based approaches for fall detection are comparatively simple algorithms that recognize a fall whenever input values exceed predefined thresholds. Sometimes, the thresholds are integrated into a set of rules [3]. In this work, we do not distinguish between rule-based and threshold-based approaches as in our opinion both use rules that define thresholds for one or various measured variables. Fall detection algorithms rely on different sensors, mainly accelerometer or gyroscope [1]–[3]. Different authors propose approaches with smartphones, e.g., [21], [22]. The preceding approaches only use one type of a sensor: accelerometer or gyroscope. However, some authors incorporate combinations of sensors, e.g., [13], [23], or [24]. Detailed reviews on algorithms for threshold-based fall detection can be found in [3], [4], and [25].

In contrast, machine learning approaches based on pattern recognition are more complex and sophisticated compared to threshold-based approaches [1]. In literature, different machine learning procedures can be found for fall detection. For example, Gjoreski *et al.* compared Naive Bayes, Support Vector Machine (SVM), J48 decision trees, and Random Forest. They show that the Random Forest performed best for post-fall body posture recognition [8]. However, other authors also proposed to use SVM [26]–[29], Artificial Neural Network (ANN) [30]–[32], k-NN [11], [12], [27], Decision Trees [9], [33], [34], Naive Bayes [35], Hidden Markov Model [36], or Fuzzy Frequent Pattern Mining [37].

Both, threshold-based and machine learning approaches for fall detection, suffer from overfitting to a (sometimes rather small) dataset as comparisons of datasets in [11] and [12] have shown. Hence, they might achieve almost perfect accuracy for their test data but might fail for other test subjects. Additionally, as we focus on wearables, the user might change the position of the device, which influences the choice of an algorithm as the motion patterns are changed.

In this paper, we propose a self-adaptive system that is able to adapt the algorithm for fall detection on the fly. This includes determining the current sensor position based on the user's movement pattern. Whereas the fact that a sensor's position is an important design consideration for a fall detection algorithm is acknowledged in research, to the best of our knowledge, there is no approach that adapts the fall detection algorithm to the sensor's position. However, this work should neither be a comparison of algorithms for fall detection nor claim to integrate all available algorithms for fall detection. Rather, it offers a flexible framework for adapting the fall detection procedure. For comparisons of algorithms (cf. [8], [11]–[13], [23], [27], [38], [39]) or overviews of existing algorithms (cf. [1], [3], [4], [13]) the reader is refereed to the literature.

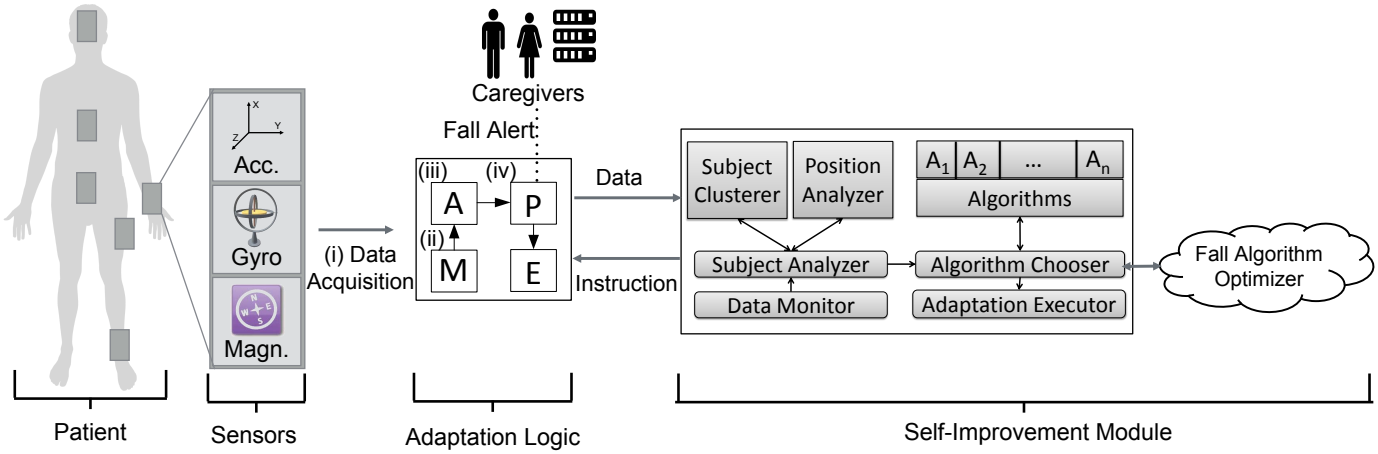

Figure 2. Design of the self-adaptive fall detection system. The Fall Algorithm Optimizer is a central, single component that is connected to several fall detection instances, each composed of sensors, adaptation logic, and self-improvement module. The adaptation logic's MAPE elements represent the (ii) data preprocessing, (iii) fall detection, and (iv) fall alert.

III. Self-adaptive Fall Detection

Two classes of wearables can be identified for fall detection: (i) dedicated devices and (ii) smartphones. Both have in common that their algorithms are designed for a specific position and optimized for a set of test subjects. However, this might not be realistic as such wearables can be worn at different positions or by different people. Hence, there is potential to improve the performance of the fall detection system and its algorithms at runtime. For this purpose, we designed our system as a self-adaptive system.

Next, we explain the concepts of self-adaptive software systems. After, we present the system design for (i) our self-adaptive fall detection system as well as (ii) the self-improvement module that detects position changes and adapts the fall detection algorithm accordingly. Figure 2 shows the design of our system for self-adaptive fall detection.

A. Self-adaptive Systems in a Nutshell

Self-adaptive systems enable adaptation of software at runtime as reaction to changes in their resources or in their environment [40]. Therefore, such systems integrate an adaptation logic that controls the managed resources. Within the adaptation logic, the MAPE control cycle **M**onitors the managed resources as well as their environment through sensors, **A**nalyzes whether the current system performance could be increased through adaptation, **P**lans the change of system parameters, the system structure, or context-adaptation, and **E**xecutes the adaptation on the managed resources [41]. Mapping to fall detection, the MAPE components capture the data (M), analyze whether a fall happened (A), and plan the reaction (P), e.g., informing caregivers in case of a fall (E). Established concepts to reason about adaptations use models, rules, goals, or utility functions in the MAPE cycle [40].

