Received August 20, 2019, accepted September 29, 2019, date of publication October 16, 2019, date of current version November 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2947739

A Machine Learning Approach for Fall Detection Based on the Instantaneous Doppler Frequency

ALI CHELLI, (Member, IEEE), AND MATTHIAS PÄTZOLD, (Senior Member, IEEE)
Faculty of Engineering and Science, University of Agder, 4898 Grimstad, Norway

Corresponding author: Ali Chelli (ali.chelli@uia.no)

This work was supported by the WiCare Project through the Research Council of Norway under Grant 261895/F20.

ABSTRACT Modern societies are facing an ageing problem that is accompanied by increasing healthcare costs. A major share of this ever-increasing cost is due to fall-related injuries, which urges the development of fall detection systems. In this context, this paper paves the way for the development of radio-frequency-based fall detection systems, which do not require the user to wear any device and can detect falls without compromising the user's privacy. For the design of such systems, we present an activity simulator that generates the complex path gain of indoor channels in the presence of one person performing three different activities: slow fall, fast fall, and walking. We have developed a machine learning framework for activity recognition based on the complex path gain. Additionally, we propose a novel method that accurately estimates the instantaneous Doppler frequency (IDF) from the complex path gain. Then, we extract six features from the IDF and provide the feature vector as input to the classifier, which has to predict the user's activity. We assess the recognition accuracy of four different classification algorithms: K-nearest neighbors (KNN), decision tree, artificial neural network (ANN), and cubic support vector machine (SVM). Our analysis reveals that the KNN, decision tree, ANN, and cubic SVM achieve an overall recognition accuracy of 86.1%, 94%, 98.9%, and 99.9%, respectively. The best performing algorithm, cubic SVM, has a fall detection accuracy of 100% with zero false alarms and zero undetected falls, which represents the best achievable performance. By comparing our fall detection system with existing ones in the literature, we demonstrate the superiority of our proposed solution.

INDEX TERMS Fall detection, machine learning, activity recognition, instantaneous Doppler frequency, complex path gain, feature extraction, hypothesis testing.

I. INTRODUCTION

Advances in the diagnosis and treatment of diseases are leading to increasing life expectancy in modern societies, which has resulted in an aging population and new societal challenges. In western countries, most of these elderly people prefer to live in their own houses and are therefore prone to fall accidents. Statistics show that falls are the main cause of fatal and non-fatal injuries for older people [1]. Every 19 minutes, an older adult dies from a fall, while 3 million elders are treated for fall injuries [2]. In 2015, the total cost of fall-related medical care exceeded 50 billion dollars [3]. After a fall, rapid medical care can significantly diminish the potential damage from fall injuries, which could reduce the number of casualties from falls and lower healthcare costs. Therefore, fall detection systems that can detect and report falls as fast as possible are of great importance. In recent years, a plethora of fall detection systems have been developed using different approaches. Existing fall detection systems can be divided into three main classes [4]: (i) context-aware systems, (ii) wearable device-based systems, and (iii) radio-frequency (RF)-based systems.

Context-aware systems leverage sensors deployed in specific monitoring areas. Monitoring sensors include mainly cameras, microphones, and pressure sensors. In camera-based fall detection systems, a classification algorithm is applied to recorded video to detect falls [5]. Although such an approach has a high fall detection accuracy, it suffers from many limitations, such as high deployment costs, limited monitoring area, and violation of the user's privacy.

In wearable-based fall detection systems, the user must wear a device capable of collecting acceleration data, while the user is performing daily activities. This recorded acceleration can be analyzed in real time by an activity recognition

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Zhang.

algorithm to determine if a fall has occurred. The use of wearable devices for activity recognition and fall detection has been extensively investigated in the literature [6]–[8]. Wearable-based fall detection systems have numerous advantages: (i) they can detect falls without violating the user's privacy, (ii) they have a low cost, and (iii) they have an unlimited monitoring area. However, if the user forgets to wear the device or avoids wearing the device for comfort reasons, fall detection becomes impossible, which is the biggest disadvantage of wearable-based systems.

The third category of fall detection systems is based on RF techniques and are therefore referred to as RF-based fall detection systems. These systems are device-free and do not require the user to wear a device. These systems do not compromise the users' privacy, but have a limited monitoring area. In RF-based fall detection systems, a machine learning algorithm analyzes the received RF signal to extract the fingerprint of human activity and to detect falls. In [9], falls are detected based on the channel state information (CSI) obtained from a Wi-Fi network card. A three-dimensional (3D) motion tracking system using wireless signals to detect falls of a single person is developed in [10]. Several research studies take advantage of the wide spread use of Wi-Fi systems to develop solutions for indoor localization, gesture recognition, and motion detection, while few focus on fall detection. Numerous studies leverage Wi-Fi signals for indoor localization [11], [12]. The variations in the received signal strength of Wi-Fi are utilized for gesture recognition in [13]. The authors of [14] propose a method for human tracking based on wireless signals, but this method cannot be used for human activity recognition.

In this paper, our primary objective is to recognize human activity and detect falls by applying machine learning techniques on the instantaneous Doppler frequency (IDF). This objective is achieved through the following steps. First, we develop an activity simulator that generates the complex path gain of indoor channels in the presence of a single moving person. This RF-activity simulator is developed by building on the results of [15]. In our model, the person is replaced by a single moving scatterer representing the movement of the head. We simulate mainly 3 activities: walking, slow fall, and fast fall. The activity simulator generates time-series of complex path gains pertaining to each of these activities by randomizing the trajectory of the participants, their heights, and their speeds within typical interval ranges. We propose a signal processing technique to estimate the IDF from the generated complex path gain. Using supervised learning, we train classifiers to recognize the type of performed activity based on the IDF. We evaluate the recognition accuracy of four classification algorithms: K-nearest neighbors (KNN), decision tree, artificial neural network (ANN), and cubic support vector machine (SVM).

Note that the IDF has never been used before for activity recognition. As the user performs an activity, his/her body parts move and cause a Doppler shift in the received radio signal, which is captured by the IDF. The latter illustrates how the Doppler shift varies as time progresses. This time-variant aspect of the IDF is due to the time-variant nature of the speed, angles of departure, and angles of arrival. As such, the IDF contains the fingerprints of the user's movement and represents a promising data source to recognize the user's activity accurately. The IDF can be estimated from the complex path gain as explained in Section IV. A model of the complex path gain is provided in Section III together with the physical meaning of the complex path gain and how it can be measured using a Wi-Fi network card.

The main contributions of our paper are as follows:

- We develop an activity simulator that generates the complex path gain of indoor channels in the presence of one person performing three different activities: slow fall, fast fall, and walking. This approach represents a paradigm shift from the current experimental-based design to a software-based design. In software-based design, mathematical models are used to describe the trajectory of the body for various activities. An RF-activity simulator, takes this trajectory as input and generates the corresponding radio signal that contains the fingerprint of the user movement. By using a software based approach, it is not needed to carry real-field experiments to collect RF-data associated with different activities and users, as this data can be generated by the activity simulator. One of the main advantages of this software-based approach that it allows speeding up the development of RF-based human activity recognition systems.
- We propose a novel method to extract the IDF with high accuracy from the complex path gain. The IDF can be utilized to predict accurately the user' activity. The use of the IDF to detect human activity is motivated by its high sensitivity to the movement compared to the amplitude of the complex path gain.
- We build a machine learning framework for activity recognition based on the IDF.
- We assess the recognition accuracy of four different classification algorithms: KNN, decision tree, ANN, and cubic SVM.
- We consider different subsets of features and investigate their impact on the activity recognition accuracy and precision with respect to the four classification algorithms.
- We demonstrate that the KNN, decision tree, ANN, and cubic SVM algorithms achieve an overall accuracy of 86.1%, 94%, 98.9%, and 99.9%, respectively. These results are achieved by extracting only six features from the IDF, which makes the proposed solution accurate, while its computational costs are low.

The remainder of the paper is organized as follows. Section II provides an overview of the proposed activity simulator, while the expression of the complex path gain is derived in Section III. In Section IV, we discuss the pre-processing methods used to estimate the IDF. Section V describes the machine learning framework and how we use it to recog-

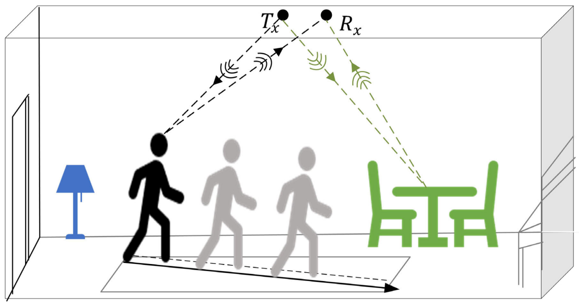


FIGURE 1. A typical 3D indoor propagation scenario with one moving person walking straight forward.

nize different types of activities. In Section VI, we assess the performance of the proposed framework and discuss the obtained experimental results. Finally, Section VII offers concluding remarks.

II. ACTIVITY SIMULATOR OVERVIEW

The aim of this paper is to develop a machine learning framework that recognizes the user activity based on the complex path gain. Testing of such machine learning solutions requires to perform many experimental trials with different persons. Moreover, a huge amount of data must be collected, processed, and stored. This experimental-based approach is time consuming and costly. For instance, the data collection process in [9] involved 10 participants, each performing 4 activities at eight different locations. At each location, the activity is repeated 25 times. Considering that the data collected in each trial lasts for 1 minute, the total recorded data has a duration of 8000 minutes, which corresponds to 5.5 days of actual recorded data. If we consider that we collect in each day 4 hours of recording, the data collection would last 34 days. Clearly the data collection process is time consuming. To solve this problem, we use a software-based approach in which an RF-activity simulator can generate synthetic data that emulates the complex path gain of an indoor 3D environment in the presence of a single moving person as illustrated in Fig. 1. The RF-activity simulator takes as input the user' trajectory and provides as output the complex path gain. In this section, we provide an overview of the activity simulator and describe how the data was generated for different activities.

In our simulator, a person is modelled by a single scatterer representing the head.[1] As the user performs his activity, the position of this scatterer changes over time. In the literature, we find head trajectory models for some activities, such as walking [16]. This trajectory is fed to the RF-activity simulator, which uses the plane wave propagation theory to compute the complex path gain at the receiver. To make the synthetic data generic and realistic, we consider in our simulation 30 different participants with different heights, walking with various speeds along different paths. These participants perform three activities: walking, slow fall, and fast fall. During the data collection, only one person moves inside a room with a size of 5 m by 10 m.

Inside the room, we have static objects (e.g., furniture and walls) and one moving person. The room is equipped with a transmitter and a receiver denoted by T_x and R_x, respectively, operating in the 2.4 GHz Wi-Fi band. The transmitter T_x emits electromagnetic waves that propagate in the indoor environment. These waves are reflected by the objects located in the room before arriving at the receiver R_x as shown in Fig. 1. All the objects in the room are static except for the moving person. A Doppler shift of the transmitted signal is caused by the person's movement which affects the received RF signal. Thus, the received signal contains the fingerprints of the user activity. With signal processing techniques, we can extract the IDF from the complex path gain. Finally, by applying machine learning, we can recognize the user activity based on the IDF.

