Computing (2026) 108:65
https://doi.org/10.1007/s00607-026-01651-y

REGULAR ARTICLE

A lightweight AI system for real-time elderly fall detection

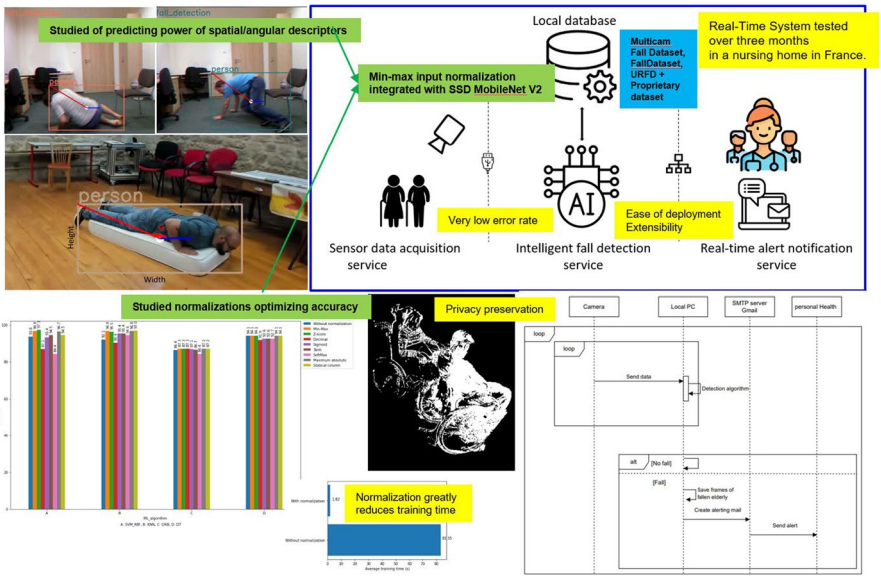
Moustafa Fayad[1] · Mohammed Amine Merzoug[2] · Ahmed Mostefaoui[3] · Ernesto Damiani[4] · Gabriele Gianini[5] · Réda Yahiaoui[1]

Received: 17 June 2025 / Accepted: 18 March 2026 / Published online: 9 April 2026

Abstract Elderly fall detection is a critical healthcare challenge that demands accurate, reliable, and real-time solutions suitable for deployment in sensitive environments, such as nursing homes. Although significant advances have been made in sensing technologies and machine learning, the role of data preprocessing, particularly feature normalization, remains underexplored, despite its substantial impact on model performance. This paper presents a lightweight vision-based fall detection system that has been deployed and evaluated in a real-world elderly care facility, addressing a key limitation of previous work that often lacks real-life validation. Central to the approach is a systematic investigation into the effect of eight normalization techniques on the performance of four representative classification models. The results show that appropriate normalization, particularly Min–Max and Z-score normalization, leads to substantial improvements in classification accuracy and F1 score, while also reducing training time by a factor of 45. Based on these findings, the proposed system incorporates min–max normalization into a hybrid architecture that combines SSD MobileNet V2 with geometric heuristics for real-time fall detection. The system is privacy-aware, non-intrusive, and practical for real-world healthcare deployment.

Extended author information available on the last page of the article

Graphical abstract



Keywords Elderly fall detection · Human activity recognition · Vision-based monitoring · Intelligent health monitoring

Mathematics Subject Classification 68T01

1 Introduction

The global demographic landscape is undergoing a transformation, characterized by an unprecedented surge in the elderly population. According to the WHO [1], the number of vulnerable elderly people is increasing dramatically compared to other age groups, increasing the incidence of chronic diseases, falls, and household accidents with severe consequences. Elderly falls have also been highlighted to lead to increased mortality rates [2, 3]. In particular, reports show that falling is the second most significant cause of unintentional injury mortality [1, 4]: 684,000 people die every year from falls, and those over 60 years of age experience the highest number of fatal falls.

Examples of situations in which elderly people may be isolated and unable to receive immediate assistance after a fall include (i) Private residences, particularly for elderly individuals living alone or in isolated areas with limited social interaction. (ii) Nighttime or during sleeping hours: when family members or caregivers may not be available to provide immediate emergency assistance. (iii) Hospital wards: elderly patients in hospitals can fall while trying to get out of bed without assistance, especially at night when the levels of staff may be lower. (iv) Nursing and retirement homes where elderly people can be at risk due to isolation and limited supervision.

To address this public health issue, the research has focused on developing *fast and reliable* fall detection systems that are capable of ensuring minimal false alarms. Furthermore, a fall detection system must also be *affective-aware*: it has to minimize anxiety through timely alerts, respect privacy to reduce psychological discomfort, and employ a non-intrusive design and devices that foster a sense of dignity and safety. By addressing both physical risk and emotional impact, an affective-aware system contributes to a holistic, human-centered approach to elderly care.

The initial stage of fall detection involves capturing the relevant parameters during the movement of the subject [5]. In the literature, a variety of data sources have been explored, including visual, accelerometer, Wi-Fi signal [6–9]. Devices that allow the acquisition of such data are classified into two main types: (i) body sensors (such as accelerometers, gyroscopes, magnetometers, and RFID) and (ii) external ambient sensors (including piezoelectric, passive infrared, microphones, and cameras) [8, 10].

Regardless of the data source, substantial research efforts have focused on developing AI models for fall detection [11] and pre-impact [12–14]. However, despite advances in technology, elderly fall detection systems continue to face challenges such as accuracy, reliability, complexity, privacy concerns, and user acceptance.

1.1 Research motivation

The data generated by sensor devices cannot be directly used by machine learning models in their raw format. This poses challenges for extracting relevant information and significantly affects performance. Various factors can contribute to the decrease in data quality, including aberrant measurements, errors in variable distribution, missing variables, and the presence of duplicate or redundant data [15, 16]. In particular, data normalization mitigates the dominance of large-magnitude features by rescaling the data while preserving its underlying distribution and relative proportions.

In this regard, several studies have examined different techniques that are applicable at different stages of fall detection models. However, to our knowledge, no studies have investigated the impact of feature normalization (scaling) on classification performance in the context of fall detection. Existing research focuses primarily on feature preprocessing tasks, such as segmentation/feature representation [17–19].

In this paper, we address this issue by investigating the impact of eight normalization techniques on the performance of a fall detection system and identify the most effective normalization methods. The results show that proper normalization substantially affects classification effectiveness, improving both accuracy and the F1 score, while also generating a significant reduction in training time, up to a factor of *45*.

Motivated also by the opportunity to experiment with the real-world deployment of a fall detection system in an elderly care facility (see the next subsection), we developed a fall detection system based on camera devices.

The development of such a system presented several challenges:

- ***Accuracy***: the system often struggles to reliably distinguish between falls and activities of daily living (ADLs), leading to false alarms (false positives) or, more critically, missed falls (false negatives). Striking the right balance between sensitivity (detecting actual falls) and specificity (minimizing false alarms) is still

difficult.

- ***Reliability***: any fall detection system faces reliability problems. For example, inconsistent performance in different environments (e.g., cluttered rooms vs. open spaces), sensor malfunctions, missed falls due to occlusions or poor lighting conditions.
- ***Integration***: integrating fall detection systems with existing healthcare systems or emergency response services can be challenging, potentially leading to delays or inefficiencies in alerting emergency responders.
- ***Complexity***: fall detection systems can be too complex or cumbersome to operate or maintain by operators, leading to low adoption rates or user dissatisfaction. The designed system must be lightweight and easy to install and possibly not require the use of devices, which many elderly users find uncomfortable or stigmatizing.
- ***User acceptance***: ensuring user acceptance and participation with fall detection systems can be difficult, particularly if users perceive them as intrusive. The proposed systems must be aware of privacy (avoid streaming or storing identifiable video data). This promotes dignity and reduces the stress associated with constant surveillance, especially in sensitive environments such as bedrooms.

Addressing these issues requires a multidisciplinary approach and collaboration between researchers, developers, healthcare professionals, and end-users to develop solutions that are accurate, reliable, user-friendly, and ethically sound.

Importantly, one main limitation of machine learning-based fall detection systems is the absence of real-world evaluation [20]. Although off-line laboratory tests can offer encouraging results, the shift to an intricate real-life environment, such as nursing homes, introduces numerous challenges. Conducting real-time evaluations of fall detection systems in authentic real-world settings is essential to (i) overcome the challenges inherent in such environments and (ii) ensure user acceptance of these systems.

1.2 Methodology and contributions

This research contributes to ongoing efforts in elderly fall detection by introducing novel approaches and algorithms that aim to improve detection accuracy, reduce false alarms, and enhance overall system performance in real-world settings. In particular, this paper presents a real-time vision-based elderly fall detection system that has been deployed and evaluated in a nursing home in France. The architecture of the proposed system consists of three main subsystems: (i) an energy-efficient image acquisition service; (ii) an accurate fall detection service; and (iii) a real-time notification service to alert the concerned monitoring staff or family members. The proposed fall detection approach integrated into the system combines geometric features with thresholding to improve the efficiency and accuracy of fall detection. Real-time evaluation of the system was carried out in the private room of an elderly couple over a period of

three months at Jean XXIII Housing Establishment for Dependent Elderly People (EHPAD[1]) in Montferrand-le-Château (France).