However, uncertainty at runtime can lead to incompleteness of goals, rules, or models as well as non-optimized utility functions. Self-improvement is *the adjustment of the adaptation logic to handle former unknown circumstances or changes in the environment or the managed resources* [18]. This addresses the aforementioned issues. The self-improvement module in our fall detection system detects a position change of the wearable device and adapts the fall detection algorithm accordingly. In the following, we explain the adaptation logic and the self-improvement module of our fall detection system.

B. Fall Detection System

Nearly every fall detection system uses acceleration-based information as these signals are the most reliable information that can be used to detect a fall. Some authors add other sensor types such as a gyroscope to complement the acceleration data. The position of the sensors influences the patterns of this data. Reliability, usability, and acceptability are all strongly influenced by the device placement on the patient's body [38]. Different works evaluated that a device placement close to the body's center of gravity provides the most reliable sensor measurements [3], [8]. As a smartphone offers the required sensors and can be worn near to the body's center, many authors propose a smartphone-based solution to fall detection. However, smartphone-based fall detection solutions still have some drawbacks: (i) it is not always worn by the user (especially, when she is at home), (ii) if it is in the pocket then there is some disturbing movement, and (iii) it can not be worn everywhere, e.g., in the shower. A smartwatch, an often proposed alternative, is worn all the time. However, it has the disadvantage that arms are the most moving body part which influences the fall detection algorithm's quality. Therefore, other authors rely on dedicated devices for fall detection. Our system model is flexible enough to support both alternatives. Casilari *et al.* identified that wearing up to three devices is more accurate, however, more might decrease sensitivity [21]. As wearing multiple devices is not practical for daily life purposes, we rely on a single device in this paper.

The literature specifies a sequence of actions for fall detection [2], [42]: (i) Data acquisition, (ii) Data preprocessing, (iii) Fall detection, and (iv) Fall alert. Our system supports all of

these actions. In the following, we will explain them in more detail. These activities are highlighted in Figure 2.

The sensors are seen as managed resources in the system model. The adaptation logic collects the raw data from the sensors (cf. (i) data acquisition). Within the adaptation logic's monitor, the data is aggregated into windows and features are extracted (cf. (ii) data preprocessing). As next step in the adaptation logic, the analyzer is triggered. Using the aggregated features and specified algorithm, the analyzer divides ADLs from falls (cf. (iii) fall detection). In case of a detected fall, the planner decides corresponding actions – e.g., an acoustic signal or a notification of caregivers – and the executor triggers these actions (cf. (iv) fall alert). Since falls occur indoors as well as outdoors, it is often not sufficient to only inform caregivers about what happened but also where and when it happened. Therefore, a fall alert message should always contain the patient's location and time information. An intermediary device sends the fall alert message. A smartphone perfectly suits as intermediary as it is equipped with a GPS module and integrates wireless communication technology such as Bluetooth or Wi-Fi to connect to the wearable fall detector. Further, it can communicate with caregivers.

C. Adding Self-Improvement

As shown in Figure 2, we add a module for self-improvement to the system. There are two main purposes for the self-improvement module. First, the position of wearables for fall detection might change over time. This changes the pattern of the movement data. As a result, it might be possible, that the fall detection algorithm's performance is decreased as these algorithms are optimized for a specific data pattern. The self-improvement module's *Subject Analyzer* is responsible for detecting the current sensor position for a subject. Therefore, a subject's physiological characteristics – which can influence her movement patterns – are taken into account. Second, after detecting the current sensor position, the most suitable algorithm is chosen and its parameters are optimized. This is the responsibility of the self-improvement module's planner - the *Algorithm Chooser*. Therefore, the Algorithm Chooser integrates rules that specify the best algorithm depending on the sensor position and the characteristics of the person who wears the fall detection device and returns an algorithm that improves the system performance for the new position of the device. On the one hand, a metric for optimization can be to reduce the likelihood of recognizing non-falls as falls, i.e., false alarms. On the other hand it is also possible to accept false alarms to a certain degree to increase the number of correctly recognized falls.

Additionally, the *Fall Algorithm Optimizer* can optimize the parameters of algorithms – e.g., improves the thresholds depending on the captured data – or learns new rules for the matching of the sensor's position and subject category to fall detection algorithms. The Fall Algorithm Optimizer is offered as a web service in the cloud. As several self-improvement modules can use the service, it relies on a large database.

IV. Implementation

The previous section presented the design of our approach for a self-adaptive fall detection system. The system is separated into three parts: (i) sensors send data from accelerometers to (ii) the adaptation logic that integrates the fall detection which might be adjusted by (iii) the self-improvement module. We used the FESAS framework for implementing the adaptation logic as well as the self-improvement module. The FESAS framework offers support for developers of self-adaptive systems [17].

In this section, we present the implementation of our self-adaptive fall detection system. The implementation of the Fall Algorithm Optimizer is excluded for this paper.

A. Fall Detection

The adaptation logic's approach is reusable for other fall detection devices. It only relies on a stream of raw sensor data of the x-, y- and z-axis. We defined interfaces for the interaction between adaptation logic and the sensors as well as the self-improvement module, so that other systems might be customized to be plugged into our approach for adaptation of the fall detection algorithm. As fall detection algorithms, we implemented (i) two threshold-based algorithms from [13] and (ii) machine learning algorithms based on [32], [28], and [33] using the WEKA machine learning framework [43]. Additionally, we considered different configurations of SVM, k-NN, Random Forest, and J48 decision trees for comparison.