The height L of each participant is randomly generated and evenly distributed in the interval [1.5 m, 1.9 m]. The stride length L_s can be computed as $L_s = 0.413 * L$ which implies that L_s is uniformly distributed in the interval [0.62 m, 0.78 m]. Each participant follows a path composed of 60 straight lines. The start and end points of each line are randomly generated. Hence, different participants follow different paths which makes the collected data more generic and more realistic. For each participant, the complex path gain is recorded for 120 s and then divided into 30 buffers of length 4 s. Each data buffer is labeled with the corresponding actual activity.

For the walking activity, the subject moves with a constant speed along a straight line for 2 s, then changes the direction of motion and walks along a new straight line for 2 s. For the activity "walking", the path of each participant is composed of 20 straight lines with random starting and ending points. Along this path, the speed of the participant is kept constant. In each walking scenario, the speed is the outcome of a random generator. The speed values are uniformly distributed in the interval [0.8 m/s, 1.2 m/s] [17].

A fall comprises three main stages: (i) the pre-fall, (ii) the fall, and (iii) the post-fall. In the pre-fall stage, the participant is walking with a constant speed, while for the post-fall stage the participant is lying on the ground with zero velocity. During the fall stage, the body posture changes gradually from vertical to horizontal and the speed increases until reaching its maximum value right before the body hits the ground. The walking speed in the pre-fall phase for a fast fall is equal to 0.9 m/s and for slow fall is 1.2 m/s. The duration of the fast fall and the slow fall are equal to 1 s and 2 s [18], respectively. The maximum speeds for the fast fall and the slow fall are equal to 2.6 m/s and 1.8 m/s, respectively [19]. These parameters results in a maximum achievable Doppler shift of 19.2 Hz for walking, 28.8 Hz for slow fall, and 41.6 Hz for fast fall.

Using the described mobility model, we can compute the time-variant coordinates of the persons as they move inside

[1] In our investigation, both the transmitter and the receiver are attached to the ceiling. Therefore, as the person moves, the head movement has the dominant effect on the radio waves, since it is the nearest body part to the transmitter and the receiver.

the room. Knowing the positions of the transmitter T_x and the receiver R_x, we can determine the Doppler frequency associated with the human movement and subsequently compute the complex path gain characterizing the wireless link from the transmitter T_x to the receiver R_x. This complex path gain is the output of the RF-activity simulator, which is provided as input for the machine learning framework. This latter has to predict the person's activity based on the complex path gain.

III. COMPLEX PATH GAIN

The room is equipped with a transmitter T_x and a receiver R_x having as coordinates (x^T, y^T, z^T) and (x^R, y^R, z^R), respectively. The transmitter T_x emits electromagnetic waves that propagate in the indoor environment. These waves are reflected by the objects located in the room before arriving at the receiver R_x. With this setup, we can measure the CSI of the indoor propagation channel. The complex path gain can be determined from this measured CSI. During the data collection, all objects in the room are static, while a single participant is moving inside the room. We model this moving person by a single moving scatterer S^M corresponding to the movement of the head.

For the walking activity, the person moves with a constant speed along a path consisting of 20 straight lines. The start and the end points of each line are generated randomly. For a given line in this path, the direction of motion α_v and the speed of motion v_h in the horizontal plane are constants. If the person walks along a new line, the direction of motion α_v is updated to a new value, while the speed v_h in the horizontal $x - y$ plane remains the same.

During walking, the time-variant positions $x(t)$ and $y(t)$ along the x- and y-axis of the moving scatterer S^M can be expressed as

$$x(t) = x^M + v_h \cos(\alpha_v)\, t \tag{1}$$

$$y(t) = y^M + v_h \sin(\alpha_v)\, t \tag{2}$$

where (x^M, y^M) denotes the initial position of S^M in the $x - y$ plane at time $t = 0$. The time-variant position $z(t)$ of the scatterer S^M (the head) along the z-axis can be written as [16]

$$z(t) = h_{\text{step}} \cos\left(2\pi f_{\text{step}} t\right) + h_{\text{head}} \tag{3}$$

where h_{head}, h_{step}, and f_{step} stand for the body height, the step height of the head during walking, and the walking frequency, respectively. The walking frequency can be expressed as $f_{\text{step}} = v_h/L_s$, where L_s denotes the stride length [17]. The value of the stride length is proportional to the body height. The vertical speed $v_v(t)$ along the z-axis can be determined by deriving $z(t)$ with respect to t, i.e., $v_v(t) = dz(t)/dt = -2\pi f_{\text{step}} h_{\text{step}} \sin(2\pi f_{\text{step}} t)$. It is worth mentioning that the validity of the head trajectory model in (3) was confirmed by fitting the trajectory model to real-world data obtained from tracking the head trajectory using video recording [16].

In [16], the head trajectory was obtained by processing successive frames captured by a camera as the person walks. Applying video processing techniques, the real head trajectory data was computed [16]. A sinusoidal model for the movement of the head was then assumed with three model parameters. A fitting problem was formulated by defining an objective function that measures the error between the model and the actual data [16]. The optimal values of the model parameters were determined by minimizing the objective error function using non-linear optimization methods [16]. The head trajectory model obtained through this fitting process is provided in (3). The simulated walking trajectories are generated according to equations (1), (2), and (3), which makes the simulated trajectories fit with real-world data.

We model the static objects in the room, such as walls and furniture, by N fixed scatterers S_n^F $(n = 1, 2, \ldots, N)$. Thus, the channel transfer function $H(t, f')$ can be expressed as

$$H(t, f') = c_m \exp\left[j\theta_m(t)\right] + \sum_{n=1}^{N} c_n \exp\left(j\theta_n\right) \tag{4}$$

where the symbols c_m and c_n denote the path gains associated with the moving scatterer and nth fixed scatterer, respectively. The phases θ_n are independent identically distributed random variables with uniform distribution in the interval $[0, 2\pi)$. The phase $\theta_m(t)$ associated with the moving scatterer is time-variant and can be computed as [20]

$$\theta_m(t) = \theta_m - 2\pi f' \tau_m'(t) \tag{5}$$

where θ_m represents the initial phase at $t = 0$, which is uniformly distributed in the interval $(0, 2\pi]$. The symbol f' refers to the frequency, while $\tau_m'(t)$ is the time-variant delay associated with the moving scatterer S^M. Alternatively, the phase can be expressed in terms of the IDF $f_m(t)$ as [21]

$$\theta_m(t) = 2\pi \int_{-\infty}^{t} f_m(u)du = \theta_m + 2\pi \int_{0}^{t} f_m(u)du. \tag{6}$$

The IDF $f_m(t)$ can be determined as [15]

$$\begin{aligned} f_m(t) = -f_{\max}(t) \Big\{ &\cos\left(\beta_v(t)\right) \Big[\cos\left(\beta^T(t)\right) \\ &\cos\left(\alpha^T(t) - \alpha_v\right) + \cos\left(\beta^R(t)\right) \cos\left(\alpha_v - \alpha^R(t)\right) \Big] \\ &+ \sin\left(\beta_v(t)\right) \Big[\sin\left(\beta^T(t)\right) + \sin\left(\beta^R(t)\right) \Big] \Big\} \end{aligned} \tag{7}$$

where

$$f_{\max}(t) = \frac{v(t)}{c_0} f' \tag{8}$$

is the maximum Doppler frequency, with c_0 being the speed of light. The term $v(t)$ refers to the speed of motion, which can be expressed as $v(t) = \sqrt{v_v(t)^2 + v_h(t)^2}$. The symbols $\beta^T(t)$, $\alpha^T(t)$, $\beta^R(t)$, $\alpha^R(t)$, and $\beta_v(t)$ denote the elevation angle of departure, azimuth angle of departure, elevation angle of arrival, azimuth angle of arrival, and vertical angle of motion, respectively. All these angles are illustrated in Fig. 2. According to [15], all these angles can be computed knowing the positions of the transmitter T_x and receiver R_x, the time variant position $(x(t), y(t), z(t))$, and speed $v(t)$ of the scatterer S^M. The 3D geometrical model provided in Fig. 2 is a more abstract representation of the 3D propagation scenario illustrated in Fig. 1.

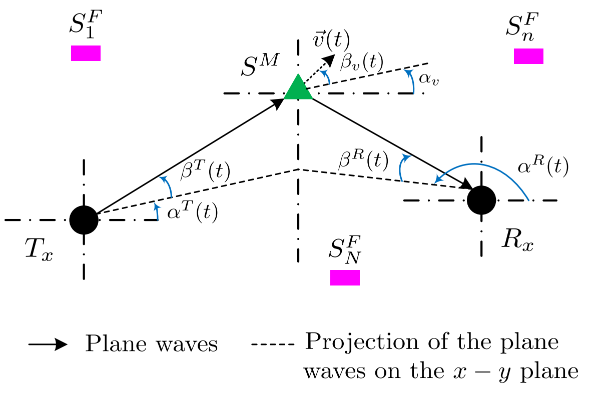


FIGURE 2. A 3D geometrical model of indoor propagation scenario with N fixed scatters S_n^F ($n = 1, 2, \ldots, N$) and one moving scatterer S^M.

Using a Wi-Fi network card, it is possible to collect CSI data that contains the fingerprint of the user's activity. In general, the collected CSI data contains K rows each of which is associated with a subcarrier k, which will be denoted by $\mu_k(t)$. In fact, the kth row of the CSI data corresponds to the channel transfer function evaluated at the kth subcarrier f'_k, i.e., $\mu_k(t) = H(t, f'_k)$. The expression of f'_k is given by $f'_k = f'_0 + k\Delta f'$, where f'_0 stands for the carrier frequency, k denotes the subcarrier index, and $\Delta f'$ refers to the subcarrier bandwidth. Henceforth, the quantity $\mu_k(t)$ will be referred to as the complex path gain. For ease of notation, the subscript k will be dropped and the complex path gain will be denoted as $\mu(t)$. The expression of $\mu(t)$ can be obtained as

$$\mu(t) = c_m \exp\left[j\theta_m(t)\right] + \sum_{n=1}^{N} c_n \exp\left(j\theta_n\right). \tag{9}$$

Our assumptions about the speed profile, body height, head trajectory during the walk, maximum speed during the fall, and the duration of the fall are all supported by measurements published in the literature [16], [18], [19]. To make our synthetic data generic, we simulated the activity of 30 participants with different heights, different walking speeds, and following different paths while performing their activities. This makes the data statistically uncorrelated and reflects the diversity generally observed in real-world data. All these facts support the argument that our synthetic data is a reasonable approximation of real-word data.

In this paper, we use a software-based design as opposed to experimental-based design adopted in existing studies. In software-based design, mathematical models are used to describe the trajectory of the body for various activities. An RF-activity simulator, takes this trajectory as input and generates the corresponding radio signal that contains the fingerprint of the user movement. This RF-activity simulator is flexible and allows generating RF-data pertaining to different activities, speed profiles, and users' height. One of the main advantages of this software-based approach that it allows speeding up the development of RF-based human activity recognition systems.

IV. DATA PRE-PROCESSING

A. DATA FILTERING

Using the developed RF-activity simulator, we generate the complex path gain data associated with the activity of 30 different individuals. These data contain the fingerprint of the activity performed by the user. To determine which activity the user is performing during the data collection, we must first pre-process the measured complex path gain $\mu(t)$. As it is obvious from (9), the complex path gain $\mu(t)$ comprises the contribution of both moving and fixed scatterers. Intuitively, the classification accuracy would improve if it were based solely on the complex path gain component associated with the moving scatterer. Consequently, it is important to remove the multipath components pertaining to the fixed scatterers.