The empirical evaluation has been divided into two parts. In the first experiment, we investigated the impact of eight normalization techniques(namely Min–Max, Z-score, Decimal Scaling, Sigmoid, Tanh, Softmax, Maximum Absolute, and Statistical Column normalization) on the performance (accuracy, F1 score, and training time) of four commonly used classifiers in fall detection: SVM-RBF (Support Vector Machine with Radial Basis Function), KNN (K-Nearest Neighbors), GNB (Gaussian Naive Bayes), and DT (Decision Tree). These models were chosen to ensure a balance between probability-based methods (GNB and DT) and distance/kernel-based methods (SVM-RBF and KNN), enabling a comprehensive evaluation of how different feature normalization techniques influence classification performance.

The first experiment was carried on as follows: (i) dataset selection, (ii) data preprocessing and preparation, (iii) hyperparameter tuning, and (iv) evaluation of each optimized model with normalized and unnormalized data. To ensure a thorough analysis and fair comparison, we (i) selected two public fall data sets of different sizes but with the same imbalance ratio: URFD (University of Rzeszow Fall Detection) [21] and UP-Fall [22], and (ii) tuned the hyperparameters of each model using grid search.

The results obtained in the paper confirm that data normalization significantly accelerates the training of machine learning models. Furthermore, the study identifies the most efficient normalization techniques (in terms of performance) in the context of elderly fall detection, as well as those with the lowest performance. Specifically, the experiments indicate that min–max, z-score, and maximum absolute outperform the other normalization techniques, while softmax exhibits the poorest performance. Sections 3.3 and 3.4 provide a comprehensive analysis of the results obtained.

In the second experiment, building on the results of the first, we investigated the integration of min-max normalization with the SSD MobileNet V2 model (Single Shot MultiBox Detector) [23, 24]. The proposed hybrid fall detection service, integrated into the system, combines geometric features with a thresholding mechanism to enhance model performance and accuracy by standardizing the range of input data. In particular, the geometric features, notably the bounding box dimensions, have demonstrated their relevance in a previous study that analyzed the combination of 255 geometric and pixel-based features [25]. After detecting a person, we analyze variations in the bounding box using two parameters: δ, which captures changes in the height and width of the detected individual's bounding box across frames, and θ, which measures angle variation around the person (detailed in Sect. 4.3.2 and Fig. 6). To enhance the robustness of our detection model, we apply min-max normalization to θ, enabling standardized comparison across frames. This hybrid method integrates deep learning with traditional computer vision techniques to accurately detect falls based on comprehensive spatial and angular dynamics.

The threshold values for δ and θ were empirically determined using the public data sets Multicam Fall Dataset [26] and FallDataset [27], distinct from those used in the previous experiment (i.e., URFD [21] and UP-Fall [22]) to ensure robustness.

[1] EHPAD: Etablissement d'Hebergement pour Personnes Agées Dépendantes (English: Accommodation Establishment for Dependent Elderly People).

The results obtained show that the system respects the privacy of individuals, making it suitable for deployment in real-world environments such as nursing or retirement homes. In addition, the tests indicate proper operation of the system, promising performance (with a low observed error rate under the evaluated conditions), as well as ease of deployment and scalability.

1.3 Structure of the paper

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 investigates the impact of feature normalization on fall detection, followed by a detailed experimentation, performance evaluation, and a discussion of the findings. Section 4 presents the requirements of the system for accurate real-time fall detection, describes the design and architecture of the proposed system with its key services, and discusses implementation, performance evaluation, and real-world deployment. Finally, Sect. 5 concludes the paper and outlines the directions for future work.

2 Related work

The problem of fall detection has attracted considerable attention from researchers. Several works have examined techniques that can be deployed at different stages of the life cycle of fall detection. However, there are no studies on the impact of feature normalization. For instance, the research community has fundamentally focused on feature pre-processing (segmentation, representation domain, etc.). In this section, we will not compare the performance of existing fall detection work but rather present an exhaustive review to position this work.

Data segmentation aims to carry out processing according to specific criteria, enabling the identification of relevant information and an accurate estimation of the model's decisions. Aziz et al. [17] evaluated the impact of window size and lead time on the sensitivity and specificity of a preimpact fall detector: window size denotes the duration of sensor data utilized to analyze and identify a fall, typically measured in seconds (it is of the order of a few seconds); lead time refers to the interval between detecting a fall and the moment of impact, usually expressed in milliseconds (for example, 500 ms). A sufficiently large window size ensures comprehensive analysis, while an optimal lead time allows for timely intervention. Kinematic data were acquired using an Opal model, which contains a triaxial accelerometer and a gyroscope operating at 128 Hz for 15 s per trial. The study assessed 18 features related to triaxial acceleration, velocity, and angular velocity and used an Support Vector Machine with Radial Basis Functions (SVM-RBF), with 10-fold cross-validation. During the experiment, window sizes and lead times varied between 0.125–1.125 and 0.0625–1.125 s, respectively, with an increment step of 0.0625s for each. The system exhibited high performance, achieving sensitivity greater than 95% and specificity greater than 90% for combinations of window sizes between 0.125 and 1 s and lead times from 0.0625 to 0.1875 s.

Data representation allows for the assessment of how various domains (temporal, frequency, differential, etc.) and the relationships between variables affect performance. Wagner et al. [18] investigated the impact of five regularized numerical differentiation methods on the performance of a Naive Bayes-based fall detector. These methods included central difference, regularized central difference, Kalman filter, Tikhonov regularization, and smoothing approximation followed by analytical differentiation. The study focused on three features related to the individual's center of mass: maximum downward vertical velocity, maximum horizontal velocity, and maximum total velocity. The evaluation was carried out using the leave-one-out cross-validation procedure. For the URFD data set [21], the regularized version of the central difference method demonstrated superior performance by increasing the true positive rate and reducing the false positive rate compared to the other methods.

The authors in [19] explored the performance of four classifiers (Naive Bayes, KNN, J48, and Random Forest) separately and in an ensemble based on the stacking of classifiers. They calculated eight features from each segment of accelerometer data provided by central one-second values around the peak. These features include the resultant, variance, standard deviation, Euclidean norm, root mean square, kurtosis, skewness, and geometric mean. The results obtained demonstrated that the stacking-based ensemble had a superior performance (89% sensitivity and 95% specificity) compared to the other classifiers.

The studies referenced above [17–19], together with others [29, 30, 32, 33], all summarized in Table 1, demonstrate the substantial impact of various techniques on the performance of fall detection systems. However, a significant research gap remains regarding the effect of feature scaling techniques on classification performance. To contribute in this direction, this study employed four types of classifier in conjunction with eight normalization methods, offering a thorough comparison with nonnormalized features. This comprehensive analysis aims to improve our understanding of how feature normalization influences the performance and effectiveness of fall detection algorithms.

3 Feature normalization approaches

The performance of machine learning models is significantly influenced by the effort invested in the data pre-processing phase. In this section, we examine the impact of feature normalization on the training time, accuracy, and F1 score of fall detection models.

3.1 Normalization transform definitions

Normalization mitigates the influence of large-scale features while simultaneously preserving the distribution and ratios of the original data set. The considered monotonic data normalization transforms are defined in Table 2.

Table 1 Related work summary

Work	Sensors	Impact variables	Actuation process	Used Dataset(s)	Most efficient configuration	Performance
[17]	Tri-axial accelerometer Tri-axial gyroscope	Window size (0.125 : 1.125 : 0.0625) Lead time (0.0625 : 1.125 : 0.0625)	SVM-RBF with 10-fold cross-validation	Proprietary dataset	Window size (0.125 : 1) Lead time (0.0625 : 0.1875)	> 95% sensitivity > 90% specificity
[18]	Kinect	Central difference Regularized central difference Kalman filter Tikhonov regularization - Smoothing approximation	Naive Bayes with leave-one-out cross-validation	URFD [21]	Regularized central configuration	AUC > 98%
[19]	Smartphone accelerometer	Separate classifiers Stacked classifiers	Naive Bayes KNN J48 Random Forest	FallADL [28]	Stacking	89% sensitivity 95% specificity
[29]	Doppler	Single-window spectrograms Multi-window spectrograms Wigner-Ville distribution Wavelet transform	Stacked auto-encoder with softmax regression	Proprietary dataset	Multi-window spectrograms	90% success rate
[30]	Tri-axial accelerometer	50% overlapping sliding window (0.5 : 5 : 0.5) Impact-defined window (0.5 : 2.5 : 0.5)	SVM-RBF KNN CART Naive Bayes with 5-fold cross-validation	Proprietary dataset SisFall [31]	Sliding window (0.5) Impact defined window (1.5) SVM-RBF (*) KNN (**)	(*) 94% accuracy (**) 98% accuracy
[32]	IMU Optical heart rate	Separate sensor Fusion data	ANN KNN Random Forest XGBoost	Proprietary dataset	Fusion data Random Forest	93% accuracy
[33]	Tri-axial accelerometer	Number and duration of sliding window	SVM-RBF with 5-fold cross-validation	ErciyesUni [34] FallAllD [35] SisFall [31]	W1 (0.5 or 1) W2 (5.5 or 3.75) W3 (5.5 or 3.75)	96.1<= F-score <= 99.7

3.2 Experimentation

To study the impact of data normalization on fall detection model performance, we performed the following four steps: (i) dataset selection, (ii) data pre-processing, (iii) hyperparameter tuning, and (iv) evaluation of optimized models with normalized and unnormalized data (see Fig. 1).