The algorithms rely on time windows with feature vectors rather than the unprocessed raw data. The essential idea behind generating windows from a time-dependent data stream is to compute feature vectors that represent the performed activity in a more general way. Feature vectors aggregate the raw data into specific metrics, e.g., the mean, correlation coefficient, or energy value. The window size usually depends on the kind of activities which should be recognized. Considering existing works, a size between one and three seconds is most common [44]. However, there is no agreed set of features but reviewing existing works highlights a common base in context of time and frequency domain features. Following these conclusions, we use windows which overlap by half and have a length of one second. We consider the most common time- and frequency-based features that were used in the domain (see Table I). Time-based feature values are transformed into frequency-based ones by applying Discrete Fourier transform.

Table I
Summary of considered feature methods.

Time
Correlation coefficient (Pearson), entropy (Shannon), gravity (roll, pitch), mean, mean absolute deviation, interquartile range (type R-5), kurtosis, median, standard deviation (SD), variance, horizontal SD magnitude, sum vector magnitude, acceleration range
Frequency
Energy (Fourier, Parseval)

Especially gravity-based features are promising as usually a device has an acceleration of $\sim 9.81 \frac{m}{s^2}$ due to the gravitational field. However, this changes if the device is in free fall. For that reason, we separate the acceleration and gravitational force with a low-pass filter to derive the gravity vector. The implementation of the window manager and the feature extraction is based on [10].

B. Self-Improvement Module

The implementation of the self-improvement module is based on recent work from [18]. As described in the previous section, the self-improvement module detects the current sensor position and adapts the fall detection algorithm accordingly. Therefore, the adaptation logic regularly sends the generated windows to the self-improvement module's monitor.

As explained in Section III, the analyzer is designed for modularity: It can incorporate different analyzing modules for different reasoning purposes that might be triggered simultaneously or sequentially. We implemented two sub-modules: the Subject Clusterer and the Position Analyzer. First, the *Subject Clusterer* groups the subject into a subject cluster. Therefore, we used the data of the subjects to build a clustering model based on the subjects' age and Body Mass Index (BMI) using X-Means. Second, the *Position Analyzer* detects the current sensor position. We treat position detection as a multi-class classification problem with the target classes chest, waist, and thigh. In this context, we trained a subject-independent classifier, hence, we considered all available training data expect the target subject to train a classification model that derives the sensor's position. For avoiding oscillations due to a single wrong classification, a position change is only considered after having detected the new position twice.

If the Position Analyzer detects that the sensor position for capturing data was changed, it triggers the Algorithm Chooser. We implemented the rule-based Algorithm Chooser using the rule engine *EasyRules*[1]. Rules are defined in a spreadsheet which is processed by the *Apache POI* library[2]. We used the implemented algorithms and fall detection datasets to analyze the best algorithm for a specific sensor position. Section V presents the results of the analysis.

V. Evaluation

We evaluated our approach across multiple datasets, subjects, sensor positions, and device types to demonstrate that our self-adaptive fall detection system can deal with heterogeneity. Therefore, we conducted a series of experiments, using public available data to make our results comparable and reproducible. The outline for the experiments is as follows:

- Experiment 1: We tested how well different fall detection algorithms perform for different datasets and evaluated whether the results can be generalized.
- Experiment 2: Based on our findings from Experiment 1, we tested whether the performance of the algorithms depends on the sensor position on the subjects' body. We introduced self-adaptation to autonomously select the best configuration for this position.
- Experiment 3: To test the applicability under real world conditions, we created routines for subjects from different datasets and ran our system on unmodified raw data. Therefore, we changed the sensor position at runtime and evaluated how well our system adapts to these changes.

In the following, we provide aggregated results for each experiment. Individual results for each experiment and the preprocessed data are publicly available[3].

A. Experiment Setup

For the experiments, we used multiple publicly available fall datasets. These datasets were recorded, labeled, and provided by different authors from different institutions. Thus, they are neither standardized in *what* they include nor *how* the data is recorded and stored. This results in a great variance regarding the content, the extent, as well as the quality of the data.

Among all available datasets we selected those with labeled data for falls and ADL. We excluded datasets with missing information about the subjects as well as datasets that were significantly smaller than others. Table II shows the datasets.

Table II
Datsets used for the evaluation

Dataset	Device Type	Position	Frequency	Subjects
MMSys [23]	SensorTag	Chest, Thigh	100 Hz	32
UMA [21]	SensorTag Smartphone	Chest, Waist Thigh	20 Hz 200 Hz	10
UniMiB SHAR [12]	Smartphone	Thigh	50 Hz	30
SisFall [13]	Self-built	Waist	200 Hz	23

From these datasets, we excluded 11 subjects because they did not perform falls, 4 subjects because of missing data, and 2 subjects as they contained significantly less data. We used accelerometer raw data. Further, we excluded wrist and ankle data, as these positions were only available in one dataset.

For each dataset we segmented the raw data in windows and computed feature vectors based on three-dimensional acceleration data (cf. Table I). We manually inspected the feature vectors and compared them among each other. It became apparent that these feature vectors show a large variance across the datasets but are rather homogeneous within one dataset. This turns out to be a major challenge when working with multiple datasets (cf. [11], [12]). Differences during the process of recording and preprocessing the data reduce their comparability. Further, subjects from different datasets performed different activities. As the datasets vary in size, we randomly selected 500 windows from each subject. The ratio of fall windows to non-fall windows is $20 : 80$. In the following experiments, we use these windows to compile test and training sets for various machine learning algorithms.

[1] https://github.com/j-easy/easy-rules

[2] https://poi.apache.org/

[3] A complete overview of all results and further F-measures can be found here: https://sensor.informatik.uni-mannheim.de#results2018hips

Table III
F_2-MEASURES OF THE APPROACHES FOR EACH DATASET USING LEAVE-ONE-SUBJECT-OUT. GREY CELLS INDICATE A DIFFERENCE LARGER THAN 0.1 BETWEEN SETTINGS.