Note that the contribution of the fixed scatterers to the channel gain is an unknown constant term. This implies that this term is a zero-frequency component. Thus, we can remove the contribution of all fixed scatterers by applying a high-pass filter to the complex path gain $\mu(t)$. To this end, we use a Chebyshev filter of Type II [22] with a stopband attenuation of 40 dB and a stopband frequency of 0.05 Hz. The choice of a Type II Chebyshev filter is motivated by two main reasons. First, Type II Chebyshev filters are sharper than Butterworth filters, which makes it possible to filter out the contribution of all fixed scatterers [22]. Second, Type II Chebyshev filters can extract the contribution of the moving scatterers to the complex path gain $\mu(t)$ with very minor distortions, since Type II Chebyshev filters have no ripples for frequencies larger than the passband frequency [22]. Subsequently, the filtered complex path gain $\hat{\mu}(t)$ is a good approximation of the contribution of the moving scatterer, i.e.,

$$\hat{\mu}(t) = \mu(t) * h(t) \approx c_m \exp\left[j\theta_m(t)\right], \tag{10}$$

where $h(t)$ refers to the impulse response of the Chebyshev filter of Type II.

B. IDF ESTIMATION

In this section, we propose a method for estimating the IDF of non-stationary signals. Our starting point is the filtered complex path gain $\hat{\mu}(t)$, which can be regarded as a mono-component signal with time-variant frequency. Our proposed estimation method comprises the following steps: leftmargin=0.5cm

1) Extract the phase $\hat{\theta}_m(t)$ of the signal $\hat{\mu}(t)$.
2) Determine an estimate $\hat{f}_m(t)$ of the IDF as $\hat{f}_m(t) = \frac{d(\hat{\theta}_m(t))}{2\pi\, dt}$.
3) Apply a Gaussian smoothing filter to remove the ripples in the estimated IDF $\hat{f}_m(t)$. The obtained IDF after smoothing is denoted as $\tilde{f}_m(t)$.

Next, we numerically evaluate the accuracy of the proposed estimation method. We consider three data buffers associated with a walking, slow fall, and fast fall scenario. Fig. 3 illustrates both the estimated and the exact IDF $f_m(t)$ for a walking, slow fall, and fast fall scenario. This figure shows

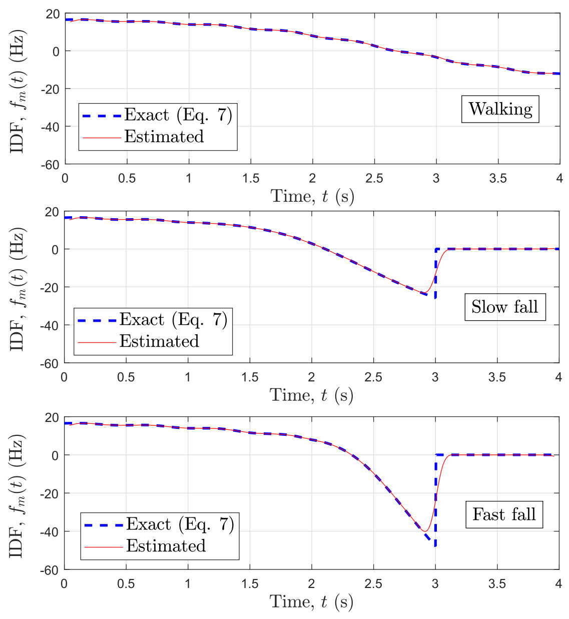


FIGURE 3. IDF $f_m(t)$ for a walking, slow fall, and fast fall scenario.

the high accuracy of the proposed estimation method in determining the IDF $f_m(t)$. For the slow fall scenario in Fig. 3, the participant walks for 1 s, and then starts falling. The fall lasts for 2 s, i.e., from $t = 1$ s to $t = 3$ s. For the fast fall scenario, the participant walks for 2 s, and then starts falling. The fall lasts for 1 s, i.e., the fall time interval is [2 s, 3 s]. After the fall, the person remains on the ground without any movement from $t = 3$ s to $t = 4$ s. In this post-fall phase, the IDF $f_m(t)$ becomes zero, since the participant does not move. For the walking scenario, the IDF is positive for $t \in [1\text{ s}, 2.6\text{ s}]$, which implies that in this phase the user is moving towards the transmitter/receiver.[2] The IDF is negative for $t \in [2.6\text{ s}, 4\text{ s}]$, which indicates that the user is moving away from the transmitter/receiver.

V. MACHINE LEARNING FRAMEWORK

This work aims to recognize three different human activities based on the recorded complex path gain. The considered activities are walking, slow fall, and fast fall. Towards our objective, we use a supervised machine learning approach and train a set of classification algorithms to recognize human activity. The input signal provided to the classifier is a recorded complex path gain in an indoor environment where a single person performs three activities. The various building blocks of the activity recognition scheme are illustrated in Fig. 4.

The first block is the RF-activity simulator described in Sections II and III. The output of the activity simulator is the complex path gain $\mu(t)$ that captures the impact of the fixed scatterers in the indoor environment as well as the impact of the moving person. The complex path gain $\mu(t)$ is then fed to the data pre-processing block. This block first

[2]In our simulation scenario, the transmitter and the receiver are collocated.

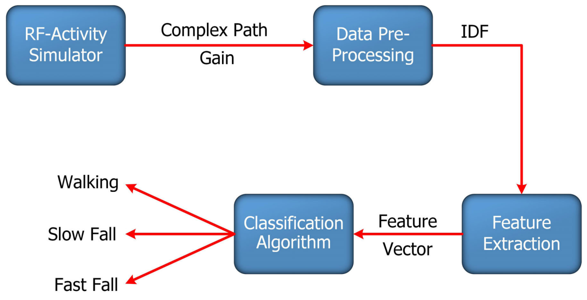


FIGURE 4. Machine learning framework for activity recognition.

TABLE 1. List of features.

Feature name	Feature number
Mean	Feature 1
Variance	Feature 2
RMS	Feature 3
Maximum of the absolute value	Feature 4
IDF slope during fall	Feature 5
IDF slope after fall	Feature 6

removes the impact of fixed scatterers using a high-pass filter, as described in Section IV-A. Afterwards, we estimate the IDF using the method proposed in Section IV-B. The IDF becomes the input signal for the feature extraction block, which extracts six features from the IDF, namely the mean value, variance, root mean square (RMS), maximum of the absolute value, slope during fall, and slope after fall. These features are then stacked in a vector, called feature vector. The classification algorithm determines the type of the performed activity based on the feature vector. In the following, we discuss in more detail the set of extracted features and explain how statistical tools can be used to determine to what extent these features can improve the recognition accuracy. Additionally, we describe the classification algorithm block.

A. FEATURE EXTRACTION

If the classification algorithm uses the raw data of the IDF to determine the type of the performed activity, the obtained results would have very poor accuracy. To deal with this issue, we must extract a set of features that captures a quantitative description of each activity and allows us to distinguish different activities. We extract six features from the IDF. These features and their corresponding feature number are listed in Table 1.

The third feature, which is the RMS of the IDF $\tilde{f}_m(t)$ can be determined as

$$\tilde{f}_m^{rms} = \sqrt{\frac{1}{T}\int_0^T \left[\tilde{f}_m(t)\right]^2 dt} \qquad (11)$$

where T is the length of the buffer which is equal to 4 s.

The fifth feature that we extract from the IDF is the IDF slope during fall which is denoted as S_{fall}. During the fall, the speed increases until the maximum value is reached just before the body touches the ground. This coincides with an increasing absolute value of the IDF that reaches its maximum as the body touches the ground. By determining the time instant t_f, at which the absolute value of the IDF is maximum, we can localize the time interval during which the fall occurs. This fall time interval is located right before the time instant t_f. Since the fall interval is larger or equal to 1 s, we can easily select two time instants t_{f_0} and t_{f_1} that belong to the fall interval. Subsequently, we can compute the slope of the IDF during the fall S_{fall}. In fact, the IDF during the fall is well approximated by a linear function whose slope can be determined with two values of the IDF measured at two distinct time instants t_{f_0} and t_{f_1} belonging to the fall phase, i.e.,

$$S_{\text{fall}} = \frac{\tilde{f}_m(t_{f_1}) - \tilde{f}_m(t_{f_0})}{t_{f_1} - t_{f_0}}. \tag{12}$$

For a walking scenario, there is actually no fall event and the feature slope during the fall does not reflect a meaningful physical parameter. However, we compute this feature for a walking scenario in the way described above. Our expectation is that the absolute value of the slope would be large for falls and small for walking scenarios. Moreover, larger absolute values of the slope are expected for fast falls compared to slow falls. This feature will improve the activity recognition accuracy as it will be shown in Section VI.

The IDF slope after the fall is the sixth feature that we extract from the IDF. This feature is referred to as $S_{\text{post-fall}}$. After the fall, the person remains on the ground and the value of the IDF becomes equal to zero. The post-fall interval is located in the time domain right after the instant t_f. Hence, by selecting two time instants t_{f_2} and t_{f_3} that belong to the post-fall interval, we can compute the slope after fall as

$$S_{\text{post-fall}} = \frac{\tilde{f}_m(t_{f_3}) - \tilde{f}_m(t_{f_2})}{t_{f_3} - t_{f_2}}. \tag{13}$$

Fig. 5 illustrates the histogram of the variance of the IDF $\tilde{f}_m(t)$ for walking, slow fall, and fast falls. This figure shows that the variance of $\tilde{f}_m(t)$ is mostly below 40 Hz^2 for walking. For slow fall and fast fall, the values of the variance of the IDF $\tilde{f}_m(t)$ are mainly within the intervals [10 Hz^2, 100 Hz^2] and [30 Hz^2, 210 Hz^2], respectively. Note that for fast fall, we obtain larger variance values compared to walking and slow fall. This is due to the fact that the IDF $\tilde{f}_m(t)$ has a larger dynamic range for fast falls compared to walking and slow falls.

Fig. 6 depicts the histogram of the maximum value of $|\tilde{f}_m(t)|$ for the activities walking, slow fall, and fast fall. From this figure, one can see that the maximum value of $|\tilde{f}_m(t)|$ is between 0 and 20 Hz for the activity walking, while for slow falls and fast falls the range of the maximum value of $|\tilde{f}_m(t)|$ is in the intervals [8 Hz, 24 Hz] and [12 Hz, 38 Hz], respectively. In fact, for fast falls, the velocity reaches larger values compared to slow fall and walking scenarios, which result in much larger values of $|\tilde{f}_m(t)|$.