Table 2 Normalization techniques used in this work

Method	Formula	Description
Min-max [36]	$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$	The values are linearly rescaled so that the minimum becomes 0 and the maximum becomes 1
Z-score [36]	$x' = \frac{x - \mu}{\sigma}$	Measures deviations from the mean in units of standard deviation; linear transformation
Decimal norm. [36]	$x' = \frac{x}{10^j}$	j is the smallest integer such that $\max(\lvert x' \rvert) < 1$
Sigmoid [37, 38]	$x' = \frac{1}{1 + e^{-x}}$	Non-linear mapping into the interval (0,1) via an S-shaped function
Tanh [37]	$x' = \frac{1 - e^{-2x}}{1 + e^{-2x}}$	Non-linear mapping similar to sigmoid; outputs in (-1,1)
Softmax [37]	$x'_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$	Transforms a vector into a probability distribution
Max-abs scaling [39]	$x' = \frac{x}{\max(\lvert x \rvert)}$	Rescales features so that the maximum absolute value becomes 1
Statistical norm. [40]	$x'_i = \frac{x_i - C}{10C}$ $C = \frac{1}{10+n} \sum_{j=1}^{n} x_j$	Transformed feature values sum to 1

3.2.1 Dataset selection

We considered the publicly available data sets URFD [21] and UP-Fall [22] because (i) they are rich in vision and movement information, and (ii) they share the same imbalance ratio (approximately a factor 5). The imbalance ratio [41, 42] is defined as $IR = N_-/N_+$, where N_- is the number of majority (or negative) observations in the dataset, and N_+ is the number of minority (or positive) observations. In our case, N_- is the number of both ADLs (activities of daily living) and fall-like, and N_+ is the number of fall instances. Table 3 summarizes the details of the data sets considered.

The URFD data set [21] was built using two Microsoft Kinect cameras and two accelerometers (PS Move (60 Hz) and x-IMU (256 Hz) devices). All RGB and depth images were synchronized with the corresponding motion data based on timestamp values. Camera 0 was positioned parallel to the floor at an approximate height of 1 m, while camera 1 was configured for the placement of the ceiling 2.5 *m* above the floor. Accelerometers were mounted on the waist or near the pectoral muscles, which are considered optimal positions on the human body to detect falls [43, 44]. URFD contains data from 70 activities (30 falls and 40 daily and fall-like activities) performed by 5 volunteers (actors). The activities (falling while standing and sitting in a chair, walking, sitting, lying on the floor, picking, lifting, or placing objects on the floor, tying shoelaces, bending to the left, and squatting) took place in conventional indoor spaces such as offices and classrooms.

This study used features extracted from depth sequences captured by Camera 0. Specifically, in the first experiment, we analyzed a total of 11,544 samples: 9741 from the negative class and 1803 from the positive class. The extracted features

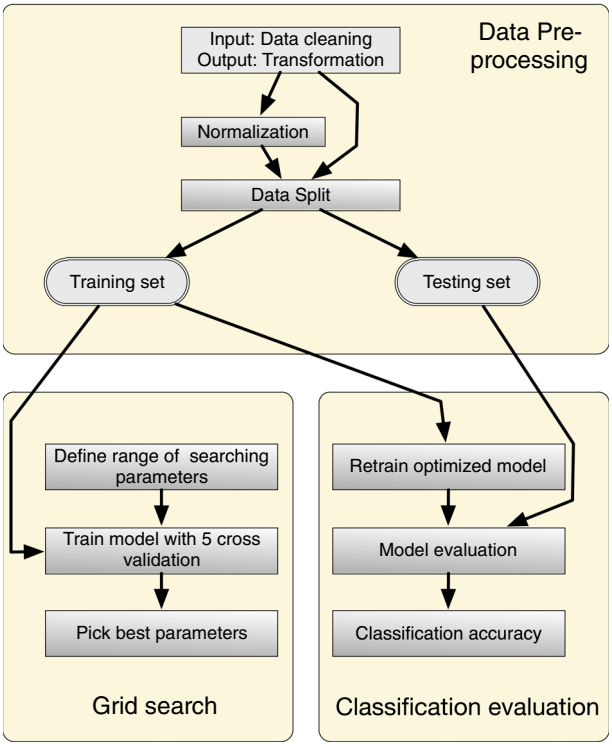


Fig. 1 Schema of the normalization impact experiment

can be categorized into two main groups: (i) geometric features, which focus on the spatial and proportional aspects of the detected individual. (ii) Pixel-based features, which analyze the distribution and occupancy of pixels within depth images.

The UP-Fall dataset [22] includes 11 activities in which measurements were acquired using 14 different sensors (namely, 2 Microsoft LifeCam Cinema cameras, 5 IMUs (inertial measurement units), 1 NeuroSky electroencephalograph (EEG) Mind-

Table 3 Description of the used URFD and UP-Fall datasets

Dataset	Year	Activities	Sensors	Actors	IMU position	Camera position	IR
URFD [21]	2014	70 (30 falls and 40 ADLs)	2 Kinect cameras 1 PS Move 1 x-IMU	5	Near the pelvis (waist)	Parallel to the floor (1 m) and ceiling configuration (2.5 m)	$\frac{9,741}{1,803} \approx 5$
UP-Fall [22]	2019	11 (5 falls and 6 ADLs)	2 Microsoft LifeCam 5 IMU 1 ECG NeuroSky MindWave 6 infrared	17	Ankle, pocket, waist, neck and wrist	Frontal view (1.82 m) Lateral view (1.82 m)	$\frac{248,727}{45,951} \approx 5$

Wave headset and 6 infrared). UP-Fall was collected with the collaboration of 17 volunteers (young, healthy men and women) aged between 18 and 24, with an average height of 1.66 m and an average weight of 66.8 kg. These volunteers simulated 5 falls and 6 ADLs, with 3 attempts each. UP-Fall combines over 850 GB of body, ambient, and visual sensor data from various activities (falling forward with hands, falling forward with knees, falling backward, falling to the side, falling while sitting on a chair, walking, standing, sitting, picking up an object, jumping and lying down).

We used the 294,678 available motion samples (248,727 from the negative class and 45,951 from the positive class) acquired from the sensors positioned at the waist (3-axis accelerometer and 3-axis gyroscope).

3.2.2 Data pre-processing

We performed pre-processing on both the features and output data to optimize them for the employed fall detection classifiers. This included steps such as data cleaning, data normalization, and output transformation.

- ***Data cleaning***: involved validating the correct structure of the data, which included eliminating duplicate entries, outliers, and missing values.
- ***Data normalization***: the numerical values of the attributes have been scaled by applying the eight considered data normalization techniques presented in Sect. 3.1. The aim of this step is to improve the quality of features by avoiding the dominance of those of large magnitude over the model's performance. We also considered an additional experimental process without data normalization to compare it with the above techniques.
- ***Output transformation***: fall detection is typically addressed as a binary classification problem (1 for fall, 0 for activities of daily living or fall-like activities) [45]. However, in the URFD dataset, the position or status of the subjects is labeled into three groups for each measurement: (– 1) indicating the person is not lying on the ground, (0) when the person begins to fall, and (1) for post-fall (i.e., the person is lying on the ground). In our study, we modified the labels: we replaced the output (– 1) with (0), and (0, 1) was unified as (1). For the UP-Fall dataset, the output labels range from 1 to 10. The first five labels correspond to different types of simulated falls by the volunteers, and the last five correspond to ADLs. We redefined the labels: we mapped the first 5 to 1 (fall) and the remainder to 0 (ADLs and fall-like observations).

After preprocessing, each data set was split into two subsets: 70% for training and 30% for evaluation (Fig. 1).

3.2.3 Hyperparameter tuning

The performance of machine learning models depends crucially on the setting of their hyperparameters (e.g., learning rate, batch size, number of epochs, regularization parameters, number of hidden layers). Those can be manually adjusted by a trial and error process; however, this approach is time consuming and yields typically subop-

Table 4 Hyperparameter grid search configuration for the optimized models (SVM-RBF, KNN, GNB, and DT)

Model	Hyperparameter	Hyperparameter range	Step	No. of elements	Scale
SVM-RBF	C	$[2^{-5} : 2^{15}]$	2^2	11	Logarithmic
	gamma	$[2^{-13} : 2^3]$		10	
KNN	N_neighbors	[1 : 40]	1	40	Linear
GNB	Var_smoothing	$[10^{-9} : 10^2]$	$10^{0.11}$	100	Logarithmic
DT	Max_depth	[2 : 19]	1	18	Linear
	Min_samples_split	[2 : 19]		18	
	Min_samples_leaf	[1 : 9]		9	

timal results. Automatic exploration of the hyperparameter space can be carried out using various approaches, including random search, grid search, genetic algorithms, differential evolution optimization, and Bayesian optimization [46]: the methods offer different trade-offs between simplicity of implementation, scalability with the dimension of the hyperparameter space, and guarantees of finding an optimal solution [47]. In the present work, due to its exhaustiveness, ease of implementation, and reproducibility [48] we used grid search. This approach checks all combinations of hyperparameters (using some form of regular spacing if the values of a hyperparameter are numerical) and compares performance. For each combination of values, the model is trained and a score is calculated. Then, the hyperparameter combination that yields the highest score is selected. During hyperparameter optimization, we used a five-fold cross-validation on the training data. We used the grid proposed by scikit-learn [49]. Table 4 provides a summary of the hyperparameter configurations (range of values, step, number, and type of scale). Following this tuning phase, we successfully identified the optimal parameters for each classifier.