Case		SVM	ANN	RF	k-NN	J48	TB
Fall	MMSys	0.55	0.73	0.80	0.63	0.78	0.68
	UMA	0.43	0.71	0.77	0.66	0.72	0.10
	UniMiB SHAR	0.34	0.45	0.64	0.60	0.61	0.20
	SisFall	0.50	0.54	0.75	0.65	0.70	0.62
ADL	MMSys	0.94	0.96	0.96	0.95	0.95	0.89
	UMA	0.84	0.90	0.93	0.92	0.93	0.35
	UniMiB SHAR	0.76	0.89	0.90	0.86	0.87	0.46
	SisFall	0.93	0.93	0.92	0.88	0.91	0.83
avg.		0.66	0.77	0.83	0.77	0.81	0.52

Table IV
F_2-MEASURES OF THE APPROACHES **CROSS** DATASETS USING LEAVE-ONE-SUBJECT-OUT. GREY CELLS INDICATE A DIFFERENCE LARGER THAN 0.1 BETWEEN SETTINGS.

Case		SVM	ANN	RF	k-NN	J48	TB
Fall	MMSys	0.30	0.52	0.80	0.62	0.77	0.21
	UMA	0.44	0.68	0.76	0.65	0.73	0.22
	UniMiB SHAR	0.41	0.63	0.65	0.59	0.62	0.28
	SisFall	0.33	0.48	0.74	0.66	0.68	0.24
ADL	MMSys	0.89	0.92	0.96	0.93	0.95	0.88
	UMA	0.87	0.92	0.94	0.91	0.93	0.78
	UniMiB SHAR	0.84	0.86	0.89	0.86	0.87	0.69
	SisFall	0.83	0.85	0.91	0.88	0.89	0.81
avg.		0.61	0.74	0.83	0.76	0.81	0.51

We kept the amount of preprocessing as low as possible. As some fall data was only coarsely labeled, we needed to adjust the labels manually. Therefore, we used the energy of a window as measure and defined each window as a non-fall when the sum of the three energy values was below 10, which is comparable to standing or lying.

B. Experiment 1: Cross-Datasets Fall Detection

In the first experiment, we evaluated the performance of multiple machine learning algorithms for fall detection. We focused on a *leave-one-subject-out* approach where we, first, investigated each dataset independently. This means that we only trained the models on data of $n-1$ subjects of one dataset and tested the model on the remaining subject. Subsequently, we also used data from the other datasets as training set to see how well the models perform across datasets. For both settings, we trained a single classifier for each subject.

Table III illustrates that the fall recognition remains stable for a certain classifier across the datasets. However, it strikes that especially for *UniMiB SHAR* falls are recognized with a lower accuracy (about -10 %). Preliminary experiments showed that some falls in a certain dataset had a significant lower intensity, i.e., were more similar to common activities concerning the acceleration patterns. Overall, the Random Forest classifier performed best (Falls, 75%) but the results also show that fall detection should not be underestimated with respect to the complexity. In this context, the results show that existing approaches are optimized for particular datasets but fail in classifying falls of another dataset.

In the second evaluation, we performed the comparison of algorithms across datasets. This has two effects. First, it increases the heterogeneity of the data as data from different subjects, sensors, and sensor positions are used. Second, it increases the size of the training set that the classifier is trained on. The results in Table IV show that the performance of the classifier decreases when data from multiple datasets is used in the training phase. This supports our assumption that a single classifier seems not to scale across a larger user base, i.e., *leave-one-subject-out* seems not to be an acceptable solution when dealing with heterogeneous data. Further, the results of both experiments (dataset independent and cross-datasets), indicate that Random Forest outperforms the other machine learning approaches. Thus, in the following, we focus on determining the factors that have an impact on the performance of the algorithms.

Table V
PERFORMANCE OF A POSITION-INDEPENDENT APPROACH CONCERNING RECOGNIZING A FALL FOR A CERTAIN POSITION.

Case		Precision	Recall	F_2-Measure
Fall	Chest	0.79	0.82	0.81
	Thigh	0.73	0.72	0.72
	Waist	0.80	0.73	0.75
avg.		0.76	0.75	0.76
ADL	Chest	0.96	0.95	0.96
	Thigh	0.92	0.92	0.92
	Waist	0.92	0.94	0.93
avg.		0.93	0.93	0.94

C. Experiment 2: Position-Aware Fall Detection

In Experiment 1, we neglected the fact that the algorithms were trained and tested on data of different sensor positions at the same time. However, users can wear the sensors at different positions on their body. Positions near the body axis are rather stable compared to positions on the side or limbs (e.g., hips and wrist, respectively). These show more movement, i.e., they differ in their movement patterns.

Thus, we tested a model trained on all $n-1$ subjects across all datasets on different sensor positions. Table V shows the performance of the Random Forest classifier on the three sensor positions: chest, waist, and thigh. The results indicate that the classifier performs better results for chest and waist – a first hint that hips (respectively thighs) are not the most reliable position for fall detection sensors.

Further, it supports our claim that less stable positions lead to disturbing movements of the sensors resulting in a decreasing fall detection rate. In fact, related work claimed this already in the context of physical activities [45]. We state that this is also applicable in the context of fall detection.

As the previous evaluation showed that position is an important dimension to take into account, we decided to perform an evaluation with position-aware classifiers, i.e., we trained a model for each position only with data from that position. With position-aware fall detection we focus on building a classification model for a single sensor position by training only on labeled acceleration data of this specific position. In doing so, we reduce the amount of heterogeneity in the training data and, thus, are able to train more specialized models.

This evaluation assumes that it is possible to perfectly detect the position of the sensor. Table VI outlines (i) the results of the evaluation of the position-aware classifiers and (ii) the performance of these models in case that they are applied on other on-body positions, which would happen when the wrong sensor position was classified. The results show that optimizing classifiers for a position improves the fall detection rate (cf. Table V and Table VI). As we distinguish only between three positions and Fall versus ADL, even this slight improvement ($\sim$ 1-2%) is notable. The *thigh*-aware classifier achieved a significantly lower precision for chest and waist data but also vice versa. This is another support for our claim that sensor data from thighs is less reliable, i.e., hips do lie. Therefore, a distinction between these body areas is beneficial.