Fig. 7 provides the histogram of the slope S_{fall} of the IDF for the activities walking, slow fall, and fast fall. This figure shows that this parameter is negative. Actually, during the fall, the head of the person moves towards the ground and away from the transmitter and the receiver which are located at the ceiling. This leads to decreasing values of the Doppler frequency and a negative slope of the IDF. The slope values are within the intervals [0 Hz/s, −12 Hz/s] for walking, [−8 Hz/s, −40 Hz/s] for slow falls, and [−20 Hz/s, −80 Hz/s] for fast falls. Note that the absolute value of the slope reaches larger values for fast falls compared

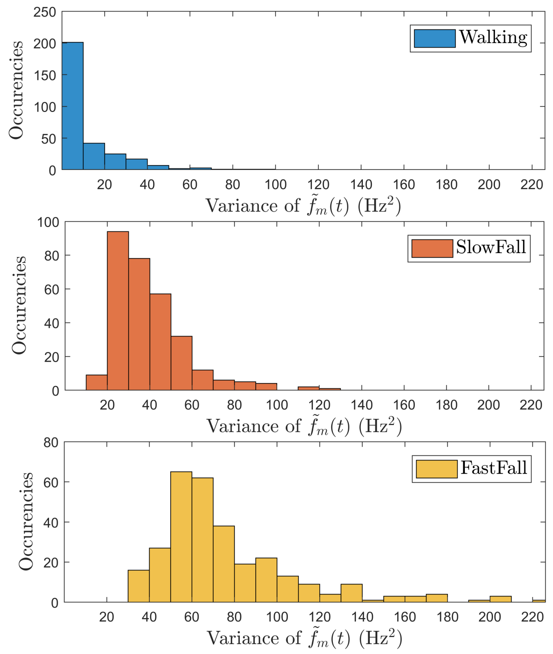


FIGURE 5. Histogram of the variance of the IDF $\tilde{f}_m(t)$ for the activities walking, slow fall, and fast fall.

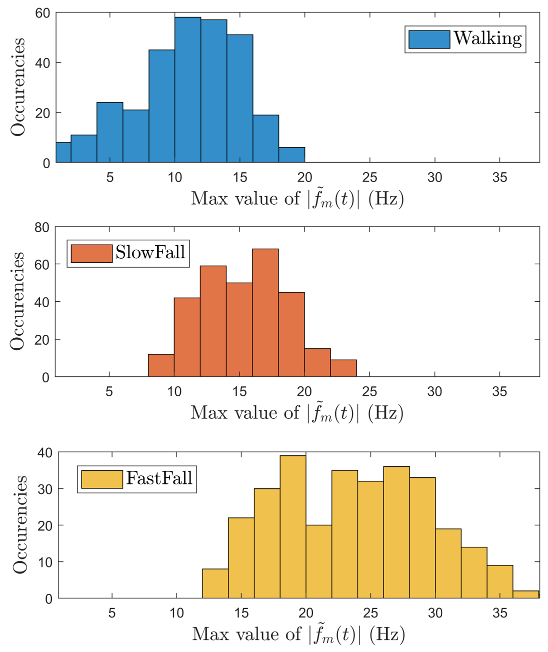


FIGURE 6. Histogram of the maximum value of $|\tilde{f}_m(t)|$ for the activities walking, slow fall, and fast fall.

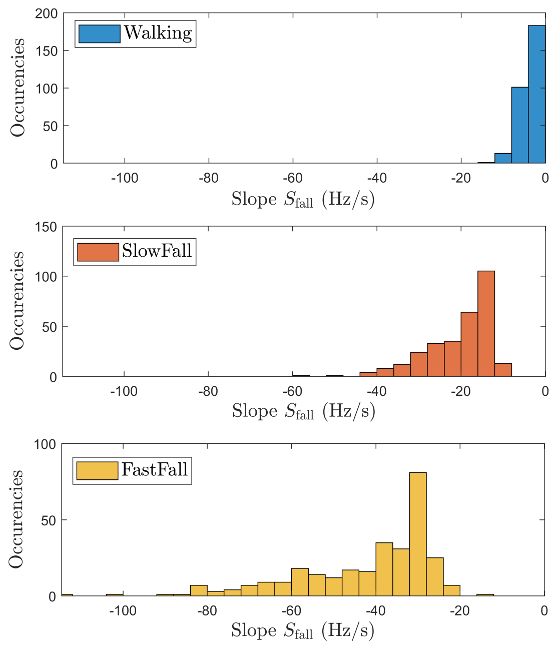


FIGURE 7. Histogram of the slope S_{fall} of the IDF for the activities walking, slow fall, and fast fall.

to slow falls. This is due to the fact that the speed during the fall is larger for fast falls compared to slow falls. The mean value of the slope equals -20 Hz/s and -42 Hz/s for slow and fast falls, respectively.

B. STATISTICAL ANALYSIS OF THE DATA

In this section, we perform a statistical analysis of the data to find the most relevant features that allow differentiating between activities. The data is divided into buffers which are labeled with an activity identity (ID). We have three activity IDs: 1, 2, and 3 corresponding to walking, slow fall, and fast fall, respectively. First, let us introduce some useful notation. Let $\mathcal{C}$ denote the set containing the three possible activity IDs, i.e., $\mathcal{C} = \{1, 2, 3\}$. The training data contains M input-output pairs $(\mathbf{x}(i), y(i))$. The vector $\mathbf{x}(i) \in \mathbb{R}^n$ is the feature vector associated with the ith training example. The scalar $y(i) \in \mathcal{C}$ indicates the class of the ith training example with $i = 1, \ldots, M$. The vector $\mathbf{x}(i) = \{x_1(i), \ldots, x_j(i), \ldots, x_n(i)\}$, where $x_j(i)$ is the value of feature j for the ith training example. Let $\mathbf{x}_j^c = \{x_j^c(1), \ldots, x_j^c(i), \ldots, x_j^c(M)\}$ where $x_j^c(i) = \{x_j(i)|y(i) = c \in \mathcal{C}\}$. The set $\mathbf{x}_j^c$ contains all the values of the feature j for a given class c. Thus, the sets $\mathbf{x}_j^1$, $\mathbf{x}_j^2$, and $\mathbf{x}_j^3$ contain all the values of the feature j pertaining to the classes walking, slow fall, and fast fall, respectively.

To determine if a feature j allows a classifier to distinguish two classes c_1 and c_2, it suffices to show that there is a significant statistical difference between the two vectors $\mathbf{x}_j^{c1}$ and $\mathbf{x}_j^{c2}$. Towards this aim, the statistical framework of hypothesis testing can be invoked [23]. This problem is similar to the problem of comparing two population means [23]. The vectors $\mathbf{x}_j^{c1}$ and $\mathbf{x}_j^{c2}$ can be regarded as two separate random samples from two populations. Let us denotes by μ_j^{c1} and μ_j^{c2} the true means of feature j for classes $c1$ and $c2$, respectively. The sample means $\bar{\mu}_j^{c1}$ and $\bar{\mu}_j^{c2}$ of feature j for classes $c1$ and $c2$ can be computed as

$$\bar{\mu}_j^{c1} = \frac{1}{M} \sum_{i=1}^{M} x_j^{c1}(i) \tag{14}$$

$$\bar{\mu}_j^{c2} = \frac{1}{M} \sum_{i=1}^{M} x_j^{c2}(i). \tag{15}$$

It is well known that the sample mean is an unbiased estimator of the true mean [23]. An unbiased estimator $[\hat{\sigma}_j^{c1}]^2$ of the first population variance $[\sigma_j^{c1}]^2$ is obtained as [23]

$$[\hat{\sigma}_j^{c1}]^2 = \frac{1}{M-1} \sum_{i=1}^{M} (x_j^{c1}(i) - \bar{\mu}_j^{c1})^2. \tag{16}$$

Similarly, an unbiased estimator $[\hat{\sigma}_j^{c2}]^2$ of the second population variance $[\sigma_j^{c2}]^2$ can be determined as

$$[\hat{\sigma}_j^{c2}]^2 = \frac{1}{M-1} \sum_{i=1}^{M} (x_j^{c2}(i) - \bar{\mu}_j^{c2})^2. \tag{17}$$

The null hypothesis H_0 states that the two population means are the same, i.e., $H_0 : \bar{\mu}_j^{c1} - \bar{\mu}_j^{c2} = 0$. The alternative hypothesis H_a states that the two population means are different, i.e., $H_a : \bar{\mu}_j^{c1} \neq \bar{\mu}_j^{c2}$. The test statistic for comparing the two means (also known as the z_{score}) can be expressed as [23]

$$z_{\text{score}} = \frac{\bar{\mu}_j^{c1} - \bar{\mu}_j^{c2}}{\sqrt{\frac{[\hat{\sigma}_j^{c1}]^2 + [\hat{\sigma}_j^{c2}]^2}{M}}}. \tag{18}$$

The p_{value} of the test statistic can be obtained as

$$p_{\text{value}} = 2Q(|z_{\text{score}}|). \tag{19}$$

where $Q(\cdot)$ is the complementary cumulative distribution function of the standard normal distribution known as the Q-function. If the p_{value} is less than the significance level α of the hypothesis test, i.e., $p_{\text{value}} < \alpha$, we say that we have enough evidence to reject H_0, and thus, there is a significant statistical difference between the vectors $\mathbf{x}_j^{c1}$ and $\mathbf{x}_j^{c2}$. This implies that the feature j allows us to distinguish the classes $c1$ and $c2$. Otherwise, if $p_{\text{value}} > \alpha$, we do not have enough evidence to reject H_0, which means that there is no significant statistical difference between the vectors $\mathbf{x}_j^{c1}$ and $\mathbf{x}_j^{c2}$. Thus, we cannot distinguish the classes $c1$ and $c2$ using feature j. The value of the significance level α is typically set to 0.01. This corresponds to a 99% confidence interval.

For our data we obtain the following P_{value} matrix

$$P_{\text{value}} = \begin{bmatrix} 0.0028 & 0.0059 & 0.8413 \\ 0 & 0 & 0 \\ 0.0020 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.5606 & 0.1956 & 0 \end{bmatrix} \tag{20}$$

where the first column $P_{\text{value}}(j, 1)$ provides the p_{value} associated with the feature j [see Table 1] for the activities walking and slow fall (activity ID 1 and 2). The second column $P_{\text{value}}(j, 2)$ shows the p_{value} related to the activities walking and fast fall, while the third column $P_{\text{value}}(j, 3)$ represents the p_{value} for the activities slow and fast fall. Row j in the matrix P_{value} coincides with feature j $(j = 1, \ldots, 6)$. The list of feature names and their corresponding feature numbers are provided in Table 1.

Most of the elements of the matrix P_{value} are smaller than $\alpha = 0.01$ expect for $P_{\text{value}}(1, 3)$, $P_{\text{value}}(6, 1)$, and $P_{\text{value}}(6, 2)$ which are larger than α. The term $P_{\text{value}}(1, 3)$ represents the p_{value} associated with the mean value of the IDF for the activities slow fall and fast fall. Since $P_{\text{value}}(1, 3) > \alpha$, there is no significant statistical difference between the mean value of the IDF for the activities slow and fast fall. However, this first feature (i.e. the mean value of the IDF) allows us to distinguish walking from slow fall and walking from fast fall, since $P_{\text{value}}(1, 1)$ and $P_{\text{value}}(1, 2)$ are less than α.

The last row in the P_{value} matrix is associated with the last feature, i.e., the slope of the IDF after the fall. We note that this feature does not allow us to distinguish walking and fast fall as well as walking and slow fall, since there is no significant statistical difference between these activities in terms of the Feature 6. To explain this, let us introduce the following notation. Let us denote by I_W the interval indicating the range of values of Feature 6 associated with the activity walking. Similarly, let us denote by I_{SF} and I_{FF} the interval indicating the range of values of Feature 6 associated with the activities slow fall and fast fall, respectively.