Table 5 Performance comparison of SVM-RBF, KNN, GNB, and DT classifiers (with and without normalization) using the URFD and UP-Fall datasets: (OP = optimized hyperparameters, accuracy denoted by *, F1 score by **)

Classifier	SVM-RBF						KNN						GNB						DT					
Dataset	URFD						URFD			UP-Fall			URFD			UP-Fall			URFD			UP-Fall		
OP | Metric | Delta		*	**	*	**			*	**		*	**		*	**		*	**			*	**		*	**
Technique																										
Without normalization	2^{15} 2^{-15}	93.8 77.3		0 0		7	92.1 71.4		0 0	10	88.7 49		0 0	0.46	86.4 49.4		0 0	4.64e-5	86.8 35.6		0 0	11 2 2	**91.1** **79.6**		0 0	19 6 2	**94.9** 82.6		0 0
Min-max	2^{15} 8	96.8 89.5		3 12.2		1	96.8 89.3		4.7 17.9	4	96.7 89.2		8 40.2	2.15	**87.3** 53.8		0.9 4.4	0.02	**86.9** **36.5**		0.1 0.9	14 2 2	**94.1** **79.6**		0 0	19 6 2	**94.9** 82.6		0 0
Z-score	2^5 2	**97.3** **91**		3.5 13.7		1	96.5 87		4.4 17	4	96.4 88		7.7 39	3.59	**87.3** 53.3		0.9 3.9	0.17	86.8 36.1		0 0.5	14 2 2	**94.1** **79.6**		0 0	19 6 2	**94.9** 82.6		0 0
Decimal	2^{15} 2^3	87.1 56.4		-6.7 -20.9		21	90.4 62.3		-3.7 -9.1	3	96.7 89.3		8 40.3	0.36	**87.3** 53.7		0.9 4.3	0.003	**86.9** **36.5**		0.1 0.9	15 2 11	91.9 70.8		-2.2 -8.8	16 13 2	94.3 82.3		-0.6 -0.3
Sigmoid	2^{15} 2^3	93.4 75.4		-0.4 -1.9		4	95.4 83.7		3.3 12.3	6	91.5 68		2.8 19	0.6	87.1 **55.7**		0.7 6.3	1e-9	84.5 1.3		-2.3 -34.3	14 1 7	92.6 75.1		-1.5 -4.5	19 4 2	**94.9** 82.6		0 0
Tanh	2^{13} 2^3	94.5 80		0.7 2.7		4	95.4 84		3.3 12.6	4	92.6 73.1		3.9 24.4	2.15	86.7 38.7		0.3 -10.7	1e-9	84.9 0		-2.4 -35.0	14 1 7	92.6 75.1		-1.5 -4.5	19 2 2	94.8 82.5		-0.1 -0.1
Softmax	2^{-5} 2^{-15}	84.4 0		-9.4 -77.3		5	94.6 80.8		-2.5 -9.4	14	**97** **90.2**		8.3 41.2	0.1 0	84.4 49.4		-2 -7.3	7.74	84.4 0		-2.4 -35.0	19 2 5	92.7 75.1		-1.4 -4.5	8 4 1	84.5 1.4		-10.4 -81.2
Maximum absolute	2^{15} 2^3	96.7 89		2.9 11.7		1	96.8 89.5		4.7 18.1	4	96.7 89.2		8 40.2	1.67	87.2 54.7		0.8 5.3	0.03	**86.9** 35.9		0.1 0.3	14 2 2	**94.1** **79.6**		0 0	19 6 2	**94.9** 82.7		0 0.1
Statistical column	2^{15} 2^3	94.5 80.4		0.7 3.1		1	**97** **90**		4.9 18.6	8	93.1 74.7		4.4 25.5	1	87.2 53		0.8 3.6	4.64e-5	86.7 34.3		-0.1 -1.3	14 2 2	**94.1** **79.6**		0 0	19 2 2	**94.9** **82.8**		0 0.2

In bolface the best performing

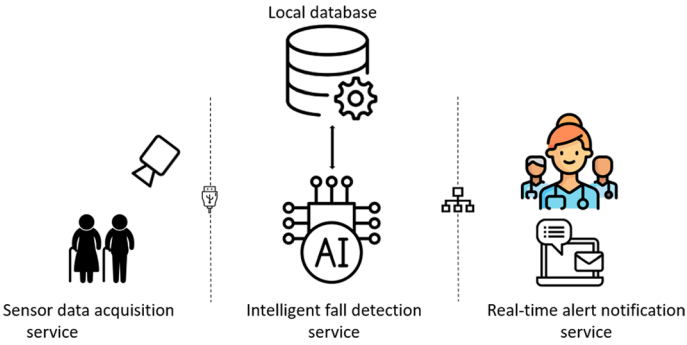


Fig. 2 Overall architecture of the proposed fall detection system. The data storage/logging is used to store fall data for future analysis

3.3 Performance evaluation results

To assess the performance of each model, we consider training time, the F1 score, and accuracy metrics. F1 score is the harmonic mean of precision ($p \equiv TP/(TP+FP)$) and recall ($r \equiv TP/(TP+FN)$), that is, $F1 \equiv 2pr/(p+r)$ Accuracy is the fraction of correctly predicted observations compared to the total number of observations ($a \equiv (TP+TN)/(TP+TN+FP+FN)$).

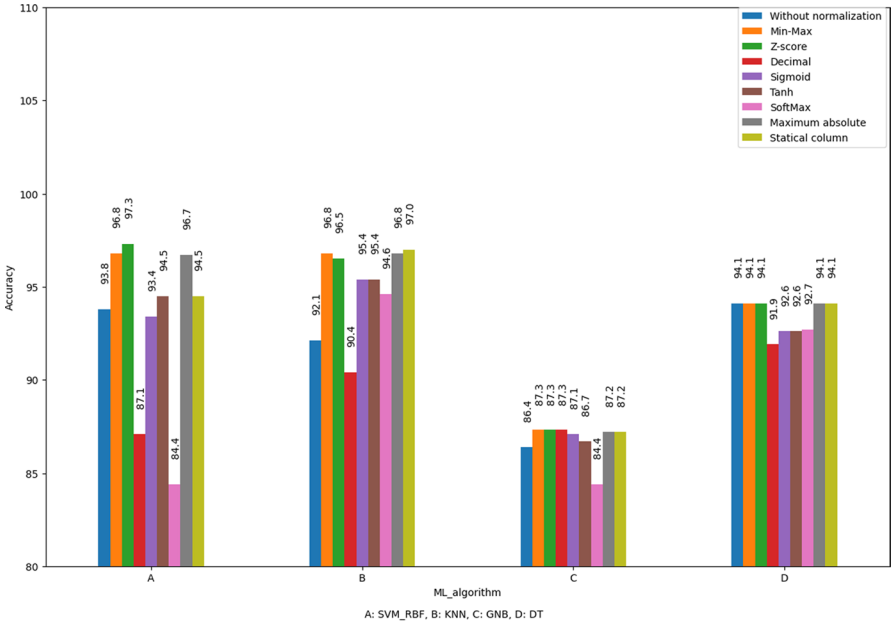


Fig. 3 Accuracy of SVM-RBF, KNN, GNB, and DT classifiers (with and w/out normalization) using the URFD dataset

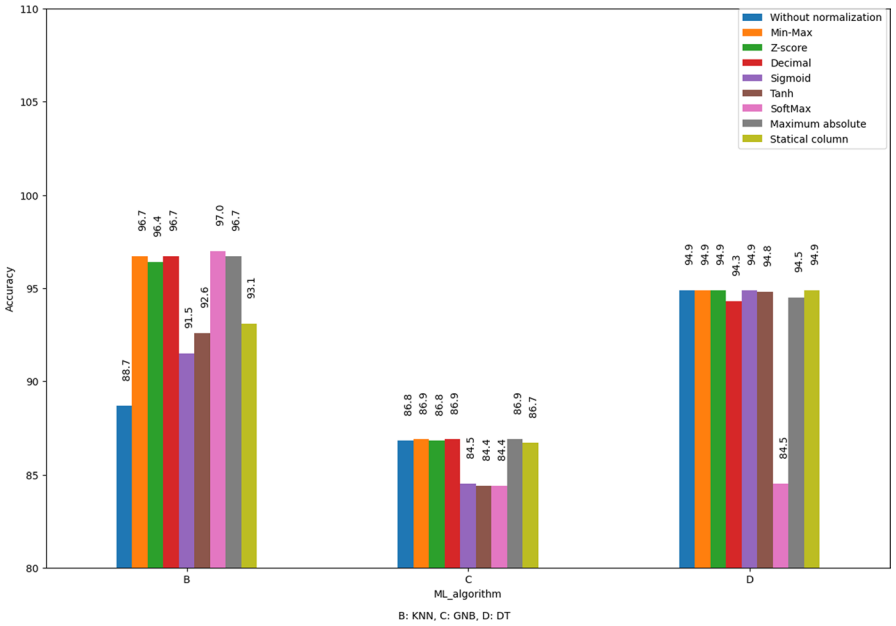


Fig. 4 Accuracy of KNN, GNB, and DT classifiers (with and w/out normalization) using the UP-Fall dataset

We used grid search and 5-fold cross-validation to determine the optimal hyperparameters for each model. Subsequently, as illustrated in Fig. 1, each classifier was evaluated using these optimal hyperparameters in the test set. This evaluation was carried out on the test set after retraining the optimized models on the entire training set without cross-validation. It should be noted that the SVM-RBF model was excluded from experiments involving the UP-Fall dataset due to its considerable training time requirements, especially when dealing with large datasets [50].