So far, we have assumed that the sensor position is given. As this assumption does not hold true in a real world setting, we need to determine the current position at runtime. The *Position Analyzer* (cf. Section IV-B) recognizes the current position using a Random Forest classifier. Table VII shows the results for the position recognition of waist, chest, and thigh sensor data. The best performance is achieved for detecting the thigh. This is not surprising, as the thigh has a different movement pattern than the torso-related waist and chest and, hence, can be easier distinguished. We have conducted the same test with ANN and k-NN algorithms that provided the same results.

D. Experiment 3: Self-Adaptive Fall Detection

In addition to the previous experiments that focused on individual aspects of fall detection algorithms, we evaluated the performance of our overall self-adaptive fall detection system. Therefore, we used as input the raw data of a subset of subjects from the UMA [21] and MMSys [23] datasets and performed the whole fall detection procedure, that is, (i) generate windows and compute feature vectors at runtime, (ii) recognize the sensor position based on these features, (iii) select the best classifier for this position, and (iv) classify the windows into *Falls* and *ADLs*. The datasets represent traces from people with different physiological attributes to show that the system works for a wide range of subjects. For each subject, there exist multiple data streams from different device positions. We manually merged these streams using data from different sensor positions alternately to simulate changes of the sensor position at runtime, e.g., when the subject moves the smartphone from the front chest pocket to the trousers pocket.

We ran two evaluations with the generated traces to show the benefits of our approach. As baseline, we used all fall detection algorithms for each subject without using the self-improvement module, i.e., the fall detection algorithm did not recognize the sensor position and, thus, did not change the classifier for detecting falls. For comparison, we ran the overall process of our system, including the position detection and classifier selection with the Random Forest algorithm. Table VIII shows the results of this comparison. As Random Forest outperformed the other algorithms in all measures, we applied a rule set which adapts the settings of the Random Forest classifier according to the identified position (RF_{adapt}). We want to emphasize the fact that for our rather small and non-optimized set of algorithms, Random Forest was superior. This might be different for other sets of algorithms which results in a need for adjusting the rule base. Our systems provides this possibility and can select the optimal fall detection algorithm at runtime.

For a detailed discussion of our evaluation, Figure 3 shows the results of the overall evaluation of our self-adaptive fall detection system for one single subject. For this subject, only accelerometer data for the two positions chest and thigh is available. The center of the figure shows the three-dimensional raw acceleration data of one subject performing multiple

Table VI
POSITION-AWARE FALL DETECTION ACROSS DATASETS. (M_C STANDS FOR A MODEL TRAINED ON CHEST DATA ONLY.)

		M_C		M_T		M_W	
Case		**P**	**R**	**P**	**R**	**P**	**R**
Fall	**Chest**	0.82	0.82	0.60	0.81	0.79	0.82
	Thigh	0.54	0.69	0.73	0.74	0.43	0.85
	Waist	0.80	0.80	0.58	0.74	0.78	0.74
ADL	**Chest**	0.96	0.96	0.89	0.92	0.94	0.95
	Thigh	0.84	0.88	0.92	0.92	0.66	0.77
	Waist	0.95	0.95	0.86	0.90	0.93	0.92

Table VII
PERFORMANCE OF THE POSITION ANALYZER IN DETECTING THE SENSOR POSITION

Class	**Precision**	**Recall**	**F-Measure**
Chest	0.78	0.73	0.75
Thigh	0.88	0.85	0.87
Waist	0.76	0.87	0.81
avg.	0.81	0.82	0.81

Table VIII
EVALUATION OF DIFFERENT ALGORITHMS IN THE SELF-ADAPTIVE SYSTEM

Case		**ANN**	**RF**	**k-NN**	**J48**	$\textbf{RF}_{\textbf{adapt}}$
Fall	**Precision**	0.73	0.33	0.08	0.31	0.33
	Recall	0.38	0.88	0.80	0.79	0.93
	F_2**-Measure**	0.42	0.66	0.28	0.60	0.68
ADL	**Precision**	0.97	1.00	0.99	0.99	1.00
	Recall	0.99	0.95	0.73	0.95	0.95
	F_2**-Measure**	0.99	0.96	0.77	0.96	0.96