In fact, for slow fall and fast fall, the value of this Feature 6 should be equal to zero. However, due to estimation errors of the IDF, we find that the mean value of Feature 6 is equal to 0.1867 for slow fall and -1.25 10^{-6} for fast fall, while the variance of Feature 6 is 0.6289 for slow fall and 7.39 10^{-6} for fast fall. For the activity walking, we find that the mean value and the variance of Feature 6 are equal to 0.3433 and 21.1, respectively. Based on these facts, we can conclude that both I_{SF} and I_{FF} are included in the interval I_W, i.e., $I_{SF} \subset I_W$ and $I_{FF} \subset I_W$.

This is confirmed by Fig. 8 that provides the histogram of Feature 6, i.e., the parameter $S_{\text{post-fall}}$ of the IDF, for the activities walking, slow fall, and fast fall. Fig. 8 shows that the range of values of $S_{\text{post-fall}}$ for slow fall and fast fall is included in the interval of values of $S_{\text{post-fall}}$ for the activity walking, i.e., $I_{SF} \subset I_W$ and $I_{FF} \subset I_W$. In other words, the values of the parameter $S_{\text{post-fall}}$ pertaining to the activities slow fall and fast fall cannot be distinguished from their counterpart associated with the activity walking. This explains why the values of $P_{\text{value}}(6, 1)$ and $P_{\text{value}}(6, 2)$ are larger than α.

Note that 15 elements of the matrix P_{value} are smaller than the significance level α. Thus, using the proposed features, the classifier should be able to distinguish the three considered activities (walking, slow fall, and fast fall) with a good accuracy.

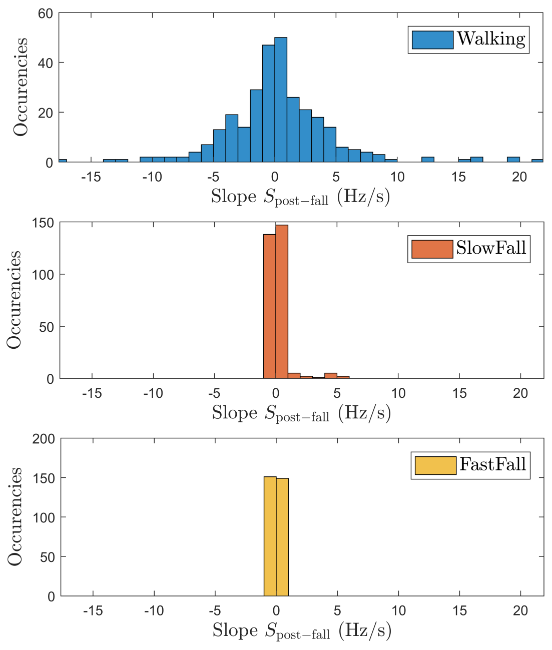


FIGURE 8. Histogram of the slope $S_{\text{post-fall}}$ of the IDF for the activities walking, slow fall, and fast fall.

C. CLASSIFICATION ALGORITHM

Classification algorithms follow a supervised learning approach to recognize different types of activities. The supervised learning approach consists of two phases: the training phase and test phase. First, the data is divided into buffers which are labeled with three activity IDs: 1, 2, and 3 corresponding to walking, slow fall, and fast fall, respectively. Each data buffer has a length of 4 s and contains the recording of the complex path gain, while a person is performing one of the three activities.

This data is pre-processed to remove the impact of fixed scatterers and to estimate the IDF, which contains the fingerprint of the user activity. We extract six features from the IDF and arrange them into two subsets: Subset A and Subset B. Subset A contains the mean value, variance, RMS, and maximum absolute value. Subset B comprises the features of Subset A augmented with the slopes of the IDF during fall and after fall. The feature vectors associated with Subset A and Subset B have a length of 4 and 6, respectively.

During the training phase, the classification algorithm is exposed to the labeled training data. This means that, for each data buffer in the training data, we provide the corresponding feature vector together with the activity ID. In this way, the classification algorithm can tune its internal parameters such that its recognition accuracy is maximized.

After the training phase, the trained classifier is exposed to the test data. In the test phase, for each buffer of the test data, the classifier is provided with the corresponding feature vector without the activity ID. The trained classifier uses the feature vector to determine the probability that the data buffer pertains to one of the three possible activities. If the activity class c $(c = 1, 2, 3)$ has the highest probability, the classifier decides that the performed activity has the ID c. The correctness of the classifier decision can be determined using the data buffer label. For each buffer in the test data, the classifier predicts the performed activity. At the end of the test phase, we can assess the performance of the trained classifier as well as the accuracy and precision of its predictions.

There are two main approaches for evaluating the performance of a classification algorithm, namely the holdout method and the cross-validation method. In the holdout method, the data is divided into two parts: the training data set and the testing data set. For instance, 70% of the data is used for training, while the remaining 30% are used for testing. In general, the samples belonging to the training set and the test set are chosen randomly. Thus, if we do two consecutive evaluations of the classifier's performance, we might obtain two different results, since the samples belonging to the training and test set have changed randomly in the second evaluation compared to the first one.

To obtain a more accurate performance evaluation, we use the cross-validation method. In this method, the data is divided into K folds. In the first iteration, the first fold is used for testing and the remaining $K-1$ folds are used for training. In the kth iteration, the k-th fold is used for testing, while the remaining $K-1$ folds are used for training. In each iteration, the performance of the classification algorithm is evaluated and then the average performance over the K iterations is computed. By using averaging, we ensure that the obtained performance using K-folds cross-validation is more accurate compared to the holdout method which is more prone to randomness.

In this paper, we evaluated the performance of four classification algorithms, namely KNN, decision tree, ANN, and cubic SVM. Principles and background information about these classification algorithms can be found in [24].

VI. EXPERIMENTAL RESULTS

The performance of the proposed activity recognition framework is evaluated in this section. In our performance evaluation, we use a 10-fold cross-validation. We assess the performance of four classification algorithms: KNN, decision tree, ANN, and cubic SVM. Moreover, we compare the performance of our proposed fall detection solution to existing fall detection systems in the literature.

A. KNN ALGORITHM

The KNN algorithm takes as input an unlabeled feature vector associated with a test example and aims to recognize the class of this test example. The KNN algorithm determines the K closest training examples to the test example in the multi-dimensional feature space. We use the Euclidian distance as a distance metric to identify the K closest neighbors. KNN measures the Euclidian distance between the feature vector of the test example and the feature vectors associated with all the training examples, and then selects the K nearest neighbors. Using a majority voting, KNN determines the most common class among the K closest neighbors and labels the test example with that class. In our investigation, the parameter K of the KNN classifier is set to 100.

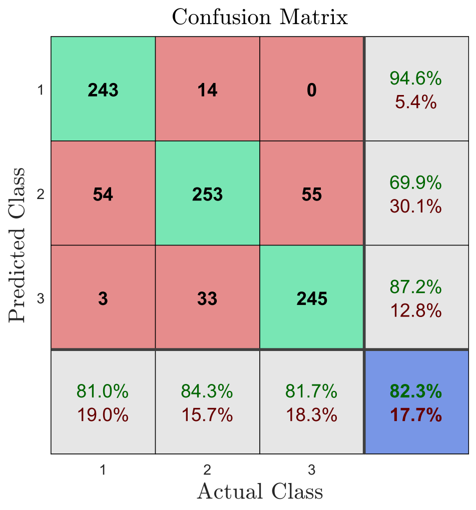


FIGURE 9. Confusion matrix of the KNN algorithm obtained using the features from Subset A.

The confusion matrix of the KNN algorithm is provided in Fig. 9. This confusion matrix is obtained using the features from Subset A. The labels 1, 2, and 3 at the bottom of the confusion matrix correspond to the activities walking, slow fall, and fast fall, respectively. While the labels 1, 2, and 3 at the bottom of the confusion matrix indicate the actual class, the labels 1, 2, and 3 on the side of the confusion matrix refer to the class predicted by the classifier. The diagonal elements of the matrix indicate the number of correct classifications, while the off-diagonal elements show the number of misclassified cases.

The first column contains the number of actual walking cases that are correctly classified as well as those that are misclassified. For instance, the top left element of the matrix indicates that in 243 cases the classifier predicted the activity correctly as walking. The second element of the first column indicates that in 54 cases, the KNN algorithm misclassified walking as slow fall. The KNN algorithm misclassified walking as fast fall in 3 cases as shown in the third row of the first column of the confusion matrix in Fig. 9. Based on these numbers, we can evaluate the classification accuracy for the activity walking to 81% as shown in the bottom cell of the first column. The first line of the confusion matrix illustrates the precision of the algorithm in predicting the activity walking. From the first line of the confusion matrix, we observe that in 14 cases, the KNN algorithm predicted the activity as walking, while the actual activity is slow fall. The activity walking is never misclassified as fast fall. This yields a classification precision of 94.6% for the activity walking.

The last column of the confusion matrix shows the classification precision of the algorithm, while the last row contains the classification accuracy of the algorithm. Note that the precision and the accuracy of the classification have two different meanings. The classification precision focuses on the predicted activity. For a particular activity, the precision

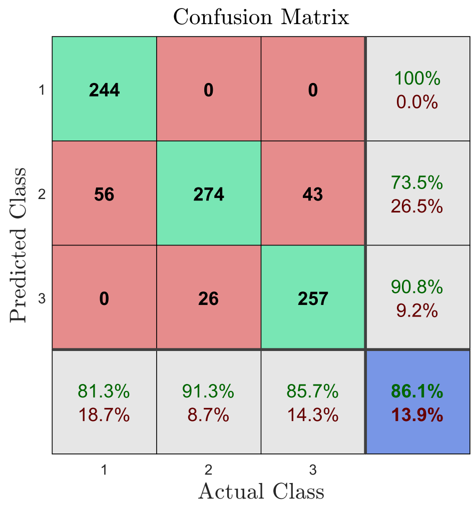


FIGURE 10. Confusion matrix of the KNN algorithm obtained using the features from Subset B.

quantifies the percentage of correct classifications out of the buffers *predicted* to belong to that certain activity. In contrast, the classification accuracy focuses on the actual activity and indicates the percentage of successful classifications out of the *actual* buffers belonging to a particular class.

For the KNN algorithm, we obtain a classification accuracy of 81%, 84.3%, and 81.7% for the activities walking, slow fall, and fast fall, respectively. The classification precision of the algorithm is equal to 94.6%, 69.9%, and 87.2% for the activities walking, slow fall, and fast fall, respectively. The overall accuracy of the classifier is equal to 82.3%

Fig. 10 illustrates the confusion matrix of the KNN algorithm using the features from Subset B. Compared to Fig. 9, where the features from Subset A are utilized, we notice that the overall accuracy increases from 82.3% to 86.1%. Moreover, the classification accuracy for the activities walking, slow fall, and fast fall is enhanced by 0.3%, 7%, and 4%, respectively, if the features from Subset B are used instead of the features from Subset A. Additionally, the precision in predicting the activities walking, slow fall, and fast fall is improved by 5.4%, 3.6%, and 3.6%, respectively, compared to Fig. 9.