3.3.1 Training time

To analyze the impact of data normalization on training time, we applied the ensemble learning hard voting algorithm. We stack the optimized models with the hyperparameters detailed in Table 5 for each data set. The results obtained indicate a significant improvement: a factor of 45 reduction in the average training time: from 83.35 s without normalization to 1.82 s with normalization.

3.3.2 F1 score

Normalization methods significantly affected the F1 score in various models and datasets (Table 5). For example, the maximum absolute normalization method increased the KNN F1 score by 18. 1% in the URFD database and 40. 2% in the UP-Fall. Similarly, min-max normalization improved the KNN F1 score by 17. 9% on URFD

and by 40.2% on UP-Fall. Furthermore, the z-score method improved the F1 score of SVM-RBF by 13. 7% in URFD and KNN by 17% in the same data set.

In contrast, some normalization methods have a negative impact on performance. In particular, the softmax method drastically reduced the F1 score of SVM-RBF by 77. 3% and that of GNB by 49. 4% in URFD. Decimal normalization also had adverse effects, decreasing the F1 score of SVM-RBF by 20.9% and of KNN by 9. 1% in URFD. The sigmoid and tanh normalization methods produced mixed results, showing modest improvements in some cases while causing deterioration in others.

3.3.3 Accuracy

The results obtained presented in Table 5 and Figs. 3 and 4 show the accuracy of the SVM-RBF, KNN, GNB, and DT classifiers (with and without normalization) using the URFD and UP-Fall datasets. When analyzing these results, we notice that the considered data normalization techniques impact the accuracy of classifiers as follows when compared to the non-normalization approach:

- SVM-RBF: accuracy difference ranging from −9.4 to 3.5% in URFD.
- KNN: accuracy difference ranging from −1.7 to 4.9% in URFD and from 2.8 to 8.3% in UP-Fall.
- GNB: accuracy difference ranging from -2% to 0.9% in URFD and from −2.4 to 0.1% in UP-Fall.
- DT: accuracy difference ranging from −2.2 to 0% in URFD and −10.4 to 0% in UP-Fall.

For example, with the SVM-RBF model, the results show that data normalization improves accuracy by 3.5% compared to without normalization (a significant improvement in healthcare). Specifically, the accuracy differences for each normalization technique applied to the SVM-RBF model are the following: softmax (– 9.4%), decimal (-6. 7%), sigmoid (– 0.4%), statistical column (0.7%), tanh (0.7%), absolute maximum (2.9%), min-max (3.0%) and z score (3.5%). Detailed results, including optimized hyperparameters, accuracy and differences in accuracy for each normalization technique in the SVM-RBF, KNN, GNB, and DT classifiers with the URFD and UP-Fall datasets, are presented in Table 5.

As shown in Table 5 and Figs. 3 and 4, data normalization leads to a significant improvement in accuracy for SVM-RBF (3.5% in URFD) and KNN (4.9% in URFD and 8.3% in UP-Fall). In contrast, for GNB, we observe a slight improvement (0.9% in URFD and 0.1% in UP-Fall) and no improvement for DT. Note that the accuracy of models with unnormalized data is not always the lowest when compared to that with normalized data. For example, for SVM-RBF in URFD, the accuracy without normalization is 93.8%, and with softmax normalization it is 84. 4% (a decrease in accuracy of 9.4%). As a second example, for DT in UP-Fall, the accuracy without normalization is 94.9% versus. 84.5% for softmax (that is, a decrease of 10.4%).

Table 6 Evaluation of normalization techniques for SVM-RBF, KNN, GNB, and DT classifiers using the URFD and UP-Fall datasets (7 cases, excluding SVM-RBF with UP-Fall): (+) indicates improvement, (-) deterioration, and (0) no impact

Metric	Accuracy			F1 score		
Impact	+	-	**0**	+	-	**0**
Min-max	5	0	2	5	0	2
Z-score	4	0	3	5	0	2
Decimal	3	4	0	3	4	0
Sigmoid	3	3	1	3	3	1
Tanh	4	3	0	3	4	0
Softmax	2	5	0	2	5	0
Maximum Absolute	5	0	2	6	0	1
Statical column	4	1	2	5	1	1

3.4 Discussion of the results

The experiments demonstrate that min-max and z-score normalization yield the highest accuracy results and that softmax has, in most cases, the poorest performance. The performance degradation observed across datasets can be attributed to the behavior of certain normalization methods that excessively compress the range of feature values, thereby reducing the discriminative power of key descriptors. In contrast, Min–Max and Z-score normalization effectively preserved the relative importance of the features within their original scale while maintaining their semantic interpretability. This stability contributed to more consistent and transferable performance across heterogeneous datasets.

Specifically, considering the seven cases that involve four classifiers (SVM-RBF, KNN, GNB, and DT) and two datasets (URFD and UP-Fall), while excluding the SVM-RBF model from experiments with the UP-Fall dataset, the experiments demonstrate that three normalization methods (min-max, z-score, and maximum absolute) do not negatively impact performance (Table 6). As depicted in Table 6, among these, the Min-Max method shows a particularly stable impact on both metrics (accuracy and F1 score), with 5 improvements, 0 deterioration, and 2 neutral results. In contrast, the softmax method resulted in the most significant deterioration in classifier performance.

In general, normalization methods that fit data to comparable scales enable machine learning algorithms (such as SVM-RBF and KNN) to better distinguish features in the data. Min-max normalization transforms data to a fixed range between 0 and 1, while Z-score standardizes data based on the mean and standard deviation, centering data around zero with unit variance. These transformations allow for more uniform processing of features, thereby improving model performance. The maximum absolute normalization scales values to a fixed range by dividing by the absolute maximum value, making it less sensitive to extreme values and contributing to more stable performance. In contrast, methods like softmax, which uses an exponential function, can be highly susceptible to outliers, disproportionately magnifying deviations. This may result in models that are less robust to variations in the data, which explain the drastic decreases in performance observed.

4 Real-time fall detection system development

This section presents the real-time fall detection system developed for elderly care and deployed in a real-world nursing home environment. Before delving into the implementation details, we outline the key functional and nonfunctional requirements that such a comprehensive system should meet to ensure effective implementation and deployment in real-world settings.

4.1 System requirements for real-time fall detection

4.1.1 Functional requirements

- ***Continuous energy-efficient monitoring***: Real-time fall detection requires continuous monitoring of the environment, whether through video surveillance, wearable sensors, or other devices, even during low activity periods.
- ***Real-time fall detection***: The system must accurately detect falls as they occur without any significant delay.
- ***Immediate notification***: After detecting a fall, the system must immediately notify caregivers or emergency services (within seconds) to ensure prompt intervention, and provide information on the individual's location and condition to facilitate a coordinated response. The system must also include false alarm mitigation mechanisms.
- ***Multi-camera support***: The system must integrate multiple cameras (e.g. in a building with multiple rooms) to provide comprehensive coverage.
- ***Data logging***: The system must log all fall detection events and provide reports for analysis and evaluation.
- ***User interface***: The system must offer a user-friendly interface for easy monitoring and management.
- ***System customization***: End users must be able to customize notification preferences, alert thresholds, and other system settings to tailor the system to their specific needs and preferences.

4.1.2 Non-functional requirements

- ***Performance (high accuracy and low latency)***: the system must be able to accurately detect falls in varying lighting conditions, camera angles and environments (e.g., cluttered rooms, open spaces). The system must achieve *high accuracy* in fall detection, minimizing false negatives and false positives, and thus minimizing false alarms. In addition, fall detection and notification must be done rapidly with *low latency* to ensure timely responses.
- ***High-availability***: the system must be able to monitor the environment 24/7 and at any time of day or night. Availability refers to the proportion of time that a system is operational and accessible to users and other systems (e.g., five-nine availability 99. 999%).

- ***Reliability***: refers to the probability that a system will function correctly under specified conditions for a given period of time. An elderly fall detection system must be reliable and robust. Regardless of hardware and/or software failures, the system must be able to continue to deliver its main functionalities, and recovery must be instantaneous and transparent (without data loss or service interruption).
- ***Scalability***: the system must maintain the required low-latency performance in the face of increasing volumes of data and users (multiple cameras, large amounts of sensor data, high throughput, etc.).
- ***Data privacy and security***: the data collected for fall detection is personal and sensitive. The system must ensure compliance with the relevant regulations and standards that govern healthcare and data privacy. For instance, in Europe, to address and harmonize data protection and privacy, the EU has specified a set of strict regulations, the GDPR (General Data Protection Regulation), to which all digital businesses and services using personal sensitive data must comply. One of the main problems with camera-based systems is that people worry about their privacy and do not like to be watched or filmed. To protect the collected data according to the consent of end-users, the latter can be offered (online) contracts that list all their preferences (e.g., who can access their data, for how long, etc.) and allow them to access and modify these preferences at any time. The system must also provide measures (confidentiality, integrity, and availability of information) to ensure data privacy and security and to protect sensitive data captured by the system.