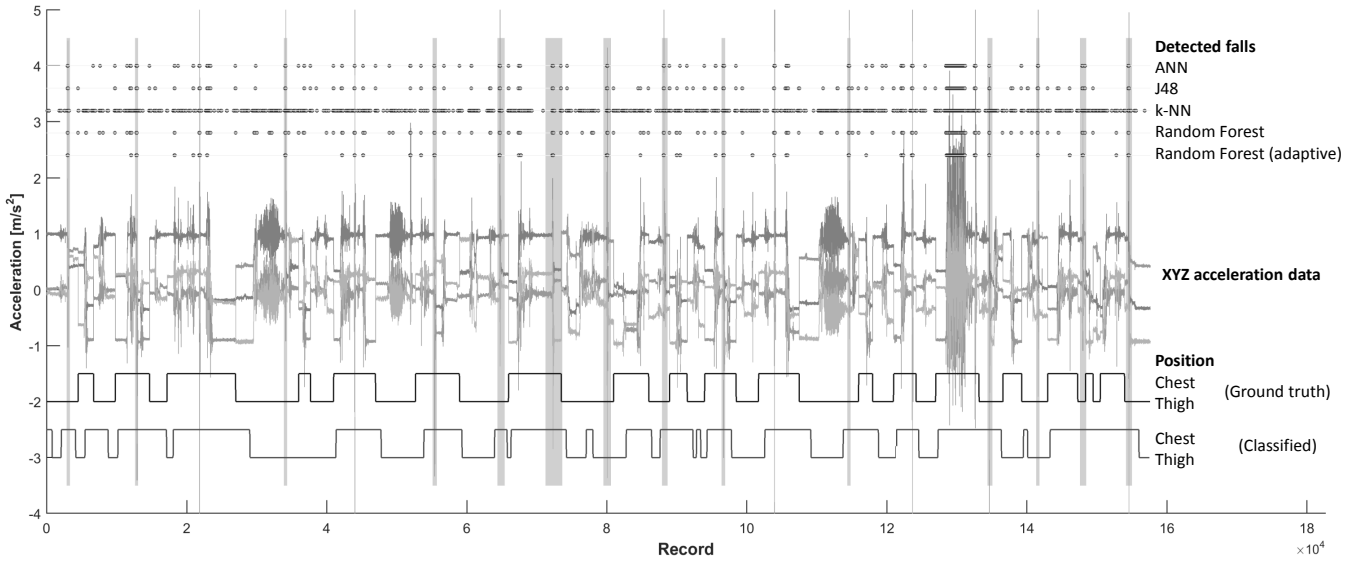

Figure 3. Results of the evaluation of the overall self-adaptive fall detection system. The figure shows the three-dimensional raw acceleration data of one subject performing multiple ADLs and falls. Data records that are labeled as *Fall* are highlighted with gray transparent boxes. Notice that for each fall multiple records are labeled as *Fall.* The marker on the upper third of the plot shows the frames that were detected as *Fall* by the respective fall detection algorithm. The cornered lines below the raw data show the current position of the sensor (*Ground truth*) and the position recognized by our system (*Classified*). The results show that the adaptive fall detection algorithm outperforms the static algorithms and that the position recognition works well on the underlying data. It detects at least one record for each fall event.

ADLs and falls. Records labeled with 'Fall' are highlighted with a transparent gray rectangle. The markers above indicate the records that are classified as 'Fall' by the respective fall detection algorithm. The cornered lines below the raw data show the current position of the sensor ('Ground truth') and the position recognized by our system ('Classified'). False alarms can be detected by slight extensions of the algorithms. One possible solution would be to check whether there is continuous movement after a detected fall. If the subject continues moving in a normal way, the detected fall can be considered a false alarm. Our algorithms try to avoid missed falls and rather allow some false alarms. Table IX shows the results of the evaluation of the position recognizer in the self-adaptive system. The interpretation of these results is twofold: First, the numbers indicate that sensors positioned at a subject's thigh perform worse than sensors at a subject's chest or waist. Second, even though the recall is below 0.7 for all positions, it does not mean that more than 30 percent of falls have not been detected. As one can see in Figure 3, each fall consists of multiple fall windows. If the algorithm evaluates at least one of these fall windows, the fall is detected. This happens for each fall.

Table IX
EVALUATION RESULT FOR THE POSITION RECOGNIZER IN THE SELF-ADAPTIVE SYSTEM.

Class	Precision	Recall	F_2-Measure
Chest	0.66	0.58	0.60
Thigh	0.32	0.27	0.28
Waist	0.56	0.69	0.66
avg.	0.51	0.52	0.51

VI. DISCUSSION

In this paper, we investigated the reliability of fall detection cross publicly available datasets. As shown in the previous sections, this problem is not trivial as several solutions might perform reliably within a certain dataset but not for others. The results show that the sensor position have an essential influence on the fall detection performance. In this context, the publicly available datasets also considered different definitions of a fall which makes it hard to directly compare existing results. A common distinction is whether the subject remains on the ground is part of a fall or not. In this work, we only focused on the transition from standing to lying on the ground.

However, even if the considered datasets are heterogeneous concerning the acceleration patterns, our results show that especially stable sensor positions are suitable. Indeed, the chest was the best performing position, followed by the waist and thigh. However, this is in contrast to the distribution of smart-watches, smart-glasses, and smart-bands which are particularly attached to flexible body parts as the head, arm, and leg. Hence, as the performance decreases with the increase of movement, we point out that this is still an open issue.

As our primary goal was to investigate the different aspects and issues of fall detection, we state that our adaptive system is a suitable approach to overcome the mentioned issues, i.e., to handle the different kinds of upcoming noise but also the variability of the environment. However, there are some potential extensions for optimizing our system performance.

First, we outlined the performance of our approach in a general way. However, especially in our scenario it is neces-

sary to distinguish between precision and recall. Optimizing for precision would not be very sufficient if at the same time the recall value is getting worse as that means that we miss falls. For that reason, we state that recall is more important than precision and use the F_2 measure[3]. Hence, we believe it is acceptable to have false alarms (false positives) if this goes hand in hand with a minimized number of missed falls (false negatives). In this context, Table IX shows an interesting example were the performance of the chest and the waist are equal considering the F_2 measure but precision and recall vary significantly. In such a case, we believe the waist should be preferred as a suitable position as it miss less falls.

Second, the set of algorithms was not optimized. We chose the algorithms to have a subset of common approaches to fall detection. In this context, it was often infeasible to implement algorithms that are present in literature as many authors omit necessary details in their publication, e.g., the window sizes, or definition of features. Furthermore, not all approaches fit to our feature window, e.g., the authors of [12] rely on discrete windows with the raw sensor data instead of time-based windows. Moreover, we also identified proposed approaches during the literature review that rely on Naive Bayes. Preliminary experiments showed that these approaches were unreliable regarding fall detection. We believe that this can be attributed to the fact that Naive Bayes assumes that all features are independent. In contrast, the axis of an accelerometer but also gyroscope are highly dependent and, by implication, also the derived features. This fact highlights the already mentioned issue that some approaches seem to perform well but only in respect of certain conditions (cf. [11], [12]).

Third, we focused only on a single acceleration sensor. Modern smart devices comprise accelerometer, gyroscope, and magnetometer. In multi-sensor settings, sensor fusion techniques are necessary and may also improve the distinction between certain actions. Indeed, this also includes the usage and fusion of several sensors of different devices at different sensor positions such as dedicated wearables or smart-watches. However, authors claim that integrating too many devices might decrease the overall performance [21] as it could lead to over-fitting.