In Table 2, we provide the confusion matrix of the binary classification problem, where we classify the data into fall and non-fall classes. The fall class encompasses the activities slow fall and fast fall, whereas the non-fall class includes the activity walking.

Table 2 is computed using the confusion matrix in Fig. 10. From the binary confusion matrix in Table 2, we can determine the number of false negative (FN) and false positive (FP) classifications as well as the FN rate and FP rate for fall detection. As shown in Table 2, the number of FP classifications equals 56 and the number of FN is 0. Table 2 indicates the number of true negative (TN) and the number of true positive (TP), which refer to the number of correctly classified non-falls and the number of correctly

TABLE 2. Confusion matrix of the binary classification problem of the KNN algorithm obtained using the features from Subset B.

	Actual Non-Fall	Actual Fall
Predicted Non-Fall	244 (**TN**)	0 (**FN**)
Predicted Fall	56 (**FP**)	600 (**TP**)

classified falls, respectively. The FP rate can be computed as

$$\text{FP Rate} = \frac{\text{FP}}{\text{Number of actual non-falls}} = \frac{\text{FP}}{\text{FP}+\text{TN}} = \frac{56}{300} = 18.66\%. \tag{21}$$

The FP rate shows the percentage of false alarms. If the KNN algorithm is given a non-fall event, it misclassifies it as a fall in 18.66% of the cases. The FP rate of the KNN algorithm is relatively high which is undesirable for a fall detection system. Later in this section, we will show that other classification algorithms can overcome this deficiency and reduce the rate of false alarms.

The FN rate indicates the percentage of undetected falls by the system. The FN rate can be computed by dividing the number of FN classifications by the number of actual falls, i.e.,

$$\text{FN Rate} = \frac{\text{FN}}{\text{Number of actual falls}} = \frac{\text{FN}}{\text{TP}+\text{FN}} = \frac{0}{600} = 0\%. \tag{22}$$

Since the FN rate is 0%, this implies that there are no undetected falls. It is desirable that the fall detection system has a very low FN rate.

Using Table 2, we can compute the accuracy and precision of fall detection as follows

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{600}{600} = 100\% \tag{23}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{600}{600+56} = 91.46\%. \tag{24}$$

B. DECISION TREE ALGORITHM

The decision tree algorithm is an interpretable machine learning algorithm, as opposed to the ANN and SVM algorithms, which are considered as black boxes. A decision tree is a tree where each node represents a feature, each branch contains a decision rule, and each leaf represents a classification outcome (a class). Several branches can emerge from a feature (a node), while no branch can come out from a leaf. There are two possible ways to build a decision tree: the first approach relies on the Gini index, while the second approach uses entropy function and information gain as metrics. In this paper, we use the first approach to create our decision tree. The first step for building the tree consists of finding the *root node*, which corresponds to the feature with the smallest Gini index. At each node, the process of determining the feature with the smallest Gini index is repeated until a leaf is reached for each branch. At this stage, no further expansion is needed, which implies that the building process of the

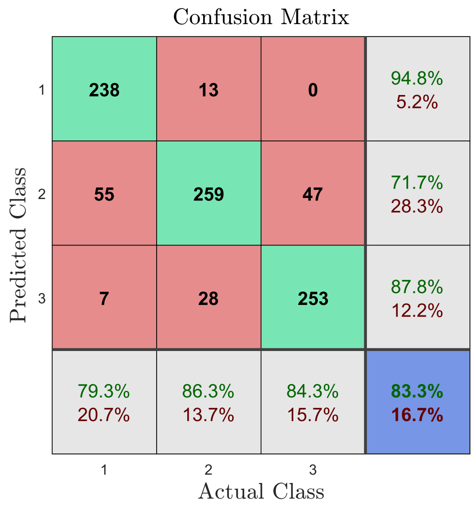


FIGURE 11. Confusion matrix of the decision tree algorithm obtained using the features from Subset A.

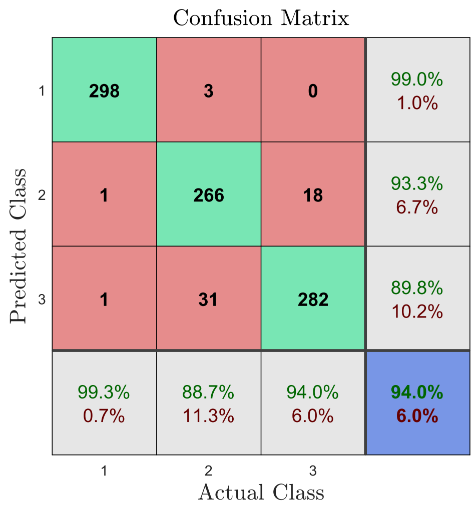


FIGURE 12. Confusion matrix of the decision tree algorithm obtained using the features from Subset B.

tree is complete. Note that the tree is built using the training data. After the training period, the built decision tree is used to classify the test data set.

Figs. 11 and 12 illustrate the confusion matrix of the decision tree algorithm when using the features from Subset A and B, respectively. We observe that the overall accuracy as well as the classification precision and accuracy per activity improve when utilizing the features from Subset B instead of those from Subset A. The overall accuracy increases from 83.3% to 94% by using the features from Subset B. The classification precision for the activities walking, slow fall, and fast fall is enhanced by 4.2%, 21.6%, and 2%, respectively. Similarly, the classification accuracy is improved by 20%, 2.4%, and 9.7% for the activities walking, slow fall, and fast fall, respectively, when using the features from Subset B instead of Subset A. By comparing Figs. 11 and 12, we notice that the number of misclassifications drop significantly. For instance, the number of walking cases that are misclassified as slow fall decreases from 55 to 1, while the number of fast falls that are misclassified as slow fall drops from 47 to 18, when using the features from Subset B.

Fig. 13 shows the decision tree of the trained classification algorithm obtained with the features from Subset B. From this tree we notice that the trained algorithm uses three features to predict the type of performed activity. These three features are the mean value of the IDF, the maximum of the absolute value of the IDF, and the IDF slope during fall, which correspond to the first, fourth, and fifth feature, respectively [see Table 1]. The first feature is denoted as Feat. 1 in Fig. 13. Similarly, the fourth and fifth features are referred to as Feat. 4 and Feat. 5, respectively.

From Fig. 13, we observe that the classification algorithm uses the following decision rules when predicting the performed activity.

1) If the value of Feature 5 (the IDF slope during fall) exceeds -10.47 Hz/s, the activity is classified as walking.
2) If Feature 5 is less than -27.27 Hz/s and Feature 4 larger than 13.96 Hz, then the activity is classified as fast fall.
3) If Feature 5 is less than -34.66 Hz/s and Feature 4 less than 13.96 Hz, then the activity is classified as fast fall.
4) If Feature 5 belongs to the interval [-34.66 Hz/s, -27.27 Hz/s) and Feature 4 less than 13.96 Hz, then the activity is classified as slow fall.
5) If Feature 5 belongs to the interval [-27.27 Hz/s, -10.47 Hz/s), Feature 4 larger than 22.25 Hz, and Feature 1 larger than -11.81 Hz, then the activity is classified as fast fall.
6) If Feature 5 belongs to the interval [-27.27 Hz/s, -10.47 Hz/s), Feature 4 larger than 22.25 Hz, and Feature 1 less than -11.81 Hz, then the activity is classified as slow fall.
7) If Feature 5 belongs to the interval [-27.27 Hz/s, -10.47 Hz/s), Feature 4 less than 22.25 Hz, and Feature 1 larger than 8.98 Hz, then the activity is classified as fast fall.
8) If Feature 5 belongs to the interval [-27.27 Hz/s, -10.47 Hz/s), Feature 4 less than 22.25 Hz, and Feature 1 smaller than 8.98 Hz, then the activity is classified as slow fall.

Table 3 shows the confusion matrix of the binary classification problem. From this table, we see that the number of FP and FN classifications is equal to 2 and 3, respectively. The FP rate can be computed using (21) and is equal to 0.66%. The FP rate coincides with the rate of false alarms. Compared with the KNN algorithm, the decision tree algorithm has a much lower FP rate. The FN rate captures the percentage of undetected falls by the system. The FN rate can be determined using (22) and is equal to 0.5%. The FN rate of the decision tree algorithm is larger than that of the KNN algorithm. In addition, the decision tree algorithm has a fall detection accuracy of 99.5% and a fall detection precision of 99.66%, which are computed using (23) and (24), respectively.

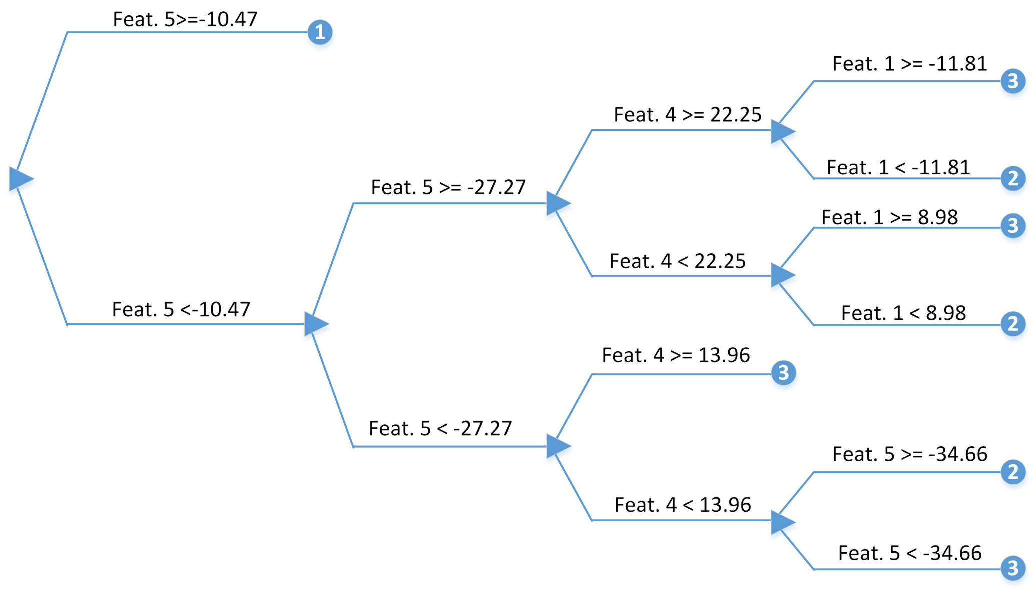


FIGURE 13. Decision tree of the trained model using the features from Subset B.

TABLE 3. Confusion matrix of the binary classification problem of the decision tree algorithm obtained using the features from Subset B.

	Actual Non-Fall	Actual Fall
Predicted Non-Fall	298 **(TN)**	3 **(FN)**
Predicted Fall	2 **(FP)**	597 **(TP)**

C. ANN ALGORITHM

The ANN classifier is a machine learning algorithm mimicking the functioning of the human brain. The ANN can be seen as a network of neurons. In this paper, the considered ANN algorithm comprises three layers: the input layer, the hidden layer, and the output layer. The hidden layer comprises 20 nodes. The hyperbolic tangent function is utilized as an activation function instead of the sigmoid function. It is well known that the use of the hyperbolic tangent function leads to a better performance compared with the sigmoid activation function. During the training phase, the optimal weights for different branches of the neural network are determined using the scaled conjugate gradient backpropagation algorithm.