The system could provide additional services to monitor the daily activities and movements of elderly individuals, such as integration with wearable sensors.

4.2 System design and architecture

The system is structured into three main components (see Fig. 2): (i) energy-efficient sensor data acquisition service (to obtain RGB images from the monitored environment), (ii) intelligent fall detection service (for image processing and information extraction), and (iii) real-time alert notification service (to alert relevant parties about detected falls).

4.2.1 Sensor data acquisition service

This service is responsible for acquiring RGB images from the monitored environment in an energy-efficient manner. The video frames are then processed for feature extraction relevant to fall detection. To be efficient and preserve resources, the service must operate in standby mode: it activates only when significant pixel changes occur, thus minimizing continuous recording. In addition, image storage and transmission must be designed and implemented to consider and ensure the privacy of seniors, which is crucial for camera-based fall detection.

4.2.2 Intelligent fall detection service

This crucial component is designed to accurately detect fall incidents in real time. Using advanced image processing and AI algorithms, this service analyzes video data captured by RGB cameras to detect potential falls with high accuracy and reliability. The system is also designed for real-time operation, continuously monitoring the video feed and processing data instantaneously, ensuring immediate detection to minimize the impact of falls.

In this work, we investigated the integration of min-max normalization with the SSD MobileNet V2 model [23, 24]. The aim was to enhance the model's latency and accuracy by standardizing the input data range. The proposed hybrid machine learning approach, which combines geometric features with thresholding, was integrated into the system to further enhance performance. The implementation details are provided in Sect. 4.3.

4.2.3 Real-time alert notification service

Real-time notifications serve as critical alerts that allow healthcare staff and family members to promptly provide the necessary assistance. To document any potential fall incident, upon fall detection, the notification service records binary images in

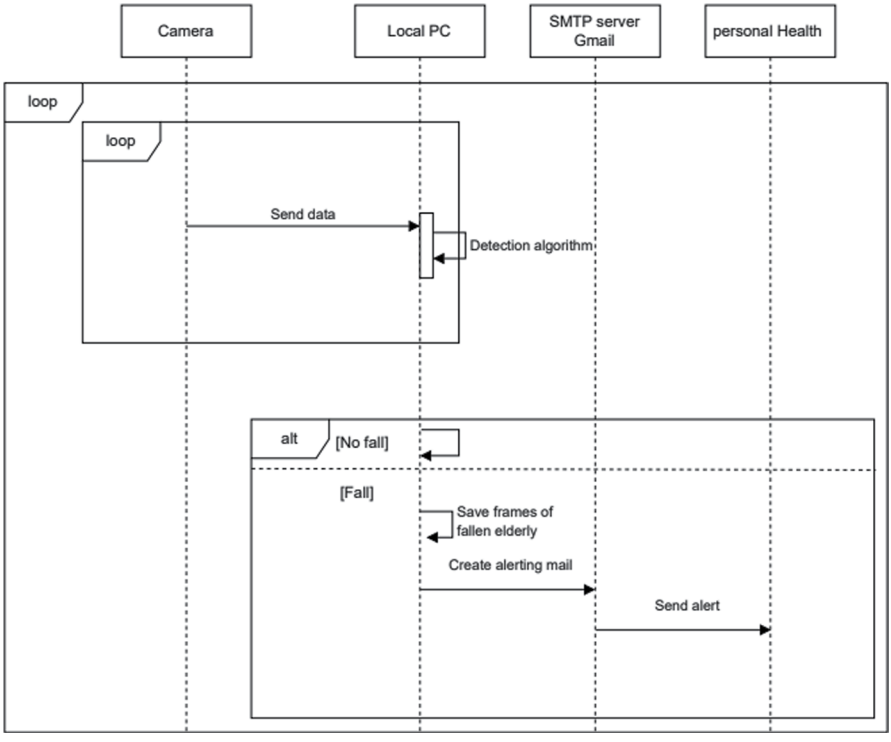


Fig. 5 UML sequence diagram of the fall detection system

its local storage (to respect privacy). Then, it sends a notification with these images to relevant monitoring parties (e.g., the nursing staff of a retirement home). In addition to email and SMS, popular instant messaging platforms and services (such as WhatsApp, Telegram, or Facebook Messenger), customized mobile applications, and alternative API architectures such as MQTT can also be used to send alerts.

4.3 System implementation and optimization

Figure 5 shows the UML sequence diagram of the system, which integrates Min-Max normalization [36] with the SSD MobileNet V2 model [23, 24].

4.3.1 Sensor data acquisition service

The system operates on a workstation equipped with an Intel Core i7 processor, 32 GB of RAM, and an NVIDIA Quadro K610M GPU, and processes video streams acquired at a frame rate of 18–22 frames per second using a Logitech camera (MSIP-REM-DZL-V-U0040 series) mounted on the wall in a fixed configuration. Each RGB frame is resized to 300 $\times$ 300 pixels to prepare it for analysis using the SSD MobileNet V2 model [23, 24]. The system loops over the detected objects classified under the "*Person*" category with a confidence score greater than 30%. From the data from the bounding box, the salient geometric characteristics are then extracted. To respect the privacy of seniors, we implemented image storage and transmission in binary mode. Specifically, we used the KNN background subtraction method [51, 52].

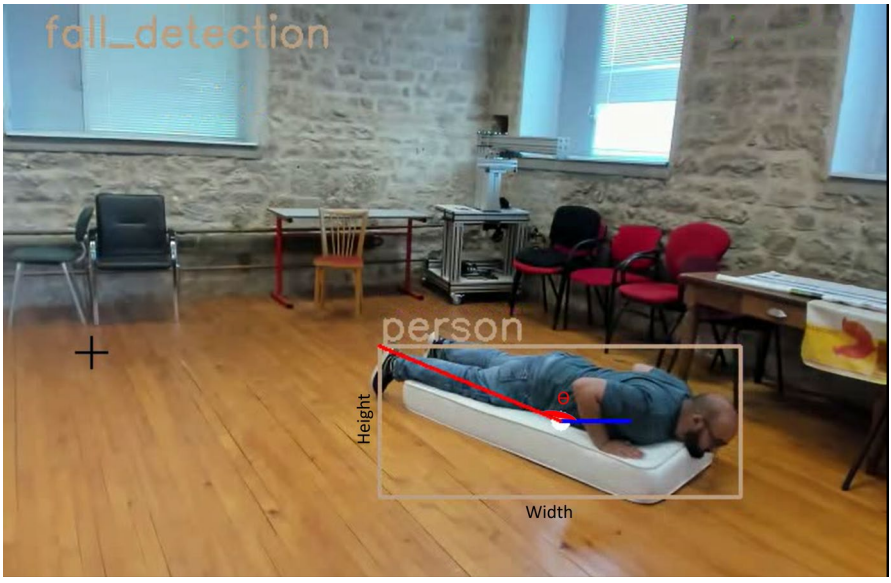


Fig. 6 Feature annotation on a sequence from our proprietary dataset: bounding box aspect ratio variation ($\delta = height - width$) and orientation angle (θ)

4.3.2 Intelligent fall detection service

To implement this core component, we used a pre-trained SSD MobileNet V2 model [23, 24] for person detection, providing a balance of speed and accuracy suitable for real-time applications. Upon successful person detection, where individuals are identified and localized using bounding boxes, the system extracts two primary geometric features: δ and θ (see Fig. 6).

The quantity δ represents the difference between the height and width of the bounding box ($\delta \equiv height - width$). Captures the aspect ratio and can be indicative of a person's posture and movement, while θ, defined by $\cos\theta = \vec{V_1} \cdot \vec{V_2} \Big/ \left(\|\vec{V_1}\| \, \|\vec{V_2}\|\right)$, is the angle formed by the horizontal vector starting from the center of the bounding box and the vector connecting the center of the bounding box to the upper left corner (see Fig. 6). It carries information on the orientation of the person and the potential tilt.

To detect falls, the system compares the normalized feature values with the predefined thresholds δ and θ (determined empirically in Sect. 4.3.3):if the values exceed their respective thresholds for a specified number of consecutive frames (also determined empirically in Sect. 4.3.3), the system concludes that a fall has occurred and promptly sends a notification alert to designated caregivers or emergency contacts, ensuring timely intervention.

4.3.3 Empirical determination of fall detection thresholds

The threshold values for δ and θ were determined by empirical analysis using two public datasets: Multicam Fall Dataset [26] and FallDataset [27]. We selected data sets different from those used in the previous normalization impact experiment (Sect. 3) to mitigate bias.

- ***Multicam Fall Dataset*** [26]: consists of simulated falls and daily activities recorded from 24 scenarios. The data is acquired by eight cameras distributed within a room, following a precise configuration, and conducted by a single volunteer.[2]
- ***FallDataset*** [27]: includes 250 video sequences capturing 192 instances of falls and 57 daily activities. The volunteers simulated these activities in four different locations under different conditions (lighting, presence of occlusion, and unorganized background).