Fourth, recently, Khan *et al.* [46] presented an overview of an alternative approach to fall detection. Rather than classifying the data into falls and ADLs, they propose to focus on outlier detection, i.e., rare or exceptional movement patterns. They argue that fall events are rare and it is difficult to capture fall data in real life situations. Due to the flexibility of our system and the ability to exchange fall detection algorithm, this could be easily integrated in our systems. Hence, a combination of fall/ADL classification with additional outlier detection might give direction to the mentioned issues.

Fifth, the approach for clustering of subjects is an open issue. Several researchers already hypothesized that physical characteristics – i.e., gender, weight, height, and fitness level – could be reliable indicators to choose group subjects for a cross-subjects model [14], [47]. However, focusing on common physical activities, this is still an open issue in respect of applicability and reliability as only few works consider this aspect [16] and to the best of our knowledge none concerning fall detection. In this work, we tried to cluster people first by age and then by their BMI to distinguish their physiologic attributes. The results of a preliminary experiment of the group-based clustering approach were not as expected. Indeed for most clusters the performance is not improved compared to the baseline. On the one hand, we might miss important information, e.g., we were not able to consider the fitness level of our subjects as this information was not available. Related work suggests to use this factor [45]. On the other hand, we focused on a two-step approach where subjects are first clustered by age and subsequently by their BMI value. This resulted in a group of five elderly only, thus, this group was not further clustered. We plan to integrate and test other clustering approaches as the leave-one-subject-out approach does not scale in respect of a larger number of people. An increasing number of people leads to a higher probability of contradictory acceleration patterns for the same action as especially children and elderly move in a different way.

VII. Conclusion

Previous works claimed that fall detection algorithms are often customized to the used datasets [11]–[13]. After analyzing different publicly available datasets for fall detection we can confirm this claim. For this analysis, we compared different fall detection algorithms on publicly available datasets (cf. Experiment 1 in Section V-B). Across all datasets, Random Forest performed best. However, the precision and recall values vary across datasets. Especially heterogeneous labeling approaches for the captured data reduces the performance of non-customized fall detection algorithms on these datasets. Furthermore, the algorithms are often optimized for a sensor position. Therefore, we additionally performed a position-aware comparison of the fall detection algorithms on the datasets (cf. Experiment 2 in Section V-C). The results indicate that knowing the position of the sensor and adjusting the algorithm accordingly is beneficial in contrast to a static algorithm. Experiment 3 (cf. Section V-D) showed that our system tackles these issues by determining the current sensor position and adapting the fall detection algorithm accordingly. Hence, our system is able to tolerate the heterogeneity of the datasets through adaptation of the fall detection algorithm.

In the preliminary experiments with subject clustering, we achieved similar performance as in the baseline tests. However, the results are influenced by the fact that we could only use the age, size, weight, and gender of the subjects neglecting potential relevant factors such as the subject's fitness level. Therefore, we plan to build an own dataset to have full control over all necessary variables. In this work, we focus on fall detection algorithms that use a time-based window with feature vectors. For future work, we plan to analyze further algorithms which rely on discrete windows with the raw sensor data instead of time-based windows, e.g., [12] or [33]. Last, improvements might be achieved by fusion of additional sensors.

REFERENCES

[1] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *BioMedical Engineering OnLine*, vol. 12, no. 1, p. 66, 2013.

[2] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.

[3] N. Pannurat, S. Thiemjarus, and E. Nantajeewarawat, "Automatic fall monitoring: a review," *Sensors*, vol. 14, no. 7, pp. 12 900–36, 2014.

[4] E. Casilari, R. Luque, and M.-J. Morón, "Analysis of Android Device-Based Solutions for Fall Detection." *Sensors*, vol. 15, no. 8, pp. 17 827–17 894, 2015.

[5] C. Y. Hsieh, C. N. Huang, K. C. Liu, W. C. Chu, and C. T. Chan, "A machine learning approach to fall detection algorithm using wearable sensor," in *Proc. ICAMSE*, 2016, pp. 707–710.

[6] D. Wild, U. S. Nayak, and B. Isaacs, "How dangerous are falls in old people at home?" *British medical journal (Clinical research ed.)*, vol. 282, no. 6260, pp. 266–8, 1981.

[7] C. Griffiths, C. Rooney, and A. Brock, "Leading causes of death in England and Wales–how should we group causes?" *Health statistics quarterly / Office for National Statistics*, no. 28, pp. 6–17, 2005.

[8] H. Gjoreski, M. Luštrek, and M. Gams, "Accelerometer placement for posture recognition and fall detection," *Proc. IE*, pp. 47–54, 2011.

[9] B. Aguiar, T. Rocha, J. Silva, and I. Sousa, "Accelerometer-based fall detection for smartphones," *Proc. MeMeA*, pp. 1–6, 2014.

[10] T. Sztyler and H. Stuckenschmidt, "On-body localization of wearable devices: An investigation of position-aware activity recognition," in *Proc. PerCom*, 2016, pp. 1–9.

[11] R. Igual, C. Medrano, and I. Plaza, "A comparison of public datasets for acceleration-based fall detection," *Medical Engineering & Physics*, vol. 37, no. 9, pp. 870 – 878, 2015.

[12] D. Micucci, M. Mobilio, and P. Napoletano, "Unimib SHAR: a new dataset for human activity recognition using acceleration data from smartphones," *CoRR*, vol. abs/1611.07688, 2016.

[13] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, "SisFall: A Fall and Movement Dataset," *Sensors*, vol. 17, no. 1, 2017.

[14] G. M. Weiss and J. W. Lockhart, "The impact of personalization on smartphone-based activity recognition," in *Proc. AAAI Workshops*, 2012, pp. 98–104.

[15] D. Coskun, O. D. Incel, and A. Ozgovde, "Phone position/placement detection using accelerometer: Impact on activity recognition," in *Proc. ISSNIP*, 2015, pp. 1–6.

[16] T. Sztyler, H. Stuckenschmidt, and W. Petrich, "Position-aware activity recognition with wearable devices," *Pervasive and Mobile Computing*, vol. 38, pp. 281–295, 2017.