Fig. 14 shows the confusion matrix of the ANN algorithm obtained with the features from Subset A. The overall accuracy reaches 94.1%, while the classification precision for the activities walking, slow fall, and fast fall is 96.9%, 90.4%, and 95.4%, respectively. The classification accuracy equals 92.3%, 94%, and 96% for the activities walking, slow fall, and fast fall, respectively. When using the features from Subset B instead of A, the overall accuracy is enhanced by 4.8% as illustrated in Fig. 15. Whereas, the classification precision is improved by 7.4% for walking, 4.3% for slow fall, and 2.7% for fast fall. The classification accuracy of the activities walking, slow fall, and fast fall is enhanced by 2.8%, 8.3%, and 2.9%, respectively, when using the features from Subset B.

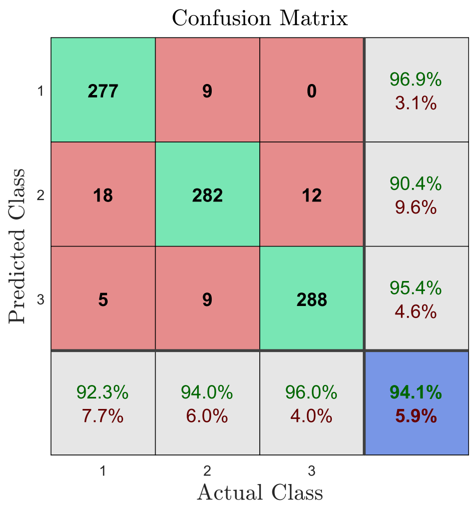


FIGURE 14. Confusion matrix of the ANN algorithm obtained using the features from Subset A.

In Table 4, we provide the confusion matrix of the binary classification problem. From this table, we see that the number of FP classifications is 1, while the FP rate is 0.33%. The number of FN classifications equals 1, whereas the FN rate is 0.16%. The ANN algorithm outperforms the decision tree algorithm by achieving lower FP and FN rates. On the other hand, the ANN algorithm has a fall detection accuracy and precision of 99.83%.

D. CUBIC SVM ALGORITHM

The first SVM algorithm was developed for binary classification problems. Subsequently, two approaches were proposed to extend SVM to multi-class classification problems: (i) the one-versus-all approach and (ii) the one-versus-one

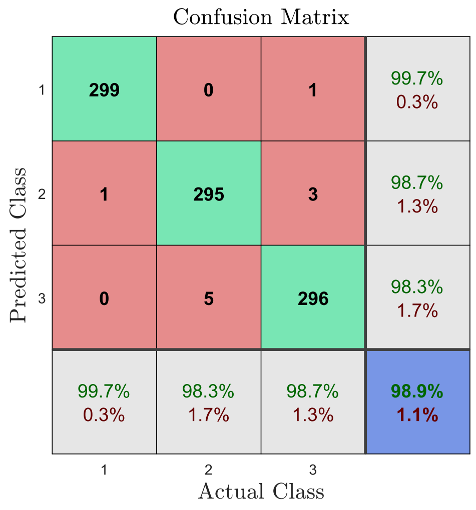


FIGURE 15. Confusion matrix of the ANN algorithm obtained using the features from Subset B.

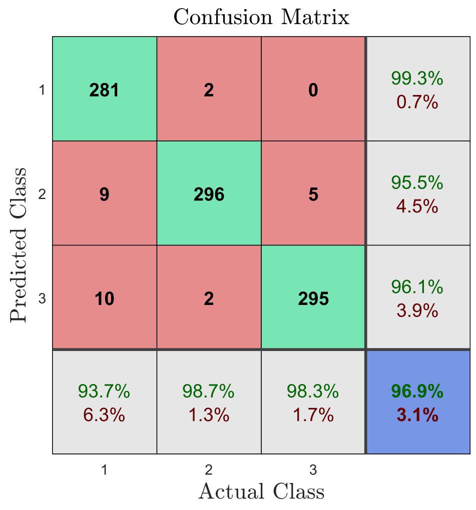


FIGURE 16. Confusion matrix of the cubic SVM algorithm obtained using the features from Subset A.

TABLE 4. Confusion matrix of the binary classification problem of the ANN algorithm obtained using the features from Subset B.

	Actual Non-Fall	Actual Fall
Predicted Non-Fall	299 (**TN**)	1 (**FN**)
Predicted Fall	1 (**FP**)	599 (**TP**)

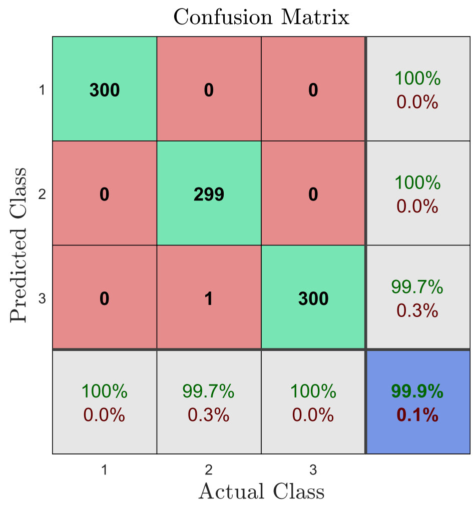


FIGURE 17. Confusion matrix of the cubic SVM algorithm obtained using the features from Subset B.

approach. In this work, the one-versus-all strategy is adopted. This strategy consists of building 3 SVM models. The first SVM model is trained with the walking data samples as positive points and all the other samples (slow falls and fast falls) as negative points. The second and third SVM models can recognize slow and fast falls, respectively. In SVM, the aim is to find the optimal boundary between different classes. This objective is achieved by first finding the support vectors for each class, and then by determining the boundary that maximizes the margin around the separating hyperplanes. This approach is efficient if the classes are linearly separable. Otherwise, a transformation is applied to the data, which allows to map it to a higher dimensional space where the classes are linearly separable. A kernel function defines in this case the inner product in the transformed higher dimensional space. In this section, the cubic SVM algorithm is used. For this algorithm, the kernel function is a polynomial of order 3.

The confusion matrix of the cubic SVM algorithm is provided in Fig. 16. This matrix is obtained using the features from Subset A. From Fig. 16, we observe that the overall accuracy of the cubic SVM algorithm reaches 96.9%. The classification accuracy for the activities walking, slow fall, and fast fall is 93.7%, 98.7%, and 98.3%, respectively. The classification precision equals 99.3%, 95.5%, and 96.1% for walking, slow fall, and fast fall, respectively.

By utilizing the features from Subset B together with the cubic SVM algorithm, we obtain the confusion matrix provided in Fig. 17. From this figure, we conclude that the use of the features from Subset B instead of those from Subset A improves the overall accuracy by 3%, while the classification precision is enhanced by 0.7%, 4.5%, and 3.6% for walking, slow fall, and fast fall, respectively. Similarly, in terms of the classification accuracy, we observe an improvement by 6.3%, 1%, and 1.7% for walking, slow fall, and fast fall, respectively. It has to be noted that when using the cubic SVM algorithm together with the features from Subset B, the overall accuracy, the classification precision, and the classification accuracy per activity are very close to 100% which is the best achievable performance.

In Table 5, we provide the confusion matrix of the binary classification problem. From this table, we see that the number of FP and FN classifications are both equal to zero. Consequently, the FP rate and the FN rate are both equal to 0%. This implies that the system has 0 false alarm, i.e., a walking activity is never misclassified as fall. Moreover, the number of undetected falls is equal to zero, i.e., a fall event is never misclassified as non-fall. The fall detection accuracy and precision reach both 100%. In this way, the cubic SVM algorithm

TABLE 5. Confusion matrix of the binary classification problem of the cubic SVM algorithm obtained using the features from Subset B.

	Actual Non-Fall	Actual Fall
Predicted Non-Fall	300 **(TN)**	0 **(FN)**
Predicted Fall	0 **(FP)**	600 **(TP)**

TABLE 6. Comparison of the overall accuracy of the four classification algorithms.

		Overall Accuracy %	
		Subset A	Subset B
Algorithms	KNN	82.3	86.1
	Decision Tree	83.3	94
	ANN	94.1	98.9
	Cubic SVM	96.9	99.9

achieves the best possible performance in terms of FN rate, FP rate, fall detection accuracy, and fall detection precision.

E. COMPARISON

In this section, we compare the performance of the four classification algorithms: KNN, decision tree, ANN, and cubic SVM. The comparison of these four algorithms in terms of overall accuracy is provided in Table 6. From this table, we see that the overall accuracy improves when using the features from Subset B instead of Subset A. This performance improvement is equal to 3.8% for KNN, 10.7% for decision tree, 4.8% for ANN, and 3% for cubic SVM. The best performance is archived by the cubic SVM algorithm with an overall accuracy of 99.9%, while the worst performance is obtained when using the KNN algorithm with an overall accuracy of 86.1%. When using the features from Subset B, the decision tree algorithm outperforms KNN by 7.9% in terms of overall accuracy, while ANN outperforms decision tree by 4.9%, and cubic SVM outperforms ANN by 1%.

When considering the binary problem of classifying fall and non-fall activities based on the features from Subset B, we obtain the following results for the FP and FN rates. The FP rate for KNN, decision tree, ANN, and cubic SVM is 18.66%, 0.66%, 0.33%, 0%, respectively. The FN rate for KNN, decision tree, ANN, and cubic SVM is 0%, 0.5%, 0.16%, 0%, respectively. The FP rate indicates the rate of false alarms, while the FN rate coincides with the rate of undetected falls. For a reliable fall detection system both the FN and FP rate must be close to 0%. We note that the cubic SVM algorithm achieves the best performance among the four algorithms with 0% FN and FP rates, which is actually the best achievable performance by any fall detection system.

In the following, we compare the performance of our proposed fall detection system to other exiting solutions in the literature. The systems that we consider as benchmarks can be categorized into wearable-based and RF-based fall detection systems. The comparison with wearable-based fall detection system is motivated by the fact that these systems are mature and can archive good performance. Many studies have investigated the performance of different fall detection algorithms using acceleration data [25], [26]. We compare the performance of our proposed machine learning framework to [25] and [26]. The choice of these two works as a benchmark is due to their high fall detection accuracy and precision.

TABLE 7. Confusion matrix of the binary classification problem of the cubic SVM algorithm obtained using the features from Subset A.

	Actual Non-Fall	Actual Fall
Predicted Non-Fall	281 **(TN)**	2 **(FN)**
Predicted Fall	19 **(FP)**	598 **(TP)**

In [25], the acceleration data is collected by two sensors attached to the participants' chests and thighs. From this collected data a feature vector of length 4 is created and provided to a decision tree algorithm. The considered classification problem in [25] is binary where fall and non-fall activities must be distinguished. The performance in terms of fall detection reaches an accuracy of 92% and a precision of 81%. In our case, the feature vector associated with Subset A has a length of 4, which is the same size as the feature vector in [25]. Thus, it would be fair to compare the results obtained with the features from Subset A to the results in [25]. In Table 7, we provide the confusion matrix of the binary classification problem obtained with the cubic SVM algorithm and the features from Subset A. Using (23), (24), and Table 7, we can compute the fall detection accuracy and precision, which are equal to 99.66% and 96.92%, respectively. Clearly, our solution outperforms the one proposed in [25] in terms of both the fall detection accuracy and precision by 7.66% and 15.92%, respectively.