The video sequences of the two datasets were analyzed to calculate the normalized thresholds. We identified the frames in which balance loss occurs in each sequence (Fig. 7); during the fall phases, the values of δ are negative; this condition is indicative of the risk of falling [53]. Regarding θ, we calculated the average value in the frames of loss of balance. In particular, we observe that it is determined that

[2] The participants signed an informed consent declaration.

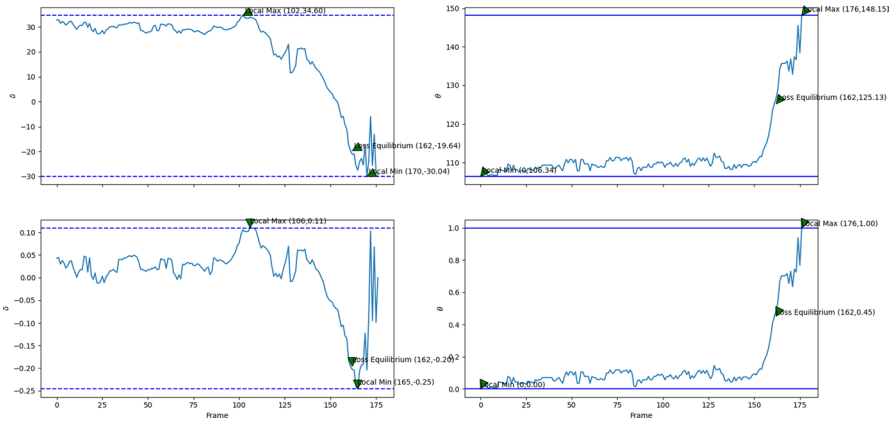


Fig. 7 Threshold variation of δ (left) and θ (right), without normalization (top) and with normalization (bottom), depending on the fall frames from the FallDataset [27] performed in a coffee room

Fig. 8 Sample experiments on δ and θ threshold validation

a fall has occurred if the following condition is met during 6 consecutive frames: $\delta < 0 \; and \; \theta_{normalized} \geq 0.33$.

We tested the threshold values using video sequences from both the URFD dataset [21] and our proprietary dataset (Fig. 8). The evaluation was conducted on 7 sequences (3 fall videos and 4 ADL videos). Under these experimental conditions, no false positives or false negatives were observed, corresponding to a 0% error rate on this limited validation set. These results indicate promising performance in distinguishing fall events from daily activities within the evaluated scenarios [54].

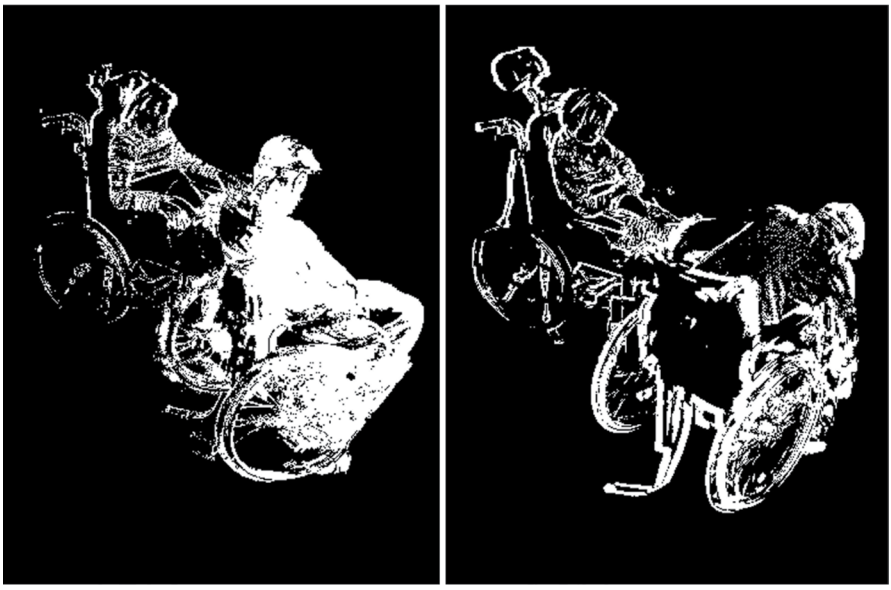


Fig. 9 The 2 false alarms received during the 3-month real-world deployment

4.3.4 Alert notification service

For an initial evaluation, we implemented the real-time alert notification service using the Gmail SMTP Server. The configuration set up was the following Server address: `smtp.gmail.com`, Port: `587`, Gmail username: `reperage.feder@gmail.com`, TLS (Transport Layer Security) required: `yes`. Note that this service can be easily replaced by others (e.g. alternative API architectures like MQTT), depending on specific deployment requirements.

4.4 Real-world system deployment

We successfully deployed the real-time fall detection system at the Jean XXIII nursing home in Besançon (France). With appropriate permissions,[3] the system was installed in the private room of an elderly couple for a period of three months (from April to June 2023[4]). This deployment aimed to evaluate the performance and reliability of the system in a real-world environment, ensuring its efficiency in providing timely alerts and support to elderly residents. The aim was also to evaluate its ease of integration into existing care routines, its usability by healthcare personnel, and its

[3] Approval for this study was obtained from the Institutional Review Board following a comprehensive review process, ensuring that all ethical considerations and privacy concerns were adequately addressed to protect participants' rights and well-being. The participants also signed an informed consent declaration.

[4] This duration was set by the nursing home and aligned with the timeline of the associated research project.

ability to operate continuously with minimal maintenance in a constrained real-life setting.

During the three-month period, no actual falls were recorded in the elderly establishment. However, we did receive two false alarms, as shown in Fig. 9. These alarms were triggered when the couple spent a day in wheelchairs due to physical fatigue. Despite these false alarms, the staff reported that the presence of the system provided reassurance to both caregivers and residents, positively contributing to well-being, and strengthening a supportive care environment.

5 Conclusions and future work

A model with a reasonable training time and high accuracy is crucial for the efficiency and reliability of real-world applications. In this paper, we introduced a novel real-time fall detection system that integrates optimal data normalization with the SSD MobileNet V2 model. This hybrid approach leverages geometric features and thresholding to enhance the accuracy and efficiency of fall detection. Implemented and evaluated in real-world settings over several months, our approach demonstrates the feasibility of a real-time fall detection system in practical environments, with encouraging results in reducing false positives and false negatives under the evaluated conditions, while enabling timely alerts.

In the paper, we also report on the impact of eight data normalization techniques on the classification performance and training time of SVM-RBF, KNN, GNB, and DT classifiers for elderly fall detection using two datasets (URFD and UP-Fall), tuning each model's hyperparameters by grid search. The experimental results indicate that suitable data normalization can significantly improve performance, improve accuracy and the F1 score, and reduce training time (the best performing are min-max, z-score and maximum absolute normalization).

In future work, we plan to extend the proposed system by incorporating additional services that collect valuable data on the movements and activities of elderly individuals. These data can provide insight into mobility patterns among older adults, facilitating the adaptation of care plans to better support their overall physical and psychological well-being. Moreover, we plan to explore transformer-based architectures, 3D convolutional neural networks (3D CNNs), and temporal modeling approaches to benchmark the proposed hybrid system against recent state-of-the-art methods.

Acknowledgements The authors sincerely thank Samuel Robbe, the Director of the Jean XXIII Nursing Home in Montferrand-le-Chateau, and all the healthcare staff for their invaluable collaboration. We express our gratitude to the elderly couple living in the nursing home for their willingness to participate, which contributed significantly to advancing research on healthy aging and the detection of falls among older adults.

Author contributions All the authors contributed to Methodology and to Investigation, M.F., M.A.M., A.M. and R.Y contributed to Conceptualization and Writing the original draft, M.F., M.A.M., and R.Y contributed to Data Curation, and to Software, M.F., A.M., E.D., G.G. contributed to Funding Acquisition and to Supervision, All authors reviewed and edited the manuscript.

Funding Open access funding provided by Università degli Studi di Milano - Bicocca within the CRUI-CARE Agreement.