[17] C. Krupitzer, F. M. Roth, C. Becker, M. Weckesser, M. Lochau, and A. Schürr, "FESAS IDE: An Integrated Development Environment for Autonomic Computing," in *Proc. ICAC*, 2016, pp. 15–24.

[18] C. Krupitzer, J. Otto, F. M. Roth, A. Frommgen, and C. Becker, "Adding Self-Improvement to an Autonomic Traffic Management System," in *Proc. ICAC*. IEEE, 2017, pp. 209–214.

[19] N. El-Bendary, Q. Tan, F. C. Pivot, and A. Lam, "Fall detection and prevention for the elderly: A review of trends and challenges," *International Journal on Smart Sensing and Intelligent Systems*, vol. 6, pp. 1230–1266, 2013.

[20] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, no. 6, pp. 10 146–10 176, 2014.

[21] E. Casilari, J. A. Santoyo-Ramón, and J. M. Cano-García, "Analysis of a smartphone-based architecture with multiple mobility sensors for fall detection," *PLOS ONE*, vol. 11, no. 12, pp. 1–17, 12 2016.

[22] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan, "Mobile phone-based pervasive fall detection," *Personal and ubiquitous computing*, vol. 14, pp. 633–643, 2010.

[23] O. Ojetola, E. Gaura, and J. Brusey, "Data set for fall events and daily activities from inertial sensors," in *Proc. MMSys*, 2015, pp. 243–248.

[24] F. Bianchi, S. J. Redmond, M. R. Narayanan, S. Cerutti, and N. H. Lovell, "Barometric pressure and triaxial accelerometry-based falls event detection," *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 6, pp. 619–627, 2010.

[25] H. A. Dau, F. D. Salim, A. Song, L. Hedin, and M. Hamilton, "Phone based fall detection by genetic programming," in *Proc. MUM*. ACM, 2014, pp. 256–257.

[26] C. Doukas, I. Maglogiannis, F. Tragkas, D. Liapis, and G. Yovanof, "Patient Fall Detection using Support Vector Machines," *Int Fed Inf Process*, vol. 247, pp. 147–156, 2007.

[27] C. Medrano, R. Igual, I. Plaza, and M. Castro, "Detecting falls as novelties in acceleration patterns acquired with smartphones," *PLOS ONE*, vol. 9, no. 4, pp. 1–9, 2014.

[28] L.-J. Kau and C.-S. Chen, "A smart phone-based pocket fall accident detection, positioning, and rescue system." *IEEE Journal of Biomedical and Health Informatics*, vol. 19, pp. 44–56, 2015.

[29] N. Shibuya, B. T. Nukala, A. Rodriguez, J. Tsay, T. Nguyen, S. Zupancic, and D. Yu-Chun Lie, "A Real-Time Fall Detection System Using a Wearable Wireless Gait Analysis Sensor and a Support Vector Machine (SVM) Classifier," *Proc. ICMU*, pp. 66–67, 2015.

[30] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini, and A. Vecchio, "A smartphone-based fall detection system," *Pervasive and Mobile Computing*, vol. 8, no. 6, pp. 883–899, 2012.

[31] H. Kerdegari, K. Samsudin, A. R. Ramli, and S. Mokaram, "Evaluation of fall detection classification approaches," *Proc. ESTCON*, vol. 1, pp. 131–136, 2012.

[32] B. T. Nukala, N. Shibuya, A. Rodriguez, J. Tsay, J. Lopez, T. Nguyen, S. Zupancic, and D. Yu-Chun Lie, "An Efficient and Robust Fall Detection System Using Wireless Gait Analysis Sensor with Artificial Neural Network (ANN) and Support Vector Machine (SVM) Algorithms," *Open Journal of Applied Biosensor*, vol. 3, pp. 29–39, 2014.

[33] O. Ojetola, E. I. Gaura, and J. Brusey, "Fall detection with wearable sensors - Safe (SmArt Fall dEtection)," *Proc. IE*, pp. 318–321, 2011.

[34] A. F. P.N., V. Viet, and C. Deok-Jai, "Semi-supervised fall detection algorithm using fall indicators in smartphone," *Proc. IMCOM*, pp. 1221–1229, 2012.

[35] Y. Choi, a. S. Ralhan, and S. Ko, "A study on machine learning algorithms for fall detection and movement classification," *International Conference on Information Science and Applications*, pp. 1–8, 2011.

[36] J. Wang, R. Chen, X. Sun, M. F. She, and Y. Wu, "Recognizing Human Daily Activities From Accelerometer Signal," *Procedia Engineering*, vol. 15, pp. 1780–1786, 2011.

[37] J. Surana, C. Hemalatha, V.Vaidehi, S. Palavesam, and M. Khan, "Adaptive learning based human activity and fall detection using fuzzy frequent pattern mining," *Proc. ICRTIT*, pp. 744–749, 2013.

[38] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jämsä, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait and Posture*, vol. 28, pp. 285–291, 2008.

[39] F. Bagalà, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, "Evaluation of accelerometer-based fall detection algorithms on real-world falls," *PLOS ONE*, vol. 7, no. 5, pp. 1–9, 05 2012.

[40] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive and Mobile Computing Journal*, vol. 17, no. Part B, pp. 184–206, 2015.

[41] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[42] X. Yu, "Approaches and principles of fall detection for elderly and patient," in *HealthCom 2008 - 10th International Conference on e-health Networking, Applications and Services*, 2008, pp. 42–47.

[43] G. Holmes, A. Donkin, and I. H. Witten, "WEKA: a machine learning workbench," in *Proc. ANZIIS*. IEEE, 1994, pp. 357–361.

[44] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014.

[45] T. Sztyler and H. Stuckenschmidt, "Online personalization of cross-subjects based activity recognition models on wearable devices," in *Proc. PerCom*, 2017, pp. 180–189.

[46] S. Khan and J. Hoey, "Review of fall detection techniques: A data availability perspective," *Medical Engineering & Physics*, vol. 39, pp. 12–22, 2017.

[47] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.