In [26], the authors built a binary classifier which can distinguish between fall and non-fall events. A feature vector of length 23 was provided to the classifier to decide if a fall has occurred or not. The performance of three classification algorithms was evaluated, namely decision tree, logistic regression, and multilayer perceptron. The best performance in [26] was achieved with the multilayer perceptron classifier which has a fall detection accuracy of 93.5% and a precision of 94.2%. In our case, the cubic SVM algorithm used together with the features from Subset B reaches a fall detection accuracy of 100% and precision of 100%, which is the best achievable performance. Compared to [26], our solution enhances both the fall detection accuracy and precision by 6.5% and 5.8%, respectively. Note as well that the size of our feature vector is much smaller than that used in [26], which implies that our solution has a lower implementation complexity and a better performance.

Additionally, we compare our results to those reported by RF-based systems. Actually, our proposed solution falls under the umbrella of RF-based fall detection systems. Therefore, it is of interest to compare our solution to RF-based fall detection systems. In [9], falls are detected based on the CSI obtained from a Wi-Fi network card. The CSI data is pre-processed then a feature vector containing seven features is built. Two classification algorithms were used to detect falls, namely One-class SVM and random forest. The authors of [9] reported only the fall detection precision and FP rate of their fall detection system. The One-class SVM algorithm has a fall detection precision of 90% and false alarm rate of 15%, while the random forest algorithm yields a 94% precision and 13% false alarm rate. In our case, we use six features and achieve a fall detection precision of 100% and false alarm rate of 0% when using the cubic SVM algorithm. Compared to the random forest classifier proposed in [9], our cubic SVM algorithm improves the fall detection precision by 6% and reduces the false alarm rate by 13%.

VII. CONCLUSION

We have developed an activity simulator that generates the complex path gain of indoor channels in the presence of one person performing three different activities: walking, slow fall, and fast fall. Using this activity simulator, we generated complex path gain data associated with 30 participants having different heights, speed profiles, and trajectories. By randomizing the height, speed, and trajectories of the participants within typical ranges, we ensure that the generated complex path gain data is generic and reflects the diversity observed in real-world measurements.

We have proposed a novel method to extract the IDF with high accuracy from the complex path gain. Moreover, we have developed a machine learning framework for activity recognition based on the IDF. We extracted six features from the IDF. Using hypothesis testing, we evaluated the usefulness of these features in distinguishing the three considered activities. With synthetic data of the RF-activity simulator, we evaluated the performance of four classification algorithms, namely KNN, decision tree, ANN, and cubic SVM, in recognizing the three considered activities. First, the algorithms were trained using the training data set, which allows tuning the internal parameters of the classifier such that their classification accuracy is maximized. Second, the trained algorithms are tested using a cross-validation method, which is known to produce accurate results compared to the holdout method.

Our investigation reveals that the cubic SVM algorithm has the best performance among the four tested algorithms. The overall accuracy for the cubic SVM reaches 99.9%, which allows it to outperform ANN by 1%, decision tree by 5.9%, and KNN by 13.8%. The classification precision of the cubic SVM algorithm for the activities walking, slow, fall, and fast fall is 100%, 100%, and 99.7%, respectively. The classification precision is 100% for walking, 99.7% for slow fall, and 100% for fast fall. In terms of fall detection, the cubic SVM achieves the best possible performance with 100% accuracy, 100% precision, 0% false alarm rate, and 0% of undetected fall rate. In addition, we have compared the performance of our system with that of existing fall detection systems in the literature and demonstrated that our system outperforms them in terms of accuracy and precision in fall detection. Besides, our system generates fewer false alarms and has a lower rate of undetected falls.

ACKNOWLEDGMENT

This article was presented at the 2019 IEEE Wireless Communication and Networking Conference (WCNC 2019), Marrakech, Morocco, April 2019.

REFERENCES

[1] G. Bergen, M. Stevens, and E. R. Burns. (Jun. 2018). *Falls and Fall Injuries Among Adults Aged ≥65 Years—United States, 2014*. [Online]. Available: https://www.cdc.gov/mmwr/volumes/65/wr/mm6537a2.htm. doi: 10.15585/mmwr.mm6537a2.

[2] Centers for Disease Control and Prevention, National Center for Injury Prevention and Control. (Aug. 2016). *Web-Based Injury Statistics Query and Reporting System (WISQARS)*. [Online]. Available: http://www.cdc.gov/injury/wisqars

[3] C. S. Florence, G. Bergen, A. Atherly, E. Burns, J. Stevens, and C. Drake, "Medical costs of fatal and nonfatal falls in older adults," *J. Amer. Geriatrics Soc.*, vol. 66, no. 4, pp. 693–698, Apr. 2018.

[4] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomed. Eng. OnLine*, vol. 12, no. 1, pp. 66, Dec. 2013.

[5] K. Sehairi, F. Chouireb, and J. Meunier, "Comparative study of motion detection methods for video surveillance systems," *J. Electron. Imag.*, vol. 26, no. 2, pp. 26–29, Apr. 2017.

[6] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, "SisFall: A fall and movement dataset," *Sensors*, vol. 17, no. 1, p. 198, Jan. 2017. doi: 10.3390/s17010198.

[7] A. Chelli and M. Pätzold, "Recognition of falls and daily living activities using machine learning," in *Proc. 29th IEEE Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Bologna, Italy, Sep. 2018, pp. 1–7.

[8] A. Chelli and M. Pätzold, "A machine learning approach for fall detection and daily living activity recognition," *IEEE Access*, vol. 7, pp. 38670–38687, 2019.

[9] Y. Wang, K. Wu, and L. M. Ni, "WiFall: Device-free fall detection by wireless networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 581–594, Feb. 2017.

[10] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller, "3D tracking via body radio reflections," in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implement.*, Seattle, WA, USA, vol. 14, Apr. 2014, pp. 317–329.

[11] M. Youssef, M. Mah, and A. Agrawala, "Challenges: Device-free passive localization for wireless environments," in *Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw. (MobiCom)*, Montreal, QC, Canada, Sep. 2007, pp. 222–229.

[12] M. Seifeldin, A. Saeed, A. E. Kosba, A. El-Keyi, and M. Youssef, "Nuzzer: A large-scale device-free passive localization system for wireless environments," *IEEE Trans. Mobile Comput.*, vol. 12, no. 7, pp. 1321–1334, Jul. 2013.

[13] H. Abdelnasser, M. Youssef, and K. A. Harras, "WiGest: A ubiquitous WiFi-based gesture recognition system," in *Proc. IEEE Conf. Comput. Commun.(INFOCOM)*, Hong Kong, Apr. 2015, pp. 1472–1480.

[14] F. Adib, D. Katabi, "See through walls with WiFi!" in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Hong Kong, Aug. 2013, pp. 75–86.

[15] A. Abdelgawwad and M. Pätzold, "A framework for activity monitoring and fall detection based on the characteristics of indoor channels," in *Proc. 87th IEEE Veh. Technol. Conf. (VTC Spring)*, Porto, Portugal, Jun. 2018, pp. 1–7.

[16] S.-U. Jung and M. S. Nixon, "Estimation of 3D head region using gait motion for surveillance video," in *Proc. 4th Int. Conf. Imag. Crime Detection Prevention (ICDP)*, London, U.K., Nov. 2011, pp. 1–6. doi: 10.1049/ic.2011.0105.

[17] T. Öberg, A. Karsznia, and K. Öberg, "Basic gait parameters: Reference data for normal subjects, 10-79 years of age," *J. Rehabil. Res. Develop.*, vol. 30, no. 2, pp. 210–223, 1993.

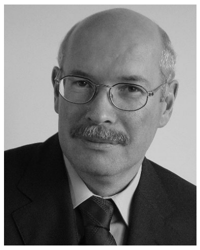
[18] A. K. Bourke, M. Torrent, X. Parra, A. Catala, and J. Nelson, "Fall algorithm development using kinematic parameters measured from simulated falls performed in a quasi-realistic environment using accelerometry," in *Proc. 35th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Boston, MA, USA, Aug. 2011, pp. 4449–4452.

[19] G. Wu, "Distinguishing fall activities from normal activities by velocity characteristics," *J. Biomech.*, vol. 33, no. 11, pp. 1497–1500, 2000.

[20] M. Pätzold and C. A. Gutierrez, "Modelling of non-WSSUS channels with time-variant Doppler and delay characteristics," in *Proc. 7th IEEE Int. Conf. Commun. Electron. (ICCE)*, Hue, Vietnam, Jul. 2018, pp. 1–6.

[21] A. Abdelgawwad and M. Pätzold, "On the influence of walking people on the Doppler spectral characteristics of indoor channels," in *Proc. 28th IEEE Int. Symp. Pers., Indoor, Mobile Radio Commun. Workshops (PIMRC)*, Montreal, QC, Canada, Oct. 2017, pp. 1–7.
[22] A. Williams and F. J. Taylor, *Electronic Filter Design Handbook*, 4th ed. New York, NJ, USA: McGraw-Hill, 2006.
[23] J. A. Rice, *Mathematical Statistics and Data Analysis*, 3rd ed. Boston, MA, USA: Thomson Brooks/Cole, 2007.
[24] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Cambridge, U.K.: Springer, 2006.
[25] O. Ojetola, E. I. Gaura, and J. Brusey, "Fall detection with wearable sensors–safe (smart fall detection)," in *Proc. 7th Int. Conf. Intell. Environ. (IE)*, Jul. 2011, pp. 318–321.
[26] I. P. E. S. Putra, J. Brusey, and E. Gaura, "A cascade-classifier approach for fall detection," in *Proc. 5th EAI Int. Conf. Wireless Mobile Commun. Healthcare (MOBIHEALTH)*, London, U.K., Oct. 2015, pp. 94–99.

ALI CHELLI (S'08–M'12) was born in Sfax, Tunisia. He received the B.Sc. degree in communications from the Ecole Superieure des Communications de Tunis (SUP'COM), in 2005, and the M.Sc. and Ph.D. degrees in information and communication technology from the University of Agder, Norway, in 2007 and 2013, respectively. He was a Postdoctoral Fellow with the King Abdullah University of Science and Technology (KAUST) and with the Norwegian Institute of Science and Technology (NTNU), Trondheim, Norway. He is currently a Postdoctoral Fellow with the University of Agder. His current research interests include machine learning, wireless communication theory, cooperative relaying, vehicle-to-vehicle communications, game theory, and channel modeling. He has been serving as a TPC member and a TPC chair in several conferences.



MATTHIAS PÄTZOLD (M'94–SM'98) received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from Ruhr University Bochum, Bochum, Germany, in 1985 and 1989, respectively, and the Habilitation degree in communications engineering from the Hamburg University of Technology, Hamburg, Germany, in 1998. From 1990 to 1992, he was with ANT Nachrichtentechnik GmbH, Backnang, Germany, where he was involved in digital satellite communications. From 1992 to 2001, he was with the Department of Digital Networks, Hamburg University of Technology. Since 2001, he has been a Full Professor of mobile communications with the University of Agder, Norway. He has authored several books and numerous technical articles. He has been actively participating in numerous conferences serving as a TPC member and the TPC chair. His publications received 14 best paper awards.