Data availability Data are not made publicly available.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. WHO (2021) Falls. https://www.who.int/news-room/fact-sheets/detail/falls. Accessed 15 Apr 2025
2. Igual R, Medrano C, Plaza I (2013) Challenges, issues and trends in fall detection systems. Biomed Eng Online 12:1–24
3. Fayad M, Mostefaoui A, Chouali S Benbernou S (2022) Toward a design model-oriented methodology to ensure qos of a cyber-physical healthcare system. Computing 104:1615–1641.
4. Turner S, Kisser R, Rogmans W (2015) Falls among older adults in the eu-28: key facts from the available statistics. EuroSafe, Amsterdam
5. Fayad M et al (2024) Impact of feature normalization on machine learning-based human fall detection
6. Mobsite S, Alaoui N, Boulmalf M, Ghogho M (2023) Semantic segmentation-based system for fall detection and post-fall posture classification. Eng Appl Artif Intell 117:105616
7. Fayad M, Mostefaoui A, Chouali S Benbernou S (2019) Fall detection application for the elderly in the family heroes system. In Proceedings of the 17th ACM International Symposium on Mobility Management and Wireless Access (pp. 17-23). https://doi.org/10.1145/3345770.3356738
8. Nooruddin S et al (2021) Sensor-based fall detection systems: a review. J Ambient Intell Human Comput 13:2735–2751 (2022).
9. Clemente J, Li F, Valero M, Song W (2019) Smart seismic sensing for indoor fall detection, location, and notification. IEEE J Biomed Health Inform 24:524–532
10. Merzoug MA, Mostefaoui A, Kechout MH Tamraoui S (2020) Deep learning for resource-limited devices. In Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks (pp. 81-87).
11. PB NN, Venkatasubramanian K Mathivanan P (2025) Fall detection for elderly people using realsense depth camera. In 2025 3rd International Conference on Smart Systems for applications in Electrical Sciences (ICSSES) (pp. 1-6). IEEE.
12. Turetta C, Ali MT, Demrozi F Pravadelli G (2025) A lightweight cnn for real-time pre-impact fall detection. In 2025 Design, Automation & Test in Europe Conference (DATE) (pp. 1-7). IEEE.
13. Kiran S, Riaz Q, Hussain M, Zeeshan M, Krüger B (2024) Unveiling fall origins: leveraging wearable sensors to detect pre-impact fall causes. IEEE Sens J 24:24086–24095
14. Chi T-H, Liu K-C, Hsieh C-Y, Tsao Y Chan C-T (2023) Prefallkd: pre-impact fall detection via cnn-vit knowledge distillation. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.
15. Munappy A, Bosch J, Olsson HH, Arpteg A Brinne B (2019) Data management challenges for deep learning. In 2019 45th Euromicro conference on software engineering and advanced applications (SEAA) (pp. 140-147). IEEE.

16. Gudivada V, Apon A, Ding J (2017) Data quality considerations for big data and machine learning: going beyond data cleaning and transformations. Int J Adv Softw 10:1–20
17. Aziz O, Russell CM, Park EJ Robinovitch SN (2014) The effect of window size and lead time on pre-impact fall detection accuracy using support vector machine analysis of waist mounted inertial sensor data. In 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (pp. 30-33). IEEE.
18. Wagner J, Mazurek P Morawski RZ (2017) Regularized numerical differentiation of depth-sensor data in a fall detection system. In 2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA) (pp. 234-236). IEEE.
19. Shrivastava R Pandey M (2021) Ensemble of multiple classifiers for accelerometer based human fall detection. In Computer Networks and Inventive Communication Technologies: Proceedings of Third ICCNCT 2020 (pp. 865-874). Singapore: Springer Nature Singapore.
20. Stone EE, Skubic M (2014) Fall detection in homes of older adults using the microsoft kinect. IEEE J Biomed Health Inform 19:290–301
21. Kwolek B, Kepski M (2014) Human fall detection on embedded platform using depth maps and wireless accelerometer. Comput Methods Programs Biomed 117:489–501
22. Martínez-Villaseñor L et al (2019) Up-fall detection dataset: a multimodal approach. Sensors 19:1988
23. Sandler M, Howard A, Zhu M, Zhmoginov A Chen L-C (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).
24. Liu W et al (2016) Ssd: single shot multibox detector. In European conference on computer vision (pp. 21-37). Cham: Springer International Publishing.
25. Fayad M, Hachani M-Y, Mostefaoui A, Chouali S, Yahiaoui R (2022) A lightweight kinect based deep learning approach, elderly fall detection. In Proceedings of the 20th ACM International Symposium on Mobility Management and Wireless Access (pp. 89-95).
26. Auvinet E, Rougier C, Meunier J, St-Arnaud A, Rousseau J (2010) Multiple cameras fall dataset. DIRO-Université de Montréal Tech Rep 1350:24
27. Charfi I, Miteran J, Dubois J, Atri M, Tourki R (2013) Optimised spatio-temporal descriptors for real-time fall detection: comparison of svm and adaboost based classification. J Electron Imag (JEI) 22:17
28. Medrano C, Igual R, Plaza I, Castro M (2014) Detecting falls as novelties in acceleration patterns acquired with smartphones. PLoS ONE 9:e94811
29. Jokanovic B, Amin MG Ahmad F (2016) Effect of data representations on deep learning in fall detection. In 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM) (pp. 1-5). IEEE.
30. Liu K-C, Hsieh C-Y, Huang H-Y, Hsu SJ-P, Chan C-T (2019) An analysis of segmentation approaches and window sizes in wearable-based critical fall detection systems with machine learning models. IEEE Sens J 20:3303–3313
31. Sucerquia A, López JD, Vargas-Bonilla JF (2017) Sisfall: a fall and movement dataset. Sensors 17:198
32. Ramachandran A, Ramesh A Karuppiah A (2020) Evaluation of feature engineering on wearable sensor-based fall detection. In 2020 International Conference on Information Networking (ICOIN) (pp. 110-114). IEEE.
33. Šeketa G, Pavlaković L, Džaja D, Lacković I, Magjarević R (2021) Event-centered data segmentation in accelerometer-based fall detection algorithms. Sensors 21:4335
34. Özdemir AT, Barshan B (2014) Detecting falls with wearable sensors using machine learning techniques. Sensors 14:10691–10708
35. Saleh M, Abbas M, Le Jeannes RB (2020) Fallalld: an open dataset of human falls and activities of daily living for classical and deep learning applications. IEEE Sens J 21:1849–1858
36. Han J, Kamber M Pei J (2011) Data mining: concepts and techniques. In: The Morgan Kaufmann (ed) series in data management systems, 3rd edn. Elsevier, p 83–124
37. Zheng A, Casari A (2018) Feature engineering for machine learning: principles and techniques for data scientists. O'Reilly Media. Inc
38. Brownlee J (2018) Develop deep learning models on theano and tensorflow using keras. Deep Learning with Python. Jason Brownlee, Melbourne
39. Galli S (2020) Python feature engineering cookbook: over 70 recipes for creating, engineering, and transforming features to build machine learning models. Packt Publishing Ltd

40. Jayalakshmi T, Santhakumaran A (2011) Statistical normalization and back propagation for classification. Int J Comput Theory Eng 3:1793–8201
41. Huang L, Zhao J, Zhu B, Chen H, Broucke SV (2020) An experimental investigation of calibration techniques for imbalanced data. Ieee Access 8:127343–127352
42. Rout N, Mishra D Mallick MK (2018) Handling imbalanced data: a survey. In International proceedings on advances in soft computing, intelligent systems and applications: Asisa 2016 (pp. 431-443). Singapore: Springer Singapore.
43. Özdemir AT (2016) An analysis on sensor locations of the human body for wearable fall detection devices: principles and practice. Sensors 16:1161
44. Ntanasis P, Pippa E, Özdemir AT, Barshan B Megalooikonomou V (2016) Investigation of sensor placement for accurate fall detection. In International Conference on Wireless Mobile Communication and Healthcare (pp. 225-232). Cham: Springer International Publishing.
45. Nahar N, Hossain MS Andersson K (2020) A machine learning based fall detection for elderly people with neurodegenerative disorders. In International Conference on Brain Informatics (pp. 194-203). Cham: Springer International Publishing.
46. Bischl B et al (2023) Hyperparameter optimization: foundations, algorithms, best practices, and open challenges. Wiley Interdiscip Rev Data Mining Knowl Discov 13:e1484
47. Yang L, Shami A (2020) On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415:295–316
48. Alibrahim H Ludwig SA (2021) Hyperparameter optimization: comparing genetic algorithm against grid search and Bayesian optimization. In 2021 IEEE congress on evolutionary computation (CEC) (pp. 1551-1559). IEEE.
49. Buitinck L et al (2013) Api design for machine learning software: experiences from the scikit-learn project. arXiv preprint arXiv:1309.0238
50. Syarif I, Prugel-Bennett A, Wills G (2016) Svm parameter optimization using grid search and genetic algorithm to improve classification performance. TELKOMNIKA (Telecom Computing Electronics and Control) 14:1502–1509
51. Zivkovic Z, Van Der Heijden F (2006) Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern Recogn Lett 27:773–780
52. Bradski G (2000) The OpenCV Library. Dr. Dobb's J Softw Tools
53. Yajai A, Rodtook A, Chinnasarn K Rasmequan S (2015) Fall detection using directional bounding box. In 2015 12th International joint conference on computer science and software engineering (JCSSE) (pp. 52-57). IEEE.
54. Fayad M et al (2023) Fall detection approaches for monitoring elderly healthcare using kinect technology: a survey. Appl Sci 13:10352

Authors and Affiliations

Moustafa Fayad[1] · Mohammed Amine Merzoug[2] · Ahmed Mostefaoui[3] · Ernesto Damiani[4] · Gabriele Gianini[5] · Réda Yahiaoui[1]

✉ Gabriele Gianini
gabriele.gianini@unimib.it

Moustafa Fayad
moustafa.fayad@univ-fcomte.fr

Mohammed Amine Merzoug
amine.merzoug@univ-batna2.dz

Ahmed Mostefaoui
ahmed.mostefaoui@univ-fcomte.fr

Ernesto Damiani
ernesto.damiani@unimi.it

Réda Yahiaoui
reda.yahiaoui@univ-fcomte.fr

[1] Université Marie et Louis Pasteur, 25030 Besançon, France

[2] University of Batna 2, 05078 Batna, Algeria

[3] Université Marie et Louis Pasteur, 90000 Belfort, France

[4] Università degli Studi di Milano, 20133 Milan, Italy

[5] Università degli Studi di Milano-Bicocca, 20126 Milan, Italy